Homework 10 Spectral

ATSC 507

Christopher Rodell

Question 1

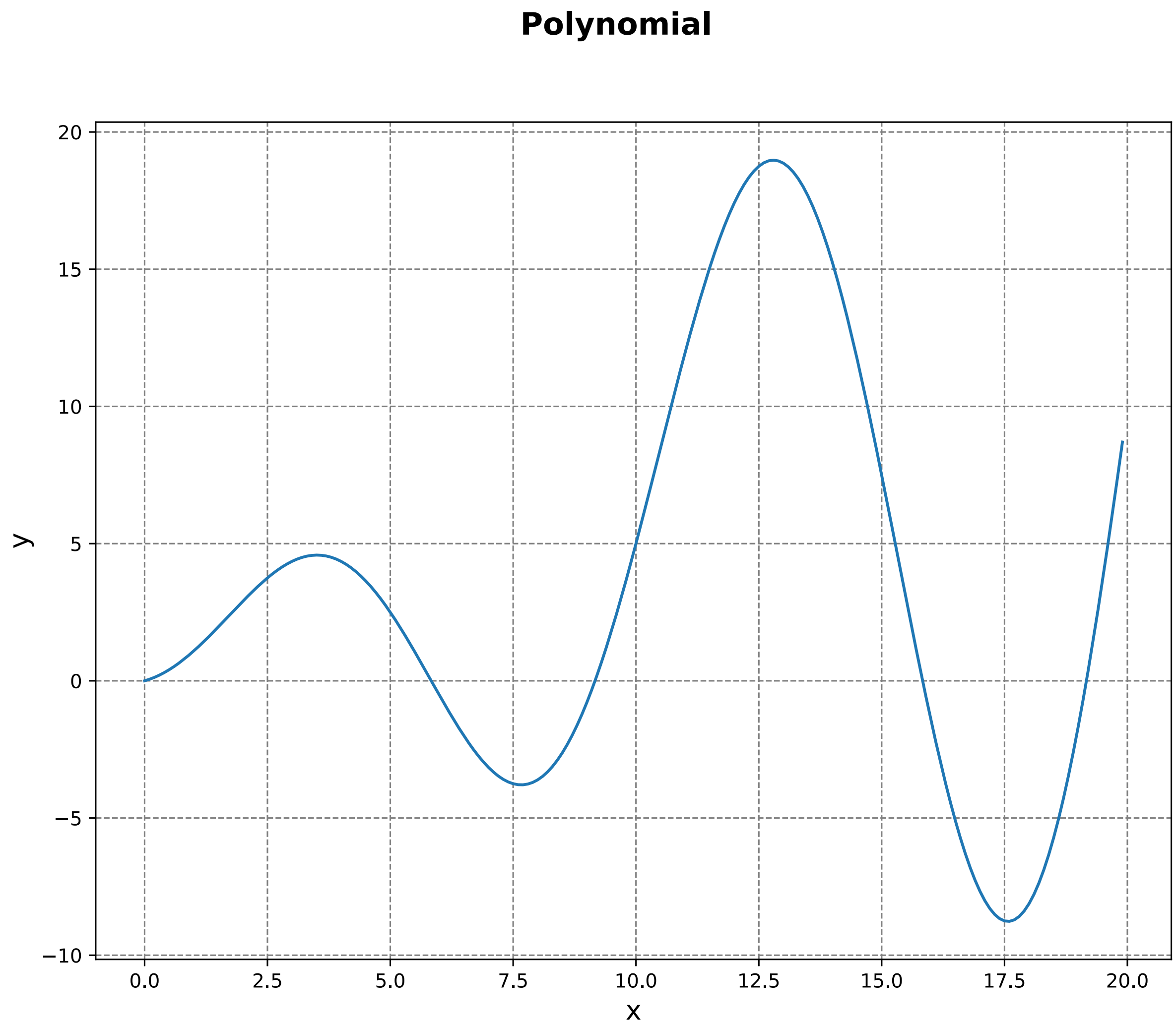
Given the polynomial

$$y = 0.5 * x + x * sin(\frac{2 * \pi * x}{10}) \qquad \text{for: } 0 \leq x \leq 20$$

- (a) plot this function

In [1]:
```
import context
import numpy as np
import matplotlib.pyplot as plt
from cr507.utils import plt_set


x = np.arange(0.,20,0.1)
y = 0.5 * x + x * np.sin((2 *np.pi * x)/ 10)


fig, ax = plt.subplots(1,1, figsize=(10,8))
fig.suptitle('Polynomial', fontsize= plt_set.title_size, fontweight="bol
d")
ax.plot(x,y)
ax.set_xlabel('x', fontsize = plt_set.label)
ax.set_ylabel('y', fontsize = plt_set.label)
ax.xaxis.grid(color='gray', linestyle='dashed')
ax.yaxis.grid(color='gray', linestyle='dashed')
# ax.legend()

# plt.show()
```

```
*****************************
context imported. Front of path:
/Users/rodell/atsc507
/private/var/folders/hc/bh1xlzfj3_n4c5gz42dbpw400000gn/T/e1167f98-a384-
43e9-b48e-3c7bd59e757c
*****************************

through /Users/rodell/atsc507/py/hw10/context.py -- pha
through /Users/rodell/atsc507/cr507/__init__.py pha II
```



Question 1 cont.

- (b) analytically integrate it, to find the exact solution.

$$f(x) = \int_0^{20} \frac{x}{2} dx + \int_0^{20} x \sin\left(\frac{\pi x}{5}\right) dx$$

Let:

$$u = x \quad v = \tfrac{5}{\pi}\cos\left(\tfrac{\pi x}{5}\right)$$
$$du = 1 \quad dv = \sin\left(\tfrac{\pi x}{5}\right)$$

$$f(x) = \Big|_0^{20} \frac{x^2}{4} - x\frac{5}{\pi}\cos\left(\frac{\pi x}{5}\right) - \left(-\frac{5}{\pi}\right)\int_0^{20} \cos\left(\frac{\pi x}{5}\right)dx$$
$$f(x) = \Big|_0^{20} \frac{x^2}{4} - \frac{5x}{\pi}\cos\left(\frac{\pi x}{5}\right) + \frac{25}{\pi^2}\sin\left(\frac{\pi x}{5}\right)$$

$$f(x) = f(20) - f(0)$$
$$f(x) = \left[\frac{20^2}{4} - \frac{5*20}{\pi}\cos\left(\frac{\pi*20}{5}\right) + \frac{25}{\pi^2}\sin\left(\frac{\pi*20}{5}\right)\right]$$
$$-\left[\frac{0^2}{4} - \frac{5*0}{\pi}\cos\left(\frac{\pi*0}{5}\right) + \frac{25}{\pi^2}\sin\left(\frac{\pi*0}{5}\right)\right]$$

$$\boxed{f(x) = 68.169}$$

Question 1 cont-

- (c) Use Gauss quadrature to numerically integrate it (using the eqs and tables in the handout, not any built-in integration function) for the following number of key points (m or n=): (i) 2 , (ii) 4 , (iii) 6 , (iv) 8 , and discuss how Gaussian quadrature converges to the exact solution. Show your work on your spreadsheet, or matlab, or your computer program. I

Use Gauss-Legendre quadrature:

$$\bar{I} = \frac{b-a}{2} \sum_{k=1}^{m} w_k f(x_k) \quad \text{to evaluate integral:} \quad I = \int_a^b f(x)dx$$

First transform

$$-1 \leqslant \xi \leqslant \mathbf{1} \quad \text{onto} \quad \mathrm{a} \leqslant x \leqslant \mathrm{b}$$

by

$$x = \frac{b+a}{2} + \frac{b-a}{2}\xi$$

Use Table A-1 Zeros and Weights for Gauss-Legendre Quadrature

m	$\pm\xi_k$	w_k
2	0.5773502692	1.0000000000
4	0.3399810436	0.6521451549
	0.8611363116	0.3478548451
6	0.2386191861	0.4679139346
	0.6612093865	0.3607615730
	0.9324695142	0.1713244924
8	0.1834346425	0.3626837834
	0.5255324099	0.3137066459
	0.7966664774	0.2223810345
	0.9602898565	0.1012285363

In [2]:
```
def guass(table):
    t = table
    a, b = 0, 20
    I_bar = []
    for i in range(len(t["Xi"])):
        x_p = (b + a)/2 + ((b - a)/2)*t["Xi"][i]
        fx_p = (0.5 * x_p) + x_p * np.sin((2 * np.pi * x_p)/ 10)
        I_p = fx_p * t["w"][i]

        x_n = (b + a)/2 + (((b - a)/2)* -t["Xi"][i])
        fx_n = (0.5 * x_n) + x_n * np.sin((2 * np.pi * x_n)/ 10)
        I_n = fx_n * t["w"][i]

        I_bar.append(I_p)
        I_bar.append(I_n)
    I_bar = np.array(I_bar)
    # print(I_bar)
    I_bar = ((b - a)/2) * np.sum(I_bar)
    return I_bar

m = ["m2", "m4", "m6", "m8"]
table = {"m2": {"Xi": [0.5773502692],
               "w": [1.0000000000]},
        "m4": {"Xi": [0.3399810436, 0.8611363116],
               "w": [0.6521451549, 0.3478548451]},
        "m6": {"Xi": [0.2386191861, 0.6612093865, 0.9324695142],
               "w": [0.4679139346, 0.3607615730, 0.1713244924]},
         "m8": {"Xi": [0.1834346425, 0.5255324099, 0.7966664774, 0.96028
98565],
               "w": [0.3626837834, 0.3137066459, 0.2223810345, 0.1012285363
]}}

I_bar_all = {}
for i in range(len(m)):
    I_bar = guass(table[m[i]])
    I_bar_all.update({m[i]: I_bar})

print("Gauss quadrature:  ", I_bar_all)
```

```
Gauss quadrature:   {'m2': 46.06414885515778, 'm4': 91.5555074549653,
'm6': 68.64561697413993, 'm8': 68.17079611506696}
```

Answer 1C

The Gauss-Legendre quadrature was remarkably accurate at with 8 points. In general, the more point used the more accurate the method becomes to the analytical solution. However, for a very large number of points run off errors, can cause a significant deterioration in accuracy.

Question 2

Search the internet fo find the type of truncation and its highest order M used for operational runs of (a) ECMWF. Relate these to Warner's L1-L4 on p49.

The table below shows the correspondence between spectral, Gaussian and latitude/longitude resolution for some ECMWF products.

Gaussian number (N)	Spectral truncation (T)	Approximate resolution in degrees [1]
N48	T63	1.875
N80	T159	1.125
N128	T255	0.75
N160	T319	0.5625
N256	T511	0.351
N320	T639	0.28125
N400	T799	0.225
N640	T1279	0.14
N1024	T2047	0.088

https://confluence.ecmwf.int/display/UDOC/What+is+the+connection+between+the+spectral+truncation+and+the +Metview+FAQ (https://confluence.ecmwf.int/display/UDOC/What+is+the+connection+between+the+spectral+truncation+and+th +Metview+FAQ)

Answer Q2

The ECMWF IFS model uses a spherical harmonic expansion of fields, truncated at a particular wavenumber. The largest spectral truncation occurs near the poles at an M = 1024. This is a much large M value or truncation value T2047 than Warren mentions. The spectral resolutions Warren mentions are much more coarse than what the ECMWF uses for its operational global forecast model.

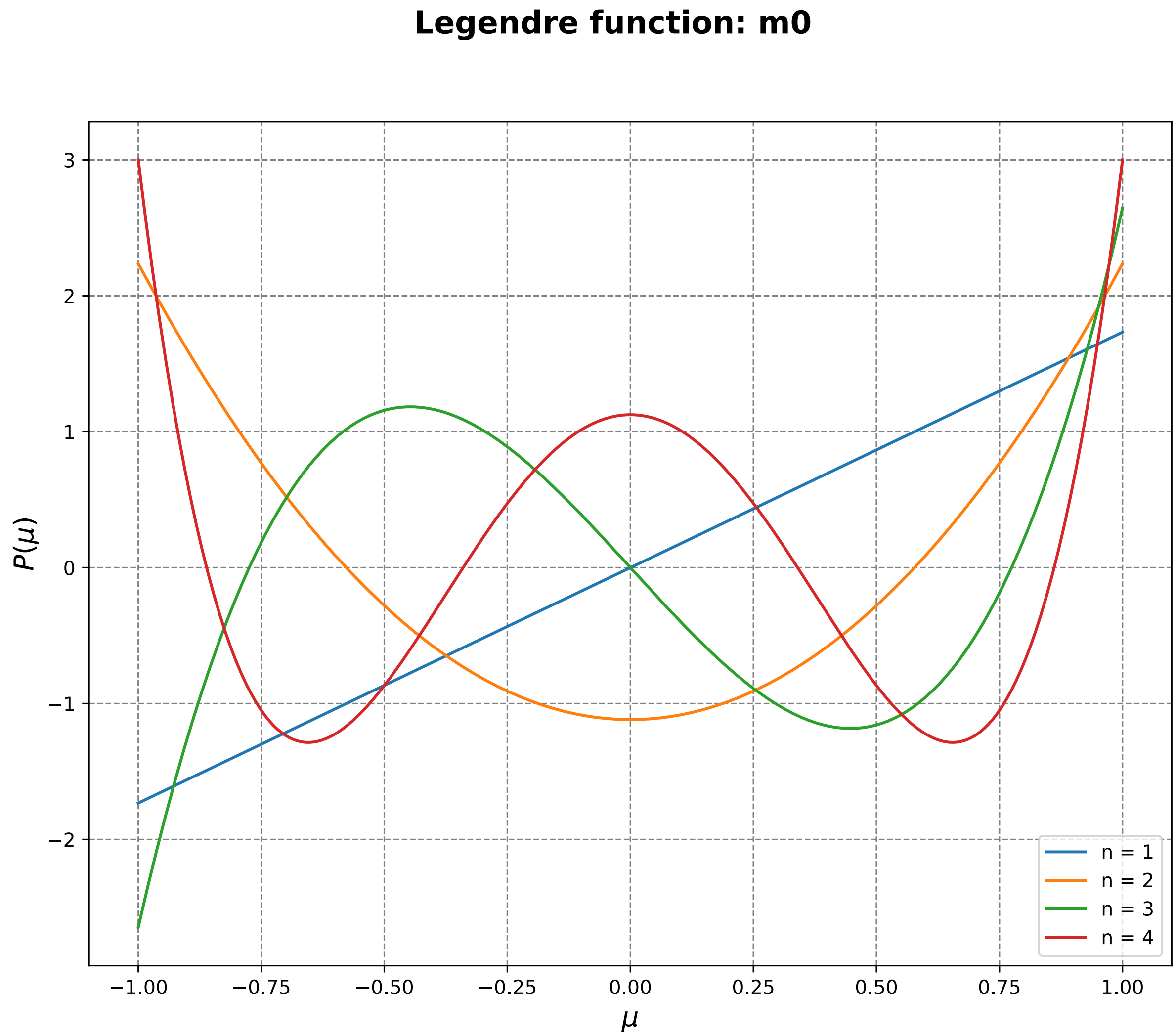
Question 3

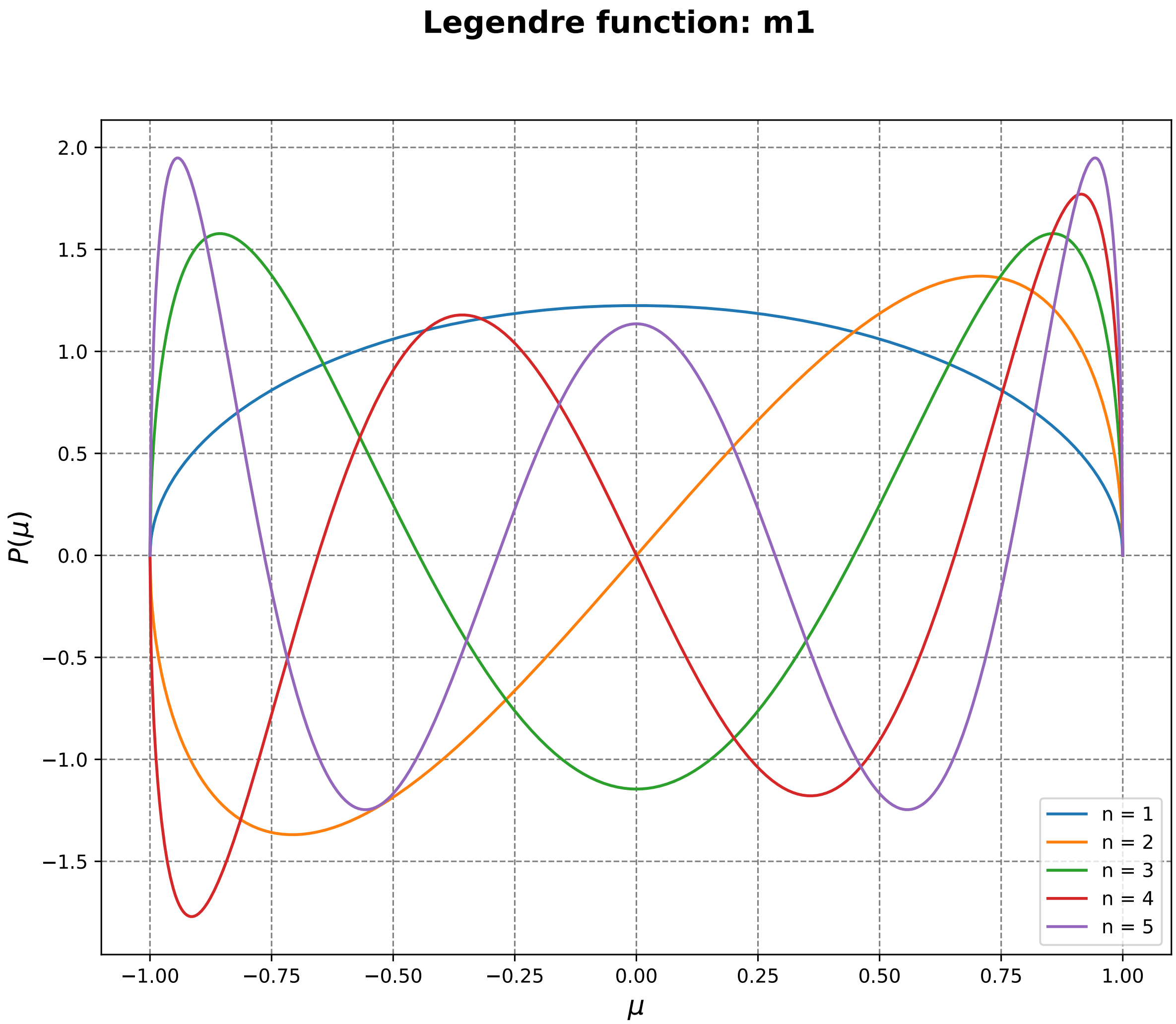
Plot eq. (4.22) of Coiffier similar to his Fig 4.2, but for: (a) m=0 with n=1 to 4, (b) m=1 with n=1 to5, and (c) m=2 with n=2 to 6.

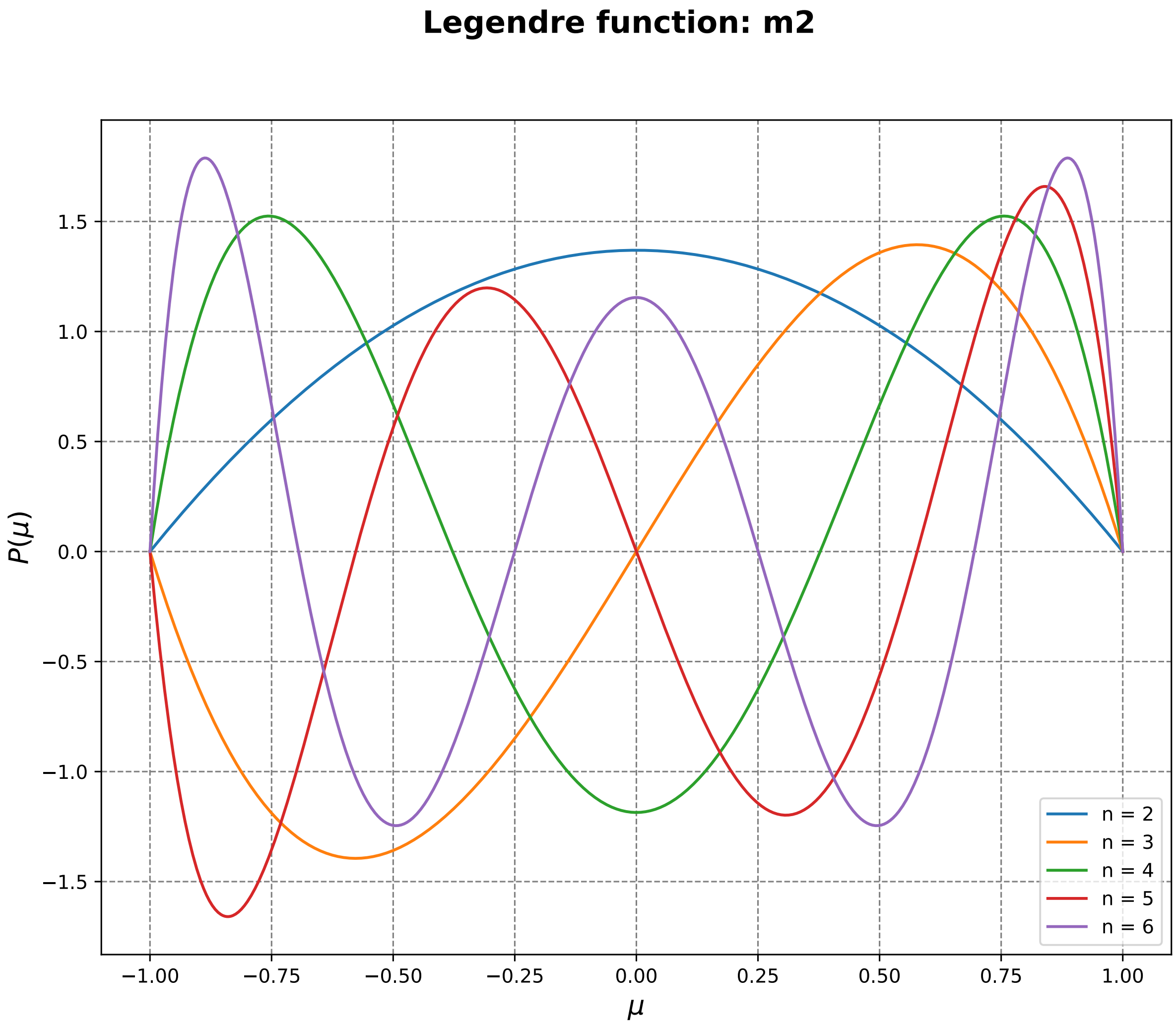
$$P_n^m(\mu) = \sqrt{(2n+1)\frac{(n-m)!}{(n+m)!}}\,\frac{\left(1-\mu^2\right)^{\frac{m}{2}}}{2^n n!}\,\frac{d^{n+m}}{d\mu^{n+m}}\left(\mu^2-1\right)^n \qquad \text{(Coiffier eq. 4.22)}$$

In [3]:
```
from sympy import *
from sympy.abc import x, mu
from scipy.special import factorial
from sympy.plotting import plot

m = ["m0", "m1", "m2"]
dictionary = {"m0": {"m": 0, "n": np.arange(1,5,1)},
              "m1": {"m": 1, "n": np.arange(1,6,1)},
              "m2": {"m": 2, "n": np.arange(2,7,1)}}


def derivatives(n,m):
    P = ((mu**2) - 1)**n
    Pprime = diff(P, mu,(m + n))
    k2 = Pprime * (1 - mu**2)**(m/2)
    return k2

def legendre(dictionary):
    d = dictionary
    m = d["m"]
    P_list = []
    for i in range(len(d["n"])):
        n = d["n"]
        k1 = ((2*n[i] + 1) * (factorial(n[i] - m)/factorial(n[i] + m)))*
*(1/2) * (1 / (2**n[i] * factorial(n[i])))
        k2 = derivatives(n[i],m)
        P = k1*k2
        P_list.append(P)
    return P_list
```

In [4]:
```
m0 = legendre(dictionary["m0"])
m1 = legendre(dictionary["m1"])
m2 = legendre(dictionary["m2"])

fig, ax = plt.subplots(1,1, figsize=(10,8))
fig.suptitle('Legendre function: m0', fontsize= plt_set.title_size, font
weight="bold")
ax.set_xlabel('$\mu$', fontsize = plt_set.label)
ax.set_ylabel('$P(\mu)$', fontsize = plt_set.label)
ax.xaxis.grid(color='gray', linestyle='dashed')
ax.yaxis.grid(color='gray', linestyle='dashed')
for i in range(len(m0)):
    lam_x = lambdify(mu, m0[i], modules=['numpy'])
    x_vals = np.linspace(-1, 1, 1000)
    y_vals = lam_x(x_vals)
    ax.plot(x_vals, y_vals, label = "n = " + str(dictionary["m0"]["n"][i
]))
    ax.legend()



fig, ax = plt.subplots(1,1, figsize=(10,8))
fig.suptitle('Legendre function: m1', fontsize= plt_set.title_size, font
weight="bold")
ax.set_xlabel('$\mu$', fontsize = plt_set.label)
ax.set_ylabel('$P(\mu)$', fontsize = plt_set.label)
ax.xaxis.grid(color='gray', linestyle='dashed')
ax.yaxis.grid(color='gray', linestyle='dashed')
for i in range(len(m1)):
    lam_x = lambdify(mu, m1[i], modules=['numpy'])
    x_vals = np.linspace(-1, 1, 1000)
    y_vals = lam_x(x_vals)
    ax.plot(x_vals, y_vals, label = "n = " + str(dictionary["m1"]["n"][i
]))
    ax.legend()



fig, ax = plt.subplots(1,1, figsize=(10,8))
fig.suptitle('Legendre function: m2', fontsize= plt_set.title_size, font
weight="bold")
ax.set_xlabel('$\mu$', fontsize = plt_set.label)
ax.set_ylabel('$P(\mu)$', fontsize = plt_set.label)
ax.xaxis.grid(color='gray', linestyle='dashed')
ax.yaxis.grid(color='gray', linestyle='dashed')
for i in range(len(m2)):
    lam_x = lambdify(mu, m2[i], modules=['numpy'])
    x_vals = np.linspace(-1, 1, 1000)
    y_vals = lam_x(x_vals)
    ax.plot(x_vals, y_vals, label = "n = " + str(dictionary["m2"]["n"][i
]))
    ax.legend()

plt.show()
```


Legendre function: m0
$P(\mu)$
μ
3
2
1
0
−1
−2
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
n = 1
n = 2
n = 3
n = 4


Legendre function: m1
$P(\mu)$
μ
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
n = 1
n = 2
n = 3
n = 4
n = 5



Question 4

Use Fourier methods similar to the class demo, so you can derive the Ordinary Differential Eq (ODE) for:

$$\partial T/\partial t + U_o \partial T/\partial x = K \partial^2 T/\partial x^2 \qquad \text{Uo and K are constants}$$

Answer Q4

$$\partial \mathrm{T}/\partial \mathrm{t} + \mathrm{U_o}\partial \mathrm{T}/\partial \mathrm{x} = \mathrm{K}\partial^2 \mathrm{T}/\partial \mathrm{x}^2 \qquad (1)$$

$$T(x,t) = \sum_m C_m(t) e^{imkx} \qquad (2)$$

Plug (2) into (1)

$$\sum_m \left[e^{imkx} \frac{\partial C_m}{\partial t} \right] + U_o \sum_m \left[C_m \frac{\partial e^{imkx}}{\partial x} \right] = K \sum_m \left[C_m \frac{\partial^2 e^{imkx}}{\partial x^2} \right]$$

$$\frac{\partial e^{imkx}}{\partial x} = imk(-e^{imkx}) \qquad \& \qquad \frac{\partial^2 e^{imkx}}{\partial x^2} = k^2 m^2 (-e^{imkx})$$

$$\sum_m \left[e^{imkx} \left(\frac{\partial C_m}{\partial t} + imkU_o C_m + k^2 m^2 K C_m \right) \right] = 0$$

This requires...

$$\frac{\partial C_m}{\partial t} + imkU_o C_m + k^2 m^2 K C_m = 0 \qquad \text{for each m}$$

$$\frac{\partial C_m}{C_m} = (-imkU_o - k^2 m^2 K)\partial t$$

$$\int_{C_m^(t=0)}^{C_m} \frac{\partial C_m'}{C_m'} = (-imkU_o - k^2 m^2 K) \int_0^t \partial t'$$

$$\ln(C_m')\Big|_{C_m(0)}^{C_m} = (-imkU_o - k^2 m^2 K)t$$

$$\ln\left(\frac{C_m}{C_m(0)} \right) = (-imkU_o - k^2 m^2 K)t$$

$$\frac{C_m}{C_m(0)} = e^{(-imkU_o - k^2 m^2 K)t}$$

or

$$\boxed{C_m(t) = C_m(t - \Delta t) e^{(-imkU_o - k^2 m^2 K)\Delta t}}$$