**РАЗРАБОТКА БИБЛИОТЕКИ МОДЕЛИРОВАНИЯ АНАЛИТИЧЕСКОГО СИГНАЛА**

***Русанов Максим***

Saint-Petersburg State University of Aerospace Instrumentation,

Saint-Petersburg, Russia

E-mail: hatrue.max@gmail.com

**Краткое описание**

В данной работе представлена программная реализация модели аналитического сигнала на языке Python 3.6.9. В статье описывается математическая модель исследуемого сигнала: строение аналитического сигнала; преобразование Гильберта и его основные свойства; описание параметров сигнала:

* Амплитуда
* Мощность
* Частота

**ВСТУПЛЕНИЕ**

Аналитический сигнал (комплексный сигнал) — используемое в теории обработки сигналов математическое представление аналогового сигнала в виде комплексной аналитической функции времени. Обычный, действительный сигнал x является при этом действительной частью аналитического представления *s(t).* Теория аналитических сигналов является развитым направлением в теории систем управления.

Как правило, для решения прикладных задач используются готовые модели исследуемого явления. В частности, если решение задачи выполняется на каком-либо языке программирования, то используются готовые библиотеки функций и методов. Это ускоряет разработку программного решения и позволяет сконцентрироваться на решении проблемы, минуя техническую реализацию предмета. Также повышается уровень абстракции, позволяющий подходить к решению более гибко.

Поэтому при подходе к задаче о моделировании аналитического сигнала был выбран такой способ. На основе уже построенных математических моделей был разработан интерфейс (библиотека функций) для моделирования сигнала и его параметров. В связи с часто встречаемой проблемой зашумления сигнала была также разработана функция сглаживания сигналов (параметров в том числе), реализованная методом «скользящего усреднения».

**МАТЕМАТИЧЕСКАЯ МОДЕЛЬ**

Формальное определение комплексного сигнала следующее:

*,* (1)

где *u(t) –* действительный сигнал, имеющий Фурье изображение; *v(t) –* сопряжённый по Гильберту сигнал.

При этом сопряженный сигнал связан с действительной частью с помощью преобразования Гильберта следующим соотношением:

. (2)

Перечислим важнейшие свойства преобразования Гильберта:

1. , где sgn(x) – знаковая функция. (3)

2. . (4)

Формула (3) даёт представление об амплитудно-частотной характеристике преобразования Гильберта, (4) формула показывает, как представляется обратное преобразование, то есть возврат к исходному сигналу.

Помимо перечисленных свойств, также немало важным является то, что Гильбертово преобразование выражается через преобразование Фурье. Этот факт играет главную роль в программной реализации модели аналитического сигнала.

Пусть *U(iω) = F[u(t)] –* изображение по Фурье действительного сигнала. Так как изображение по Фурье *s(t)* является:

. (5)

Тогда вычислить *v(t)* дополнение можно по формуле:

, (6)

где *Im(z) –* мнимая часть *z; F-1 –* обратное преобразование Фурье.

На основе комплексной модели сигнала также можно дать определения основных параметров сигнала, его амплитуды *a(t)* или мощности *с(t)*:

, (7)

и фазы:

, (8)

а также частоты:

, (9)

В данной реализации модели сигнала, присутствующий метод сглаживания работает по следующему принципу. Берётся некоторая область до и после некоторой точки и , учитывая численные значения измерений, входящих в эту область, вычисляем среднее значение. Формально это можно описать следующим образом.

Пусть имеется выборка из *N* точек {*f1, f2, …, fN*}. Тогда для нахождения среднего в окрестности выбранной точки *i* берём среднее арифметическое от M предыдущих и последующих точек, включая саму точку *i*. Тогда новые значения точек, пусть *gi*, будут вычисляться по формуле:

(10)

**ОПИСАНИЕ ПРОГРАММЫ**

При разработке модели комплексного сигнала на языке Python 3.6.9 были реализованы следующие функции, охватывающие построение сигнала сопряжённого по Гильберту, построение аналитического сигнала, вычисление основных параметров сигнала, а также сглаживание шумов.

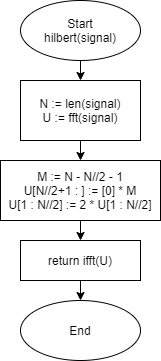
Работа ведётся следующим образом: файл библиотеки подключается в заголовке программы; пишется код с использованием ниже описанных функций; ниже пишется код, отображающий полученные данные; запускается через командную строку ОС; пользователь получает изображение (график) моделируемого сигнала. Интерфейс библиотеки состоит из следующих функций:

* *hilbert(signal) –* функция принимает массив значений сигнала, возвращает массив, представляющий значения преобразования Гильберта.
* *оrt\_signal(signal)* – функция принимает как параметр массив значений действительного сигнала и возвращает массив, представляющий значения сопряжённого сигнала.
* *analitic\_signal(signal) –* принимает массив значений сигнала, возвращает массив комплексных значений аналитического сигнала.
* *amp(signal)* – принимает массив значений аналитического сигнала, возвращает амплитуду аналитического сигнала.
* *cap(signal)* – принимает массив значений аналитического сигнала, возвращает мощность аналитического сигнала.
* *frq(signal, dx)* – принимает массив значений аналитического сигнала(signal) и шаг дискретизации сигнала(dx), возвращает частоту аналитического сигнала.
* *smoothing(signal, M)* – сглаживает входной сигнал методом *скользящего усреднения* принимает массив значений действительного сигнала(signal) и степень сглаживания (M), возвращает сглаженный сигнал.

Алгоритмическое описание:

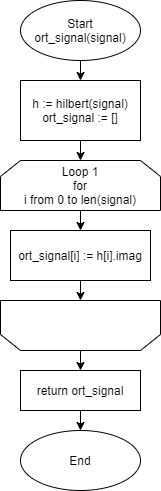
В данной части работы приведены блок-схемы выше описанных функций с комментариями, за исключением функций *amp(signal)* и *cap(signal)*, реализация которых является тривиальной:

1. *hilbert()*:



*Рис 1. Блок-схема функции преобразования Гильберта*

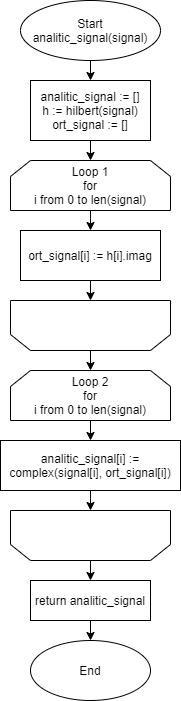
Блок *Start* указывает на начало программы и содержит название функции, а также принимаемые параметры. Далее в описании последующих функций комментирование ведётся начиная со второго блока. Второй блок: N – переменная, содержащая длину (*len()*) выборки значений сигнала; U – прямое преобразование Фурье (*fft()*) входного сигнала. Третий блок: М – центр выборки. Следующие две строки вычисляют изображение Фурье аналитического сигнала, где в соответствии с теорией при отрицательных частотах ω изображение равно нулю, а при положительных равно удвоенным значением изображением действительного сигнала. Четвёртый блок: возвращает обратное преобразование Фурье комплекснозначного сигнала. Конец алгоритма.

 2. *ort\_signal()*:

*Рис 2. Блок-схема функции ортоганального сигнала*

Второй блок: *h –* массив значений преобразованного по Гильберту сигнала; *ort\_signal –* пустой массив для хранения значений сопряженного сигнала. Третий блок: цикл переменной *i* проходящей от 0 до конца массива, в теле цикла массиву *ort\_signal* присваивается мнимая часть массива *h.* Четвёртый блок: функция возвращает массив *ort\_signal.* Конец алгоритма.

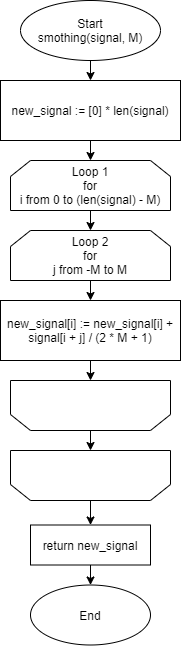
2. *analitic\_signal()*:



*Рис 3. Блок-схема функции аналитического сигнала*

Второй блок: *analitic\_signal* – массив для хранения итогового значения; *h* и *ort\_signal* несут тот же смысл, что и в предыдущем алгоритме. Третий блок: *Loop 1 –* выполняет те же функции, что и в предыдущем алгоритме. Четвёртый блок: *Loop 2 –* цикл переменной *i* проходящей от 0 до конца массива, в теле цикла массиву *analitic\_signal* присваиваются комплексные значения, где действительная часть – входной сигнал, а мнимая – сопряжённый сигнал. Пятый блок: функция возвращает массив *analitic\_signal.* Конец алгоритма.

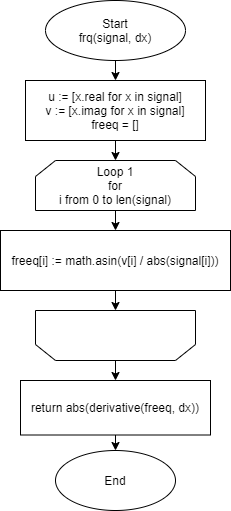
3. *smoothing()*:



*Рис 4. Блок-схема функции, реализующей сглаживание сигнала*

Второй блок: *new\_signal* – массив для хранения итогового значения. Третий блок: *Loop 1-* цикл переменной *i,* проходящей от 0 до конца массива, за исключением последних *M* точек. *Loop 1* содержит вложенный цикл *Loop 2,* который вычисляет новые усреднённые значения сигнала методом «скользящего усреднения». Четвёртый блок: функция возвращает массив *new\_signal.* Конец алгоритма.

5. *frq()*:



*Рис 5. Блок-схема функции частоты аналитического сигнала*

Второй блок: *u* – массив действительных значений аналитического сигнала*; v –* массив мнимой части аналитического сигнала*; freeq –* переменная для хранения итоговых значений частоты. Третий блок: *Loop 1 –* цикл переменной *i* проходящей от 0 до конца выборки сигнала. В теле цикла вычисляются значения фазы сигнала по формуле (8) с арксинусом. Четвёртый блок: функция возвращает производную фазы по времени, то есть значение частоты. Конец алгоритма.

Полный код программы:

**import** matplotlib**.**pyplot **as** plt

**import** numpy **as** np

**import** math **as** m

**from** scipy**.**fftpack **import** **\***

**def** hilbert\_1**(**signal**):**

N **=** **len(**signal**)**

U **=** fft**(**signal**)**

M **=** N **-** N**//**2 **–** 1

U**[**N**//**2**+**1**:]** **=** **[**0**]** **\*** M

U**[**1**:**N**//**2**]** **=** 2 **\*** U**[**1**:**N**//**2**]**

v **=** ifft**(**U**)**

**return** v

**def** derivative**(**f**,** dx**):**

df **=** **[]**

df**.**append**((**f**[**1**]** **-** f**[**0**])** **/** dx**)**

**for** i **in** **range(**1**,** **len(**f**)):**

df**.**append**((**f**[**i**]** **-** f**[**i **-** 1**])** **/** dx**)**

**return** df

**def** ort\_signal**(**signal**):**

h **=** hilbert\_1**(**signal**)**

ort\_signal **=** **[]**

**for** i **in** **range(**0**,** **len(**signal**)):**

ort\_signal**.**append**(**h**[**i**].**imag**)**

**return** ort\_signal

**def** analitic\_signal**(**signal**):**

analitic\_signal **=** **[]**

h **=** hilbert\_1**(**signal**)**

ort\_signal **=** **[]**

**for** i **in** **range(**0**,** **len(**signal**)):**

ort\_signal**.**append**(**h**[**i**].**imag**)**

**for** i **in** **range(**0**,** **len(**signal**)):**

analitic\_signal**.**append**(complex(**signal**[**i**],** ort\_signal**[**i**]))**

**return** analitic\_signal

**def** smoothing**(**signal**,** K**):**

new\_signal **=** **[**0**]** **\*** **len(**signal**)**

**for** i **in** **range(**K**,** **len(**signal**)** **-** K**):**

**for** j **in** **range(-**K**,** K**):**

new\_signal**[**i**]** **+=** signal**[**i **+** j**]** **/** **(**2**\***K **+** 1**)**

**return** new\_signal

**def** amp**(**signal**):**

**return** np**.abs(**signal**)**

**def** cap**(**signal**):**

**return** np**.abs(**signal**)** **\*** np**.abs(**signal**)**

**def** frq**(**signal**,** dx**):**

u **=** **[**x**.**real **for** x **in** signal**]**

v **=** **[**x**.**imag **for** x **in** signal**]**

freeq **=** **[]**

**for** i **in** **range(**0**,** **len(**signal**)):**

freeq**.**append**(**m**.**asin**(**v**[**i**]** **/** np**.abs(**signal**[**i**])))**

**return** np**.abs(**derivative**(**freeq**,** dx**))**

**ПРИМЕР ПОСТРОЕНИЯ СИГНАЛА**

В качестве примера работы программы был написан следующий код, моделирующий сопряженный сигнал для функции *sin(x)*. В итоге имеем следующий код:

**from** signal\_model **import** **\***

d **=** 10

discr **=** 0.0008

l **=** **int(**d **/** discr**)**

axis **=** **[(**x **\*** discr**)** **for** x **in** **range(**0**,** l**)]**

signal **=** **[**m**.**sin**(**x**)** **for** x **in** axis**]**

ort\_signal **=** ort\_signal**(**signal**)**

analitic\_signal **=** analitic\_signal**(**signal**)**

plt**.**axis**([**0**,** d**,** **-**2.5**,** 2.5**])**

plt**.**plot**(**axis**,** signal**)**

plt**.**plot**(**axis**,** ort\_signal**,** color **=** 'red'**)**

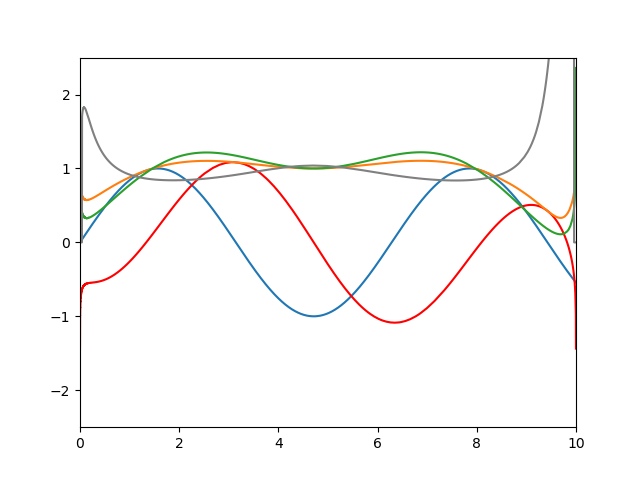
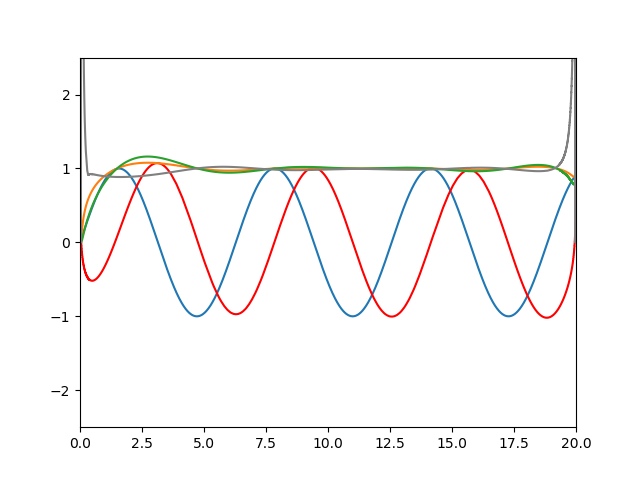
plt**.**plot**(**axis**,** amp**(**analitic\_signal**))**

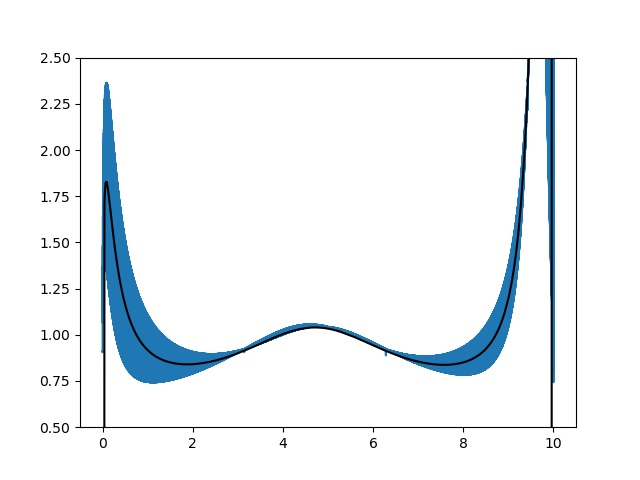
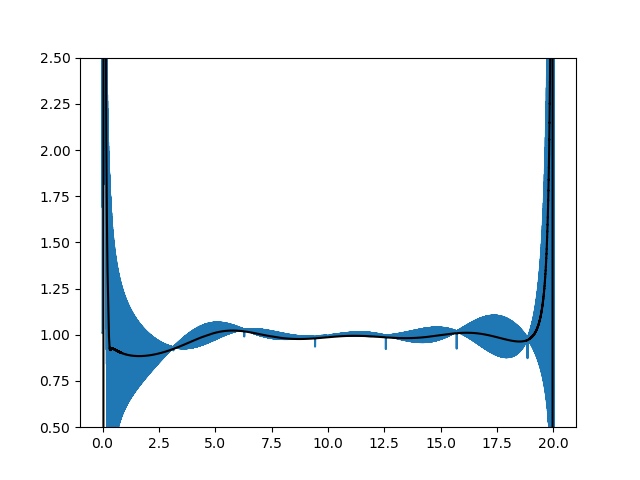
plt**.**plot**(**axis**,** cap**(**analitic\_signal**))**

plt**.**plot**(**axis**,** smoothing**(** frq**(**analitic\_signal**,** discr**),** 50**),** color **=** 'grey'**)**

plt**.**show**()**

Результат работы программы можно видеть на следующих рисунках:

*Рис 6 – 7. Результат работы программы. Синий – входной действительный сигнал; Красный – сопряженный сигнал; Оранжевый – амплитуда; Зелёный – мощность; Серый – частота.*

 В частности, как было ранее сказано, иногда при зашумлении сигнала становится необходимостью его сглаживание, что продемонстрировано ниже:

*Рис 8 – 9. Пример работы функции сглаживания. Синий – зашумлённый сигнал; Чёрный – сглаженный.*

**ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА**

[1] <https://ru.dsplib.org/content/hilbert/hilbert.html>

[2] Фалеев С. П. Расчет и моделирование устройств обработки сигналов систем управления. Учебное пособие. / Ленинградский орден Ленина электротехнический институт имени В. И. Ульянова (Ленина)

[3] Сато Юкио. Цифровая обработка сигналов. / Юкио Сато: пер. с яп. Селиной Т. Г. М. : Додэка-XXI, 2010. – 176 с. : ил. – Доп. Тит. л. яп. – ISBN 978-5-94120-251-5.