Sur Documentation

Release 1.1.dev+d99744b

Phasety

August 31, 2016

Contents

1 Contents **3**
- 1.1 Install . . . 3
- 1.2 Sur crash course . . . 3
- 1.3 Advanced examples . . . 12
- 1.4 Setup a development enviroment . . . 31

2 API reference **35**
- 2.1 sur.apps module . . . 35
- 2.2 sur.envelope module . . . 35
- 2.3 sur.envelope_sp module . . . 35
- 2.4 sur.eos module . . . 35
- 2.5 sur.manage module . . . 36
- 2.6 sur.models module . . . 36
- 2.7 sur.plots module . . . 36
- 2.8 sur.settings module . . . 36
- 2.9 sur.tools module . . . 36
- 2.10 sur.units module . . . 36
- 2.11 Module contents . . . 36

3 Indices and tables **37**

Python Module Index **39**

Sur is python library to calculate and plot envelopes and flashes for multicompound mixtures using equation of states.

It have been designed to be easy to use interactively, for example using Jupyter, although it could be used as a "back-end" to develop your own programs.

CHAPTER 1

Contents

1.1 Install

Sur is still under development (considered in *alpha* stage), but it's ready to experiment with.

Attention: By the moment, Sur only works with Python 2.7.x. It may change in future versions.

1.1.1 For Windows users

Under Windows, Sur is easily installable via a recent version of [pip](https://pip.pypa.io/en/stable/). It's bundle in recents version of Python.

- Open a command line (Start » Run » `cmd.exe`) and first, upgrade the version of pip (just in case):

```
pip install -U pip
```

- Then, in a 32bits OS:

```
pip install https://ci.appveyor.com/api/buildjobs/satlj7c3vteww2wi/artifacts/dist/sur-1.0.postf6
```

- Or for 64 bits:

```
pip install https://ci.appveyor.com/api/buildjobs/0yyuyr7mb4skhk1l/artifacts/dist/sur-1.0.postf6
```

Note: Those links are for the last stable (but still *in develop*) versions. You can try newer ones going to https://ci.appveyor.com/project/mgaitan/envelope-sur » Enviroment (x86 or x64) » Artifacts, and copy the the link to the `.whl` file)

1.1.2 For Linux users

(to do)

1.2 Sur crash course

This is a tutorial to dive in the key concepts of Sur and its usage under an interactive session.

1.2.1 Getting up

Sur has an object-oriented design. This means that the library provides *classes* to interact with. The very first step is to import the classes and some helper functions we will need.

```
In [1]: from sur import Mixture, Compound, EosSetup, EosEnvelope, EosFlash, ExperimentalEn
```

We also need to create the database and load the built-in dataset of pure components constants. By default, Sur uses a database in memory, which is not persistence.

```
In [2]: setup_database()
```

We are ready to start.

1.2.2 Define a mixture

As you know, a mixture is two or more compound which have been combined such that each substance retains its own chemical identity. A `Mixture` in sur is the same: a combination of compounds, each one with its fraction (i.e, the sum of the fractions of all the compounds in the mixture must be 1)

```
In [3]: mixture = Mixture()
```

There are several ways to add compounds to a `Mixture` instance. For instance, you can use as if it would be a dictionary.

```
In [4]: mixture["co2"] = 0.5
        mixture["n-decane"] = 0.25
        mixture["methane"] = 0.25
```

In order to facilitate the definition of interaction matrixes, we could sort each fraction by its molecular weight, no matter the order we added them.

```
In [5]: mixture.sort()
```

```
In [6]: mixture
```

```
Out[6]: [(<Compound: METHANE>, Decimal('0.25')), (<Compound: CARBON DIOXIDE>, Decimal('0.5
```

1.2.3 Setup the EoS

To simulate the envelope, we need to choice and configurate an Equation of State. This is done via an `EosSetup` instance. For example, to use an RK-PR EoS with parameters k_{ij} and l_{ij} as constants, we create a setup object like this:

```
In [7]: setup = EosSetup.objects.create(eos='RKPR', kij_mode=EosSetup.CONSTANTS, lij_mode=
```

By default, the interaction parameter between two compounds is 0.0 (however, Sur may provide a better choice when appropiated). To customize the calculation with our own parameters, we can override the default values.

```
In [8]: setup.set_interaction('kij', 'methane', 'co2', 0.1)
        setup.set_interaction('kij', 'co2', 'n-decane',  0.091)
```

```
Out[8]: <KijInteractionParameter: RKPR [<Compound: CARBON DIOXIDE>, <Compound: n-DECANE>]:
```

A setup object is independent of a mixture. It just define the equation to be used and the *bag of parameters* to tune it when we simulate an envelope for a particular mixture, but to have interaction between compounds that don't belong to a mixture is perfectly valid.

After set the interaction parameter, we probably want to see how the interaction matrix looks like for our particular compounds

```
In [9]: setup.kij(mixture)
```

```
Out[9]: array([[ 0.   ,  0.1  ,  0.   ],
               [ 0.1  ,  0.   ,  0.091],
               [ 0.   ,  0.091,  0.   ]])
```

By the way, it'd be also possible to define the whole matrix for a particular mixture at once, using the method `setup.set_interaction_matrix()`

1.2.4 Simulate the envelope

Having the mixture and the setup, we are able to obtain a the simulated envelope.

```
In [10]: envelope = mixture.get_envelope(setup)
```

In this case, `envelope` is an instance of `EosEnvelope`, i.e. an envelope that have been created via an equation of state.

Any kind of envelope has many attributes ready to be inspected. For instance, we have the array of pressure and temperature

```
In [11]: envelope.p
```

```
Out[11]: array([  1.44200000e-02,   1.64600000e-02,   2.01600000e-02,
                  2.45700000e-02,   2.97800000e-02,   3.59200000e-02,
                  4.31100000e-02,   5.14700000e-02,   6.11500000e-02,
                  7.22700000e-02,   8.49900000e-02,   9.94400000e-02,
                  1.15800000e-01,   1.34100000e-01,   1.54500000e-01,
                  1.77100000e-01,   2.02000000e-01,   2.29200000e-01,
                  2.58800000e-01,   2.90700000e-01,   3.24900000e-01,
                  4.05600000e-01,   5.23400000e-01,   6.71100000e-01,
                  8.54900000e-01,   1.08200000e+00,   1.36100000e+00,
                  1.70100000e+00,   2.11100000e+00,   2.60400000e+00,
                  3.19200000e+00,   3.88700000e+00,   4.70400000e+00,
                  5.65700000e+00,   6.75900000e+00,   8.02800000e+00,
                  9.47500000e+00,   1.11200000e+01,   1.29700000e+01,
                  1.50400000e+01,   1.73400000e+01,   1.98900000e+01,
                  2.26900000e+01,   2.57500000e+01,   2.90700000e+01,
                  3.26600000e+01,   3.65200000e+01,   4.06600000e+01,
                  4.50600000e+01,   4.97300000e+01,   5.46600000e+01,
                  5.98400000e+01,   6.52700000e+01,   7.09400000e+01,
                  7.71600000e+01,   8.40100000e+01,   9.15900000e+01,
                  1.00000000e+02,   1.09400000e+02,   1.18800000e+02,
                  1.27600000e+02,   1.35800000e+02,   1.43600000e+02,
                  1.50900000e+02,   1.54400000e+02,   1.64100000e+02,
                  1.70000000e+02,   1.75400000e+02,   1.80400000e+02,
                  1.84800000e+02,   1.88700000e+02,   1.92100000e+02,
                  1.95000000e+02,   1.97400000e+02,   1.99300000e+02,
                  2.00700000e+02,   2.01200000e+02,   2.01600000e+02,
                  2.01800000e+02,   2.02000000e+02,   2.02000000e+02,
                  2.01600000e+02,   2.01200000e+02,   2.00700000e+02,
                  2.00000000e+02,   1.99200000e+02,   1.98200000e+02,
                  1.97100000e+02,   1.95900000e+02,   1.94600000e+02,
                  1.93300000e+02,   1.90200000e+02,   1.88200000e+02,
                  1.86100000e+02,   1.83900000e+02,   1.81700000e+02,
                  1.79400000e+02,   1.74600000e+02,   1.72000000e+02,
                  1.66500000e+02,   1.60900000e+02,   1.55200000e+02,
```

```
                  1.49500000e+02,   1.43900000e+02,   1.38400000e+02,
                  1.33100000e+02,   1.28000000e+02,   1.23000000e+02,
                  1.18400000e+02,   1.13900000e+02,   1.09600000e+02,
                  1.05500000e+02,   1.01700000e+02,   9.79600000e+01,
                  9.44300000e+01,   9.10500000e+01,   8.78100000e+01,
                  8.47000000e+01,   8.17100000e+01,   7.88300000e+01,
                  7.60400000e+01,   7.33500000e+01,   7.07400000e+01,
                  6.82100000e+01,   6.57500000e+01,   6.33700000e+01,
                  6.10400000e+01,   5.87900000e+01,   5.65900000e+01,
                  5.44400000e+01,   5.23600000e+01,   5.03300000e+01,
                  4.83500000e+01,   4.64300000e+01,   4.45600000e+01,
                  4.27400000e+01,   4.09700000e+01,   3.92500000e+01,
                  3.75900000e+01,   3.59700000e+01,   3.44000000e+01,
                  3.28800000e+01,   3.14100000e+01,   2.99800000e+01,
                  2.86000000e+01,   2.72700000e+01,   2.59800000e+01,
                  2.47400000e+01,   2.35400000e+01,   2.23900000e+01,
                  2.12800000e+01,   2.02100000e+01,   1.91800000e+01,
                  1.81900000e+01,   1.72400000e+01,   1.63300000e+01,
                  1.54500000e+01,   1.46200000e+01,   1.38100000e+01,
                  1.30500000e+01,   1.23100000e+01,   1.16100000e+01,
                  1.09400000e+01,   1.03000000e+01,   9.69100000e+00,
                  9.11100000e+00,   8.55800000e+00,   8.03200000e+00,
                  7.53200000e+00,   7.05800000e+00,   6.60800000e+00,
                  6.18100000e+00,   5.77700000e+00,   5.39400000e+00,
                  5.03200000e+00])
In [12]: envelope.t
Out[12]: array([ 310.    ,  312.0833,  315.3334,  318.5817,  321.8256,  325.0618,
                 328.287 ,  331.4979,  334.6908,  337.8618,  341.0071,  344.1223,
                 347.2033,  350.2456,  353.2444,  356.1952,  359.0929,  361.9326,
                 364.7092,  367.4176,  370.0526,  375.4492,  381.889 ,  388.4249,
                 395.0497,  401.7551,  408.5312,  415.3667,  422.2482,  429.1612,
                 436.0893,  443.0145,  449.9174,  456.7773,  463.5724,  470.2797,
                 476.8758,  483.3365,  489.6373,  495.7541,  501.6625,  507.3389,
                 512.7602,  517.904 ,  522.749 ,  527.2748,  531.462 ,  535.2922,
                 538.7482,  541.8137,  544.4731,  546.7121,  548.5169,  549.8788,
                 550.8162,  551.236 ,  551.015 ,  549.9858,  547.9151,  544.911 ,
                 541.2341,  536.9976,  532.2864,  527.1669,  524.4715,  515.914 ,
                 509.8688,  503.5958,  497.1301,  490.5047,  483.751 ,  476.8995,
                 469.9794,  463.0191,  456.0457,  449.0857,  445.4863,  441.9006,
                 438.332 ,  434.7837,  431.2588,  424.291 ,  420.3194,  416.3945,
                 412.5202,  408.7   ,  404.9371,  401.2346,  397.5952,  394.0213,
                 390.5151,  383.7132,  379.7462,  375.8867,  372.1359,  368.4946,
                 364.9629,  358.2265,  354.6604,  347.9187,  341.4057,  335.2389,
                 329.4222,  323.9512,  318.8144,  313.9956,  309.4744,  305.2285,
                 301.2344,  297.4685,  293.9078,  290.5301,  287.3146,  284.2421,
                 281.2947,  278.4564,  275.7124,  273.0497,  270.4565,  267.9224,
                 265.4384,  262.9962,  260.589 ,  258.2107,  255.8562,  253.5209,
                 251.2014,  248.8943,  246.5974,  244.3085,  242.0261,  239.7491,
                 237.4765,  235.2078,  232.9426,  230.681 ,  228.423 ,  226.1689,
                 223.9189,  221.6736,  219.4335,  217.1993,  214.9716,  212.7511,
                 210.5385,  208.3345,  206.1399,  203.9553,  201.7815,  199.6192,
                 197.4689,  195.3314,  193.2072,  191.097 ,  189.0011,  186.9202,
                 184.8547,  182.8049,  180.7714,  178.7545,  176.7545,  174.7718,
```

```
                172.8065,  170.8589,  168.9294,  167.0179,  165.1248,  163.2502,
                161.3942,  159.5569,  157.7385,  155.9389,  154.1583,  152.3968,
                150.6543])
```

There is also equivalent arrays for critical points, if any. In our example there is just one point.

```
In [13]: envelope.t_cri, envelope.p_cri
```

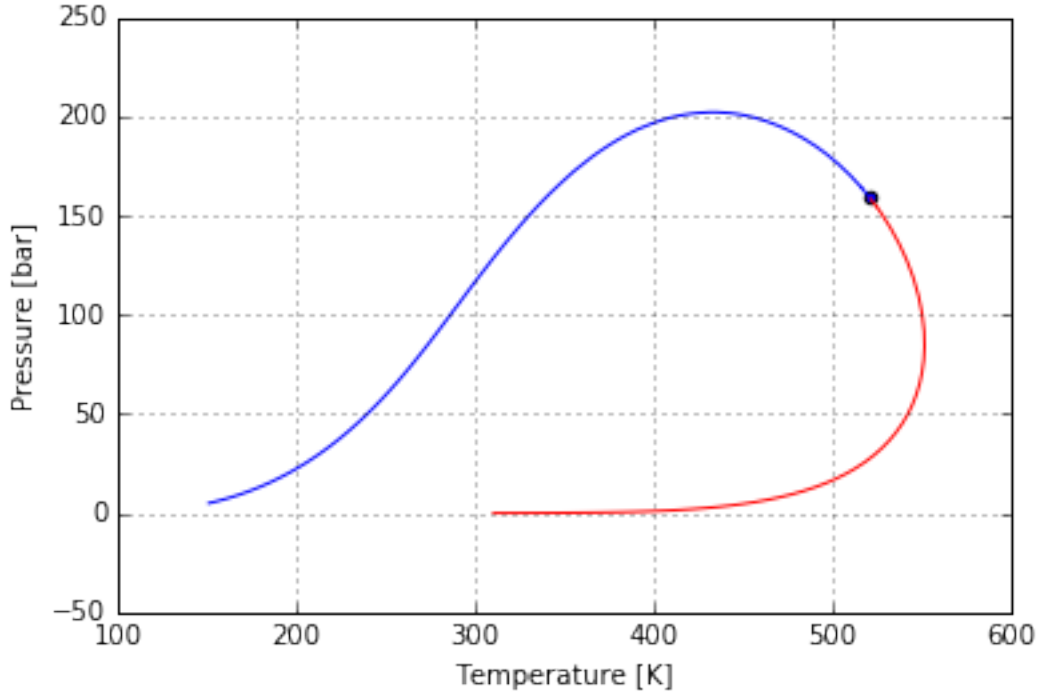
```
Out[13]: (array([ 520.3626]), array([ 159.1]))
```

1.2.5 Plotting

In addition, an envelope object is able to generate a predefined Matplotlib figure. Before run it, if you are using Sur through Jupyter, may be handy to use the *magic command* to embed matplotlib figures directly in the document (instead of raise a popup window)

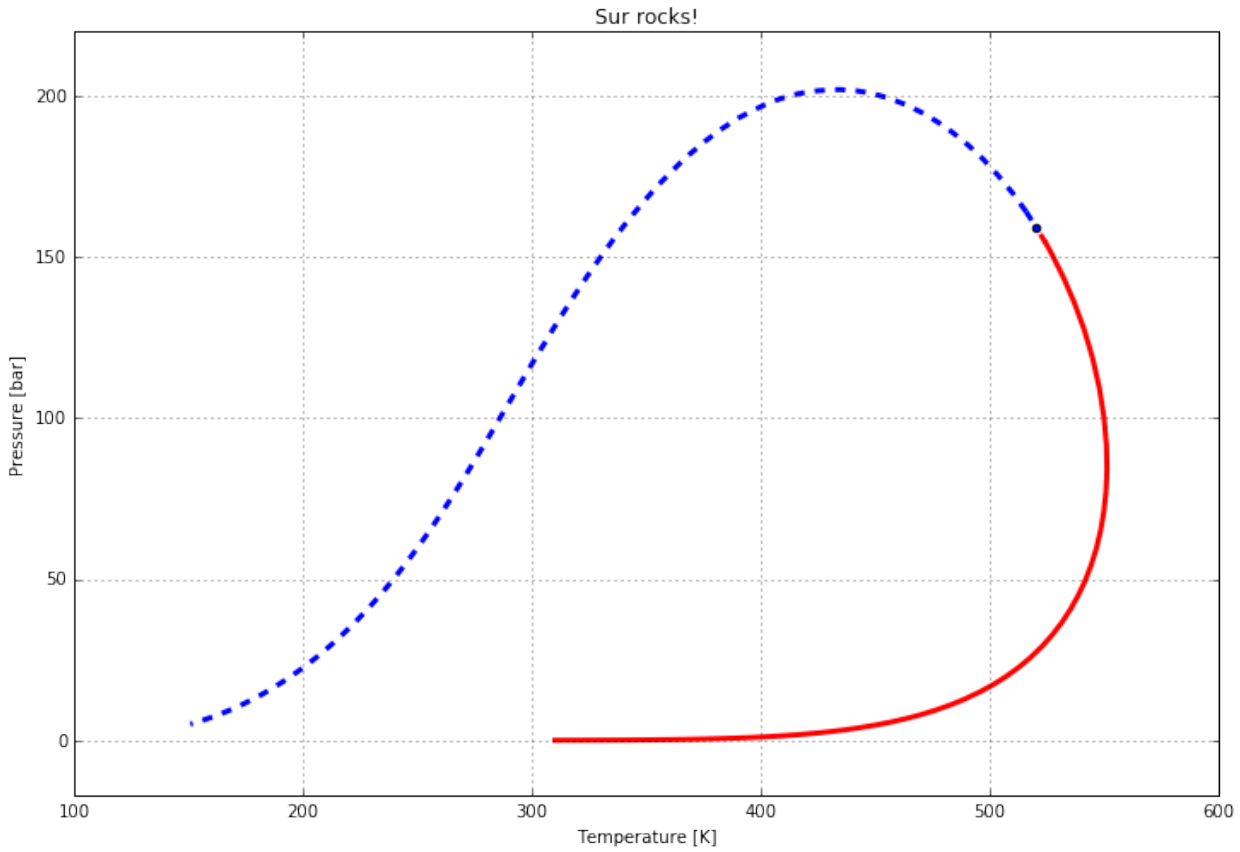
```
In [14]: %matplotlib inline
```

```
In [15]: fig = envelope.plot()
```



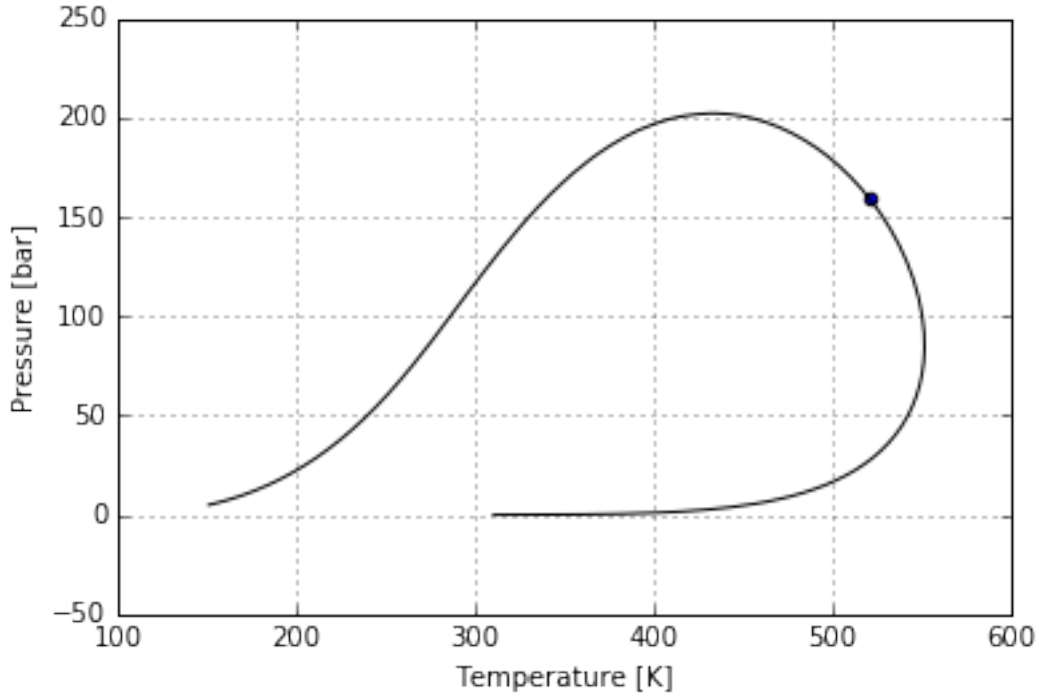
As we saved the figure object (`fig`), we can use the matplotlib API to manipulate it (changing aspect, size, span, colors..., in fact, anything we want), export, etc.

```
In [16]: from matplotlib import pyplot as plt

         ax = fig.get_axes()[0]
         ax.set_yticks([0] + ax.get_yticks()[1:])
         ax.set_ylim((-50/3, 220))
         ax.set_title('Sur rocks!')
         plt.setp(ax.lines, linewidth=3)
         plt.setp(ax.lines[1:], linestyle='--')
         fig.set_size_inches(fig.get_size_inches() * 2)
         fig
```

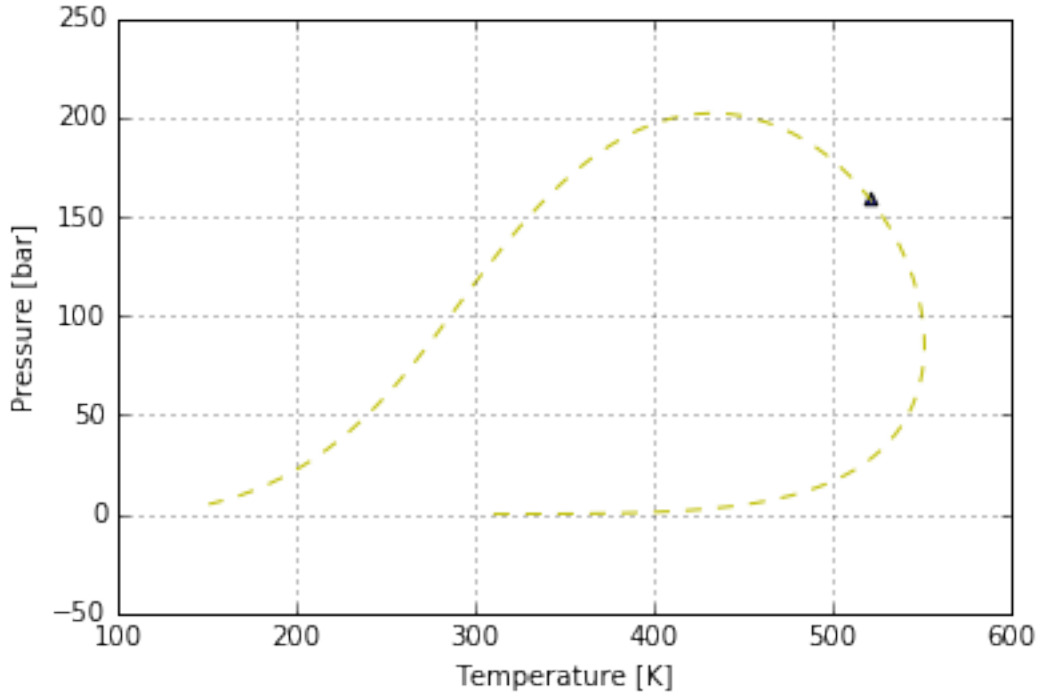


However, if that is too much power to your goal, you can pass some format directives (as in pyplot's `'plot()` <http://matplotlib.org/users/pyplot_tutorial.html>`__) directly to the method's parameters. For example:

```
In [17]: envelope.plot(format='k');
```




```
In [18]: envelope.plot(critical_point='^', format='--y');
```



1.2.6 Compare with experimental data

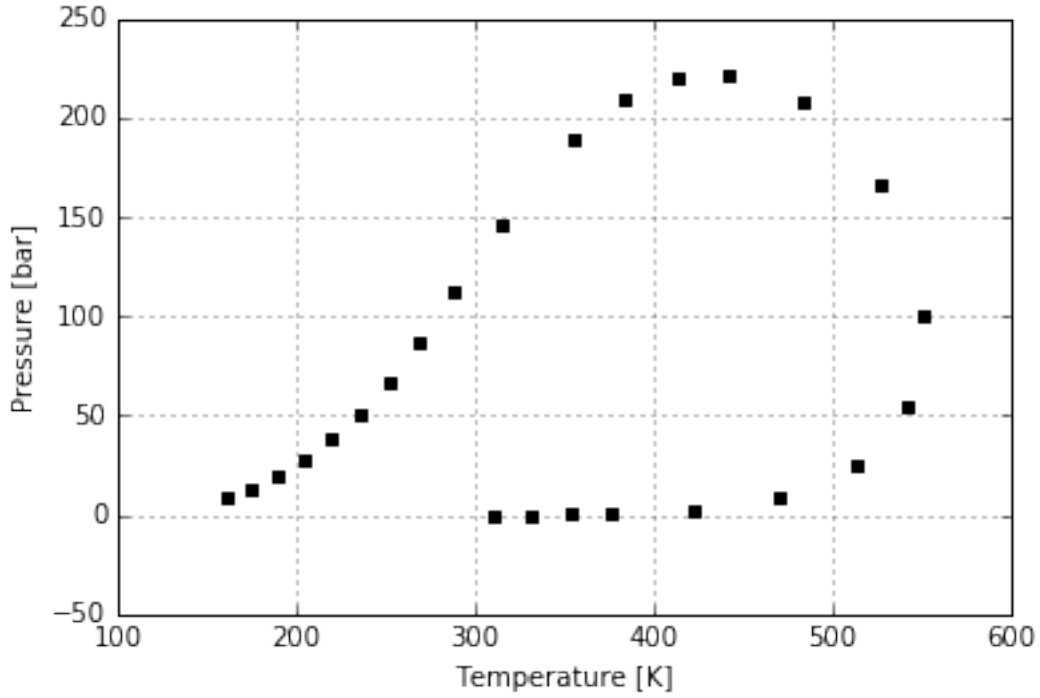
Constrant the simulation against a known experimental dataset for an evelope is possible. Here we'll mock this experimental data modifing the simulated one, but you should get it from a reliable source

```
In [19]: import numpy as np
         n = envelope.p.size

         p_experimental = envelope.p[np.arange(0, n, 7)]  * 1.1
         t_experimental = envelope.t[np.arange(0, n, 7)]
         exp_envelope = mixture.experimental_envelope(t_experimental, p_experimental)
```

In this case, the `exp_envelope` object is an instance of `ExperimentalEnvelope`. In many way it works identically than an EosEnvelope (strictly, both are subclasses of the same base class), and they share attribute and methods. Of course, the key difference for an experimental envelope is that we don't need to calculate, we just set the arrays manually.

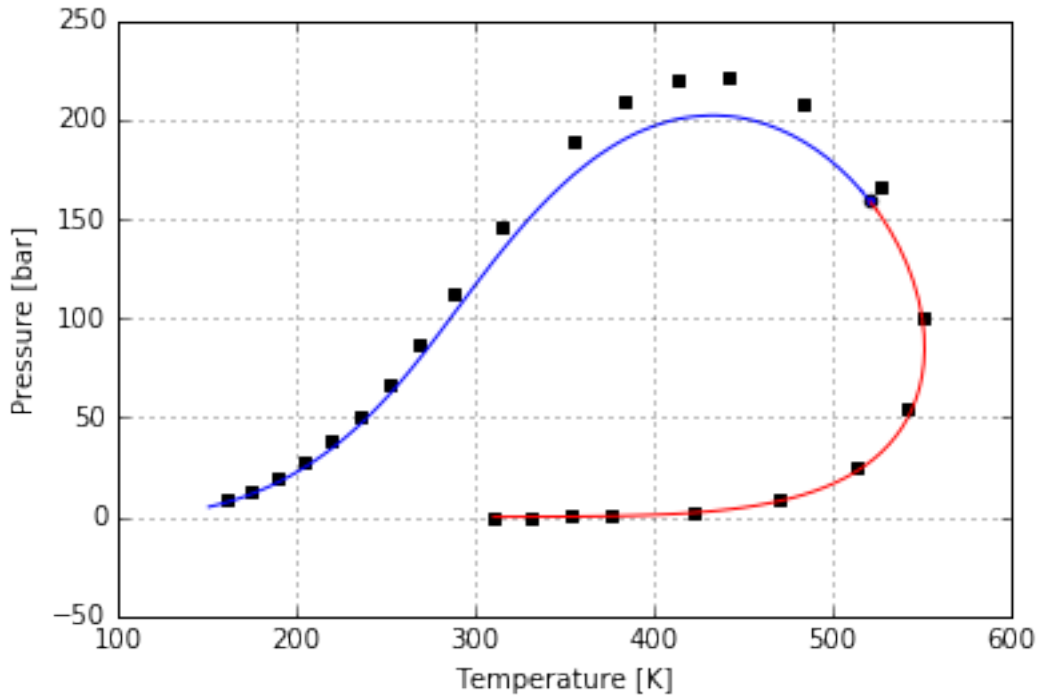
Another difference is that the default plot style is slightly different.

```
In [20]: exp_fig = exp_envelope.plot()
```



By the way, many envelope's figures can be chained, passing a base one to the method `plot()` as the first parameter. For example, here we plot both the simulated and the experimental envelopes in the same figure

```
In [21]: envelope.plot(exp_fig)
```



1.2.7 Calculate a flash

A flash represents a separation of the mixture in two with the same compounds but different fractions. In Sur, we can get a flash instance through the method `get_flash()`

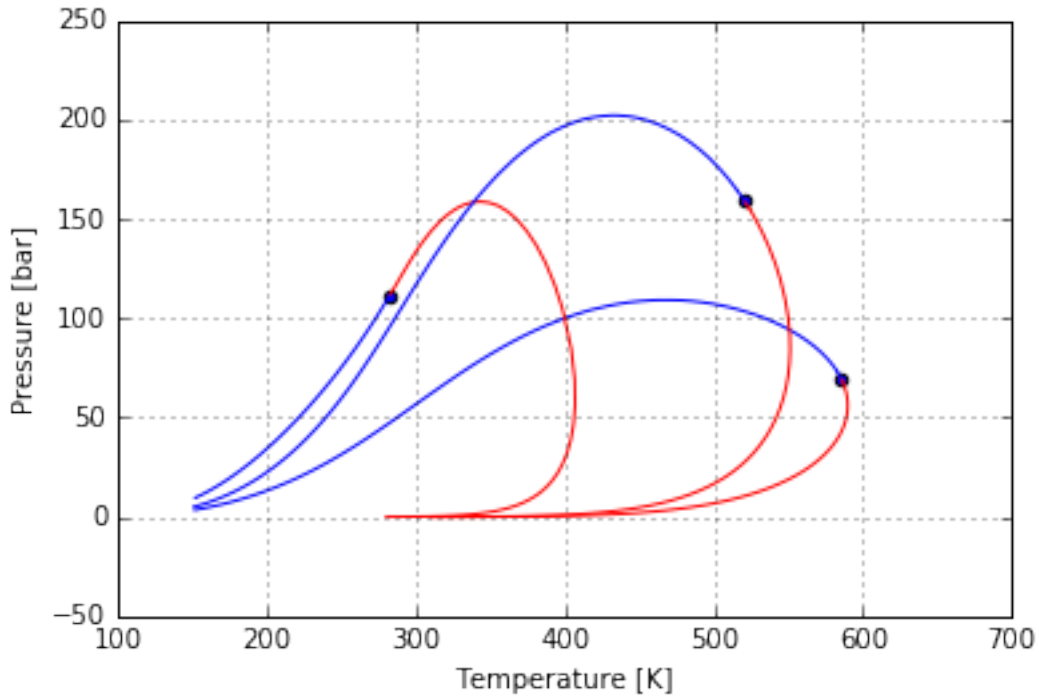
```
In [22]: flash = mixture.get_flash(setup, t=400, p=100)

In [23]: flash.vapour_mixture

Out[23]: [(<Compound: METHANE>, Decimal('0.360497')), (<Compound: CARBON DIOXIDE>, Decimal

In [24]: flash.liquid_mixture.z

Out[24]: array([ 0.116292,  0.348308,  0.5354  ])
```

Which is the same than

```
In [25]: flash.x

Out[25]: array([ 0.116292,  0.348308,  0.5354  ])
```

The `liquid_mixture` and `vapour_mixture` attributes are `Mixture` objects, meaning we can get its own evelopes and the plot all together

```
In [26]: le = flash.liquid_mixture.get_envelope(setup)
         ve = flash.vapour_mixture.get_envelope(setup)

         final_fig = envelope.plot(ve.plot(le.plot()))
```



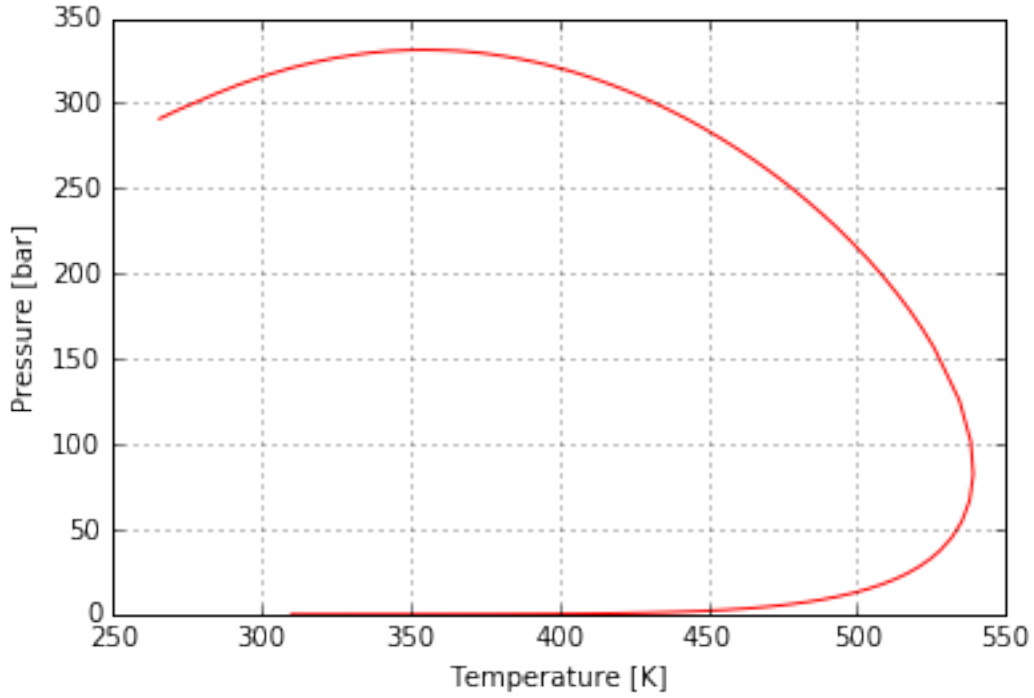
1.3 Advanced examples

Attention: These notebook examples are very raw, created as an internal practice, with no deep explanation of each step done. However, they may be useful to discover further possibilities of Sur.

1.3.1 Modeling Golzapour's synthetic oil

```
In [1]: c = """METHANE
        PROPANE
        n-PENTANE
        n-DECANE
        n-HEXADECANE"""

In [2]: f = """0.8232
        0.0871
        0.0505
        0.0198
        0.0194
        """

In [3]: from sur import Mixture, EosSetup, setup_database
        setup_database()

In [4]: m = Mixture()

In [5]: m.add_many(c, f)

In [6]: m.sort()
        m
```

```
Out[6]: [(<Compound: METHANE>, Decimal('0.8232')), (<Compound: PROPANE>, Decimal('0.0871'))

In [7]: from itertools import combinations

In [8]: setup = EosSetup.objects.create(eos='RKPR', kij_mode=EosSetup.T_DEP, lij_mode=EosSe

In [9]: for c1, c2 in combinations(m.compounds, 2):
            t = c1.tc if c1.weight < c2.weight else c2.tc
            setup.set_interaction('tstar', c1, c2, t)

In [10]: setup.tstar(m)

Out[10]: array([[   0.   ,  190.564,  190.564,  190.564,  190.564],
                [ 190.564,    0.   ,  369.83 ,  369.83 ,  369.83 ],
                [ 190.564,  369.83 ,    0.   ,  469.7  ,  469.7  ],
                [ 190.564,  369.83 ,  469.7  ,    0.   ,  617.7  ],
                [ 190.564,  369.83 ,  469.7  ,  617.7  ,    0.   ]])

In [11]: setup.set_interaction('k0', 'methane', 'propane', 0.0572)
         setup.set_interaction('k0', 'methane', 'n-pentane', 0.05616)
         setup.set_interaction('k0', 'methane', 'n-decane', 0.06891)
         setup.set_interaction('k0', 'methane', 'n-hexadecane', 0.14031);

In [12]: setup.k0(m)

Out[12]: array([[ 0.     ,  0.0572 ,  0.05616,  0.06891,  0.14031],
                [ 0.0572 ,  0.     ,  0.     ,  0.     ,  0.     ],
                [ 0.05616,  0.     ,  0.     ,  0.     ,  0.     ],
                [ 0.06891,  0.     ,  0.     ,  0.     ,  0.     ],
                [ 0.14031,  0.     ,  0.     ,  0.     ,  0.     ]])

In [13]: setup.set_interaction('lij', 'methane', 'propane', -0.00272)
         setup.set_interaction('lij', 'methane', 'n-pentane', -0.06603)
         setup.set_interaction('lij', 'methane', 'n-decane', -0.09227)
         setup.set_interaction('lij', 'methane', 'n-hexadecane', -0.12441);

In [14]: setup.lij(m)

Out[14]: array([[ 0.      , -0.00272 , -0.06603 , -0.09227 , -0.12441 ],
                [-0.00272 ,  0.      , -0.010835, -0.032481, -0.023269],
                [-0.06603 , -0.010835,  0.      , -0.02353 , -0.012501],
                [-0.09227 , -0.032481, -0.02353 ,  0.      ,  0.049878],
                [-0.12441 , -0.023269, -0.012501,  0.049878,  0.      ]])

In [15]: envelope = m.get_envelope(setup)

In [17]: %matplotlib inline

In [19]: envelope.plot();
```



Debug attributes

There is possible to see the raw input and output data that sur interchange with the `EnvelopeSur.exe` fortran program.

```
In [21]: print(envelope.input_txt)
```

```
5               NC
0.8232  0.0871  0.0505  0.0198  0.0194                  z1,z2...zNC
3                               NMODEL (1:SRK / 2:PR / 3:RKPR)
0  1                    ncomb, nTdep
METHANE(1)
190.564  45.99 0.0115478  0.116530154855 1.16          tc, pc, ohm, vc, zrat
2.30376807604   0.0304337956072   0.5   1.54083758839           ac, b, delta1, k
PROPANE(2)
369.83  42.48 0.152291  0.233012170918 1.16          tc, pc, ohm, vc, zrat
9.80216972295   0.0598748909487   1.663687   1.9574557212              ac, b, delta1, k
0.0572          k0
190.564       tstar
-0.00272               lij
n-PENTANE(3)
469.7  33.7 0.251506  0.366506659349 1.16          tc, pc, ohm, vc, zrat
20.2236971345   0.0936309815449   1.957315   2.28798764446             ac, b, delta1, k
0.05616   0.0          k0
190.564   369.83       tstar
-0.06603   -0.010835           lij
n-DECANE(4)
617.7  21.1 0.492328  0.756468369911 1.16          tc, pc, ohm, vc, zrat
56.6610724692   0.192140038284   2.239538   3.11337933794              ac, b, delta1, k
0.06891   0.0   0.0          k0
```

```
190.564   369.83   469.7       tstar
-0.09227   -0.032481   -0.02353                lij
n-HEXADECANE(5)
723.0  14.0 0.717404  1.34252502769 1.16          tc, pc, ohm, vc, zrat
116.426356444   0.341676477137   2.14291   3.90352446586                ac, b, delta1, k
0.14031   0.0   0.0   0.0          k0
190.564   369.83   469.7   617.7       tstar
-0.12441   -0.023269   -0.012501   0.049878             lij


In [22]: print(envelope.output_txt)

METHANE(1)
Tc= 190.5640   Pc =  45.9900   Vc =  0.1005   OM = 0.0115
Zc=   0.2916 Zcrat=   1.1600 Zceos=  0.3382 Vceos= 0.1165
ac=   2.3038    b =   0.0304  del1=  0.5000    k = 1.5408
PROPANE(2)
Tc= 369.8300   Pc =  42.4800   Vc =  0.2009   OM = 0.1523
Zc=   0.2775 Zcrat=   1.1600 Zceos=  0.3219 Vceos= 0.2330
ac=   9.8022    b =   0.0599  del1=  1.6637    k = 1.9575
n-PENTANE(3)
Tc= 469.7000   Pc =  33.7000   Vc =  0.3160   OM = 0.2515
Zc=   0.2726 Zcrat=   1.1600 Zceos=  0.3163 Vceos= 0.3665
ac=  20.2237    b =   0.0936  del1=  1.9573    k = 2.2880
n-DECANE(4)
Tc= 617.7000   Pc =  21.1000   Vc =  0.6521   OM = 0.4923
Zc=   0.2679 Zcrat=   1.1600 Zceos=  0.3108 Vceos= 0.7565
ac=  56.6611    b =   0.1921  del1=  2.2395    k = 3.1134
n-HEXADECANE(5)
Tc= 723.0000   Pc =  14.0000   Vc =  1.1573   OM = 0.7174
Zc=   0.2695 Zcrat=   1.1600 Zceos=  0.3127 Vceos= 1.3425
ac= 116.4264    b =   0.3417  del1=  2.1429    k = 3.9035

 Tc, Pc and Vc are given in K, bar and L/mol respectively

   K0ij MATRIX
METHANE(1)
PROPANE(2)           0.05720
n-PENTANE(3)         0.05616   0.00000
n-DECANE(4)          0.06891   0.00000   0.00000
n-HEXADECANE(5)      0.14031   0.00000   0.00000   0.00000

   T* MATRIX
METHANE(1)
PROPANE(2)         190.56400
n-PENTANE(3)       190.56400 369.83000
n-DECANE(4)        190.56400 369.83000 469.70000
n-HEXADECANE(5)    190.56400 369.83000 469.70000 617.70000

   LIJ MATRIX
METHANE(1)
PROPANE(2)          -0.00272
n-PENTANE(3)        -0.06603  -0.01083
n-DECANE(4)         -0.09227  -0.03248  -0.02353
```

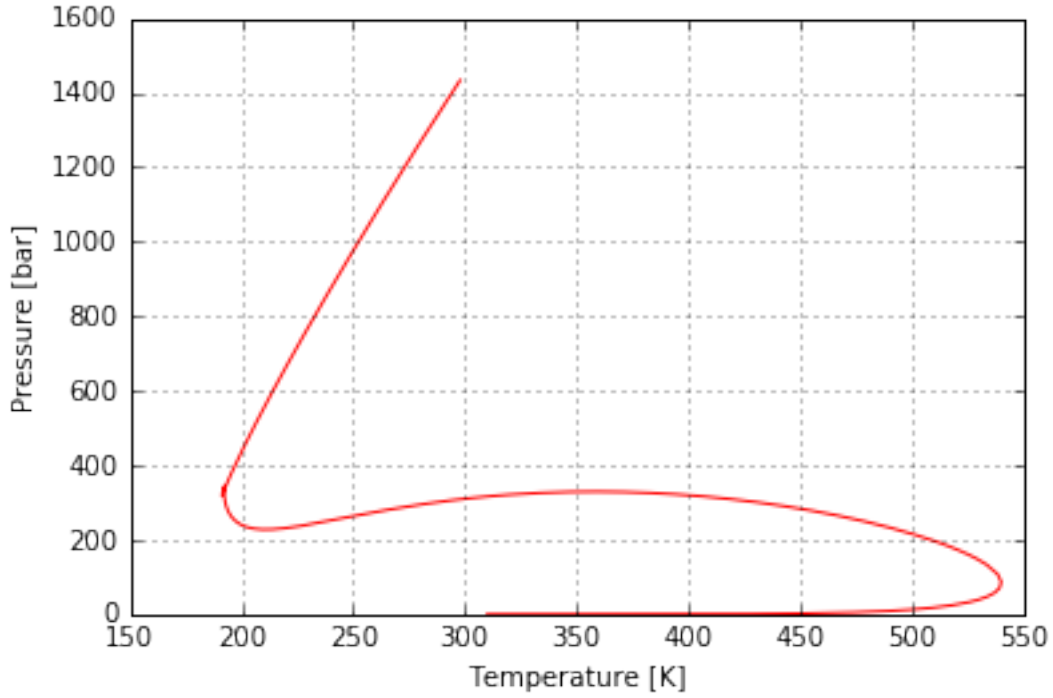
```
n-HEXADECANE(5)     -0.12441  -0.02327  -0.01250   0.04988

  Combining rules:
  0: Classical or van der Waals

Molar fractions:  0.823 0.087 0.051 0.020 0.019
  x    0.8232E+00  0.8710E-01  0.5050E-01  0.1980E-01  0.1940E-01
  y    0.8083E+00  0.9037E-01  0.5408E-01  0.2300E-01  0.2427E-01
  x    0.8232E+00  0.8710E-01  0.5050E-01  0.1980E-01  0.1940E-01
  y    0.8106E+00  0.8992E-01  0.5354E-01  0.2249E-01  0.2347E-01
  x    0.8232E+00  0.8710E-01  0.5050E-01  0.1980E-01  0.1940E-01
  y    0.8129E+00  0.8946E-01  0.5300E-01  0.2198E-01  0.2270E-01
  x    0.8232E+00  0.8710E-01  0.5050E-01  0.1980E-01  0.1940E-01
  y    0.8132E+00  0.8939E-01  0.5290E-01  0.2190E-01  0.2258E-01
  x    0.8232E+00  0.8710E-01  0.5050E-01  0.1980E-01  0.1940E-01
  y    0.8132E+00  0.8939E-01  0.5290E-01  0.2189E-01  0.2257E-01

    T(K)        P(bar)        D(mol/L)
    310.0000    0.2472E-03    0.9591E-05
    312.0835    0.3052E-03    0.1176E-04
    314.9521    0.4056E-03    0.1549E-04
    317.8332    0.5364E-03    0.2030E-04
    320.7258    0.7057E-03    0.2647E-04
    323.6288    0.9239E-03    0.3434E-04
    326.5410    0.1203E-02    0.4433E-04
    329.4613    0.1560E-02    0.5694E-04
    332.3882    0.2011E-02    0.7278E-04
    335.3203    0.2580E-02    0.9255E-04
    338.2562    0.3294E-02    0.1171E-03
    341.1944    0.4183E-02    0.1475E-03
    344.1330    0.5286E-02    0.1848E-03
    347.0704    0.6646E-02    0.2303E-03
    350.0046    0.8314E-02    0.2857E-03
    352.9337    0.1035E-01    0.3527E-03
    355.8556    0.1282E-01    0.4332E-03
    358.7683    0.1579E-01    0.5294E-03
    361.6694    0.1936E-01    0.6438E-03
    364.5565    0.2361E-01    0.7790E-03
    367.4272    0.2865E-01    0.9380E-03
    370.2788    0.3459E-01    0.1124E-02
    373.1088    0.4156E-01    0.1340E-02
    375.9142    0.4967E-01    0.1590E-02
    378.6923    0.5907E-01    0.1876E-02
    381.4399    0.6989E-01    0.2204E-02
    384.1540    0.8227E-01    0.2576E-02
    386.8314    0.9636E-01    0.2997E-02
    389.4689    0.1123E+00    0.3469E-02
    392.0630    0.1302E+00    0.3995E-02
    394.6103    0.1502E+00    0.4578E-02
    397.1074    0.1723E+00    0.5221E-02
    399.5507    0.1968E+00    0.5926E-02
    401.9367    0.2235E+00    0.6692E-02
    404.2616    0.2526E+00    0.7520E-02
    406.5220    0.2841E+00    0.8409E-02
```

```
408.7141    0.3178E+00    0.9357E-02
410.8344    0.3537E+00    0.1036E-01
412.8793    0.3917E+00    0.1142E-01
414.9101    0.4329E+00    0.1256E-01
416.9662    0.4784E+00    0.1381E-01
419.0479    0.5287E+00    0.1519E-01
421.1556    0.5843E+00    0.1670E-01
423.2897    0.6457E+00    0.1837E-01
425.4506    0.7137E+00    0.2020E-01
427.6385    0.7887E+00    0.2221E-01
429.8540    0.8717E+00    0.2442E-01
435.2429    0.1106E+01    0.3061E-01
440.6600    0.1396E+01    0.3816E-01
446.0967    0.1751E+01    0.4732E-01
451.5439    0.2185E+01    0.5835E-01
456.9909    0.2711E+01    0.7157E-01
462.4262    0.3345E+01    0.8732E-01
467.8367    0.4105E+01    0.1060E+00
473.2085    0.5011E+01    0.1280E+00
478.5263    0.6083E+01    0.1537E+00
483.7736    0.7347E+01    0.1838E+00
488.9328    0.8827E+01    0.2186E+00
493.9850    0.1055E+02    0.2589E+00
498.9102    0.1255E+02    0.3052E+00
503.6871    0.1486E+02    0.3581E+00
508.2933    0.1750E+02    0.4185E+00
512.7048    0.2053E+02    0.4871E+00
516.8961    0.2397E+02    0.5647E+00
520.8398    0.2787E+02    0.6524E+00
524.5063    0.3228E+02    0.7510E+00
527.8631    0.3724E+02    0.8617E+00
531.8966    0.4504E+02    0.1036E+01
535.3488    0.5475E+02    0.1252E+01
537.9048    0.6687E+02    0.1523E+01
539.1107    0.8209E+02    0.1866E+01
538.2987    0.1013E+03    0.2305E+01
534.4547    0.1255E+03    0.2874E+01
525.9621    0.1564E+03    0.3625E+01
522.0429    0.1675E+03    0.3905E+01
517.8064    0.1783E+03    0.4182E+01
513.2952    0.1887E+03    0.4458E+01
508.5464    0.1989E+03    0.4732E+01
503.5934    0.2087E+03    0.5004E+01
498.4657    0.2181E+03    0.5272E+01
493.1902    0.2272E+03    0.5538E+01
487.7914    0.2358E+03    0.5801E+01
482.2913    0.2441E+03    0.6061E+01
476.7103    0.2521E+03    0.6317E+01
471.0667    0.2596E+03    0.6569E+01
465.3775    0.2667E+03    0.6818E+01
459.6579    0.2734E+03    0.7063E+01
453.9218    0.2797E+03    0.7303E+01
448.1818    0.2856E+03    0.7540E+01
442.4494    0.2911E+03    0.7772E+01
```

```
    436.7347    0.2962E+03    0.7999E+01
    431.0469    0.3010E+03    0.8223E+01
    425.3940    0.3053E+03    0.8442E+01
    419.7833    0.3093E+03    0.8656E+01
    414.2209    0.3129E+03    0.8866E+01
    408.7122    0.3161E+03    0.9072E+01
    403.2617    0.3191E+03    0.9273E+01
    397.8733    0.3216E+03    0.9470E+01
    392.5500    0.3239E+03    0.9663E+01
    387.2942    0.3258E+03    0.9852E+01
    382.0379    0.3274E+03    0.1004E+02
    376.7143    0.3288E+03    0.1023E+02
    371.3183    0.3299E+03    0.1042E+02
    365.8427    0.3307E+03    0.1061E+02
    360.2784    0.3311E+03    0.1080E+02
    354.6136    0.3313E+03    0.1099E+02
    348.8328    0.3311E+03    0.1119E+02
    342.9153    0.3306E+03    0.1139E+02
    336.8332    0.3296E+03    0.1159E+02
    330.5478    0.3282E+03    0.1180E+02
    324.0036    0.3263E+03    0.1202E+02
    317.1145    0.3238E+03    0.1224E+02
    309.7443    0.3206E+03    0.1249E+02
    301.6363    0.3164E+03    0.1275E+02
    292.2243    0.3108E+03    0.1306E+02
    284.4179    0.3054E+03    0.1331E+02
    272.0585    0.2961E+03    0.1371E+02
    265.9509    0.2911E+03    0.1391E+02
    265.6858    0.2909E+03    0.1392E+02

  Number of critical points found:            0
    T(K)         P(bar)         D(mol/L)
```

1.3.2 Isochores

```
In [1]: %config InlineBackend.close_figures=False
        %matplotlib inline
        from matplotlib import interactive
        interactive(False)

In [2]: from sur import *
        setup_database()

In [3]: m = Mixture()
        m.add_many('methane propane n-pentane n-decane n-hexadecane',
                   '0.822  0.088  0.050  0.020  0.020')

        s = EosSetup.objects.create(eos='RKPR', kij_mode=EosSetup.T_DEP, lij_mode='constant


        env = m.get_envelope(setup=s)

In [4]: fig = env.plot()
        fig
```




```
In [15]: help(m.get_isochore)
```

```
Help on method get_isochore in module sur.models:

get_isochore(self, setup, v, ts, ps, t_sup, t_step=5.0, t_inf=270.0) method of sur.models.M
    Get the isochore (isoV) for this mixture, calculated using
    the setup EOS with its selected interaction parameters
    mode.
```

```
In [5]: isochore = m.get_isochore(setup=s, v=10., ts=467.01, ps=3.86, t_sup=465.0, t_step=5
```

You can get the raw input and output

```
In [6]: print(isochore.input_txt)
```

```
5               NC
5       nplus
isoV            Spec    v(k), ts and ps
0.822  0.088  0.05  0.02  0.02                  z1,z2...zNC
10.0   467.01    3.86         v(L/mol)+Ts+Ps
465.0  5.0  270.0         T(K)+dT+Tinf
3                             NMODEL (1:SRK / 2:PR / 3:RKPR)
0  1                    ncomb, nTdep
METHANE(1)
190.564  45.99 0.0115478  0.116530154855 1.16          tc, pc, ohm, vc, zrat
2.30376807604   0.0304337956072   0.5   1.54083758839           ac, b, delta1, k
PROPANE(2)
369.83  42.48 0.152291  0.233012170918 1.16          tc, pc, ohm, vc, zrat
9.80216972295   0.0598748909487   1.663687   1.9574557212               ac, b, delta1, k
0.013831          k0
190.564       tstar
-0.047134             lij
```

```
n-PENTANE(3)
469.7  33.7 0.251506  0.366506659349 1.16          tc, pc, ohm, vc, zrat
20.2236971345   0.0936309815449   1.957315   2.28798764446              ac, b, delta1, k
0.030221   0.0          k0
190.564   369.83       tstar
-0.084739   -0.010835          lij
n-DECANE(4)
617.7  21.1 0.492328  0.756468369911 1.16          tc, pc, ohm, vc, zrat
56.6610724692   0.192140038284   2.239538   3.11337933794               ac, b, delta1, k
0.068534   0.0   0.0          k0
190.564   369.83   469.7       tstar
-0.145519   -0.032481   -0.02353               lij
n-HEXADECANE(5)
723.0  14.0 0.717404  1.34252502769 1.16          tc, pc, ohm, vc, zrat
116.426356444   0.341676477137   2.14291   3.90352446586               ac, b, delta1, k
0.095928   0.0   0.0   0.0          k0
190.564   369.83   469.7   617.7       tstar
-0.121142   -0.023269   -0.012501   0.049878            lij


In [7]: print(isochore.output_txt)

  The plus fraction from component            5  is   2.000000000000000E-002
    T(K)   rho(mol/L)   P(bar)    v(L/mol)     beta       betav    xplus   yplus
    iter
  467.01  0.1000E+00  0.3860E+01  0.1000E+02  0.1000E+01  0.1000E+01
  465.00  0.1000E+00  0.3845E+01  0.1000E+02  0.9984E+00  0.9999E+00  0.898332  0.018611
  460.00  0.1000E+00  0.3792E+01  0.1000E+02  0.9950E+00  0.9998E+00  0.890882  0.015582
  455.00  0.1000E+00  0.3740E+01  0.1000E+02  0.9919E+00  0.9997E+00  0.882580  0.012958
  450.00  0.1000E+00  0.3689E+01  0.1000E+02  0.9892E+00  0.9996E+00  0.873347  0.010699
  445.00  0.1000E+00  0.3640E+01  0.1000E+02  0.9869E+00  0.9995E+00  0.863099  0.008769
  440.00  0.1000E+00  0.3591E+01  0.1000E+02  0.9848E+00  0.9994E+00  0.851757  0.007130
  435.00  0.1000E+00  0.3544E+01  0.1000E+02  0.9829E+00  0.9994E+00  0.839247  0.005750
  430.00  0.1000E+00  0.3498E+01  0.1000E+02  0.9812E+00  0.9993E+00  0.825509  0.004596
  425.00  0.1000E+00  0.3451E+01  0.1000E+02  0.9797E+00  0.9993E+00  0.810500  0.003639
  420.00  0.1000E+00  0.3406E+01  0.1000E+02  0.9783E+00  0.9992E+00  0.794204  0.002854
  415.00  0.1000E+00  0.3361E+01  0.1000E+02  0.9770E+00  0.9992E+00  0.776641  0.002215
  410.00  0.1000E+00  0.3316E+01  0.1000E+02  0.9758E+00  0.9992E+00  0.757877  0.001701
  405.00  0.1000E+00  0.3271E+01  0.1000E+02  0.9746E+00  0.9991E+00  0.738030  0.001291
  400.00  0.1000E+00  0.3227E+01  0.1000E+02  0.9734E+00  0.9991E+00  0.717280  0.000969
  395.00  0.1000E+00  0.3182E+01  0.1000E+02  0.9723E+00  0.9991E+00  0.695871  0.000719
  390.00  0.1000E+00  0.3138E+01  0.1000E+02  0.9711E+00  0.9991E+00  0.674107  0.000527
  385.00  0.1000E+00  0.3094E+01  0.1000E+02  0.9699E+00  0.9990E+00  0.652340  0.000382
  380.00  0.1000E+00  0.3050E+01  0.1000E+02  0.9687E+00  0.9990E+00  0.630950  0.000273
  375.00  0.1000E+00  0.3006E+01  0.1000E+02  0.9675E+00  0.9990E+00  0.610313  0.000193
  370.00  0.1000E+00  0.2962E+01  0.1000E+02  0.9664E+00  0.9990E+00  0.590768  0.000136
  365.00  0.1000E+00  0.2918E+01  0.1000E+02  0.9652E+00  0.9990E+00  0.572588  0.000094
  360.00  0.1000E+00  0.2874E+01  0.1000E+02  0.9641E+00  0.9989E+00  0.555955  0.000065
  355.00  0.1000E+00  0.2831E+01  0.1000E+02  0.9631E+00  0.9989E+00  0.540952  0.000044
  350.00  0.1000E+00  0.2788E+01  0.1000E+02  0.9621E+00  0.9989E+00  0.527568  0.000030
  345.00  0.1000E+00  0.2745E+01  0.1000E+02  0.9613E+00  0.9989E+00  0.515708  0.000020
  340.00  0.1000E+00  0.2703E+01  0.1000E+02  0.9604E+00  0.9989E+00  0.505214  0.000013
  335.00  0.1000E+00  0.2660E+01  0.1000E+02  0.9597E+00  0.9989E+00  0.495883  0.000008
  330.00  0.1000E+00  0.2618E+01  0.1000E+02  0.9590E+00  0.9989E+00  0.487479  0.000005
```

```
325.00  0.1000E+00  0.2576E+01  0.1000E+02  0.9583E+00  0.9988E+00  0.479748  0.000003
320.00  0.1000E+00  0.2534E+01  0.1000E+02  0.9577E+00  0.9988E+00  0.472422  0.000002
315.00  0.1000E+00  0.2493E+01  0.1000E+02  0.9570E+00  0.9988E+00  0.465218  0.000001
310.00  0.1000E+00  0.2451E+01  0.1000E+02  0.9563E+00  0.9988E+00  0.457840  0.000001
305.00  0.1000E+00  0.2409E+01  0.1000E+02  0.9556E+00  0.9988E+00  0.449965  0.000000
300.00  0.1000E+00  0.2366E+01  0.1000E+02  0.9547E+00  0.9988E+00  0.441244  0.000000
295.00  0.1000E+00  0.2324E+01  0.1000E+02  0.9536E+00  0.9988E+00  0.431283  0.000000
290.00  0.1000E+00  0.2281E+01  0.1000E+02  0.9523E+00  0.9988E+00  0.419647  0.000000
285.00  0.1000E+00  0.2237E+01  0.1000E+02  0.9507E+00  0.9988E+00  0.405877  0.000000
280.00  0.1000E+00  0.2193E+01  0.1000E+02  0.9487E+00  0.9987E+00  0.389552  0.000000
275.00  0.1000E+00  0.2147E+01  0.1000E+02  0.9460E+00  0.9987E+00  0.370439  0.000000
270.00  0.1000E+00  0.2100E+01  0.1000E+02  0.9427E+00  0.9987E+00  0.348747  0.000000
Monophasic Region:
  T(K)    rho(mol/L)   P(bar)     v(L/mol)
467.01  0.1000E+00  0.3860E+01  0.1000E+02
472.01  0.1000E+00  0.3910E+01  0.1000E+02
477.01  0.1000E+00  0.3952E+01  0.1000E+02
482.01  0.1000E+00  0.3994E+01  0.1000E+02
487.01  0.1000E+00  0.4036E+01  0.1000E+02
492.01  0.1000E+00  0.4078E+01  0.1000E+02
497.01  0.1000E+00  0.4120E+01  0.1000E+02
502.01  0.1000E+00  0.4162E+01  0.1000E+02
507.01  0.1000E+00  0.4204E+01  0.1000E+02
512.01  0.1000E+00  0.4246E+01  0.1000E+02
517.01  0.1000E+00  0.4288E+01  0.1000E+02
522.01  0.1000E+00  0.4330E+01  0.1000E+02
527.01  0.1000E+00  0.4372E+01  0.1000E+02
532.01  0.1000E+00  0.4414E+01  0.1000E+02
537.01  0.1000E+00  0.4456E+01  0.1000E+02
542.01  0.1000E+00  0.4498E+01  0.1000E+02
547.01  0.1000E+00  0.4540E+01  0.1000E+02
552.01  0.1000E+00  0.4582E+01  0.1000E+02
557.01  0.1000E+00  0.4624E+01  0.1000E+02
562.01  0.1000E+00  0.4666E+01  0.1000E+02
567.01  0.1000E+00  0.4708E+01  0.1000E+02
572.01  0.1000E+00  0.4750E+01  0.1000E+02
577.01  0.1000E+00  0.4792E+01  0.1000E+02
582.01  0.1000E+00  0.4834E+01  0.1000E+02
587.01  0.1000E+00  0.4876E+01  0.1000E+02
592.01  0.1000E+00  0.4918E+01  0.1000E+02
597.01  0.1000E+00  0.4960E+01  0.1000E+02
602.01  0.1000E+00  0.5002E+01  0.1000E+02
607.01  0.1000E+00  0.5044E+01  0.1000E+02
612.01  0.1000E+00  0.5086E+01  0.1000E+02
617.01  0.1000E+00  0.5128E+01  0.1000E+02
622.01  0.1000E+00  0.5170E+01  0.1000E+02
627.01  0.1000E+00  0.5212E+01  0.1000E+02
632.01  0.1000E+00  0.5254E+01  0.1000E+02
637.01  0.1000E+00  0.5296E+01  0.1000E+02
642.01  0.1000E+00  0.5338E+01  0.1000E+02
647.01  0.1000E+00  0.5380E+01  0.1000E+02
652.01  0.1000E+00  0.5422E+01  0.1000E+02
657.01  0.1000E+00  0.5464E+01  0.1000E+02
662.01  0.1000E+00  0.5506E+01  0.1000E+02
```

```
667.01  0.1000E+00  0.5548E+01  0.1000E+02
672.01  0.1000E+00  0.5590E+01  0.1000E+02
677.01  0.1000E+00  0.5632E+01  0.1000E+02
682.01  0.1000E+00  0.5674E+01  0.1000E+02
687.01  0.1000E+00  0.5716E+01  0.1000E+02
692.01  0.1000E+00  0.5758E+01  0.1000E+02
697.01  0.1000E+00  0.5800E+01  0.1000E+02
702.01  0.1000E+00  0.5842E+01  0.1000E+02
707.01  0.1000E+00  0.5884E+01  0.1000E+02
712.01  0.1000E+00  0.5926E+01  0.1000E+02
717.01  0.1000E+00  0.5967E+01  0.1000E+02
722.01  0.1000E+00  0.6009E+01  0.1000E+02
727.01  0.1000E+00  0.6051E+01  0.1000E+02
732.01  0.1000E+00  0.6093E+01  0.1000E+02
737.01  0.1000E+00  0.6135E+01  0.1000E+02
742.01  0.1000E+00  0.6177E+01  0.1000E+02
747.01  0.1000E+00  0.6219E+01  0.1000E+02
752.01  0.1000E+00  0.6261E+01  0.1000E+02
757.01  0.1000E+00  0.6303E+01  0.1000E+02
762.01  0.1000E+00  0.6345E+01  0.1000E+02
767.01  0.1000E+00  0.6387E+01  0.1000E+02
772.01  0.1000E+00  0.6429E+01  0.1000E+02
777.01  0.1000E+00  0.6471E+01  0.1000E+02
782.01  0.1000E+00  0.6513E+01  0.1000E+02
787.01  0.1000E+00  0.6554E+01  0.1000E+02
792.01  0.1000E+00  0.6596E+01  0.1000E+02
797.01  0.1000E+00  0.6638E+01  0.1000E+02
802.01  0.1000E+00  0.6680E+01  0.1000E+02
807.01  0.1000E+00  0.6722E+01  0.1000E+02
812.01  0.1000E+00  0.6764E+01  0.1000E+02
817.01  0.1000E+00  0.6806E+01  0.1000E+02
822.01  0.1000E+00  0.6848E+01  0.1000E+02
827.01  0.1000E+00  0.6890E+01  0.1000E+02
832.01  0.1000E+00  0.6932E+01  0.1000E+02
837.01  0.1000E+00  0.6974E+01  0.1000E+02
842.01  0.1000E+00  0.7015E+01  0.1000E+02
847.01  0.1000E+00  0.7057E+01  0.1000E+02
852.01  0.1000E+00  0.7099E+01  0.1000E+02
857.01  0.1000E+00  0.7141E+01  0.1000E+02
862.01  0.1000E+00  0.7183E+01  0.1000E+02
867.01  0.1000E+00  0.7225E+01  0.1000E+02
872.01  0.1000E+00  0.7267E+01  0.1000E+02
877.01  0.1000E+00  0.7309E+01  0.1000E+02
882.01  0.1000E+00  0.7351E+01  0.1000E+02
887.01  0.1000E+00  0.7392E+01  0.1000E+02
892.01  0.1000E+00  0.7434E+01  0.1000E+02
897.01  0.1000E+00  0.7476E+01  0.1000E+02
902.01  0.1000E+00  0.7518E+01  0.1000E+02
907.01  0.1000E+00  0.7560E+01  0.1000E+02
912.01  0.1000E+00  0.7602E+01  0.1000E+02
917.01  0.1000E+00  0.7644E+01  0.1000E+02
922.01  0.1000E+00  0.7686E+01  0.1000E+02
927.01  0.1000E+00  0.7728E+01  0.1000E+02
932.01  0.1000E+00  0.7769E+01  0.1000E+02
```

```
  937.01  0.1000E+00  0.7811E+01  0.1000E+02
  942.01  0.1000E+00  0.7853E+01  0.1000E+02
  947.01  0.1000E+00  0.7895E+01  0.1000E+02
  952.01  0.1000E+00  0.7937E+01  0.1000E+02
  957.01  0.1000E+00  0.7979E+01  0.1000E+02
  962.01  0.1000E+00  0.8021E+01  0.1000E+02
  967.01  0.1000E+00  0.8063E+01  0.1000E+02
  972.01  0.1000E+00  0.8104E+01  0.1000E+02
  977.01  0.1000E+00  0.8146E+01  0.1000E+02
  982.01  0.1000E+00  0.8188E+01  0.1000E+02
  987.01  0.1000E+00  0.8230E+01  0.1000E+02
  992.01  0.1000E+00  0.8272E+01  0.1000E+02
  997.01  0.1000E+00  0.8314E+01  0.1000E+02
 1002.01  0.1000E+00  0.8356E+01  0.1000E+02
```

los vectores t y p son recortados en dos partes, como se generan

```
In [8]: isochore.t
```

```
Out[8]: array([ 467.01,  465.  ,  460.  ,  455.  ,  450.  ,  445.  ,  440.  ,
                435.  ,  430.  ,  425.  ,  420.  ,  415.  ,  410.  ,  405.  ,
                400.  ,  395.  ,  390.  ,  385.  ,  380.  ,  375.  ,  370.  ,
                365.  ,  360.  ,  355.  ,  350.  ,  345.  ,  340.  ,  335.  ,
                330.  ,  325.  ,  320.  ,  315.  ,  310.  ,  305.  ,  300.  ,
                295.  ,  290.  ,  285.  ,  280.  ,  275.  ,  270.  ])
```

```
In [9]: isochore.p
```

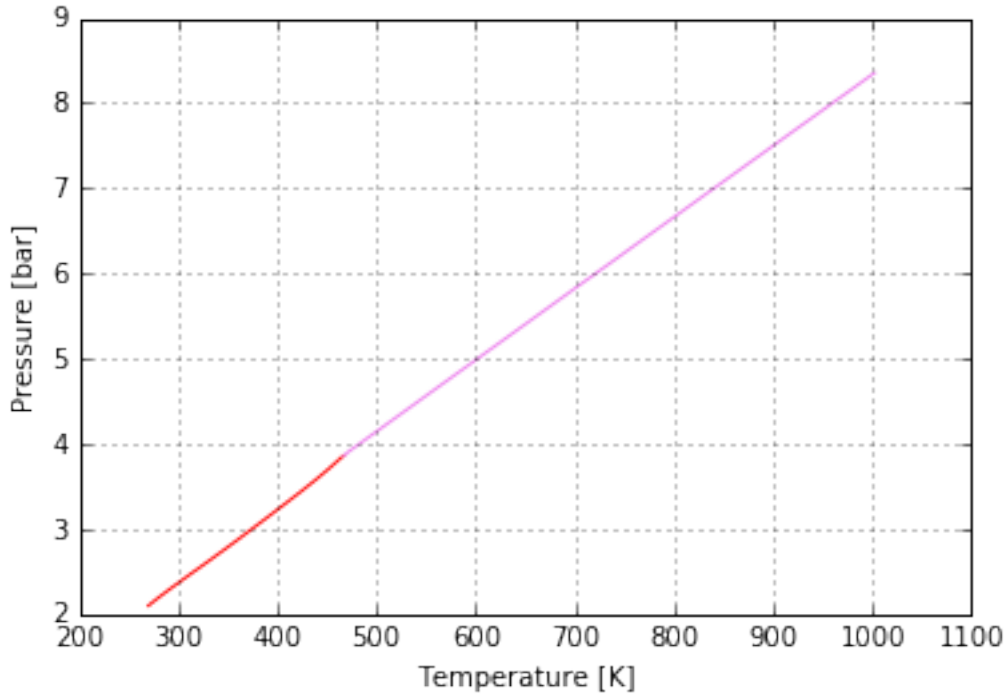
```
Out[9]: array([ 3.86 ,  3.845,  3.792,  3.74 ,  3.689,  3.64 ,  3.591,  3.544,
                3.498,  3.451,  3.406,  3.361,  3.316,  3.271,  3.227,  3.182,
                3.138,  3.094,  3.05 ,  3.006,  2.962,  2.918,  2.874,  2.831,
                2.788,  2.745,  2.703,  2.66 ,  2.618,  2.576,  2.534,  2.493,
                2.451,  2.409,  2.366,  2.324,  2.281,  2.237,  2.193,  2.147,  2.1  ])
```

```
In [10]: isochore.t_monophasic
```

```
Out[10]: array([  467.01,   472.01,   477.01,   482.01,   487.01,   492.01,
                  497.01,   502.01,   507.01,   512.01,   517.01,   522.01,
                  527.01,   532.01,   537.01,   542.01,   547.01,   552.01,
                  557.01,   562.01,   567.01,   572.01,   577.01,   582.01,
                  587.01,   592.01,   597.01,   602.01,   607.01,   612.01,
                  617.01,   622.01,   627.01,   632.01,   637.01,   642.01,
                  647.01,   652.01,   657.01,   662.01,   667.01,   672.01,
                  677.01,   682.01,   687.01,   692.01,   697.01,   702.01,
                  707.01,   712.01,   717.01,   722.01,   727.01,   732.01,
                  737.01,   742.01,   747.01,   752.01,   757.01,   762.01,
                  767.01,   772.01,   777.01,   782.01,   787.01,   792.01,
                  797.01,   802.01,   807.01,   812.01,   817.01,   822.01,
                  827.01,   832.01,   837.01,   842.01,   847.01,   852.01,
                  857.01,   862.01,   867.01,   872.01,   877.01,   882.01,
                  887.01,   892.01,   897.01,   902.01,   907.01,   912.01,
                  917.01,   922.01,   927.01,   932.01,   937.01,   942.01,
                  947.01,   952.01,   957.01,   962.01,   967.01,   972.01,
                  977.01,   982.01,   987.01,   992.01,   997.01,  1002.01])
```

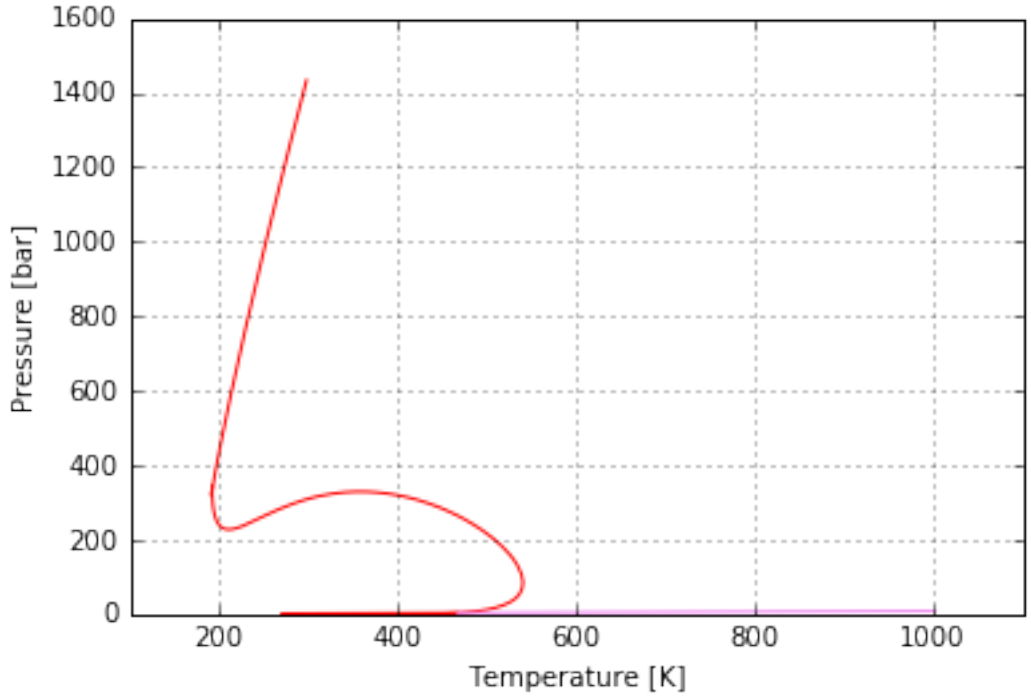
```
In [11]: isochore.p_monophasic
```

```
Out[11]: array([ 3.86 ,  3.91 ,  3.952,  3.994,  4.036,  4.078,  4.12 ,  4.162,
                 4.204,  4.246,  4.288,  4.33 ,  4.372,  4.414,  4.456,  4.498,
                 4.54 ,  4.582,  4.624,  4.666,  4.708,  4.75 ,  4.792,  4.834,
                 4.876,  4.918,  4.96 ,  5.002,  5.044,  5.086,  5.128,  5.17 ,
                 5.212,  5.254,  5.296,  5.338,  5.38 ,  5.422,  5.464,  5.506,
                 5.548,  5.59 ,  5.632,  5.674,  5.716,  5.758,  5.8  ,  5.842,
                 5.884,  5.926,  5.967,  6.009,  6.051,  6.093,  6.135,  6.177,
                 6.219,  6.261,  6.303,  6.345,  6.387,  6.429,  6.471,  6.513,
                 6.554,  6.596,  6.638,  6.68 ,  6.722,  6.764,  6.806,  6.848,
                 6.89 ,  6.932,  6.974,  7.015,  7.057,  7.099,  7.141,  7.183,
                 7.225,  7.267,  7.309,  7.351,  7.392,  7.434,  7.476,  7.518,
                 7.56 ,  7.602,  7.644,  7.686,  7.728,  7.769,  7.811,  7.853,
                 7.895,  7.937,  7.979,  8.021,  8.063,  8.104,  8.146,  8.188,
                 8.23 ,  8.272,  8.314,  8.356])

In [12]: isochore.rho, isochore.rho_monophasic,

Out[12]: (array([ 0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1]),
          array([ 0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,
                  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1,  0.1]))

In [13]: isochore.plot()
```



También se le puede pasar una figura ya generada, para que superponga el nuevo plot

```
In [14]: isochore.plot(fig)
```



1.3.3 Advanced plots

```
In [1]: %config InlineBackend.close_figures=False
        %matplotlib inline
        from matplotlib import interactive
        interactive(False)

        from sur import *
        setup_database()

        m = Mixture()
        m.add_many("methane co2 n-decane", "0.25 0.50 0.25")
```

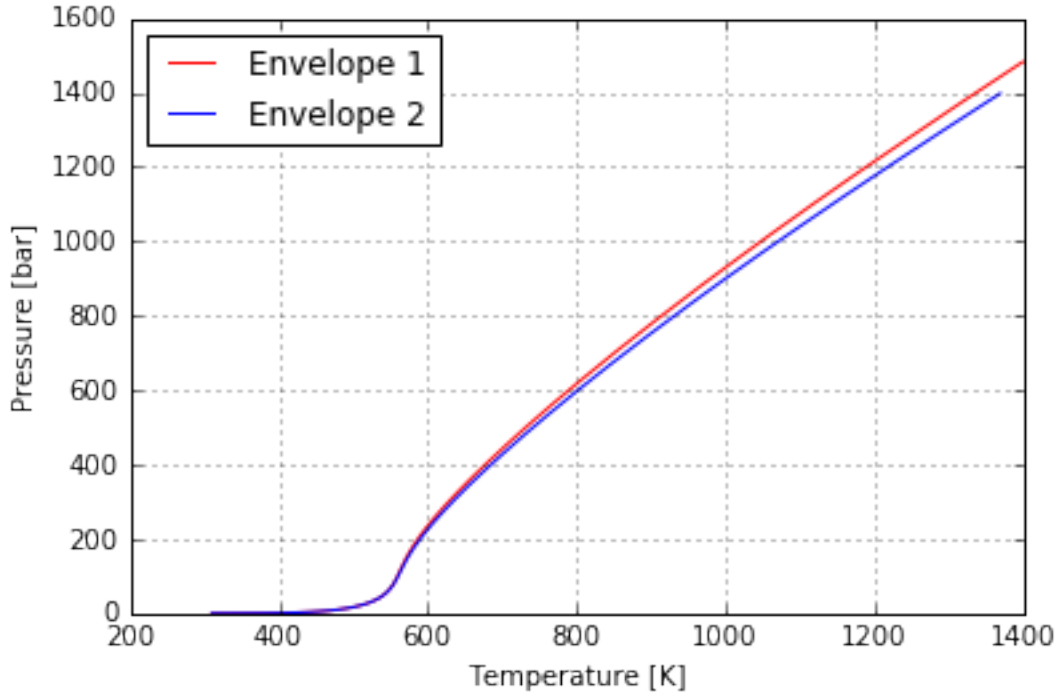
```
In [2]: s1 = EosSetup.objects.create(eos='RKPR', kij_mode=EosSetup.CONSTANTS, lij_mode=EosS
        s1.set_interaction('kij', 'methane', 'co2', .1)
        s1.set_interaction('kij', 'co2', 'n-decane',  0.091)
        s1.set_interaction('lij', 'co2', 'n-decane',  -0.90)
        env1 = m.get_envelope(s1, label="Envelope 1")
```

```
In [3]: s2 = EosSetup.objects.create(eos='RKPR', kij_mode=EosSetup.CONSTANTS, lij_mode=EosS
        s2.set_interaction('kij', 'methane', 'co2', .11)
        s2.set_interaction('kij', 'co2', 'n-decane',  0.081)
        s2.set_interaction('lij', 'co2', 'n-decane',  -0.93)
        env2 = m.get_envelope(s2, label="Envelope 2")
```

```
In [4]: from sur.plots import multiplot
```

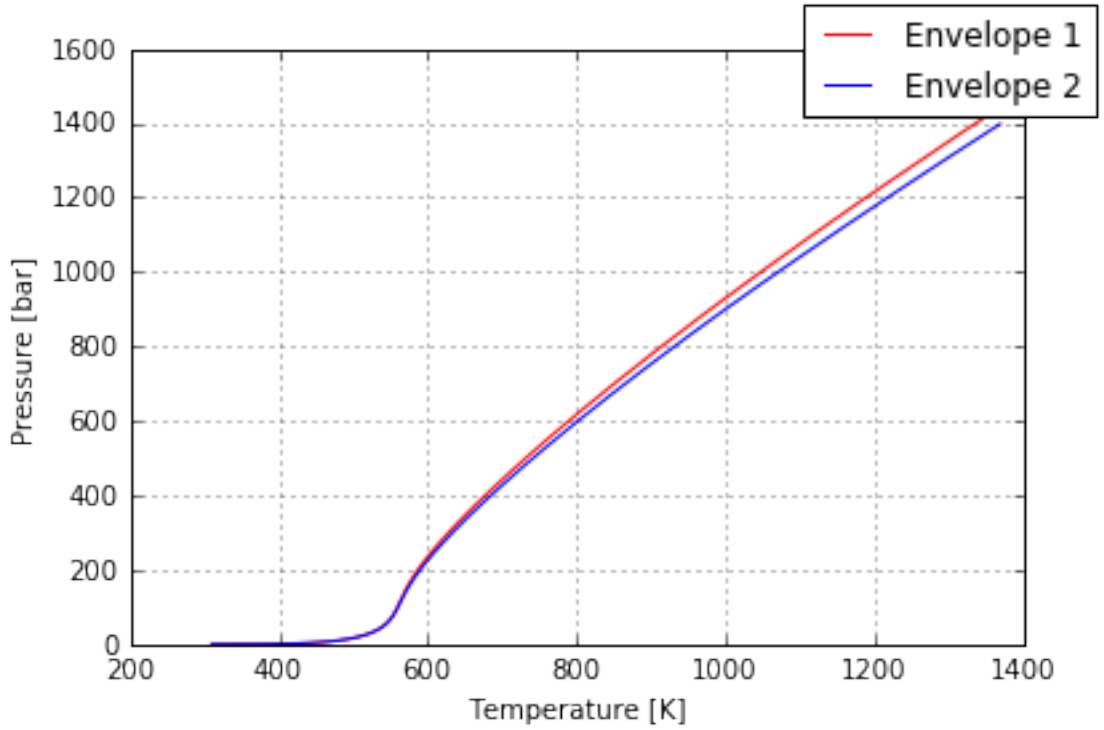
```
In [6]: fig = multiplot([env1, env2], legends='best')
```

```
In [7]: fig
```

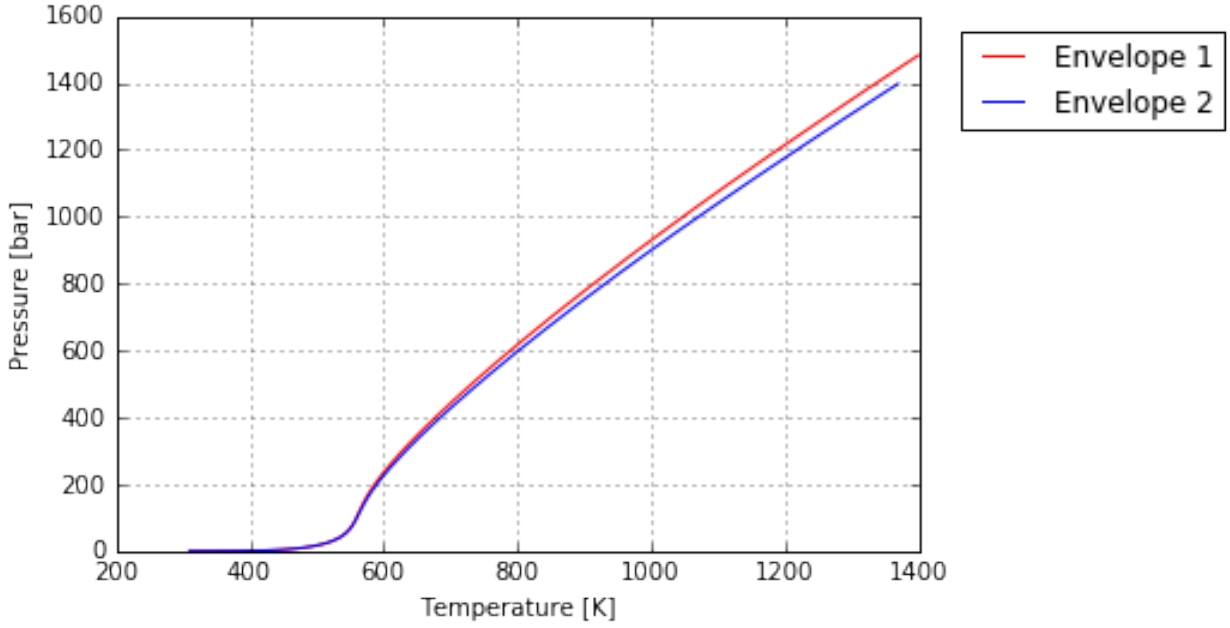


You can use the `bbox_to_anchor` keyword argument to place the legend partially outside the axes and/or decrease the font size.

```
In [8]: ax = fig.get_axes()[0]
        ax.legend(bbox_to_anchor=(1.1, 1.1))
        fig
```

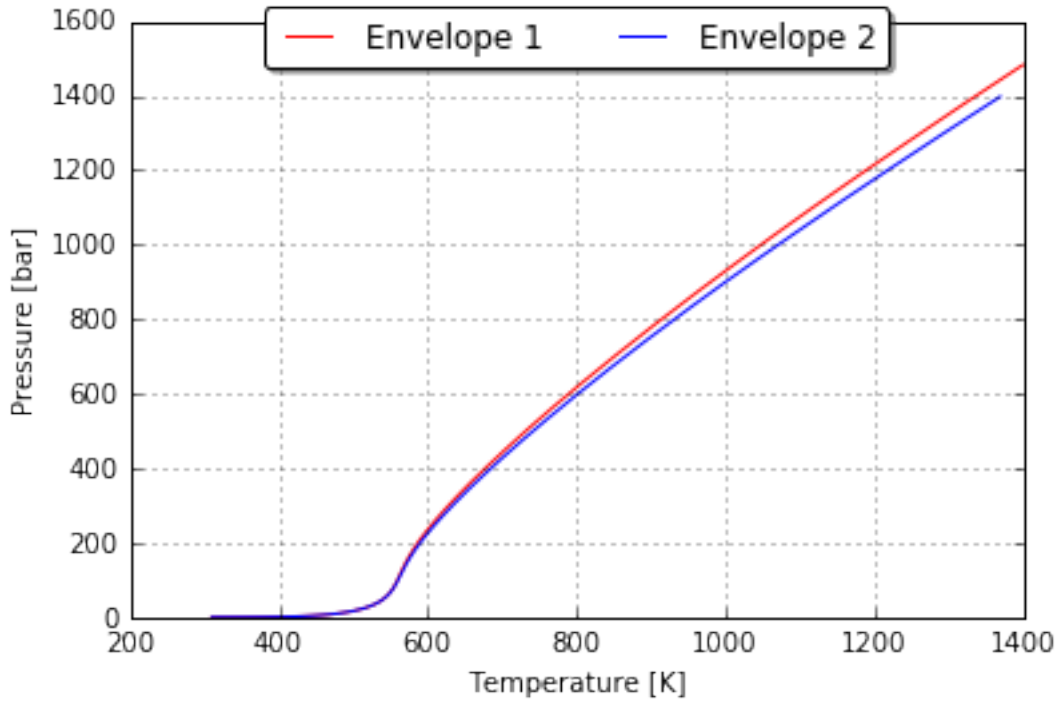



```
In [9]: ax.legend(bbox_to_anchor=(1.4, 1))
        fig
```



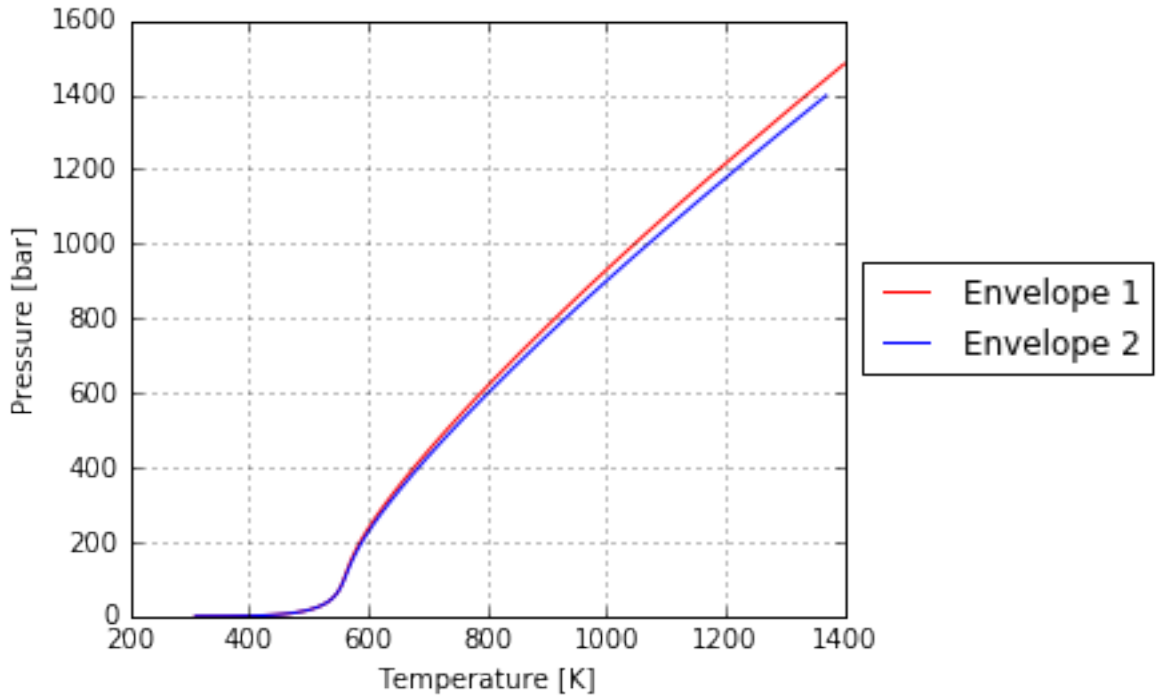
Similarly, you can make the legend more horizontal and/or put it at the top of the figure (I'm also turning on rounded corners and a simple drop shadow):

```
In [10]: ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.05),
                   ncol=3, fancybox=True, shadow=True)

         fig
```



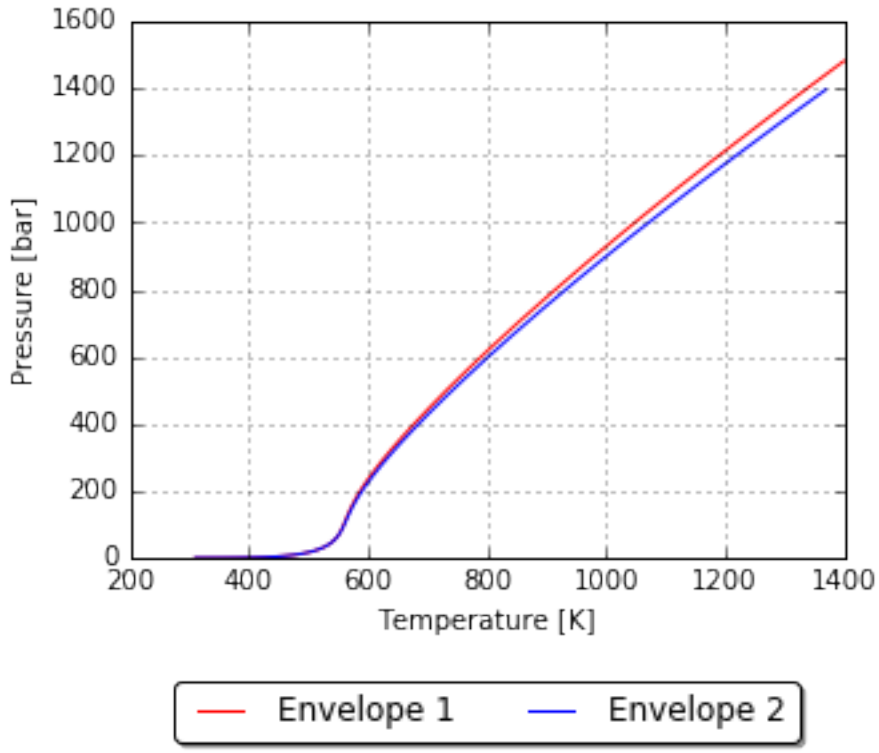
Alternatively, you can shrink the current plot's width, and put the legend entirely outside the axis of the figure

```
In [11]: # Shink current axis by 20%
         box = ax.get_position()
         ax.set_position([box.x0, box.y0, box.width * 0.8, box.height])

         # Put a legend to the right of the current axis
         ax.legend(loc='center left', bbox_to_anchor=(1, 0.5))
         fig
```



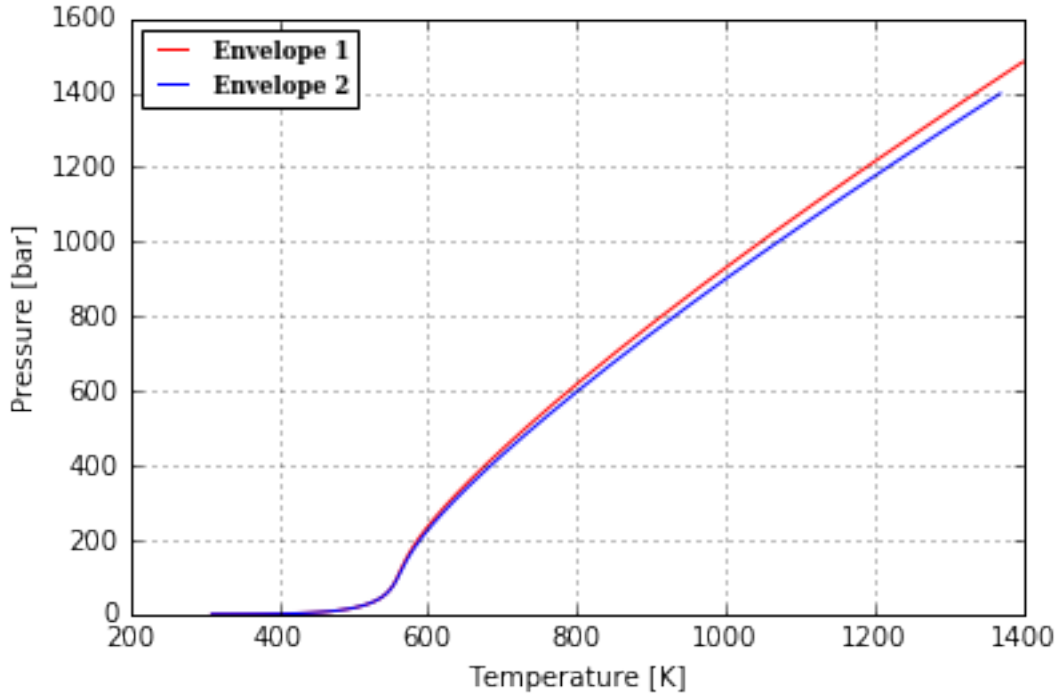
And in a similar manner, you can shrink the plot vertically, and put the a horizontal legend at the bottom

```
In [12]: # Shink current axis's height by 10% on the bottom
         box = ax.get_position()
         ax.set_position([box.x0, box.y0 + box.height * 0.1,
                          box.width, box.height * 0.9])

         # Put a legend below current axis
         ax.legend(loc='upper center', bbox_to_anchor=(0.5, -0.2),
                   fancybox=True, shadow=True, ncol=5)
         fig
```



Configure fonts

```
In [13]: # reset figure
         fig = multiplot([env1, env2], legends='upper right')
         ax = fig.get_axes()[0]

         from matplotlib.font_manager import FontProperties

         fontP = FontProperties()
         fontP.set_size('small')
         fontP.set_family('serif')
         fontP.set_weight('bold')

         ax.legend(loc='best', prop=fontP)
         fig
```



1.4 Setup a development enviroment

This document is a guide to install an enviroment to contribute to the development of Sur library. It's fully based on modern Ubuntu Linux distributions (particularly Ubuntu 15.10) but should be straight forward follow this on any other linux distribution.

1.4.1 Install system packages

1. Install system packages:

```
$ sudo apt-get install git python-dev python-pip wine gfortran \
                       g++ libfreetype6-dev libpng12-dev xclip
$ sudo pip install virtualenvwrapper
```

Note: To code, of course you'll also need an editor. Could be anyone you prefer! Some people prefers simple ones like `gedit` (installed by default) or geany. Some other prefers something more geeky like `vim` or even more complete tools like `pydev` or `ninja-ide`.

1.4.2 Make a virtual enviroment

We use virtualenv (using virtualenvwrapper) to isolate our enviroment for other projects and system-wide python packages

2. Setup virtualenvwrapper

   2.1 Create a directory to hold your virtualenvs:

```
mkdir ~/.virtualenvs
mkdir ~/projects
mkdir ~/.pip_download_cache
```

2.2 Edit you `~/.bashrc` adding these lines:

```
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
export PROJECT_HOME=$HOME/projects      #folder for new projects. Could be what you want
```

2.3 And reload that:

```
$ source ~/.bashrc
```

3. Now create the project sur:

```
$ mkproject sur
```

This will create a new virtualenv in the WORKON_HOME binded to a project directory in PROJECT_HOME

Note: Next times, when you want to active the sur's virtualenv you'll run:

```
$ workon sur
```

When you want to deactive the virtualenv, on any path

```
(sur)~/projects/sur$ deactivate
```

1.4.3 Checkout the code

sur's git repo isn't public, so you need to have credentials to read and/or write it. Please if you don't have a Bitbucket account, create one:

1. Go to https://bitbucket.org/account/signup/ and sign up
2. Let me know (gaitan@gmail.com) your username and ask for dev permissions on Sur. If you don't have an user on bitbucket also let me know first. I'll send and invitation
3. Setup a ssh-key

```
$ ssh-keygen
$ xclip -sel clip < ~/.ssh/id_rsa.pub

and paste this on
https://bitbucket.org/account/user/<your_userr>/ssh-keys/
```

4. Then we go:

```
(sur)~/projects/sur$ git clone git@bitbucket.org:phasety/envelope-sur.git .
(sur)~/projects/sur$ git checkout develop
```

5. Remember to configure your identity (so, your future great code will be recognized):

```
(sur)$ git config --global user.name "Juan Perez"
(sur)$ git config --global user.email perez@phasety.com
```

1.4.4 Install in dev mode

(sur)~/projects/sur$ pip install -U numpy pip jupyter nose (sur)~/projects/sur$ pip install -e .

Tip: The flag `-e` on the last command means this package will be *editable*. Every change on the code of Sur will impact automatically. It's the same than `python setup.py develop`

Then check if it was installed:

```
(sur) $ ipython

In [1]: import sur    # may take few seconds to load
```

Run tests:

```
(sur) $ nosetests
```

And open an example notebook for lib usage:

```
(sur) $ jupyter notebook examples/basic_envelope.ipynb
```

¡Happy coding!

CHAPTER 2

API reference

2.1 sur.apps module

2.2 sur.envelope module

2.3 sur.envelope_sp module

2.4 sur.eos module

class sur.eos.**SRK**

Bases: sur.eos.CubicModel

Soave modification of Redlich–Kwong EOS

MODEL_ID = 1

MODEL_NAME = 'SRK'

class sur.eos.**PR**

Bases: sur.eos.CubicModel

Peng–Robinson equation of state

MODEL_ID = 2

MODEL_NAME = 'PR'

class sur.eos.**RKPR**

Bases: sur.eos.CubicModel

RKPR equation of state

MODEL_ID = 3

MODEL_NAME = 'RKPR'

classmethod from_constants(*tc*, *pc*, *acentric_factor*, *vc=None*, *del1=None*, *rhoLsat_t=()*, *pvdat=()*)

Given the constants of a pure compound, returns a tuple of arrays with the same constants adjusted and the arrays of correspond model's parameters.

Constants requires tc, pc, and acentric_factor as mandatory and vc or del1 or rhoLsat_t=(rhoLsat, T).

If a point pvdat = (P, T) is given, it's used as a constraint to calculate the vapor pressure

Returns (constants, model_parameters)

Where:

```
constants = array([tc, pc, acentric_factor, vc])
models_parameters =  array([ac, b, del1, rk])
```

classmethod `from_parameters`(*ac*, *b*, *del1*, *rk*)
Given the model's parameters, returns a tuple of arrays with the pure compound constants adjusted and the the ajusted model's array of parameters.

Returns (constants, model_parameters)

Where:

```
constants = array([Tc, Pc, acentric_factor, Vc])
models_parameters =  array([ac, b, del1, rk])
```

`sur.eos.`**`get_eos`**(*model*)

2.5 sur.manage module

2.6 sur.models module

2.7 sur.plots module

Plot shortcuts utilities

`sur.plots.`**`multiplot`**(*envelopes=None*, *experimental_envelopes=None*, *formats=None*, *critical_point='o'*, *experimental_colors=None*, *experimental_markers=None*, *legends=None*)
merge the plot of multiples envelopes

2.8 sur.settings module

2.9 sur.tools module

2.10 sur.units module

2.11 Module contents

`sur.`**`data`**(*a*)

`sur.`**`setup_database`**()
this is a hackish trick.

It setup the django enviroment through setup_environ(settings)

Then populates the database but, instead of fixtures, it dumps db on disk ('disk') to a memory one ('default'), and then syncs the missing tables on the latter.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

S

sur, 36
sur.eos, 35
sur.manage, 36
sur.plots, 36
sur.settings, 36

D

data() (in module sur), 36

F

from_constants() (sur.eos.RKPR class method), 35
from_parameters() (sur.eos.RKPR class method), 36

G

get_eos() (in module sur.eos), 36

M

MODEL_ID (sur.eos.PR attribute), 35
MODEL_ID (sur.eos.RKPR attribute), 35
MODEL_ID (sur.eos.SRK attribute), 35
MODEL_NAME (sur.eos.PR attribute), 35
MODEL_NAME (sur.eos.RKPR attribute), 35
MODEL_NAME (sur.eos.SRK attribute), 35
multiplot() (in module sur.plots), 36

P

PR (class in sur.eos), 35

R

RKPR (class in sur.eos), 35

S

setup_database() (in module sur), 36
SRK (class in sur.eos), 35
sur (module), 36
sur.eos (module), 35
sur.manage (module), 36
sur.plots (module), 36
sur.settings (module), 36