1 MFT series expansion of the Ising model

```
import numpy as np
import matplotlib.pylab as plt
from scipy.optimize import curve_fit

def bw_free_energy(n, m, h, kb, T):
    # Compute the Bragg-Williams free energy for the Ising model to n terms
    # Reduced unit: qJ = 1.0, Tc = 1.0
    Gn = -m**2 / 2
    Gn -= h * m

    taylor_sum = 0
    for k in range(1, n+1):
        taylor_sum += m**(2*k) * (1/(2*k-1) - 1/(2*k))
    Gn += kb * T * taylor_sum

    return Gn

def bw_first_derivative(n, m, h, kb, T):
    Gn = -m
    Gn -= h

    taylor_sum = 0
    for k in range(1, n+1):
        taylor_sum += 2*k*m**(2*k - 1) * (1/(2*k-1) - 1/(2*k))
    Gn += kb * T * taylor_sum

    return Gn

def bw_binary_search(n, h, kb, T, m1, m2, target_accuracy=1e-3):
    # Employ the binary search method to find the minimum of the
    #  approximate free energy
    accuracy = 1.0
    while accuracy > target_accuracy:
        f1 = bw_first_derivative(n, m1, h, kb, T)
        f2 = bw_first_derivative(n, m2, h, kb, T)
        assert f1 * f2 < 0
        m = (m1 + m2) / 2
        fmid = bw_first_derivative(n, m, h, kb, T)
        if fmid == 0.0:
            return m
        if fmid * f1 > 0:
            m1 = m
        else:
            m2 = m
        accuracy = np.abs(m1 - m2)

    return (m1 + m2) / 2

def exact_free_energy(m, h, kb, T):
    G = -m**2 / 2
    G -= h * m
    G += (1/2) * kb * T * ((1+m)*np.log(1+m) + (1-m)*np.log(1-m))
    return G
```

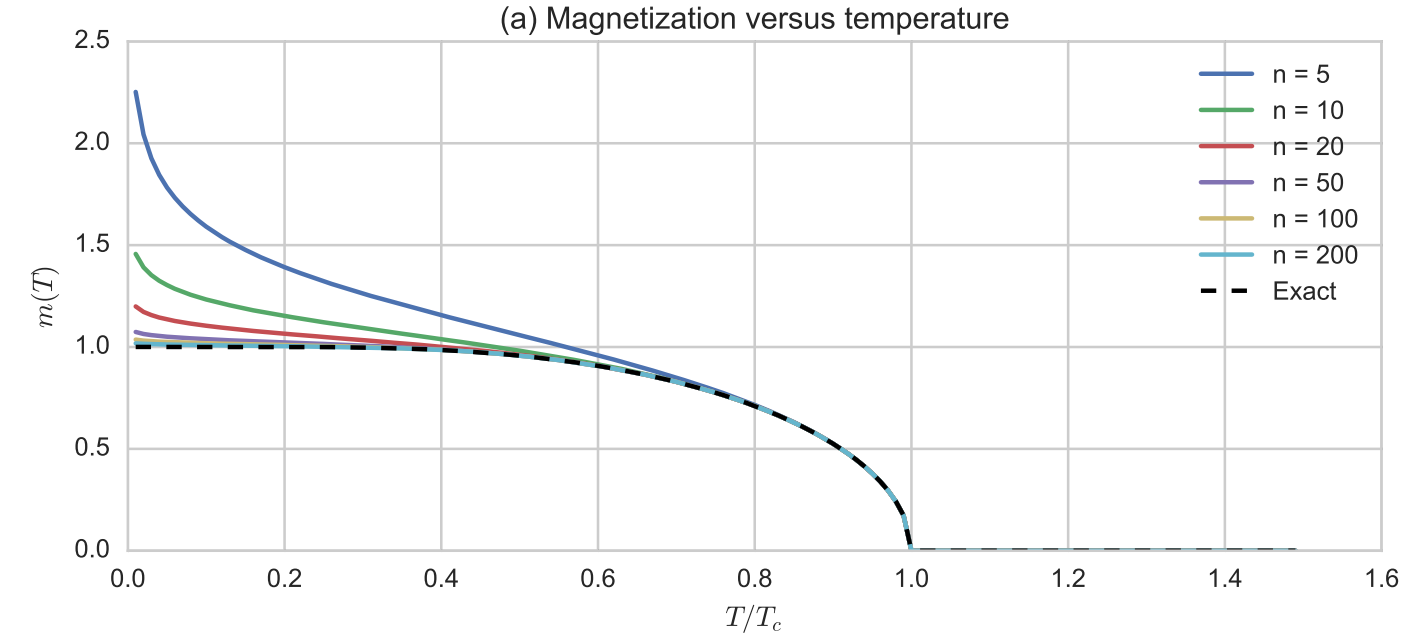
```
def exact_magnetization(m, h, kb, T):
    # The Curie-Weiss self-consistent solution for m(T)
    return np.tanh((1/(kb*T)) * (m + h))

def exact_binary_search(h, kb, T, m1, m2, target_accuracy=1e-3):
    # Employ the binary search method to find the solution to the
    #  exact magnetization at a given temperature
    accuracy = 1.0
    while accuracy > target_accuracy:
        f1 = m1 - exact_magnetization(m1, h, kb, T)
        f2 = m2 - exact_magnetization(m2, h, kb, T)
        assert f1 * f2 < 0
        m = (m1 + m2) / 2
        fmid = m - exact_magnetization(m, h, kb, T)
        if fmid == 0.0:
            return m
        if fmid * f1 > 0:
            m1 = m
        else:
            m2 = m
        accuracy = np.abs(m1 - m2)

    return (m1 + m2) / 2

if __name__ == '__main__':
    h = 0
    kb = 1.0
    n = 100
    m = np.linspace(-1, 1, 200)

    fig, axes = plt.subplots(figsize=(8,11), nrows=3)

    T_list = np.arange(0.01, 1.5, 0.01)

    exact_root_list = []
    for T in T_list:
        # If T < Tc, the root is nontrivial
        if T < 0.99999:
            exact_root = exact_binary_search(h, kb, T, 0.1, 2)
        else:
            exact_root = exact_binary_search(h, kb, T, -0.5, 0.5)
        exact_root_list.append(exact_root)
    exact_root_list = np.array(exact_root_list)

    n_plot = [5, 10, 20, 50, 100, 200]
    for n in n_plot:
        bw_root_list = []
        for T in T_list:
            if T < 0.99999:
                bw_root = bw_binary_search(n, h, kb, T, 0.1, 3)
            else:
                bw_root = bw_binary_search(n, h, kb, T, -0.5, 0.5)
            bw_root_list.append(bw_root)
```

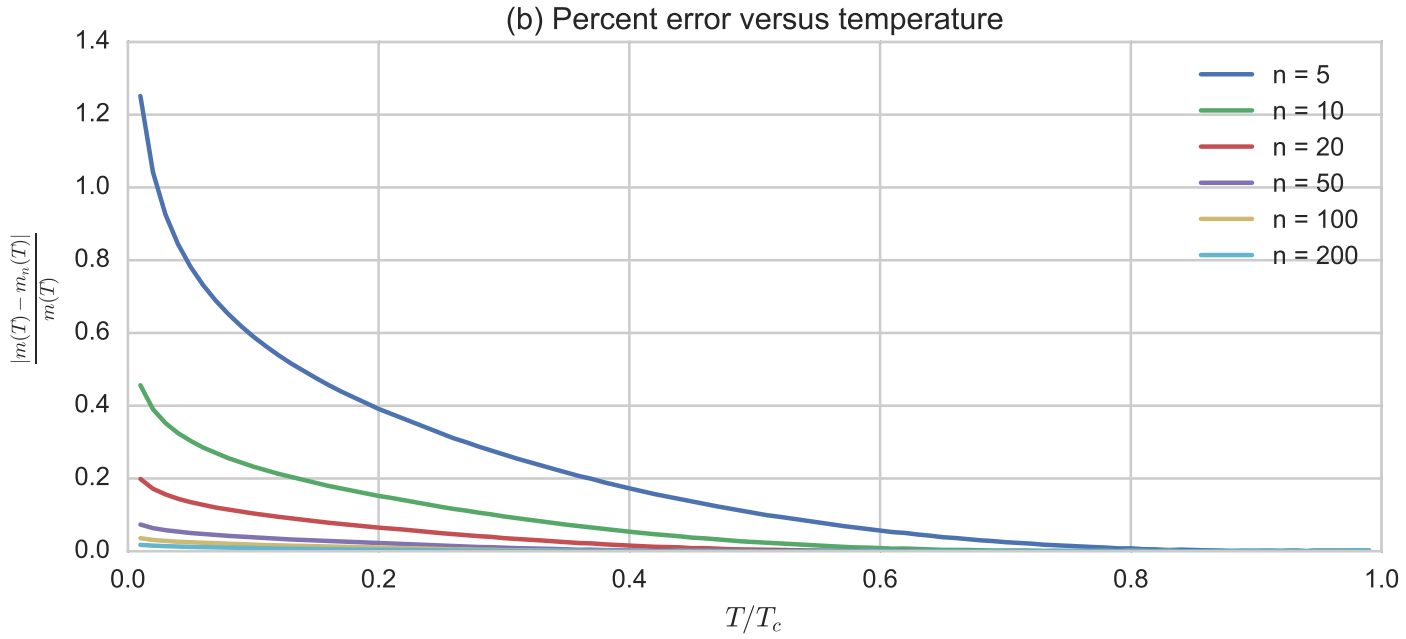
```
    bw_root_list = np.array(bw_root_list)
    diff = np.abs(bw_root_list - exact_root_list)
    error = diff / exact_root_list
    axes[0].plot(T_list, bw_root_list, label='n = %d' % n)
    axes[1].plot(T_list, error, label='n = %d' % n)
axes[0].plot(T_list, exact_root_list, 'k--', label='Exact')

axes[0].set_ylabel('$m(T)$')
axes[0].set_xlabel('$T/T_c$')
axes[1].set_ylabel(r'$\frac{|m(T) - m_n(T)|}{m(T)}$')
axes[1].set_xlabel('$T/T_c$')

axes[0].set_title('(a) Magnetization versus temperature')
axes[1].set_title('(b) Percent error versus temperature')

axes[0].legend()
axes[1].legend()

max_error_list = []
n_list = np.logspace(1, 3, 21)[:16]
for n in n_list:
    n = int(n)
    print(n)
    bw_root_list = []
    exact_root_list = []
    for T in T_list:
        if T < 0.99999:
            bw_root = bw_binary_search(n, h, kb, T, 0.1, 3)
            exact_root = exact_binary_search(h, kb, T, 0.1, 2)
        else:
            bw_root = bw_binary_search(n, h, kb, T, -0.5, 0.5)
            exact_root = exact_binary_search(h, kb, T, -0.5, 0.5)
        bw_root_list.append(bw_root)
        exact_root_list.append(exact_root)

    exact_root_list = np.array(exact_root_list)
    bw_root_list = np.array(bw_root_list)
    diff = np.abs(bw_root_list - exact_root_list)
    error = diff / exact_root_list
    error = error[np.isfinite(error)]
    max_error_list.append(np.max(error))

power_scaling = lambda n, a, p, c: a * n**p + c
popt, pcov = curve_fit(power_scaling, n_list, max_error_list,
                       [1, -0.5, n_list[0]])

axes[2].scatter(n_list, np.array(max_error_list))
axes[2].plot(n_list, power_scaling(n_list, *popt), 'k--',
             label='Error $\sim n^{{{0:.2f}}}$'.format(popt[1]))
axes[2].set_xscale('log')
axes[2].set_yscale('log')
axes[2].set_xlim([1e1, 1e3])
axes[2].set_ylabel(r'Max$\left(\frac{|m(T) - m_n(T)|}{m(T)}\right)$')
axes[2].set_xlabel('n')
axes[2].set_title('(c) Max error versus number of terms in expansion')
```

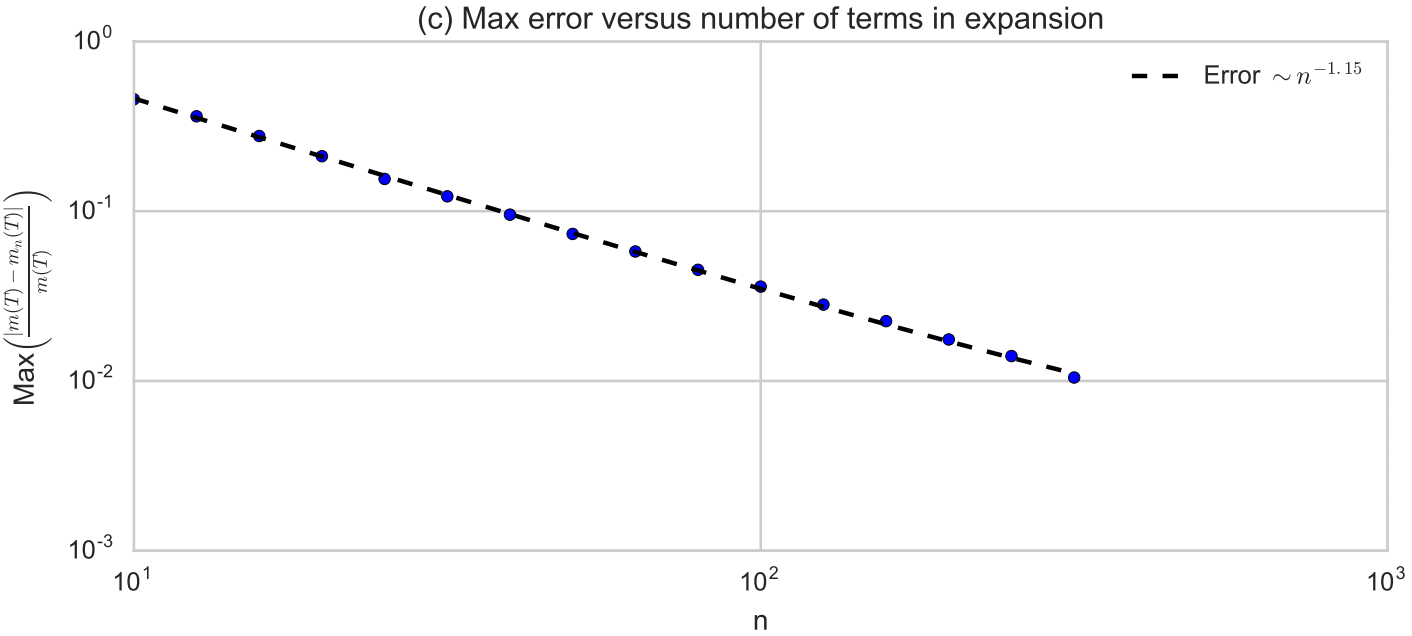
```
    axes[2].legend()

    fig.tight_layout()
    fig.savefig('ising.pdf')
```

The figure produced by the above code is on the next page.

When approximating the Ising magnetization, the largest deviation from the exact solution occurs at low values of T, as can be seen in (a) and (b). To determine the required number of terms n to reach agreement within 1%, the maximum error was computed at values ranging from $n = 10$ to 1000, but ended prematurely around $n \approx 300$ due to an overflow error (m^n terms being the culprit). Fortunately, this is also around the number of terms required to be within 1% of the exact solution. Additionally, the error was found to scale as approximately $n^{-1.15}$.


(a) Magnetization versus temperature
$m(T)$
T/T_c
n = 5
n = 10
n = 20
n = 50
n = 100
n = 200
Exact


(b) Percent error versus temperature
$\frac{|m(T)-m_n(T)|}{m(T)}$
T/T_c
n = 5
n = 10
n = 20
n = 50
n = 100
n = 200


(c) Max error versus number of terms in expansion
$\mathrm{Max}\left(\frac{|m(T)-m_n(T)|}{m(T)}\right)$
n
Error $\sim n^{-1.15}$