test_statistical_physics

November 18, 2022

0.1 test_statistical_physics.py

[1]:
```python
"""
test_statistical_physics.py

omec.class_type = ""
omec.__init__()
omec.solver.verbose = True
commands = ["solve", "NewtonsLaw2", a]
print(omec.process(commands))
"""
import copy
import sys
import os
lstPaths = ["../src"]
for ipath in lstPaths:
    if ipath not in sys.path:
        sys.path.append(ipath)
from libsympy import *
from statistical_physics import *
# print(sys.version)
# print(sys.path)
# Execute jupyter-notebook related commands.
exec(open('libnotebook.py').read())
```

```
libsympy is loaded.
libnotebook is loaded.
```

0.1.1 Settings

[2]:
```python
### Settings
class sets:
    """
    Setttings class.

    Instead of settings class, settings nametuble might be used.
    Settings = namedtuple("Settings", "type dropinf delta")
    sets = Settings(type="symbolic", dropinf=True, delta=0.1)
```

```
    """
    global dictflow, test_all

    def __init__(self):
        pass

    # File settings
    input_dir  = "input/statistical_physics"
    output_dir = "output/statistical_physics"

    # Plotting settings
    plot_time_scale = {1:"xy", 2:"xz", 3:"yz"}[3]

    # Execution settings.
    test_all = {0:False, 1:True}[1]
    dictflow = {100:"get_formulary",
                310:"1D_1/2_paramagnet_way1", 311:"1D_1/2_paramagnet_way2",
                331:"1D_simple_harmonic_oscillator", 332:"",
                430:"monoatomic_ideal_gas",
                710:"ideal_gas_canonical"}
    flow = [dictflow[i] for i in [310,311]]
    if test_all: flow = [dictflow[i] for i in dictflow.keys()]
```

[ ]:
```
print("Test of the {0}.".format(sets.flow))
```

0.1.2 get_formulary

[ ]:
```
### get_formulary
if "get_formulary" in sets.flow:
    ostat.class_type = "micro_canonical_discrete_distinguihable"
    ostat.__init__()
    ostat.get_formulary()

    ostat.class_type = "micro_canonical_discrete_indistinguihable"
    ostat.__init__()
    ostat.get_formulary()

    ostat.class_type = "micro_canonical_continuous_indistinguihable"
    ostat.__init__()
    ostat.get_formulary()
```

0.2 3 Paramagnets and Oscillators

0.2.1 A Spin-1/2 Paramagnet Way1

[4]:
```
### A Spin-1/2 Paramagnet Way1
if "1D_1/2_paramagnet_way1" in sets.flow:
    print("A Spin-1/2 Paramagnet Way1")

    ostat.class_type = "micro_canonical_discrete_distinguihable"
    ostat.__init__()
    ostat.solver.verbose = True
    [mu,B] = symbols('mu B', real=True)
    xreplaces = {g:1, engF:mu*B*(2*i-3), j:1, n:2}
    display("Single particle partition function:", ostat.Zsp)
    """
    res = ostat.Zsp.xreplace(xreplaces)
    display(simplify(res.doit()))
    """
    commands = ["xreplace", "ostat.Zsp", xreplaces]
    ostat.process(commands)
    Zsp = simplify(ostat.result.doit())
    display(Zsp)

    commands = ["xreplace", "ostat.U", xreplaces]
    ostat.process(commands)
    U = simplify(ostat.result.doit())
    display(U)

    commands = ["xreplace", "ostat.Cv", xreplaces]
    ostat.process(commands)
    Cv = simplify(ostat.result.doit())
    display(Cv)

    commands = ["xreplace", "ostat.S", xreplaces]
    ostat.process(commands)
    S = simplify(ostat.result.doit())
    display(S)

    commands = ["xreplace", "ostat.F", xreplaces]
    ostat.process(commands)
    F = simplify(ostat.result.doit())
    display(F)

    commands = ["xreplace", "ostat.M", xreplaces]
    ostat.process(commands)
    M = simplify(ostat.result.doit())
    display(M)
```

```
simplify(ostat.M.evalf(subs=xreplaces).doit())
```

A Spin-1/2 Paramagnet Way1

'Single particle partition function:'

$$Z_{sp}(\varepsilon(i), T) = \sum_{i=j}^{n} g(i) e^{-\frac{\varepsilon(i)}{Tk_B}}$$

'xreplace ostat.Zsp {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2}'

Eq(Z_sp(varepsilon(i), T), Sum(g(i)*exp(-varepsilon(i)/(T*k_B)), (i, j, n)))(xreplace, {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2})

$$Z_{sp}(B\mu\left(2i-3\right), T) = \sum_{i=1}^{2} e^{-\frac{B\mu(2i-3)}{Tk_B}}$$

$$Z_{sp}(B\mu\left(2i-3\right), T) = 2\cosh\left(\frac{B\mu}{Tk_B}\right)$$

'xreplace ostat.U {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2}'

Eq(U(T), N*T**2*k_B*Derivative(log(Sum(g(i)*exp(-varepsilon(i)/(T*k_B)), (i, j, n))), T))(xreplace, {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2})

$$U(T) = NT^2 k_B \frac{\partial}{\partial T} \log\left(\sum_{i=1}^{2} e^{-\frac{B\mu(2i-3)}{Tk_B}}\right)$$

$$U(T) = -BN\mu\tanh\left(\frac{B\mu}{Tk_B}\right)$$

'xreplace ostat.Cv {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2}'

Eq(C_v(T), Derivative(N*T**2*k_B*Derivative(log(Sum(g(i)*exp(-varepsilon(i)/(T*k_B)), (i, j, n))), T), T))(xreplace, {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2})

$$C_v(T) = \frac{\partial}{\partial T} NT^2 k_B \frac{\partial}{\partial T} \log\left(\sum_{i=1}^{2} e^{-\frac{B\mu(2i-3)}{Tk_B}}\right)$$

$$C_v(T) = \frac{4B^2N\mu^2 e^{\frac{2B\mu}{Tk_B}}}{T^2 k_B \left(e^{\frac{4B\mu}{Tk_B}} + 2e^{\frac{2B\mu}{Tk_B}} + 1\right)}$$

'xreplace ostat.S {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2}'

Eq(S(T), N*T*k_B*Derivative(log(Sum(g(i)*exp(-varepsilon(i)/(T*k_B)), (i, j, n))), T) + N*k_B*log(Sum(g(i)*exp(-varepsilon(i)/(T*k_B)), (i, j, n))))(xreplace, {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2})

$$S(T) = NTk_B \frac{\partial}{\partial T} \log \left(\sum_{i=1}^{2} e^{-\frac{B\mu(2i-3)}{Tk_B}} \right) + Nk_B \log \left(\sum_{i=1}^{2} e^{-\frac{B\mu(2i-3)}{Tk_B}} \right)$$

$$S(T) = \frac{N \left(-B\mu \left(e^{\frac{2B\mu}{Tk_B}} - 1 \right) + \log \left(\left(2\cosh\left(\frac{B\mu}{Tk_B}\right) \right)^{Tk_B \left(e^{\frac{2B\mu}{Tk_B}} + 1 \right)} \right) \right)}{T \left(e^{\frac{2B\mu}{Tk_B}} + 1 \right)}$$

'xreplace ostat.F {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2}'

Eq(F(T, B), -N*T*k_B*log(Sum(g(i)*exp(-varepsilon(i)/(T*k_B)), (i, j, n))))(xreplace, {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2})

$$F(T,B) = -NTk_B \log \left(\sum_{i=1}^{2} e^{-\frac{B\mu(2i-3)}{Tk_B}} \right)$$

$$F(T,B) = -\log \left(\left(2\cosh\left(\frac{B\mu}{Tk_B}\right) \right)^{NTk_B} \right)$$

'xreplace ostat.M {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2}'

Eq(M(T, B), -Derivative(-N*T*k_B*log(Sum(g(i)*exp(-varepsilon(i)/(T*k_B)), (i, j, n))), B))(xreplace, {g(i): 1, varepsilon(i): B*mu*(2*i - 3), j: 1, n: 2})

$$M(T,B) = -\frac{\partial}{\partial B} \left(-NTk_B \log \left(\sum_{i=1}^{2} e^{-\frac{B\mu(2i-3)}{Tk_B}} \right) \right)$$

$$M(T,B) = N\mu \tanh\left(\frac{B\mu}{Tk_B}\right)$$

0.2.2 A Spin-1/2 Paramagnet Way2

[5]:
```
### A Spin-1/2 Paramagnet Way2
if "1D_1/2_paramagnet_way2" in sets.flow:
    print("A Spin-1/2 Paramagnet Way2")

    ostat.class_type = "micro_canonical_discrete_distinguihable"
    ostat.__init__()
    ostat.solver.verbose = True
    [mu,B] = symbols('mu B', real=True)
    xreplaces = {g:1, engF:mu*B*(2*i-3), j:1, n:2}
    display("Single particle partition function:", ostat.Zsp)

    Zsp = simplify(ostat.Zsp.evalf(subs=xreplaces).doit())
    U   = simplify(  ostat.U.evalf(subs=xreplaces).doit())
```

```
    Cv  = simplify( ostat.Cv.evalf(subs=xreplaces).doit())
    S   = simplify(  ostat.S.evalf(subs=xreplaces).doit())
    F   = simplify(  ostat.F.evalf(subs=xreplaces).doit())
    M   = simplify(  ostat.M.evalf(subs=xreplaces).doit())

    # list(map(display, [Zsp,U,Cv,S,F,M]))
    display(Zsp,U,Cv,S,F,M)
```

```
A Spin-1/2 Paramagnet Way2

'Single particle partition function:'
```

$$Z_{sp}(\varepsilon(i), T) = \sum_{i=j}^{n} g(i) e^{-\frac{\varepsilon(i)}{Tk_B}}$$

$$Z_{sp}(B\mu(2i-3), T) = 2\cosh\left(\frac{B\mu}{Tk_B}\right)$$

$$U(T) = -BN\mu\tanh\left(\frac{B\mu}{Tk_B}\right)$$

$$C_v(T) = \frac{4B^2N\mu^2 e^{\frac{2B\mu}{Tk_B}}}{T^2k_B\left(e^{\frac{4B\mu}{Tk_B}} + 2e^{\frac{2B\mu}{Tk_B}} + 1\right)}$$

$$S(T) = \frac{N\left(-B\mu\left(e^{\frac{2B\mu}{Tk_B}} - 1\right) + \log\left(\left(2\cosh\left(\frac{B\mu}{Tk_B}\right)\right)^{Tk_B\left(e^{\frac{2B\mu}{Tk_B}}+1\right)}\right)\right)}{T\left(e^{\frac{2B\mu}{Tk_B}} + 1\right)}$$

$$F(T, B) = -\log\left(\left(2\cosh\left(\frac{B\mu}{Tk_B}\right)\right)^{NTk_B}\right)$$

$$M(T, B) = N\mu\tanh\left(\frac{B\mu}{Tk_B}\right)$$

0.2.3 An Array of 1-D Simple Harmonic Oscillators

[6]:
```
### An Array of 1-D Simple Harmonic Oscillators
if "1D_simple_harmonic_oscillator" in sets.flow:
    print("An Array of 1-D Simple Harmonic Oscillators")

    ostat.class_type = "micro_canonical_discrete_distinguihable"
    ostat.__init__()
    ostat.solver.verbose = False
    [h,nu,theta] = symbols('h nu theta', real=True, positive=True)
    xreplaces = {g:1, engF:(i+S(1)/2)*h*nu, j:0, n:inf, (h*nu)/kB:theta}
```

```
    display("Single particle partition function:", ostat.Zsp)

    commands = ["xreplace", "ostat.Zsp", xreplaces]
    ostat.result = ostat.process(commands).rhs
    Zsp = simplify(ostat.result.doit())
    display(Zsp)

    commands = ["xreplace", "ostat.U", xreplaces]
    ostat.result = ostat.process(commands).rhs
    U = simplify(ostat.result.doit())
    display(U)

    commands = ["xreplace", "ostat.Cv", xreplaces]
    ostat.result = ostat.process(commands).rhs
    Cv = simplify(ostat.result.doit())
    display(Cv)
```

```
An Array of 1-D Simple Harmonic Oscillators


       ␣
↪---------------------------------------------------------------------------


        TypeError                                 Traceback (most recent call␣
↪last)

        Input In [6], in <cell line: 2>()
          7 ostat.solver.verbose = False
          8 [h,nu,theta] = symbols('h nu theta', real=True, positive=True)
    ----> 9 xreplaces = {g:1, engF:(i+S(1)/2)*h*nu, j:0, n:inf, (h*nu)/kB:theta}
         10 display("Single particle partition function:", ostat.Zsp)
         12 commands = ["xreplace", "ostat.Zsp", xreplaces]


        TypeError: 'Equality' object is not callable
```

0.3 An Array of 3-D Simple Harmonic Oscillators todo

0.4 4 Indistinguishable Particles and Monatomic Ideal Gases

0.4.1 Monoatomic Ideal Gas

[3]:
```
### Monoatomic Ideal Gas
if "monoatomic_ideal_gas" in sets.flow:
    print("Monoatomic Ideal Gas")

    ostat.class_type = "micro_canonical_continuous_indistinguihable"
    ostat.__init__()
```

```python
ostat.solver.verbose = False
[h,nu,theta] = symbols('h nu theta', real=True, positive=True)
xreplaces = {i:eng, g:4*m*pi*V*(2*m*eng)**(S(1)/2)/(h**3), engF:eng}
display("Single particle partition function:", ostat.Zsp)

commands = ["xreplace", "ostat.Zsp", xreplaces]
ostat.result = ostat.process(commands).rhs
Zsp = simplify(ostat.result.doit())
display(Zsp)

commands = ["xreplace", "ostat.U", xreplaces]
ostat.result = ostat.process(commands).rhs
U = simplify(ostat.result.doit())
display(U)

commands = ["xreplace", "ostat.S", xreplaces]
ostat.process(commands)
S = simplify(ostat.result.doit())
display(S)
```

```
Monoatomic Ideal Gas
```

```
'Single particle partition function:'
```

$$Z_{sp}(\varepsilon(i), T) = \int\limits_0^\infty g(i) e^{-\frac{\varepsilon(i)}{T k_B}}\, d\varepsilon$$

$$Z_{sp}(\varepsilon, T) = \int\limits_0^\infty \frac{4\sqrt{2}\pi V m^{\frac{3}{2}} \sqrt{\varepsilon} e^{-\frac{\varepsilon}{T k_B}}}{h^3}\, d\varepsilon$$

$$\frac{2\sqrt{2}\pi^{\frac{3}{2}} T^{\frac{3}{2}} V k_B^{\frac{3}{2}} m^{\frac{3}{2}}}{h^3}$$

$$U(T) = N T^2 k_B \frac{\partial}{\partial T} \log\left(\int\limits_0^\infty \frac{4\sqrt{2}\pi V m^{\frac{3}{2}} \sqrt{\varepsilon} e^{-\frac{\varepsilon}{T k_B}}}{h^3}\, d\varepsilon\right)$$

$$\frac{3 N T k_B}{2}$$

$$S(T) = N T k_B \frac{\partial}{\partial T} \log\left(\int\limits_0^\infty \frac{4\sqrt{2}\pi V m^{\frac{3}{2}} \sqrt{\varepsilon} e^{-\frac{\varepsilon}{T k_B}}}{h^3}\, d\varepsilon\right) + N k_B \log\left(\int\limits_0^\infty \frac{4\sqrt{2}\pi V m^{\frac{3}{2}} \sqrt{\varepsilon} e^{-\frac{\varepsilon}{T k_B}}}{h^3}\, d\varepsilon\right) - k_B \log(N!)$$

$$S(T) = \frac{k_B \left(2N \log\left(\frac{2\sqrt{2}\pi^{\frac{3}{2}} T^{\frac{3}{2}} V k_B^{\frac{3}{2}} m^{\frac{3}{2}}}{h^3}\right) + 3N - 2\log\left(\Gamma\left(N+1\right)\right)\right)}{2}$$

0.5 7 Electrons in Metals

0.5.1 The Ideal Gas in the Canonical Ensemble

[4]:
```
#---The Ideal Gas in the Canonical Ensemble
if "ideal_gas_canonical" in sets.flow:
    print("The Ideal Gas in the Canonical Ensemble")

    ostat.class_type = "canonical"
    ostat.__init__()
    ostat.solver.verbose = True

    ostat.ZN  = Eq( ostat.ZN.lhs, ostat.subformulary.Z_Ideal_Gas)
    display(ostat.F)
```

```
The Ideal Gas in the Canonical Ensemble
```

$$F(N,T,B) = -Tk_B \log\left(\frac{V^N \left(\frac{2\pi T k_B m}{h^2}\right)^{\frac{3N}{2}}}{N!}\right)$$

[ ]:
```
```