Lab_assignment_4

November 15, 2022

1 Lab Assignment 4

1.1 Pranith Praveen, s2097010

1.2 Task 1 (5 marks)

Give your implementation of the `plot_approx` and `approx_fourier` functions from Lab 4.

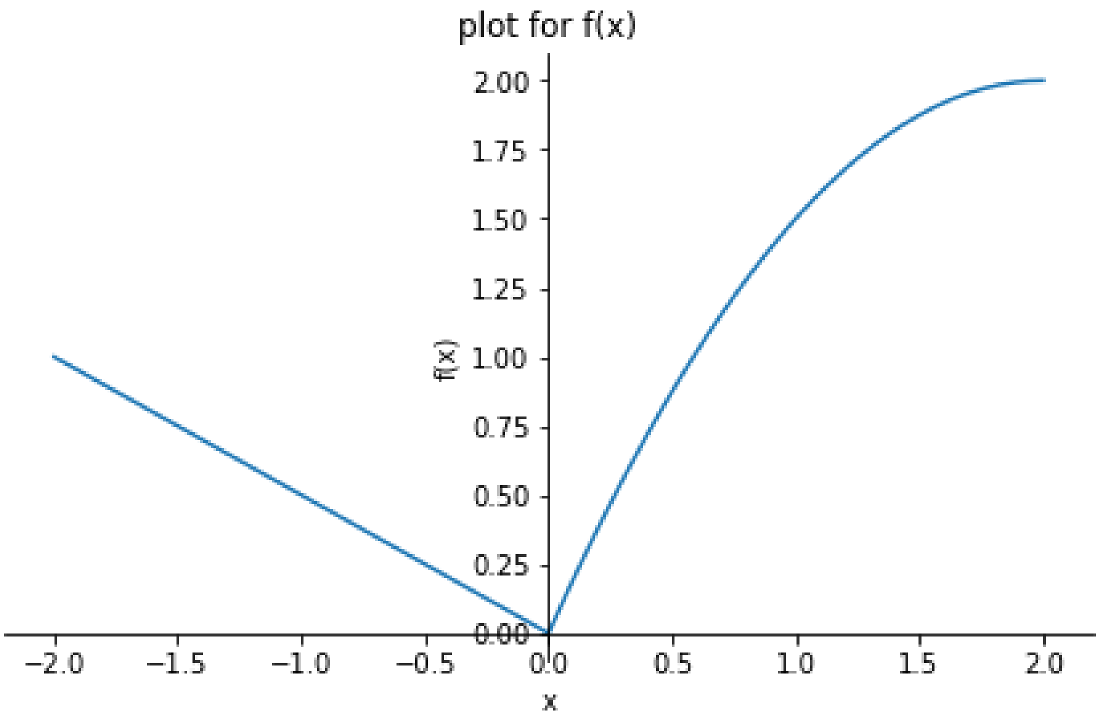
Use them to produce a plot of a Fourier series approximation of the function defined by

$$f(x) = \begin{cases} -\frac{1}{2}x & -2 \leq x < 0 \\ 2x - \frac{1}{2}x^2 & 0 \leq x < 2 \end{cases} \qquad f(x+4) = f(x)$$

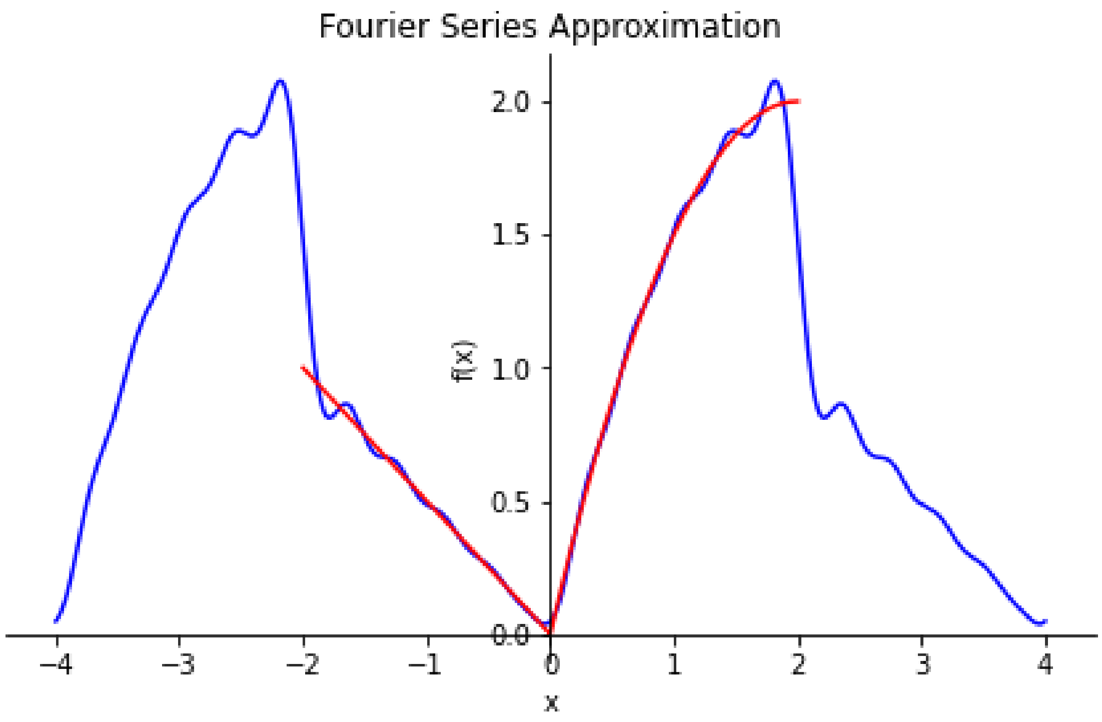
using the first 10 terms of the Fourier series.

Also include a piecewise plot of $f(x)$ for a single interval of periodicity.

[1]:
```
import sympy as sym
import sympy.plotting as sym_plot
sym.init_printing()
from IPython.display import display_latex

x,n = sym.symbols('x, n')


f = sym.Piecewise((2*x-x**2/2, x>=0), (-x/2, x<0))

sym_plot.plot(f, (x,-2,2), title = 'plot for f(x)')


def approx_fourier(f, L, num_terms):

    a0 = sym.Rational(1,L)*sym.integrate(f, (x, -L, L))
    an = sym.Rational(1,L)*sym.integrate(f*sym.cos(n*sym.pi*x/L), (x, -L, L))
    bn = sym.Rational(1,L)*sym.integrate(f*sym.sin(n*sym.pi*x/L), (x, -L, L))
    f_approx = a0/2 + sym.Sum(an*sym.cos(n*sym.pi*x/L)+bn*sym.sin(n*sym.pi*x/
↪L), (n,1,num_terms))
    return f_approx.doit()
def plot_approx(f, L, num_terms):
    f_approx = approx_fourier(f, L, num_terms)
    f_plot = sym_plot.plot((f_approx,(x,-2*L,2*L)), (f,(x,-L,L)), show  = False␣
↪, title = 'Fourier Series Approximation')
    f_plot[0].line_color = "blue"
    f_plot[1].line_color = "red"
    return f_plot
```



[2]:
```
approx_fourier(f, 2, 10)
plot_approx(f, 2, 10).show()
```



1.3 Task 2 (5 marks)

Solve Exercise 3.1 from Lab 4 , but with the initial condition

$$u(x,0) = f(x) = \begin{cases} 1 & L/2-1 < x < L/2+1 \\ 0 & \text{otherwise.} \end{cases}$$

Note that to ensure the code runs in reasonable time, you should use $L = 10$ and run the animation for $0 \leq t \leq 20$, with only 2 frames per second. You should use at least 200 terms of the series solution in order to obtain a good approximation. Describe the behaviour of the solution.

[17]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

L = 10
a = 1
m = (L/2)-1
p = (L/2) +1
f = sym.Piecewise((1, (m<x)), (1, (p>x)),(0,(x>p)),(0,(x<m)))


cn = sym.Rational(2,L)*sym.integrate(f*sym.sin(n*sym.pi*x/L), (x, 0, L))
```

```
t = sym.symbols('t')
u_symbolic = sym.Sum(cn.simplify()*sym.sin(n*sym.pi*x/L)*sym.cos(n*sym.pi*a*t/
 ↪L), (n,1,250))
#u_symbolic = u_symbolic.doit()

fps = 2 # number of frames per second

fig, ax = plt.subplots()

x_vals = np.linspace(0,L,200)

print(len(x_vals))

u = sym.lambdify([x, t], u_symbolic, modules='numpy')

# set up the initial frame
line, = ax.plot(x_vals, u(x_vals,0), 'k-')
plt.plot(x_vals,u(x_vals,0),'r:')
plt.xlabel('x')
plt.ylabel('u')
plt.ylim(-1.2,1.2)
plt.close()

# add an annotation showing the time (this will be updated in each frame)
txt = ax.text(0, 0.9, 't=0')

def init():
    line.set_ydata(u(x_vals,0))
    return line,

def animate(i):
    line.set_ydata(u(x_vals,i/fps))  # update the data
    txt.set_text('t='+str(i/fps)) # update the annotation
    return line, txt


ani = animation.FuncAnimation(fig, animate, np.arange(1, fps*20),␣
 ↪init_func=init,
                              interval=10, blit=True, repeat=False)


u_symbolic
```

```
200
```

[17]:

$$\sum_{n=1}^{250} \begin{cases} \frac{2\cdot(1-\cos(\pi n))\sin\left(\frac{\pi n x}{10}\right)\cos\left(\frac{\pi n t}{10}\right)}{\pi n} & \text{for } (n > -\infty \vee n > 0) \wedge (n > -\infty \vee n < \infty) \wedge (n > 0 \vee n < 0) \wedge (n < 0 \vee n < \\ 0 & \text{otherwise} \end{cases}$$

```
[18]: from IPython.display import HTML
      HTML(ani.to_jshtml())
```

```
[18]: <IPython.core.display.HTML object>
```

The solution is periodic for every 20 t and produces a square wave

1.4 Submission instructions

After producing the animation (as in the lab), you should also use the following line of code to produce an mp4 file of your animation. The file should then appear alongside the .ipynb file in your Jupyter file list. Note, if you are not using noteable, you will need to install ffmpeg. You can see this page for instructions https://ffmpeg.org/.

```
[19]: ani.save('hdeq_lab4_task2.mp4', writer='ffmpeg', fps=20)
```

Alternatively, you can use the code

```
[6]: ani.save('hdeq_lab4_task2.gif', writer='Pillow ', fps=20)
```

```
MovieWriter Pillow  unavailable; using Pillow instead.
```

To save the simulation as a gif. Upload four files on gradsecope the .pdf, the .ipynb, the .py and the .mp4 (or .gif).

```
[ ]:
```