**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **25/05/2020** | **Name:** | **Mamatha.m** |
| **Course:** | **Digital signal processing** | **USN:** | **4AL16EC035** |
| **Topic:** | **Introduction to Fourier Series & Fourier Transform** | **Semester & Section:** | **6 B** |
| **Github Repository:** | Mamatha84 |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  **Introduction to Fourier Series & Fourier Transform**  Fast forward two hundred years, and the fast Fourier transform has become the cornerstone of computational mathematics, enabling real-time image and audio compression, global communication networks, modern devices and hardware, numerical physics and engineering at scale, and advanced data analysis.  Simply put, the fast Fourier transform has had a more significant and profound role in shaping the modern world than any other algorithm to date. With increasingly complex problems, data sets, and computational geometries, simple Fourier sine and cosine bases have given way to tailored bases, such as the data-driven SVD. In fact, the SVD basis can be used as a direct analogue of the Fourier basis for solving PDEs with complex geometries. In addition, related functions, called wavelets, have been developed  for advanced signal processing and compression efforts.  **Fourier series**  A fundamental result in Fourier analysis is that if f(x) is periodic and piecewisesmooth, then it can be written in terms of a Fourier series, which is an infinitesum of cosines and sines of increasing frequency. In particular, if f(x) is 2\_-periodic, it may be written as:    Fourier series approximation to a hat function  % Define domain  dx = 0.001;  L = pi;  x = (-1+dx:dx:1)\*L;  n = length(x); n quart = floor(n/4);  % Define hat function  f = 0\*x;  f(nquart:2\*nquart) = 4\*(1:nquart+1)/n;  f(2\*nquart+1:3\*nquart) = 1-4\*(0:nquart-1)/n;  plot(x,f,’-k’,’LineWidth’,1.5), hold on  % Compute Fourier series  CC = jet(20);  A0 = sum(f.\*ones(size(x)))\*dx;  fFS = A0/2;  for k=1:20  A(k) = sum(f.\*cos(pi\*k\*x/L))\*dx; % Inner product  B(k) = sum(f.\*sin(pi\*k\*x/L))\*dx;  fFS = fFS + A(k)\*cos(k\*pi\*x/L) + B(k)\*sin(k\*pi\*x/L);  plot(x,fFS,’-’,’Color’,CC(k,:),’LineWidth’,1.2)  end  **Fourier series for a discontinuous hat function**  dx = 0.01; L = 10;  x = 0:dx:L;  n = length(x); nquart = floor(n/4);  f = zeros(size(x));  f(nquart:3\*nquart) = 1;  A0 = sum(f.\*ones(size(x)))\*dx\*2/L;  fFS = A0/2;  for k=1:100  Ak = sum(f.\*cos(2\*pi\*k\*x/L))\*dx\*2/L;  Bk = sum(f.\*sin(2\*pi\*k\*x/L))\*dx\*2/L;  fFS = fFS + Ak\*cos(2\*k\*pi\*x/L) + Bk\*sin(2\*k\*pi\*x/L);  end  plot(x,f,’k’,’LineWidth’,2), hold on  plot(x,fFS,’r-’,’LineWidth’,1.2)  Fourier series using python  # #### test that it works with real coefficients:  from numpy import linspace, allclose, cos, sin, ones\_like, exp, pi, \  complex64, zeros  def series\_real\_coeff(a0, a, b, t, T):  """calculates the Fourier series with period T at times t,  from the real coeff. a0,a,b"""  tmp = ones\_like(t) \* a0 / 2.  for k, (ak, bk) in enumerate(zip(a, b)):  tmp += ak \* cos(2 \* pi \* (k + 1) \* t / T) + bk \* sin(  2 \* pi \* (k + 1) \* t / T)  return tmp  t = linspace(0, T, 100)  f\_values = f(t)  a0, a, b = fourier\_series\_coeff\_numpy(f, T, 52)  # construct the series:  f\_series\_values = series\_real\_coeff(a0, a, b, t, T)  # check that the series and the original function match to numerical precision:  assert allclose(f\_series\_values, f\_values, atol=1e-6)  # #### test similarly that it works with complex coefficients:  def series\_complex\_coeff(c, t, T):  """calculates the Fourier series with period T at times t,  from the complex coeff. c"""  tmp = zeros((t.size), dtype=complex64)  for k, ck in enumerate(c):  # sum from 0 to +N  tmp += ck \* exp(2j \* pi \* k \* t / T)  # sum from -N to -1  if k != 0:  tmp += ck.conjugate() \* exp(-2j \* pi \* k \* t / T)  return tmp.real  f\_values = f(t)  c = fourier\_series\_coeff\_numpy(f, T, 7, return\_complex=True)  f\_series\_values = series\_complex\_coeff(c, t, T)  assert allclose(f\_series\_values, f\_values, atol=1e-6) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **25/5/2020** | **Name:** | **Mamatha.m** | |
| **Course:** | **Python** | **USN:** | **4al16ec035** | |
| **Topic:** | **Fixing Programming Errors**  **Application 3: Build a Website**  **Blocked** | **Semester & Section:** | **6 b** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **To fix programming errors:**   1. Read the error from the beginning. The first line tells you the location of the error. So, the error happened in script1.py(that was the name of my script), on line 1. Now you know where the error occurred. For your convenience you also have the line that caused the error printed out in the second line of the error message. 2. Next, look at the error type. In this case the error type is a SyntaxError. That means you have written something that doesn’t follow the Python syntax rules. So, now you have an idea of what error you are dealing with. For an overview of possible Python error types you can look [here](https://docs.python.org/3/library/exceptions.html#concrete-exceptions). 3. Look at the details of the error. On the right of SyntaxError you have the detailed information about the error. In this case this information is "invalid syntax"and you also have an arrow character pointing upward. That error is pointing to the colon character. The arrow is trying to say that the colon doesn’t belong there. 4. Time to use your logic. Now, the Python interpreter gave you all the information that a robot can give. Now it’s your turn as a human to use your logic to fix the error. So, Python executes a script from top to bottom, line by line, and reads each line from left to right. In this case it started to read the first line and it detected round brackets after the assignment operator. That means you are creating a tuple. That’s fine. But then after you write the first item (“Name” in this case) you were supposed to write a comma to separate that item from the next item, but you used a colon instead, so the interpreter is saying that a colon is not syntactically correct to use with round brackets. Therefore, you should make up your mind to either write a tuple like data = ("Name", "John", "Surname", "Smith")or a dictionary of key-value pairs like data = {"Name":"John", "Surname":"Smith".  The decision is up to you. In this case though I believe the programmer meant to write a dictionary, so I am going to replace the round brackets with curly brackets because I know a dictionary is defined through curly brackets.   **Working of host file in website blocker:**  # Run this script as root  import time  from datetime import datetime as dt  # change hosts path according to your OS  hosts\_path = "/etc/hosts"  # localhost's IP  redirect = "127.0.0.1"  # websites That you want to block  website\_list =  ["www.facebook.com","facebook.com",  "dub119.mail.live.com","www.dub119.mail.live.com",  "www.gmail.com","gmail.com"]  while True:  # time of your work  if dt(dt.now().year, dt.now().month, dt.now().day,8)  < dt.now() < dt(dt.now().year, dt.now().month, dt.now().day,16):  print("Working hours...")  with open(hosts\_path, 'r+') as file:  content = file.read()  for website in website\_list:  if website in content:  pass  else:  # mapping hostnames to your localhost IP address  file.write(redirect + " " + website + "\n")  else:  with open(hosts\_path, 'r+') as file:  content=file.readlines()  file.seek(0)  for line in content:  if not any(website in line for website in website\_list):  file.write(line)  # removing hostnmes from host file  file.truncate()  print("Fun hours...")  time.sleep(5) | | | |
|  | | | |