Carter, Problem 2-7

Here's the Dieterici equation of state:

In [54]:
```
# Gotcha's:  make sure you declare variables and use * everywhere there is a multi
plication
P,v,T, a,b,R=var('P v T a b R')

# Define the r.h.s. of the equation of state as an "expression"
Dieterici=(R*T/(v-b))*exp(-a/(R*T*v))
show(Dieterici)
```

Out[54]:
$$-\frac{RTe^{\left(-\frac{a}{RTv}\right)}}{b-v}$$

The critical point occurs at unique values of T, v and P such that

1. The Dieterici equation of state is satisfied,
2. $(\partial P/\partial v)_T = 0$, the slope of the P-V curve is 0 at the critical point, and
3. $(\partial^2 P/\partial v^2)_T = 0$, the concavity of the P-V curve is 0 at the critical point,

The values that simultaneously solve these three equations are $(P, T, v) = (P_c, T_c, v_c)$.

First approach

Define the three equations, and solve them simultaneously. What could be simpler??

In [55]:
```
# First equation - the equation of state
eq1 = P==Dieterici
show(eq1)
```

Out[55]:
$$P = -\frac{RTe^{\left(-\frac{a}{RTv}\right)}}{b-v}$$

In [56]:
```
# Second equation - first derivative of P wrt V == 0:
firstderiv = diff(Dieterici,v)
eq2 = firstderiv==0
show(eq2)
```

Out[56]:
$$-\frac{RTe^{\left(-\frac{a}{RTv}\right)}}{(b-v)^2} - \frac{ae^{\left(-\frac{a}{RTv}\right)}}{(b-v)v^2} = 0$$

In [57]:
```
# Third equation - second derivative of P wrt V == 0
secondderiv = diff(firstderiv,v)
eq3 = secondderiv == 0
show(eq3)
```

Out[57]:
$$-\frac{2\,RTe^{\left(-\frac{a}{RTv}\right)}}{(b-v)^3} + \frac{2\,ae^{\left(-\frac{a}{RTv}\right)}}{(b-v)v^3} - \frac{2\,ae^{\left(-\frac{a}{RTv}\right)}}{(b-v)^2v^2} - \frac{a^2e^{\left(-\frac{a}{RTv}\right)}}{RT(b-v)v^4} = 0$$

```
In [58]: solve(   [eq1,eq2,eq3], P, v, T)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-58-f650b6e88ba3> in <module>()
----> 1 solve(   [eq1,eq2,eq3], P, v, T)

/ext/sage/sage-8.3_1804/local/lib/python2.7/site-packages/sage/symbolic/relation
.pyc in solve(f, *args, **kwds)
   1133             s = []
   1134
-> 1135     sol_list = string_to_list_of_solutions(repr(s))
   1136
   1137     # Relaxed form suggested by Mike Hansen (#8553):

/ext/sage/sage-8.3_1804/local/lib/python2.7/site-packages/sage/symbolic/relation
.pyc in string_to_list_of_solutions(s)
    578     from sage.structure.sequence import Sequence
    579     from sage.calculus.calculus import symbolic_expression_from_maxima_s
tring
--> 580     v = symbolic_expression_from_maxima_string(s, equals_sub=True)
    581     return Sequence(v, universe=Objects(), cr_str=True)
    582

/ext/sage/sage-8.3_1804/local/lib/python2.7/site-packages/sage/calculus/calculus
.pyc in symbolic_expression_from_maxima_string(x, equals_sub, maxima)
   2157             _augmented_syms = {}
   2158     except SyntaxError:
-> 2159         raise TypeError("unable to make sense of Maxima expression '%s'
in Sage"%s)
   2160     finally:
   2161         is_simplified = False

TypeError: unable to make sense of Maxima expression '[if(union(if(_SAGE_VAR_a!=
0,(e^2*_SAGE_VAR_b!=0)and(4*e^2*_SAGE_VAR_b^4!=0)and(4*e^2*_SAGE_VAR_a*_SAGE_VAR
_b^6!=0),union()),if(_SAGE_VAR_a!=0,(_SAGE_VAR_v*e^(_SAGE_VAR_a/(_SAGE_VAR_R*_SA
GE_VAR_T*_SAGE_VAR_v))-_SAGE_VAR_b*e^(_SAGE_VAR_a/(_SAGE_VAR_R*_SAGE_VAR_T*_SAGE
_VAR_v))!=0)and(_SAGE_VAR_v^4*e^(_SAGE_VAR_a/(_SAGE_VAR_R*_SAGE_VAR_T*_SAGE_VAR_
v))-2*_SAGE_VAR_b*_SAGE_VAR_v^3*e^(_SAGE_VAR_a/(_SAGE_VAR_R*_SAGE_VAR_T*_SAGE_VA
R_v))+_SAGE_VAR_b^2*_SAGE_VAR_v^2*e^(_SAGE_VAR_a/(_SAGE_VAR_R*_SAGE_VAR_T*_SAGE_
VAR_v))!=0)and(_SAGE_VAR_R*_SAGE_VAR_T*_SAGE_VAR_v^7*e^(_SAGE_VAR_a/(_SAGE_VAR_R
*_SAGE_VAR_T*_SAGE_VAR_v))-3*_SAGE_VAR_R*_SAGE_VAR_T*_SAGE_VAR_b*_SAGE_VAR_v^6*e
^(_SAGE_VAR_a/(_SAGE_VAR_R*_SAGE_VAR_T*_SAGE_VAR_v))+3*_SAGE_VAR_R*_SAGE_VAR_T*_
SAGE_VAR_b^2*_SAGE_VAR_v^5*e^(_SAGE_VAR_a/(_SAGE_VAR_R*_SAGE_VAR_T*_SAGE_VAR_v))
-_SAGE_VAR_R*_SAGE_VAR_T*_SAGE_VAR_b^3*_SAGE_VAR_v^4*e^(_SAGE_VAR_a/(_SAGE_VAR_R
*_SAGE_VAR_T*_SAGE_VAR_v))!=0),union())),if(_SAGE_VAR_a!=0,[_SAGE_VAR_P==(e^(2*I
*pi*z137-2)*_SAGE_VAR_a)/(4*_SAGE_VAR_b^2)],union()),union())]' in Sage
```

Rats! This doesn't work! Somehow it's too complicated for SageMath!

But notice that `eq2` and `eq3` do not involve P at all. Perhaps we can solve those 2 for T and v, and then substitute those values back into `eq1` to get P. We could also divide out the exponential factor in `eq2` and `eq3` to make those a bit easier to solve. ...This works!

But in the meantime, I accidentally found **an even easier approach**: If you just give the variables P, v, and T to `solve(...)` **in a different order**, with P last, it will work right away! Like this...

In [59]:
```
solve(   [eq1,eq2,eq3], v, T, P)
```

Out[59]:
```
[[v == b, T == 0, P == r24], [v == 2*b, T == 1/4*a/(R*b), P == 1/4*a*e^(-2)/b^2]
]
```

The first solution, with $T = 0$, and $v = b$ (where $(v - b)$ is in the denominator!) Doesn't sound too attractive.

The second solution is what we want, and matches the solution that Carter gives:

$$v_c = 2b; \quad T_c = \frac{a}{4Rb}; \quad P_c = \frac{a}{4b^2}e^{-2}.$$

Finally, we wish to calculate

$$RT_c/(P_c v_c)$$

In [65]:
```
vc=2*b
Tc=a/(4*R*b)
Pc=(a/(4*b^2))*e^(-2)

simplify(R*Tc/(Pc*vc))
```

Out[65]:
```
1/2*e^2
```

In [66]:
```
# evaluate the numerical value of the previous cell
n(_)
```

Out[66]:
```
3.69452804946533
```

3.6945: This is in the middle of the values in the table of problem 2-6, and close to the value for CO_2! And interestingly, that value is independent of the values of a and b.