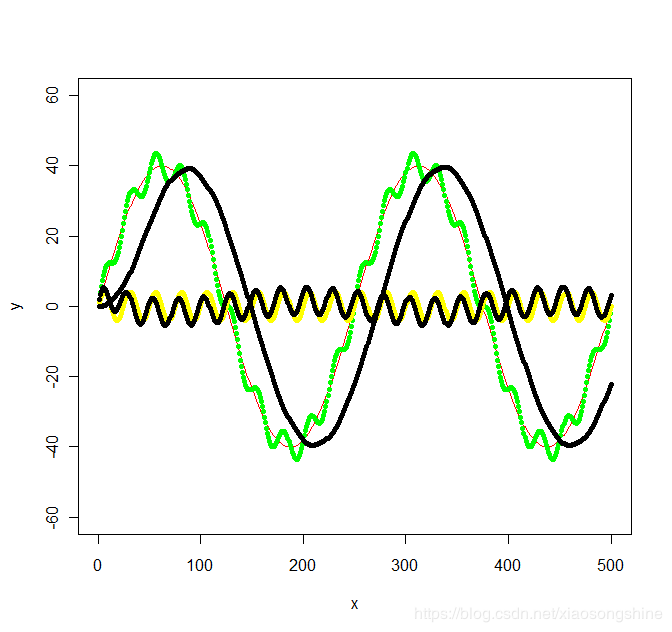
**Фильтрация данных**

**1. Введение**

 В задачах обработки и отображения данных Python часто используется для фильтрации. Рассмотрим, как возможно использовать Python для реализации простой обработки данных на основе модуля **scipy** в реальных задачах: **1. Фильтр нижних частот, 2. Фильтр верхних частот, 3. Полосовой фильтр, 4. Полосовой ограничительный фильтр.** Фильтрация нижних частот реализует удаление сигналов выше определенной пороговой частоты; фильтрация верхних частот для удаления сигналов ниже определенной частоты; полосовая фильтрация относится к комбинации аналогичных низких и высоких частот; для сохранения сигналов вне указанной полосы и подавление сигналов в этой полосе.

Это позволяет устранить некоторые мешающие сигналы. Возьмем для примера фильтрацию нижних частот. Отобразим на экране график функции некоторого сигнала с частотой 1 Гц. Но обнаружим, что в сигнале формы много шумов. Эти шумы составляют сотни или тысячи Гц. Это бесполезные шумы для наших целей. Мы можем отфильтровать сигнал, удаляя те частоты (отсчеты), которые превышают определенный порог, используя фильтр нижних частот. Форма графика сигнала будет более гладкой.



**2. Фактические боевые учения**

Используем модуль scipy, который можно установить с помощью следующей команды: pip install scipy

**1) Фильтрация нижних частот**

Предположим здесь, что частота дискретизации (количество отсчетов/значений в единицу времени) составляет 1000 Гц, а максимальная частота самого сигнала составляет 500 Гц. Требуется отфильтровать частотные компоненты выше 400 Гц, то есть частота среза составляет 400 Гц. wn = 2 \* 400/1000 = 0,8. Wn = 0,8

from scipy import signal

b, a = signal.butter(8, 0.8, 'lowpass') # Значение 8 указывает порядок фильтра

filtedData = signal.filtfilt(b, a, data) # данные - сигнал, который нужно отфильтровать

**2) Фильтр высоких частот**

Предположим, что частота дискретизации сигнала составляет 1000 Гц, а максимальная частота самого сигнала составляет 500 Гц. Требуется отфильтровать частотные компоненты ниже 100 Гц, то есть частота среза равна 100 Гц. wn = 2 \* 100/1000 = 0,2. Wn = 0,2

from scipy import signal

b, a = signal.butter(8, 0.2, 'highpass') # Конфигурационный фильтр 8 указывает порядок фильтра

filtedData = signal.filtfilt(b, a, data) # данные - это сигнал, который нужно отфильтровать

**3) Полосовой фильтр**

Предположим здесь, что частота дискретизации составляет 1000 Гц, а максимальная частота самого сигнала составляет 500 Гц. Требуется отфильтровать частотные компоненты ниже 100 Гц и выше 400 Гц, то есть частота среза составляет 100, 400 Гц, тогда wn1 = 2 \* 100 /1000=0,2, Wn1 = 0,2; wn2 = 2 \* 400/1000 = 0,8, Wn2 = 0,8. Wn = [0,02,0,8]

from scipy import signal

b, a = signal.butter(8, [0.2,0.8], 'bandpass') # Конфигурационный фильтр 8 указывает порядок фильтра

filtedData = signal.filtfilt(b, a, data) # данные - это сигнал, который нужно отфильтровать

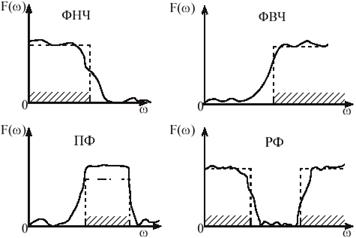
**4) Полосовой режекторный фильтр**

Предположим, что частота дискретизации составляет 1000 Гц, а максимальная частота самого сигнала составляет 500 Гц. Требуется отфильтровать частотные компоненты выше 100 Гц и ниже 400 Гц, то есть частота среза составляет 100, 400 Гц, тогда wn1 = 2 \* 100 /1000=0,2, Wn1 = 0,2; wn2 = 2 \* 400/1000 = 0,8, Wn2 = 0,8. Wn = [0,02,0,8]. Этот фильтр работает аналогично полосовому фильтру с той разницей, что частоты в полосе пропускания удаляются, а вне полосы пропускания - удаляются.

from scipy import signal

b, a = signal.butter(8, [0.2,0.8], 'bandstop') # Число Параметр 8 указывает порядок фильтра

filtedData = signal.filtfilt(b, a, data) # данные - это сигнал, который нужно отфильтровать



**3. Описание функцию фильтрации**

scipy.signal.filtfilt(b, a, x, axis=-1, padtype='odd', padlen=None, method='pad', irlen=None)

Входные параметры:

b: числитель вектора коэффициентов фильтра

a: знаменатель вектора коэффициентов фильтра

x: массив данных для фильтрации.

axis: ось массива данных x для фильтрации.

padlen: количество элементов, продолжающих X на обоих концах оси до применения фильтра. Это значение должно быть меньше количества фильтруемых элементов -1. (Тип Int или None)

method: метод обработки фронта сигнала. Когда используется метод pad, сигнал заполняется; padtype и padlen определяются типом заполнения, а irlen игнорируется. Когда method - "gust", используется метод Gustafson, а padtype и padlen игнорируются. {"Pad", "gust"}

irlen: Если для метода задано значение «gust», irlen указывает длину импульсной характеристики фильтра. Если irlen равно None, то любая часть импульсной характеристики игнорируется. Для длинных сигналов указание irlen может значительно улучшить производительность фильтра. (Тип Int или None)

Выходные параметры:

y: массив отфильтрованных данных

(2) Конструктор фильтра (только фильтр Баттерворта).

scipy.signal.butter(N, Wn, btype='low', analog=False, output='ba')

Входные параметры:

N: порядок фильтра

Wn: нормализованная частота среза. Формула расчета Wn = 2 \* частота среза / частота дискретизации. (Примечание: согласно теореме дискретизации, частота дискретизации должна быть более чем в два раза больше максимальной частоты самого сигнала, чтобы восстановить сигнал. Частота среза должна быть меньше максимальной частоты самого сигнала, поэтому Wn должно быть от 0 до 1). При построении полосового фильтра или полосового фильтра Wn представляет собой список длиной 2.

btype: тип фильтра {«lowpass», «highpass», «bandpass», «bandstop»},

вывод: тип вывода {‘ba’, ‘zpk’, ‘sos’},

Выходные параметры:

b, a: числитель (b) и знаменатель (a) вектора полиномиальных коэффициентов БИХ-фильтра. output = 'ba'

z, p, k: нули, полюса и системное усиление передаточной функции БИХ-фильтра. output = 'zpk'

sos: представление БИХ-фильтра в разрезе второго порядка. output = 'sos'

**Задание на практическую работу**

1. Сгенерировать два вектора значений (f1 и f2), показывающий изменение синусоидального сигнала.

Параметры сигналовследующие: длительность – 2 сек, количество значений в 1 сек (частота дискретизации) – 1000,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вариант** | **Частота для f1, Гц** | **Частота для f1** | **Амплитуда для f1** | **Амплитуда для f2** |
| 1 | 10 | 47 | 1 | 2 |
| 2 | 60 | 26 | 0,5 | 2,5 |
| 3 | 50 | 26 | 2 | 1,5 |
| 4 | 47 | 30 | 2,5 | 2,5 |
| 5 | 26 | 47 | 1,5 | 3 |
| 6 | 26 | 10 | 2,5 | 1,5 |
| 7 | 30 | 60 | 3 | 2,5 |
| 8 | 44 | 50 | 1,5 | 1,5 |
| 9 | 50 | 47 | 2 | 2,5 |
| 10 | 15 | 10 | 1 | 3 |

1. Отобразить одновременно графики f1 и f2.
2. Наложить на данные значения шум:
   1. Случайно сгенерированные значения (амплитуда шума в 5 раз меньше наименьшей амплитуды из двух сигналов).
   2. Низкочастотный синусоидальный сигнал (частота в 10 раз меньше наименьшей частоты из двух сигналов).
3. Отобразить на графике полученные зашумленные векторы значений.
4. Применить к зашумленных сигналам все виды фильтром. Параметры фильтрации постараться выбрать самостоятельно с целью максимального подавления шумов.
5. Отобразить на графике одновременной исходный вектор f1, зашумлённый и отфильтрованный.
6. Отобразить на графике одновременной исходный вектор f2, зашумлённый и отфильтрованный.
7. Сложить два сигнала f1 и f2, вывести график. Применяя последовательно разные виды фильтров постараться максимально подавить сигнал f2 для наилучшего отображения сигнала f1.

**Примечание (уравнение синусоидального сигнала)**

U = A sin2πƒt,

где А - амплитуда сигнала, ƒ - частота (в герцах). Синусоидальный сигнал показан на рис. 1.17. Иногда бывает полезно переместить начало координат (t = 0) в точку, соответствующую произвольному моменту времени; в этом случае в выражение для синусоидального напряжения следует включить фазу

U = A sin(2πƒt +*ϴ*).