Построение Фурье-преобразования

Фурье-преобразование позволяет найти коэффициенты для периодического сигнала с периодом Т. Формула Фурье-преобразования имеет следующий вид

[clip_image002[12]](http://lh3.googleusercontent.com/-sMoNCKkLous/VTgGr8A0tBI/AAAAAAAAA58/67C2Usm5XGs/s1600-h/clip_image002%25255B12%25255D%25255B3%25255D.png)

Здесь f(t)- это реальный сигнал. Подлежат определению коэффициенты an, bn.

Расчетные соотношения таковы

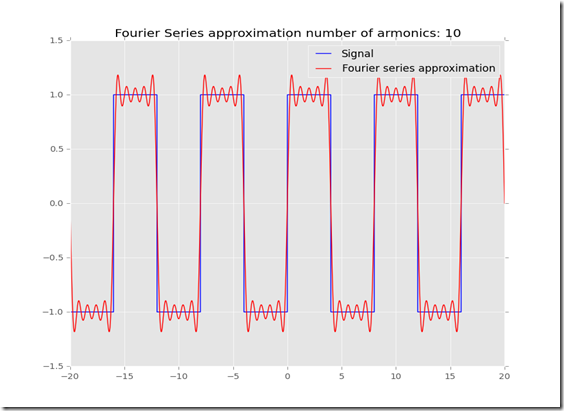
[clip_image002[10]](http://lh3.googleusercontent.com/-Ushn75YhbO0/VTgGszLVwKI/AAAAAAAAA6I/9Zwgzinc06k/s1600-h/clip_image002%25255B10%25255D%25255B3%25255D.png)

[clip_image002[16]](http://lh3.googleusercontent.com/-htRtKXSNo6c/VTgGuMST72I/AAAAAAAAA6Y/69B7D4M1VFU/s1600-h/clip_image002%25255B16%25255D%25255B3%25255D.png)

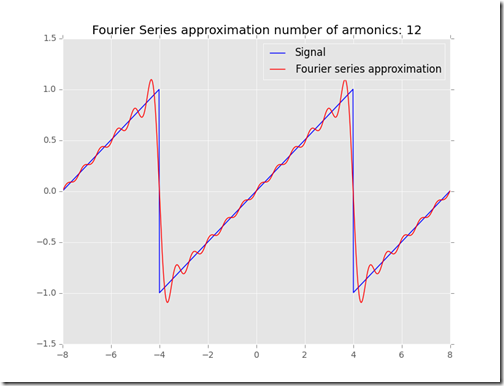
[clip_image002[14]](http://lh3.googleusercontent.com/-mPUlyoWGcTM/VTgGu7gCA3I/AAAAAAAAA6o/n2mlmqWqbRc/s1600-h/clip_image002%25255B14%25255D%25255B3%25255D.png)

[clip_image004](http://lh3.googleusercontent.com/-64oDM5M1AT4/VTgGwJUN3_I/AAAAAAAAA64/RDAh5SNxWco/s1600-h/clip_image004%25255B4%25255D.png)

Ниже показано, что даже первые (n=10) неплохо аппроксимируют импульс:

[](http://lh3.googleusercontent.com/-qWSLIDDib7M/VTgGxeTAzPI/AAAAAAAAA7M/bLguQgIcLcY/s1600-h/figure_1%25255B5%25255D.png)

Аналогично, пилообразный сигнал:

[](http://lh3.googleusercontent.com/-oog4sV2fC7E/VT6tfV9AU8I/AAAAAAAAA7k/UCaCeg-fv0s/s1600-h/figure_1%25255B4%25255D.png)

Все это можно продемонстрировать с помощью следующей программы

|  |  |
| --- | --- |
|  | import numpy as np |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | plt.style.use("ggplot") |
|  |  |
|  | # Setup |
|  | x\_ = np.linspace(-20,20,10000) |
|  |  |
|  | T = 8 |
|  | armonics = 10 |
|  |  |
|  | def squareWave(x): |
|  | global T |
|  | lowerBoundLeft = (-T/2) |
|  | lowerBoundRight = 0 |
|  | upperBoundLeft = 0 |
|  | upperBoundRight = (T/2) |
|  | one = 1 |
|  | negativeOne = -1 |
|  |  |
|  | while True: |
|  | if (x >= lowerBoundLeft) and (x <= lowerBoundRight): |
|  | return negativeOne |
|  | elif (x >= upperBoundLeft) and (x <= upperBoundRight): |
|  | return one |
|  | else: |
|  | lowerBoundLeft -= T/2 |
|  | lowerBoundRight -= T/2 |
|  | upperBoundLeft += T/2 |
|  | upperBoundRight += T/2 |
|  | if one == 1: |
|  | one = -1 |
|  | negativeOne = 1 |
|  | else: |
|  | one = 1 |
|  | negativeOne = -1 |
|  |  |
|  | # Bn coefficients |
|  | def bn(n): |
|  | n = int(n) |
|  | if (n%2 != 0): |
|  | return 4/(np.pi\*n) |
|  | else: |
|  | return 0 |
|  |  |
|  | # Wn |
|  | def wn(n): |
|  | global T |
|  | wn = (2\*np.pi\*n)/T |
|  | return wn |
|  |  |
|  | # Fourier Series function |
|  | def fourierSeries(n\_max,x): |
|  | a0 = 0 |
|  | partialSums = a0 |
|  | for n in range(1,n\_max): |
|  | try: |
|  | partialSums = partialSums + bn(n)\*np.sin(wn(n)\*x) |
|  | except: |
|  | print("pass") |
|  | pass |
|  | return partialSums |
|  |  |
|  |  |
|  | y = [] |
|  | f = [] |
|  | for i in x\_: |
|  | y.append(squareWave(i)) |
|  | f.append(fourierSeries(armonics,i)) |
|  |  |
|  |  |
|  | plt.plot(x\_,y,color="blue",label="Signal") |
|  | plt.plot(x\_,f,color="red",label="Fourier series approximation") |
|  | plt.title("Fourier Series approximation number of armonics: "+str(armonics)) |
|  | plt.legend() |
|  | plt.show() |

Собственно, вычисление Фурье-серии реализуется в двух функциях:

|  |  |
| --- | --- |
|  | def wn(n): |
|  | global T |
|  | wn = (2\*np.pi\*n)/T |
|  | return wn |
|  |  |
|  | # Fourier Series function |
|  | def fourierSeries(n\_max,x): |
|  | a0 = 0 |
|  | partialSums = a0 |
|  | for n in range(1,n\_max): |
|  | try: |
|  | partialSums = partialSums + bn(n)\*np.sin(wn(n)\*x) |
|  | except: |
|  | print("pass") |
|  | pass |
|  | return partialSums |

Найдите в них теоретические формулы.

Теперь о задании.

1. Постройте график функции Y=1/x и ее Фурье-аппроксимацию на отрезке [0,π]
2. Постройте график функции Y=x2 и ее аппроксимацию на отрезке [0,π]

Более интересна Фурье-аппроксимация дискретной функции. Вот краткая теория

|  |
| --- |
| *# Discrete Fourier Transform (DFT)*  *# FB - 20141227*  **import** random  **import** math  **import** cmath  pi2 **=** cmath**.**pi **\*** **2.0**  **def** DFT**(**fnList**):**  N **=** len**(**fnList**)**  FmList **=** **[]**  **for** m **in** range**(**N**):**  Fm **=** **0.0**  **for** n **in** range**(**N**):**  Fm **+=** fnList**[**n**]** **\*** cmath**.**exp**(-** **1**j **\*** pi2 **\*** m **\*** n **/** N**)**  FmList**.**append**(**Fm **/** N**)**  **return** FmList    **def** InverseDFT**(**FmList**):**  N **=** len**(**FmList**)**  fnList **=** **[]**  **for** n **in** range**(**N**):**  fn **=** **0.0**  **for** m **in** range**(**N**):**  fn **+=** FmList**[**m**]** **\*** cmath**.**exp**(1**j **\*** pi2 **\*** m **\*** n **/** N**)**  fnList**.**append**(**fn**)**  **return** fnList  *# TEST*  **print** "Input Sine Wave Signal:"  N **=** **360** *# degrees (Number of samples)*  a **=** float**(**random**.**randint**(1,** **100))**  f **=** float**(**random**.**randint**(1,** **100))**  p **=** float**(**random**.**randint**(0,** **360))**  **print** "frequency = " **+** str**(**f**)**  **print** "amplitude = " **+** str**(**a**)**  **print** "phase ang = " **+** str**(**p**)**  **print**  fnList **=** **[]**  **for** n **in** range**(**N**):**  t **=** float**(**n**)** **/** N **\*** pi2  fn **=** a **\*** math**.**sin**(**f **\*** t **+** p **/** **360** **\*** pi2**)**  fnList**.**append**(**fn**)**  **print** "DFT Calculation Results:"  FmList **=** DFT**(**fnList**)**  threshold **=** **0.001**  **for** **(**i**,** Fm**)** **in** enumerate**(**FmList**):**  **if** abs**(**Fm**)** **>** threshold**:**  **print** "frequency = " **+** str**(**i**)**  **print** "amplitude = " **+** str**(**abs**(**Fm**)** **\*** **2.0)**  p **=** int**(((**cmath**.**phase**(**Fm**)** **+** pi2 **+** pi2 **/** **4.0)** **%** pi2**)** **/** pi2 **\*** **360** **+** **0.5)**  **print** "phase ang = " **+** str**(**p**)**  **print**  *### Recreate input signal from DFT results and compare to input signal*  *##fnList2 = InverseDFT(FmList)*  *##for n in range(N):*  *## print fnList[n], fnList2[n].real* |

Здесь приведены реализации прямого и обратного Фурье-преобразования. При дискретном Фурье-преобразовании функция задается значениями в дискртеных точках. В нашем случае

fnList **=** **[]**

**for** n **in** range**(**N**):**

t **=** float**(**n**)** **/** N **\*** pi2

fn **=** a **\*** math**.**sin**(**f **\*** t **+** p **/** **360** **\*** pi2**)**

fnList**.**append**(**fn**)**

Функция задается точками в списке. Точки соответствуют синусоиде. Вычисление дискретных коэффициентов Фурье реализуется здесь:

**def** DFT**(**fnList**):**

N **=** len**(**fnList**)**

FmList **=** **[]**

**for** m **in** range**(**N**):**

Fm **=** **0.0**

**for** n **in** range**(**N**):**

Fm **+=** fnList**[**n**]** **\*** cmath**.**exp**(-** **1**j **\*** pi2 **\*** m **\*** n **/** N**)**

FmList**.**append**(**Fm **/** N**)**

**return** FmList

Восстановление значений функции по дискретным коэффициентам Фурье выполняется здесь

**def** InverseDFT**(**FmList**):**

N **=** len**(**FmList**)**

fnList **=** **[]**

**for** n **in** range**(**N**):**

fn **=** **0.0**

**for** m **in** range**(**N**):**

fn **+=** FmList**[**m**]** **\*** cmath**.**exp**(1**j **\*** pi2 **\*** m **\*** n **/** N**)**

fnList**.**append**(**fn**)**

**return** fnList

ЗАДАНИЕ.

1. ВЫПОЛНИТЕ ЭТОТ ПРИМЕР.
2. ОТОБРАЗИТЕ ИСХОДНУЮ ФУНКЦИЮ И ЕЕ ФУРЬЕ-АППРОКСИМАЦИЮ В ВИДЕ ГРАФИКОВ ПО ПОДОБИЮ ПЕРВОЙ ПРОГРАММЫ.
3. ТЕПЕРЬ попробуйте ввести точки, соответствующие букве G. Найдите ее Фурье-аппроксимацию. Покажите, насколько точно она воспроизводит исходную букву.