МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

І. В. Кравченко, В. І. Микитенко

ЦИФРОВА ОБРОБКА СИГНАЛІВ ТА ЗОБРАЖЕНЬ
ЧАСТИНА 1

Практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для здобувачів ступеня магістра
за освітньою програмою «Комп'ютерно-інтегровані системи та технології в приладобудуванні» спеціальності 174 (G7) «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2025

УДК 004.9 (621.398)
К77

Автори *Кравченко Ігор Володимирович*, ст. викладач кафедри комп'ютерно-інтегрованих оптичних та навігаційних систем КПІ ім. Ігоря Сікорського
Микитенко Володимир Іванович, док. техн. наук, професор, професор кафедри комп'ютерно-інтегрованих оптичних та навігаційних систем КПІ ім. Ігоря Сікорського

Рецензент *Киричук Ю.В.,* док. техн. наук, доцент, завідувач кафедри автоматизації та систем неруйнівного контролю КПІ ім. Ігоря Сікорського

Відповідальний редактор *Чиж І.Г.,* док. техн. наук, професор

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 7 від 08.05.2025 р.)
за поданням Вченої ради приладобудівного факультету
(протокол № 4/25 від 17.04.2025 р.)

К77 Кравченко, І. В. Цифрова обробка сигналів та зображень. Частина 1. [Електронний ресурс] : практикум : навч. посіб. для здобувачів ступеня магістра за освіт. програмою «Комп'ютерно-інтегровані системи та технології в приладобудуванні» спец. 174 (G7) Автоматизація, комп'ютерно-інтегровані технології та робототехніка / І. В. Кравченко, В. І. Микитенко. – Електрон. текст. дані (1 файл). – Київ : КПІ ім. Ігоря Сікорського, 2025. – 76 с.

Розглянуто технологію, методики, особливості застосування бібліотек NumPy, SciPy мови Python для обробки цифрових сигналів. Містить теоретичний матеріал, приклади та комплекс завдань для практикуму та самостійної роботи з дисципліни «Цифрова обробка сигналів та зображень».

Призначений для здобувачів ступеня магістра освітньої програми «Комп'ютерно-інтегровані системи та технології в приладобудуванні» спеціальності 174 (G7) «Автоматизація, комп'ютерно-інтегровані технології та робототехніка». Буде також корисним для студентів, науковців, інженерних працівників, які цікавляться комп'ютерною обробкою сигналів.

УДК 004.9 (621.398)

Реєстр. № НП 24/25-478. Обсяг 4.2 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

ЗМІСТ

ПОЗНАЧЕННЯ ТА СКОРОЧЕННЯ

ПЗ	- програмне забезпечення
NumPy	- Numerical Python
SciPy	- Scientific Python
АЦП	- аналогово-цифровий перетворювач
МФП	- матричний фотоприймач
FFT	- Fast Fourier Transform
ФЧС	- фільтр частотної селекції
ФНЧ	- фільтр низьких частот
ФВЧ	- фільтр високих частот
СФ	- смуговий фільтр
РФ	- режекторний фільтр
АКФ	- автокореляційна функція
ВКФ	- функція взаємної кореляції

ПЕРЕДМОВА

Навчальний посібник призначений для забезпечення інформаційними та методичними матеріалами практичних занять з дисципліни "Цифрова обробка сигналів та зображень".

Детально розглянуті можливості, особливості застосування, технології роботи з цифровими сигналами засобами мови програмування Python. Матеріал забезпечений практичними прикладами та комплексом завдань для самостійної роботи.

В практикумі 1 навчального посібника розглядаються питання застосування бібліотеки **NumPy** для роботи з одно та двовимірними масивами при обробці сигналів та зображень.

В практикумі 2 описуються засоби бібліотеки **MatplotLib** як основи для виведення інформації про сигнали в графічному вигляді.

Засобам опису імпульсних, періодичних та ймовірнісних сигналів засобами **NumPy, SciPy** присвячений практикум 3.

Практикум 4 присвячено засобам та особливостям проведення дискретного Фур'є перетворення та частотної фільтрації сигналів.

В практикумі 5 описуються засоби ПЗ для проведення вейвлет перетворень дискретних сигналів та аналізу скейлограм.

Практикум 6 містить відомості про розрахунки авто та кроскореляційих функцій імпульсних та неперервних сигналів, проведення кореляційного аналізу імовірнісних сигналів.

Практикум 7 містить відомості про різновиди цифрової згортки сигналів, зв'язок згортки з кореляцією та дискретним Фур'є перетворенням, лінійну фільтрацію сигналів.

Для проведення практичних занять необхідним є комп'ютерний клас або наявність у студентів комп'ютерів відповідним ПЗ.

Мета навчального посібника – допомога студентам в самостійному вивченні відповідних розділів навчальної дисципліни, в інформаційному та методичному забезпеченні практикуму, в набутті навичок застосування інформаційних технологій у вигляді сучасної Script мови в навчальній та науково-технічній практиці. Посібник може використовуватись для самостійної роботи та дистанційного навчання.

ПРАКТИКУМ 1. ПОЧАТОК РОБОТИ З МАСИВАМИ

Завдання роботи: вивчення можливостей ПЗ та набуття навичок в створенні та відображенні масивів даних, базових операціях з масивами.

Теоретичні положення

В мові *Python* для обробки структурованих табличних даних типу вектора, матриці передбачені вбудовані типи: список (list) та масив (array з бібліотеки **array**).

Різниця між ними полягає в тому, що масиви є одновимірними, елементами масивів можуть бути тільки числові значення всі цілі (тип 'i') або всі дійсні (тип 'd'). Списки можуть мати довільну кількість вимірів, елементами списку можуть бути дані будь-якого типу.

Перевагами списків вважається здатність зберігати дані різних типів упереміж, необмежена кількість елементів.

Основні застосування масивів включають роботу з великими обсягами даних, створення і використання матриць, виконання сортування і пошуку, а також реалізацію різних алгоритмів і структур даних. Вони компактніші, займають менше пам'яті.

Всі елементи списків, масивів пронумеровані. Нумерація починається з нуля. Індекси записуються в квадратних дужках, як в мові С. Від’ємні індекси використовуються для доступу до елементів в зворотному порядку від кінця списку. Останній елемент має індекс [-1], передостанній [-2].

Задаються значення списку одночасно прирівнюванням через квадратні дужки з розділенням комою, або поелементно в циклі.

Дії з масивами та списками виконуються вбудованими функціями, операторам та методами відповідних об’єктів.

Вбудовані функції обробки списків

Розмір списку визначається вбудованою функцією **len(name).** Для багатовимірних списків повертається один перший розмір – для двовимірних (матриць) – кількість рядків

Функція **sorted(name)** повертає відсортований список. За замовчанням – за зростанням. Визначення опційного ключа **reverse=True** сортує список за зменшенням.

Функції **min(name), max(name)** повертають найменший/найбільший елемент одновимірного списку або *рядок* двовимірного списку:

Злиття кількох структур проводиться оператором «+»:

Явне витягнення значень всіх елементів структури в окремі змінні проводиться простим прирівнюванням.

Методи списків

Метод	**Опис**
a.append(x)	Додає один елемент **x** в кінець списку **a**. Якщо **x** - список, то він додається як вкладений
a.extend(b)	Додає в кінець списку **a** всі елементи списку **b**
a.insert(i, x)	Вставляє елемент **x** на позицію з індексом **i** в список **a**
a.remove(x)	Видаляє зі списку **a** перший зліва елемент, який має значення **x.** Для матриць x: рядок-список. В разі відсутності елемента видає помилку.
a.clear()	Видаляє всі елементи зі списку **a,** робить його порожнім.
a.index(x)	Повертає індекс першого зліва елементу зі значенням **x.** В разі відсутності елемента видає помилку.
a.pop(i)	Видаляє елемент з індексом **i** зі списку **a** та повертає його значення. Без аргументу видаляє останній елемент.
a.count(x)	Рахує кількість елементів зі значенням **x.**
a.sort()	Сортує список a. За замовчанням – за збільшенням. Визначення опційного ключа **reverse=True** сортує список за зменшенням.
a.reverse()	Повертає список з оберненим порядком елементів
a.copy()	Повертає копію списку.

Програмне створення списків

Одновимірні списки

Створення порожнього списку - ініціація одновимірного списку;

Заповнення значеннями порожнього списку має проводиться методом доповнення елементів **append, extend, insert**.

Ефективніше створити список з типовим значенням, а потім змінити потрібні елементи.

Багатовимірні масиви-списки

Багатовимірні масиви-списки програмно задаються в циклах по всіх індексах: створення одновимірного списку та явний цикл по стовпцях, складний цикл-генератор по рядках, вкладені цикли-генератори.

Відображення двовимірних списків

Для зрозумілого виведення двовимірних списків слід застосувати поелементне форматоване виведення з двома циклами. Один цикл по рядках, другий – по стовпцях.

Спосіб виведення значень елементів рядка в циклі в одному рядку та переведення рядка. Для цього у внутрішньому циклі по стовпцях перелік даних оператора **print** завершується перевіркою на завершення дії з ознакою завершення end, коли всі елементи рядка вичерпано. Переведення реалізується в зовнішньому циклі оператором **print** без параметрів. Цикли можуть бути організовані явно по кількості рядків/стовпців:.

Кількість повторень в циклі можна визначати вбудованою функцією **len**:

Математичні дії з даними списків та стандартних масивів можливі тільки поелементно за відповідним номером-індексом в квадратних дужках після імені списку.

Бібліотека NumPy

Для спрощення записів, прискорення поелементних дій над масивами довільних розмірностей з однорідними даними призначена бібліотека **Numerical Python – NumPy**. Масиви бібліотеки мають власний тип numpy.ndarray, який не містить коду типу даних, як стандартні масиви array('i', [1, 2, 3]) .

Бібліотека є зовнішньою, до програми її треба підключати явно імпортуванням.

Створення масивів

Створення масиву явно проводиться зі списку за допомогою функції створення масиву **array**. Тип отриманого масиву визначається автоматично, відповідно до типу елементів у списках. За замовчанням цілі значення мають тип **int64**, дійсні – **float64**.

Примусове визначення типу елементів масиву проводиться методом **dtype**

Дані для елементів одновимірних масивів записуються як для стандартних масивів-списків через списки в квадратних дужках через кому. Для матриць рядки записуються в круглих/квадратних дужках як елементи списку забрані в квадратні дужки через кому:

Примусово визначити тип масиву можна також опціональним аргументом **dtype** функції **array.**

Типи даних NumPy

Тип	Діапазон
np.int8	-/+128
np.int16	-/+32_768
np.int32 = int	-/+2_147_483_648
np,int64= int	-/+9_223_372_036_854_775_808
np.uint8	0 .. 255
uint16	0 .. 65_535
np.uint32	0.. 4_294_967_295
np.uint64	0 .. 18_446_744_073_709_551_615
np.float16	16 біт: 10 знаків мантиса, 5 – порядок
np.float32	32 біт: 23 знаків мантиса, 8 – порядок
np.float64 = float	64біт: 52 знаків мантиса, 11 – порядок
np.complex64	2X32 bit
np.complex128 = complex	2X64 bit

Виведення розмірності масиву a проводиться методом **a.ndim**. Виведення розмірів масиву a проводиться методом **a.shape**

Методи для створення масивів з початковим вмістом

Створення одновимірних масивів

Для створення одновимірних масивів з регулярними елементами призначені методи `linspace()`, `arange ()`, `logspace()`, `geomspace()`.

Метод `linspace()` в якості обов'язкових аргументів має початок діапазону та кінець діапазону. Кількість потрібних елементів `num=`, тип елементів `dtype=`, виведення останньої точки `endpoint=` задаються опціонально. За замовчанням `num=50`, `dtype=float`, `endpoint=True` (враховувати останню точку), Створює лінійну прогресію.

Метод `arange ()` в якості обов'язкових аргументів має початок діапазону та кінець діапазону. Крок елементів `k=` та тип елементів `dtype=` задаються опціонально. За замовчанням крок 1, тип елементів int32. Створює лінійну прогресію.

Метод `logspace()` в якості обов'язкових аргументів має початок діапазону та кінець діапазону. Кількість потрібних елементів `num=`, тип елементів `dtype=`, виведення останньої точки `endpoint=`, основа логарифму `base=` задаються опціонально. За замовчанням `num=50`, `dtype=float`, `endpoint=True` (враховувати останню точку), `base=10.0`. Створює логарифмічну послідовність.

Метод `geomspace()` є подібним до `logspace()`. Створює геометричну прогресію. В якості обов'язкових аргументів має початок діапазону та кінець діапазону. Кількість потрібних елементів `num=`, тип елементів `dtype=`, виведення останньої точки `endpoint=` задаються опціонально. За замовчанням `num=50`, `dtype=float`, `endpoint=True` (враховувати останню точку).

Універсальні методи створення одно/двовимірних масивів

Метод `zeros()` створює масив, заповнений нулями, `ones()` створює масив, заповнений одиницями, `empty()` створює масив, початковий вміст якого є випадковим, `full()` створює масив з визначеним значенням елементів у вигляді сталого або списку. За замовчуванням типом створеного масиву є **float64.**

Першим аргументом всіх методів є розміри масиву. Розміри задаються через кому в дужках: рядки, стовпці тощо. Одновимірний можна задавати одним числом стовпців. Опціональний аргумент `dtype=***` явно визначає тип даних.

Значення елементів для методу `full((row,col), value <, dtype=>)` визначається після розмірів через кому.

Метод `eye(row <,col, k=, dtype=>)` створює діагональну матрицю, з нулями поза діагоналлю та одиницями на діагоналі. Опційний параметр к (за замовчанням 0) – значення на головній діагоналі, додатне – над головною діагоналлю, від'ємне – під.

Метод `diag(matr<,k=>)` виділяє з матриці-першого аргументу одновимірний масив значень діагоналі. Опційний аргумент к= визначає тип діагоналі: 0 (за замовчанням) – головна, додатне – над головною, від'ємне – під головно. З аргументом –одновимірним масивом будує квадратну діагональну матрицю з значеннями масиву по головній діагоналі.

Доступ до елементів масивів. Індексування

Доступ до значень конкретних елементів масивів проводиться через їхні індекси. Індекси починаються з нуля. Від'ємні значення означають рахування з кінця рядка/стовпця. Останній елемент має індекс -1. Індекси можуть задаватися як і для списків в окремих квадратних дужках в стилі мови С **a[1][1]**, так і в одних дужках через кому **a[1,1]** (tuple) в стилі систем комп'ютерної математики. Рекомендованим є стиль .

Визначити кілька елементів одночасно можна за допомогою діапазону індексів *slicing*. За зразком ранжування в циклах. Діапазон індексів задається трійкою значень відокремлених двокрапкою start: stop: step. Крок за замовчанням одиниця, елемент з індексом stop не обирається. Двокрапка на місці індексу означає вибір всіх елементів. Діапазони індексів для матриць працюють тільки по одній координаті.

Базові операції з масивами

Арифметичні оператори та математичні функції на масивах виконуються **поелементно.** Результатом є масив такого ж розміру, як масиви-операнди.

Арифметичні оператори мають аналоги у вигляді відповідних методів.

`+ - add(), - - subtract(), ** - power(), * - multiply(), / - divide().`

Тригонометричні методи з аргументом/результатом в радіанах

`sin(x), cos(x), tan(x)`	Синус, косинус, тангенс.
`arcsin(x), asin(x), arccos(x), acos(x), arctan(x), atan(x), atan2(x1, x2), arctan2(x1, x2)`	Арксинус, арккосинус, арктангенс
`hypot(x1, x2)`	Розрахунок гіпотенузи по катетах
`degrees(x), radians(x), deg2rad(x), rad2deg(x)`	Переведення градуси-радіани

Гіперболічні методи

`sinh(x), cosh(x), tanh(x)`	Синус, косинус, тангенс.
`arcsinh(x), asinh(x), arccosh(x), acosh(x), arctanh(x), atanh(x)`	Арксинус, арккосинус, арктангенс

Методи наближення значень

`round(a<,decimals), around(a<[, decimals)`	Математичне округлення до опційного знаку **decimals** (за замовчанням 0)
`rint(x)`	Округлення до цілого
`fix(x)`	Відкидання мантиси
`floor(x)`	Округлення в меншу сторону
`ceil(x)`	Округлення в більшу сторону
`trunc(x)`	Округлення до 0

Логарифмічні та експоненційні методи

`exp(x)`	експонента
`expm1(x)`	exp(x)-1
`exp2(x)`	2**x, power(2,x)
`log(x)`	натуральний логарифм
`log10(x)`	десятковий логарифм
`log2(x)`	двійковий логарифм

Обчислювальні методи

`sqrt(x), cbrt(x)`	корінь квадратний, кубічний
`sinc(x)`	sin(x)/x
`abs(x), absolute(x), fabs(x)`	модуль
`sign(x)`	-1, x<0 0, x=0, 1, x>0
`gcd(x1, x2)`	найбільший спільний дільник
`lcm(x1, x2)`	найменше спільне кратне
`reciprocal(x)`	обернене значення
`negative(x)`	обернення знаку
`fmod(x1, x2)`	залишок від ділення зі знаком дільника
`remainder(x1,x2), mod(x1,x2)`	абсолютне значення залишку від ділення
`modf(x)`	списки (список) з цілими (цілою) та дробовими (дробовою) частинами
`divmod(x1, x2)`	списки (список) з цілими (цілою) та залишками (залишком) від ділення

Комплексні методи

`angle(z <[, deg>)`	Фаза комплексного числа. Одиниці задаються опційним параметром deg= (за замовчанням False)
`real(z)`	виділення дійсної частини
`imag(z)`	виділення уявної частини
`conj(z), conjugate(z)`	комплексно спряжене значення
`real_if_close(a<, tol>)`	переведення в дійсне, якщо уявна частина < tol*2.22e-16 (за замовчанням 100)

Методи обробки елементів масивів

`prod(a <[, axis, dtype>]` `sum(a <[, axis, dtype>]`	добуток/сума елементів. Результат визначається опційним параметром dtype. Без опційного параметру axis дія проводиться по всіх елементах (за замовчанням). axis=0 – по рядках, axis=1 – по стовпцях
`nanprod(a <,axis, dtype>), nansum(a <, axis, dtype>)`	дії з ігноруванням нечислових NaN значень

`max(a <, axis>),`
`amax(a <, axis>),`
`min(a <, axis>),`
`amin(a <, axis>),`
	Максимальний/мінімальний елемент. Без опційного параметру axis дія проводиться по всіх елементах (за замовчанням). axis=0 – по рядках, axis=1 – по стовпцях

Методи обробки форми масивів

`a.T`	транспонування матриці
`np.rot90(a <,k>)`	поворот матриці a на 90^0 ліворуч k разів (за замовчанням 1)
`np.reshape(a, (rows,cols))` `a. reshape((rows,cols))`	зміна розмірів масиву a на (rows,cols). Добуток rows*cols в масиві-джерелі та масиві-результаті повинні співпадати
`np.resize(a, (rows,cols))` `a. resize((rows,cols))`	зміна розмірів масиву a на (rows,cols). Добуток rows*cols в масиві-джерелі та масиві-результаті можуть не співпадати. В такому випадку елементи повторюються.
`np.trim_zeros(a1<,trim='**'>)`	Прибирає нулі в одновимірному масиві. Місце прибирання визначає опційний параметр trim: ‘f’ – спереду, ‘b’ – ззаду, ‘fb’ – з обох боків (за замовчанням)
`np.ravel(a), a.ravel()`	Перетворює масив на одновимірний
`np.delete(a, Nobj<, axis=>)`	Без axis видаляє з масиву a елементи з індексами з аргументу-списку Nobj як з одновимірного. З axis видаляє вказані рядки/стовпці

`np.append(a, values<, axis>)`	Додає елементи в кінець масиву. Без axis масив перетворюється на одновимірний. З axis =0 додаються рядки.
`np.flip(a <, axis=>)`	Переставляє значення елементів по визначеній осі. За замовчанням без вказання - axis=None, по всіх осях.
`fliplr(a) – горизонтально, flipud(a) – верткально`	
`np.insert(a, obj, val <, axis=None>)`	Вставляє в матрицю a значення val між індексами obj.
`np.tile(a, reps)`	Розширює матрицю a копіюванням елементів reps разів. reps – число (розширення по стовпця) або кортеж (reps_rows, reps_colm).

Завдання до виконання

Для варіантів 1-11 розмір масиву M= N+10, де N – номер варіанту. Для варіантів більше 12 розмір масиву M= N.

В усіх завданнях:

- Провести дії в циклі поелементно стандартними засобами **Python** та з усім масивом одночасно вбудованими функціями **NumPy**. Вивести результати.
- Визначити в скільки разів відрізняються часові витрати способів. Врахувати роздільну здатність таймеру.

Завдання 1. Створити матрицю NxN з одиницями по головній та допоміжній діагоналі.

Завдання 2. Створити одиничну матрицю NxN. Розширити її до подвійного розміру нулями таким чином, щоб одиничний фрагмент знаходився по центру.

Завдання 3. Створити матрицю NxN з градієнтними значеннями від 0 до N:

а) горизонтальний  ; б) інверсний горизонтальний 

в) вертикальний  ; г) інверсний вертикальний  .

Завдання 4. Створити матрицю NxN зі зростаючим послідовно значенням елементів. Розрахувати:

а) суму всіх елементів, мінімальне значення; б) добуток всіх елементів, мінімальне значення; в) поелементний корінь квадратний, транспоновану матрицю г) множення матриці на скаляр: номер варіанту.

Завдання 5. Додатково до матриці п.4 створити матрицю з подвійними значеннями елементів матриці завдання 4. Розрахувати:

а) поелементну суму матриць; б) поелементний добуток матриць; в) поелементу різницю матиць г) поелементну частку матриць

ПРАКТИКУМ 2. ГРАФІЧНЕ ВІДОБРАЖЕННЯ СИГНАЛІВ

Завдання роботи: вивчення можливостей ПЗ та набуття навичок в побудові двомірних графіків та гістограм, створенні 3D поверхонь.

Теоретичні положення

Бібліотека **matplotlib** – ієрархічна об’єктна бібліотека виведення графіків. Створена за подобою графічної системи СКМ MATLAB.

Основні функції керування двовимірними графіками зосереджені в модулі **matplotlib.pyplot,** тривимірними – **в mpl.toolkits.mplot3d.**

Модуль **matplotlib**.**image** містить базові функції завантаження, масштабування, виведення та збереження зображень.

Бібліотека є зовнішньою, до програми її треба підключати явно імпортуванням цілком або помодульно.

Для того, щоб графіка відображалася без прямої директиви **show()** слід спробувати перевести бібліотеку в режим **inline**.

За замовчанням дії функцій бібліотеки супроводжуються виведенням протоколів – логів з додаткової інформації. Для зменшення інформації можна знизити рівень логування до рівня помилки 'Error' властивості **_log** об’єкту **_axes**.

Бібліотека обробляє дані у вигляді масивів та списків.

Графіки виводяться у автономне вікно (рис. 2.1) **Figure** – об’єкт контейнер верхнього рівня, батьківський об’єкт графіки.

Примітка. Справедливо для Python, в Notebook графіки виводяться безпосередньо в документ.

Кожне вікно містить поля графіки – канви **Axes**, назви вікна **Title**, тексту вікна **Text**.

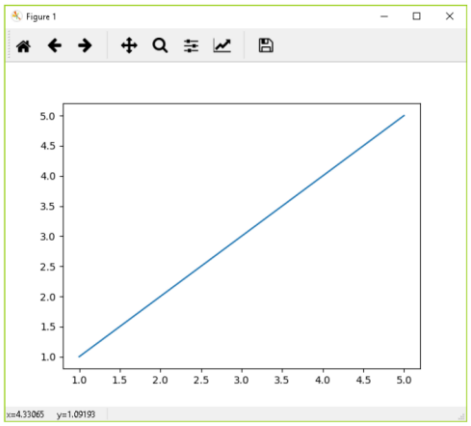


Рис. 2.1. Вікно **Figure**

Двовимірні графіки PyPlot

Канва **Axes** є контейнером другого рівня, нащадком **Figure** та батьківським об’єктом для інших графічних об’єктів: (графіків **plot** , осей **XAxis**, **YAxis,** міток осей **Xtick**, імен осей **Xlabel**, **Ylabel,** написів **Text**, сітки **Grid** тощо). Об’єкти канви **Axes** та підвікна **Subplot** можна вважати синонімами (рис. 2.2).

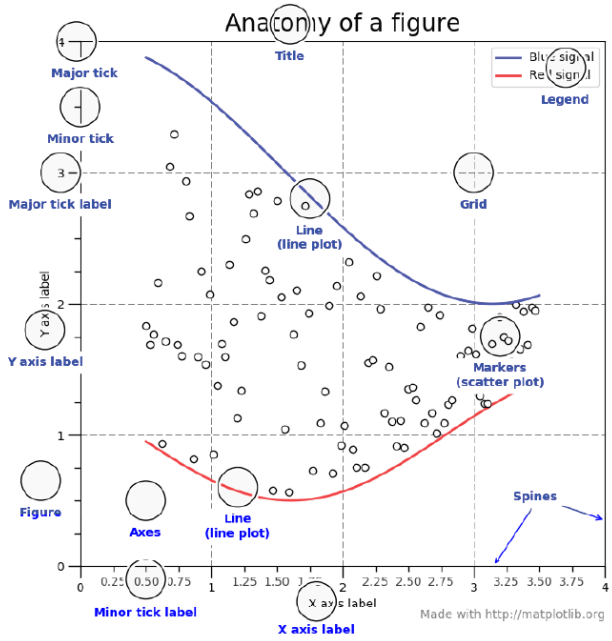


Рис. 2.2. Графічні об’єкти **Pyplot** (з відкритих джерел)

Типовий алгоритм виведення графіків засобами **Pyplot** складається з наступних кроків:

1. Створення вікна figure
2. Створення поля (ів) графіка(ів) axes
3. Підготовка масивів даних та виведення графіка (ів) на поле (я)
4. Оформлення поля та графіка
5. Виведення на екран

Виконання дій можливо суто об'єктними засобами з використанням методів та властивостей або спрощено з використанням функцій **Pyplot.**

Вікно Figure

Створення вікна не є обов'язковим. Воно створюється автоматично. Ручні дії потребуються для явного задання розмірів, розташування, назви вікна, тексту у вікні тощо.

В *Notebook* керування вікном не є повноцінним, бо канва виводиться безпосередньо в документ.

Методи (функції) **Pyplot** для **Figure:** `plt.figure()` – створює нове вікно, `plt.figtext()` – виводить текст у вікно, `plt.show()` – виводить вікно на екран, `plt.close()` – закриває вікно, `plt.Savefig()` – зберігає зображення у файл стандартного формату.

`<fighandle=>plt.figure(<arg>)` – метод створення вікна, де `fighandle` – опційний результат, ім'я змінної для ідентифікації вікна. Опційні аргументи: `num=` номер або назва вікна як рядок. За замовчанням `None`. `figsize=` двоелементний кортеж розміру вікна. Одиниця вимірювання 6.4 мм X 4.8 мм, за замовчанням `None`, `facecolor='Colorname'` рядок назви кольору фону, за замовчанням `None`, `linewidth=` товщина ліній для малювання графіків, за замовчанням 0.0.

`plt.show(<arg>]), plt.close(<arg>)` – методи виведення, закриття вікна. Опційний аргумент `arg=` – номер `num`, ім'я вікна `fighandle` , `'All'`.

`<texthandle=>plt.figtext(x, y, txt <, prop>)` – метод виведення тексту у вікно. Аргументи: `x, y` – дійсні числа координат для тексту у відносних координатах вікна, властивості шрифту `fontname='serif'/'sans-serif'/'cursive'/'fantasy'/'monospace', fontsize=float or 'xx-small'/'x-small'/'small'/ 'medium'/'large'/'x-large'/'xx-large'}`.

plt.savefig(fname, <args>) – метод збереження зображення у файл. Аргументи: fname – рядок імені файлу. Опційні аргументи: quality= ціле число якості зображення від 1 до 1000, optimize=True/False – додаткове покращення зображення для формату JPEG, progressive= True/False – прогресивне кодування для формату JPEG, facecolor='Color' рядок з назвою кольору для фону вікна, format='jpg/jpeg/bmp/png/pdf/svg' – рядок з стандартним розширенням файлу зображення.

Методи Figure

*.add_axes() – створює у вікні канву, *.add_subplot() – створює у вікні підобласть для канви, *.set() – задає властивості графічних об'єктів у вікні.

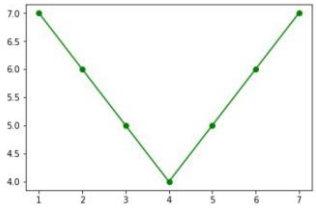
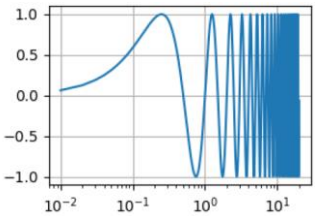
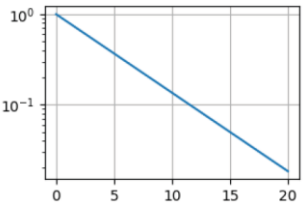
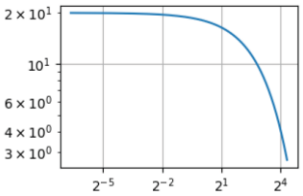
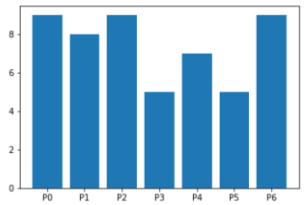
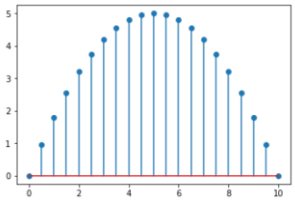
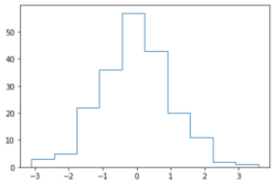
<axeshandle=>*.add_axes(<arg>), де axeshandle – опційний результат, ім'я змінної для ідентифікації канви. Аргументи-властивості канви: [left, bottom, width, height] – масив з 4-х дійсних чисел відносних розмірів канви від 0 до 1. Без визначення – канва на все вікно: polar=True/False – полярні графіки; projection=None/ 'aitoff'/ 'hammer'/ 'lambert'/ 'mollweide'/ 'polar'/ 'rectilinear' – тип осей для графіків. За замовчанням 'rectilinear'; sharex=, sharey=axeshandle – наслідування властивостей канви з вже визначених з axeshandle ; aspect='auto'/ 'equal' або число типу float – пропорції сторін; autoscale_on=, autoscalex_on=, autoscaley_on=True/False – автомасштабування графіків в канві; facecolor='Colorname' – фоновий колір; title='Title' – рядок назви над канвою; xlabel=, ylabel='Label' – назви осей; xlim=, ylim= – двоелементний масив діапазонів значень. Задання скидає aspect; xscale=, yscale= – масштаби графіків; xticklabels=, yticklabels= – масив рядків для позначок значень на осях; xticks=, yticks= – масив, послідовність координат маркерів на осях.

Канва для малювання Axes

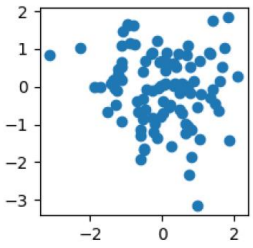
Метод (функція) **Pyplot** для створення канви з визначеними властивостями <axeshandle=>plt.axes(<arg>). Синтаксис повторює метод fig.add_axes().

<texthandle=>ax.text(x, y, 'text', <font prop>) – виведення тексту визначене місце на канві. Використання аналогічне figtext. Координати є координатами по осях канви.

Базові двовимірної графіки канви

`<handle=>ax.plot()`	Графік в декартових осях	
`<handle=>ax.semilogX()`	Логарифмічний по X графік в декартових осях	
`<handle=>ax.semilogY()`	Логарифмічний по Y графік в декартових осях	
`<handle=>ax.loglog()`	Логарифмічний графік в декартових осях	
`<handle=>ax.bar()`	Вертикальна стовпова діаграма	
`<handle=>ax.stem()`	Лінійчатий графік	
`<handle=>ax.stair()`	Графік сходинок	

<handle=>ax.scatter() Точкова діаграма



Метод <lineshandle=>ax.plot(<x>, y, <fmt, x2, y2, fmt2,...,prop) – малювання лінії(й) з визначеними параметрами. Відсутність x означає автомасштабування. Базові параметри: тип лінії, колір, маркер, – можуть визначатися в полі fmt. Будь-які властивості лінії визначаються парами «властивість=значення» як поля prop.

Формат лінії – трисимвольний рядок 'КМЛ', де перший символ К – колір, другий М – тип маркеру, Л – візерунок лінії.

Keyword	Description
-.	dash-dot line
–	dashed line
:	dotted line
-	solid line

Keyword	Description
o	circle
x	cross
D	diamond
h	hexagon
p	pentagon
+	plus
.	dot
s	square
*	star
V	down triangle
<	left triangle
>	right triangle
^	up triangle

Keyword	Description
k	black
b	blue
c	cyan
g	green
m	magenta
r	red
w	white
y	yellow

Властивості лінії: scalex=,scaley=True/False – автомасшабування графіків, color='Colorname' – рядок назви кольору; drawstyle='default'/'steps'/ 'steps-pre'/'steps-mid'/'steps-post' – тип з'єднання точок лінійно або сходинкою; label='Label' – рядок з визначенням назви лінії, яка може бути виведена додатково; linestyle= рядок з назвою або кодом як у візерунку лінії; linewidth= – товщина лінії; marker= рядок символом маркеру; markeredgecolor='Colorname' – колір контуру маркеру; markeredgewidth= товщина контуру маркеру; markerfacecolor='Colorname' – колір маркеру; markersize= – розмір маркеру.

Метод <barhandle=>ax.bar(x, height <,prop>) – малювання стовпової вертикальної діаграми з стовпцями з координатами x та висотою height. Властивості: width= - ширина стовпця в одиницях графіка. За

замовчанням 0.8; bottom= – нижнє значення стовпців. За замовчанням 0; align='center'/'edge' – горизонтальне вирівнювання стовпців. За замовчанням 'center'; color=; edgecolor=; linewidth= – товщина контуру стовпців; log=True/False – логарифмічний масштаб. За замовчанням False: facecolor= - колір стовпця; fill=True/False – заповнення стовпця кольором. За замовчанням True.

Метод <linehandle=>ax.stairs (values, edges <, param>) – малює графік у вигляді сходинок. Аргументами є масив значень values та масив координат меж сходинок edges розміром +1 від розміру масиву значень. Додатковими властивостями відносно plot є: orientation='vertical'/ 'horizontal' – орієнтація графіка; baseline= – положення осі, від якої буде вестись зафарбовування. За замовчанням 0; fill=True/False – зафарбовування області під графіком. За замовчанням False.

Метод <linehandle=>ax.stem() – малювання лінійного графіка. Застосування є подібним до ax.plot. Додатковими властивостями є linefmt='КМЛ' – колір та тип ліній. На маркер не впливає. Крім формату з ax.plot може задаватися як 'СNЛ', де N – цифра номеру кольору, markerfmt='СNЛ' – колір та тип маркеру, bottom= – ордината базової лінії. За замовчанням 0, orientation='vertical', 'horizontal'. За замовчанням 'vertical'.

Методи <linehandle=>ax.loglog(),ax.semilogx(), ax.semilogy() – малювання лінії(й) в логарифмічному форматі. Використання є аналогічним ax.plot. Додатковим параметром є base= – основа логарифму. За замовчанням 10.

Метод <gridhandle=>ax.grid(<arg>) – малювання сітки. Без аргументів вмикає/вимикає сітку. Опційні аргументи: which='minor/major'/'both' – визначає основну або дрібну сітку, linestyle= – визначає стиль ліній за правилами ax.plot. За замовчанням безперервний, color='Colorname' – колір ліній сітки, linewidth= – товщина ліній сітки. За замовчанням 0.5.

Метод <legendhandle=>ax.legend(<args>) – виведення легенди всіх існуючих на канві графіків.

Виклик без аргументів `legend ()` виводить легенди, які були призначені властивістю `legend` графіків при створені.

Виклик `legend(labels)`, де `labels` – масив рядків легенд, виводить заданий масив.

Опційні властивості: `loc='best'(0)/ 'upper left'(1)/ 'upper right'(2)/ 'lower left'(3)/'lower right'(4)/'upper center'(9)/ 'lower center'(8)/ 'center left'(6)/ 'center right'(7)/ 'center'(10)` - рядок назви або код – місце виведення легенди; `fontsize=` – розмір шрифту; `labelcolor='Colorname'` – колір шрифту; `facecolor='Colorname'` – колір фону; `edgecolor='Colorname'` – колір рамки; `title='TEXT'` – заголовок; `title_fontsizeint=` – розмір шрифту заголовку; `alignment='center'/ 'left'/ 'right'` – вирівнювання заголовку.

Визначати властивості об'єктів не обов'язково саме при їхньому створенні. Значення може бути задане або змінене методом відповідного об'єкту:

```
objecthandle.set_propertyname(value)
```

Робота з кількома полями графіків

За потреби у вікні можна створити декілька канв **Axes**. Для цього бібліотека містить кілька методів.

Метод вікна `axeshandle=fighandle.add_subplot(pos)` створює канву `axeshandle` на місті `pos`. Виклик без аргументів створює одну канву аналогічно методу `fighandle.add_axes()`. Для кількох канв позиція задається трьома числами відокремленими комами, наприклад, (2,2,1). Перша цифра – кількість рядків. Друга – кількість стовбців. Третя – позиція на якій буде побудовано графік (відлік ведеться послідовно з 1). Алгоритм дій полягає в наступному. Створюється вікно. Для першої канви викликається метод з номером канви 1. Записуються дії для канви 1. Потім для другої канви викликається метод з номером канви 2. Записуються дії для канви 2. Продовжується послідовно для потрібної кількості канв.

Метод `fighandle, axeshandles=plt.subplots(<args>)` створює одночасно і вікно і декілька канв. Канви розташовуються прямокутною мозаїкою по рядках та стовпцях. Виклик без аргументів створює

вікно та одну канву. Опційні аргументи: `nrows=` – кількість рядків; `ncols=` – кількість стовпців; `sharex=, sharey='none'/'all'/'row'/'col'` – визначає розповсюдження заданих в функції властивостей на канви. За замовчанням `'none'; squeeze=True/False` – керує типом запису результатів; `width_ratios=, height_ratios=` - масив відносних розмірів стовпців та рядків. Відмінність від методу полягає в тому, імена-посилання всіх канв створюються одночасно при одноразовому виклику функції.

Наприклад:

`fig, ax = plt.subplots(1,2)` – створює один рядок з двох канв. Доступ до канв проводиться по індексу масиву ax[], ax[]

`fig, ax = plt.subplots(2, 2)` – створює два рядки по дві канви в кожному. Доступ до канв проводиться по індексу масиву ax[0,0] ax[1,1]

`fig, (ax1, ax2) = plt.subplots(1, 2)` – створює один рядок з двох канв. Доступ до канв проводиться по іменам ax1, ax2

`fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2)` – створює два рядки по дві канви в кожному. Доступ до канв проводиться по іменам ax1 ... ax4.

Використання об’єктного програмування є ефективним у складних застосунках для забезпечення роботи з кількома вікнами та канвами. В простих випадках можна обійтися простим застосуванням функцій. Бібліотека має функції-аналоги методам практично всіх методів побудови та оформлення графіків. Наприклад, `plt.text()` – додає текст до канви, `plt.title()` – додає назву до канви, `plt.xlabel(), plt.xlabel()` – додає назву до осі, `plt.xlim(), plt.ylim()` – встановлює/зчитує діапазонів значень по осі, `plt.xscale(), plt.yscale()` – встановлює масштаб для осі, `plt.xticks(), plt.yticks()` – встановлює/зчитує мітки по осях. Дії проводяться з поточною канвою.

Функція `<axeshandle>=plt.subplot(pos,args)` створює в поточному вікні прямокутну мозаїку канв. Аргумент `pos` – трійка чисел аналогічно методу `fig.add_subplot(pos)`. Додаткові аргументи `projection= None/ 'aitoff'/ 'hammer'/ 'lambert'/ 'mollweide'/ 'polar'/ 'rectilinear'` – тип графіків. За замовчанням `'rectilinear';`

`sharex=, sharey=` – аналогічно `plt.subplots()`. Алгоритм застосування є аналогічним до методу `fig.add_subplot(pos)`.

Тривимірна графіка

Базові засоби тривимірної графіки містяться в бібліотеці `mplot3d` з модуля `mpl_toolkits`. Бібліотека поставляється разом **Python** з та потребує завантаження аналогічно іншим зовнішнім бібліотекам.

```
from mpl_toolkits.mplot3d import Axes3D
```

Бібліотека складається з чотирьох частин:

- *axes3d* – основний модуль з описом класів та методів тривимірної графіки;
- *axis3d* – додатковий модуль з розширеними методами налаштування оформлення;
- *art3d* – додатковий модуль переведення 2D об'єктів в 3D;
- *proj3d* – додатковий модуль проміжних перетворень.

Після імпорту бібліотеки у властивостях канви з'являється опція 3D графіки `projection='3d'`.

Синтаксис всіх методів тривимірної графіки однаковий. Обов'язковими аргументами є три матриці X, Y, Z. X, Y – матриці координатної сітки, Z – матриця значень у вузлах сіток. Матриця X є повторенням по рядках діапазону зміни значень уздовж осі X. Матриця Y є повторенням по стовпцях діапазону зміни значень уздовж осі Y. Розраховувати координатні матриці можна вручну. Зручніше користуватися вбудованою функцією `X, Y = np.meshgrid(X, Y)`.

Візуалізувати двовимірні масиви можна і засобами **Pyplot**.

Функція `<handle=>plt.pcolormesh(X,Y,Z <,arg>)` виводить вид згори в псевдокольорах на масив Z розмірами Row x Cols. Опційні аргументи: `cmap='Colormap'` – назви застосованої карти кольорів. За замовчанням `'viridis'; norm= asinh'/ 'linear'/ 'log'` – вмикає нормалізацію значень до діапазону 0...1 та визначає тип перерахунку; `vmin=, vmax=` – визначає мінімальне/максимальне значення, всередині яких будуть обчислюватися псевдокольори; `edgecolors='none'/ 'face'` – визначає колір контурів комірок на зображення; `alpha=` – коефіцієнт прозорості; `shading='flat'/ 'nearest'/ 'gouraud'/ 'auto'` – визначає тип інтерполяції кольорів між пікселями;

Примітка. Функція підтримує масиви-зображення у форматі RGB розміром Rows x Cols x 3 та RGBA – Rows x Cols x 4.

Колірні схеми

В **Matplotlib** зафарбовування зображень базується на різницевій схемі CIELAB: яскравість lightness L; різниця червоний-зелений a; різниця жовтий-синій b. Фарбування проводиться в поточній кольоровій схемі Colormap. Для використання пропонується широка номенклатура зафарбувань (рис. 2.3). Містяться кольорові схеми в окремому зовнішньому модулі бібліотеки та потребують завантаження:

```
import matplotlib.cm as cm
```

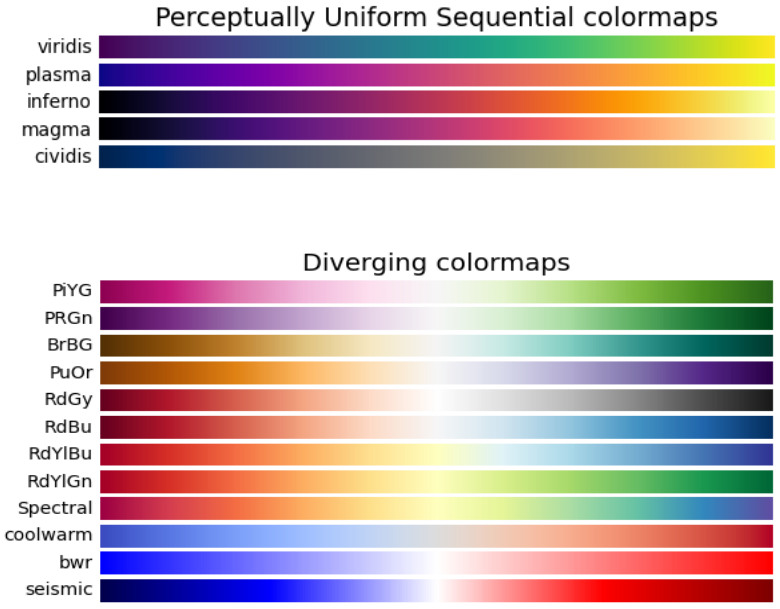


Рис. 2.3, аркуш 1. Кольорові схеми (з відкритих джерел)

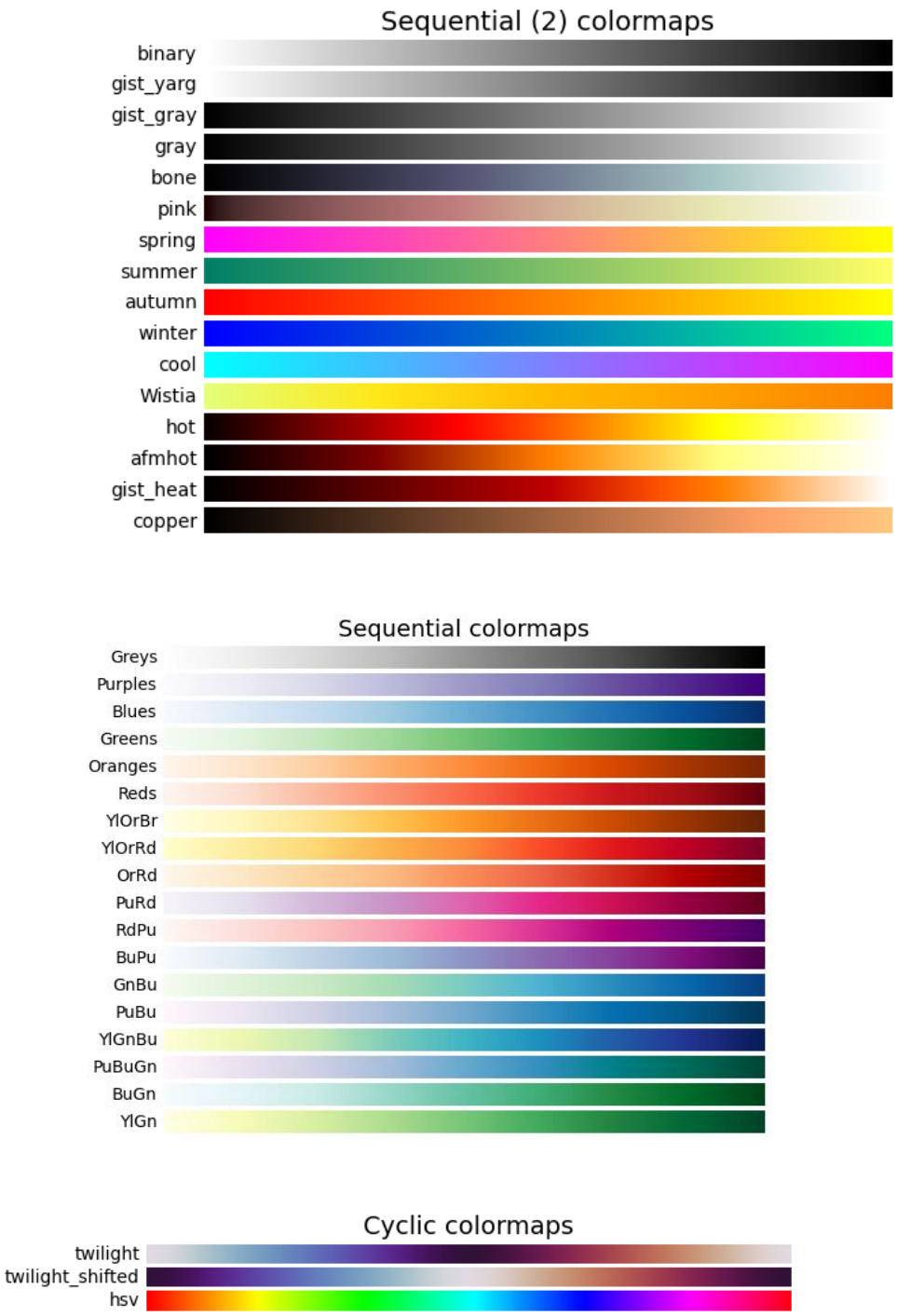


Рис. 2.3, аркуш 2. Кольорові схеми (з відкритих джерел)

Методи тривимірної графіки

Метод `<line3Dhandle=>axes3D.plot_wireframe(X, Y, Z<, args>)` – малює контурне тривимірне зображення. Опційні параметри: `rcount=, ccount=` – максимальна кількість рядків/стовпців на зображенні.

За замовчанням 50; rstride=, cstride= – крок проріджування. За замовчанням 1; color='Colorname'.

Метод <surfhandle=>=axes3D.plot_surface(X, Y, Z<, arg>) – малює зображення поверхні. Додаткові до _wireframe опційні параметри: За замовчанням rstride=cstride= 10; cmap='Colormap' - кольорова схема. За замовчанням 'viridis'; facecolors='Colorname' – масив кольорів для комірок; norm= asinh'/ 'linear'/ 'log' – вмикає нормалізацію значень до діапазону 0...1 та визначає тип перерахунку; vmin=, vmax= – визначає мінімальне/максимальне значення, всередині яких будуть обчислюватися псевдокольори; shade=True/False – тонування комірок в режимі facecolors;

Метод <linehandle=>=axes3d.plot(args) – малює лінійний просторовий графік. Аргументи: xs, ys, zs – одновимірні масиви координат; zdir='x'/ 'y'/ 'z' – напрям осі Z. За замовчанням 'z'.

Метод <contourhandle=>axes3D.contour(X, Y, Z<, arg) - малює контурний графік на визначеній площині. Опційні аргументи: extend3d=True/False – визначає необхідність розширення графіків на всі осі. За замовчанням False; zdir='x'/ 'y'/ 'z' – напрям осі Z. За замовчанням 'z'; offset= – визначає зсув площини контурних графіків перпендикулярно zdir

Метод <contourhandle=>axes3D.contourf(X, Y, Z <, args>) – малює зафарбований за кольоровою схемою малює контурний графік на визначеній площині.

Завдання до виконання

M – сума номеру варіанту N та: +10 – для N=1...5, +5 – для N=6...10, 0 – для N=11...15, -5 – для N=16...20, -10 для N=– 20...25, -15 – для N=25...30.

Завдання 1. Створити вікно розмірами MxM/2 з фоновим кольором відповідно варіанту. У вікно вивести текст прізвища, номер варіанту та канву.

Завдання 2. Побудувати лінійні графіки для M точок [-N/2…N/2] з сіткою та легенду на одній канві:

- об’єктним методом з визначенням властивостей під час створення,
- без об’єктів з форматуванням постфактум, функціями.

Осі та канву підписати.

а) $\sqrt{|x|}$, $\sqrt[3]{|x|}$, $\sqrt[4]{|x|}$ б) $e^{-|x|}$, $e^{-\sqrt{|x|}}, e^{-x^2}$

Завдання 3. Побудувати на одній канві лінійчаті графіки з сіткою та легенду для функцій п. 2:

- об'єктним методом з визначенням властивостей під час створення,
- без об'єктів з форматуванням постфактум, функціями.

Осі та канву підписати.

Завдання 4. Побудувати на одній канві стовпову діаграму та графік сходинку для першої функцій відповідного варіанту п. 2. Осі та канву підписати.

Завдання 5. Побудувати сітчасте зображення функції для NxN точок

а) [-6…6] $\sqrt{x^2+y^2}$ б) [-3…3] $sinc(x^2+y^2)$ в) [-1…1] $\cos(\pi x)\sin(\pi y)$

Завдання 6. Побудувати поверхню та двомірні проекції лінійними графіками та контурними графіками для NxN точок

а) без зафарбовування б) з зафарбовуванням

Завдання 7. Вивести чотири зображення п.5, 6, 7 мозаїкою об'єктним методом та функціями.

ПРАКТИКУМ 3. ПОБУДОВА СИГНАЛІВ

Завдання роботи: вивчення можливостей ПЗ та набуття навичок в створенні моделей, періодичних та випадкових сигналів, генерації цифрових сигналів певної форми.

Теоретичні положення

Спеціальні функції, які застосовуються для опису сигналів

Функція Хевісайда (Heaviside), функція сходинки, функція стрибка (рис. 3.1).

$$\sigma(t) = \begin{cases} 0, & t < 0 \\ 0.5, & t = 0 \\ 1, & t > 0 \end{cases} = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$$

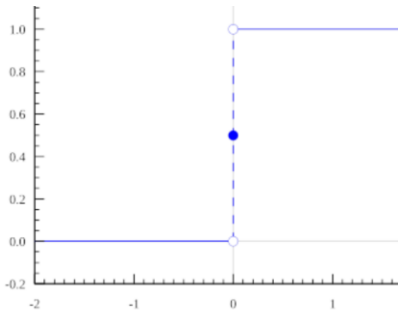


Рис. 3.1. Функція сходинки

Дельта функція Дірака, δ-функція (рис. 3.2).

$$\delta(t) = \begin{cases} \propto, & t = 0 \\ 0, & t \neq 0 \end{cases} \quad \delta(t) = \begin{cases} 1, & t = 0 \\ 0, & t \neq 0 \end{cases}$$

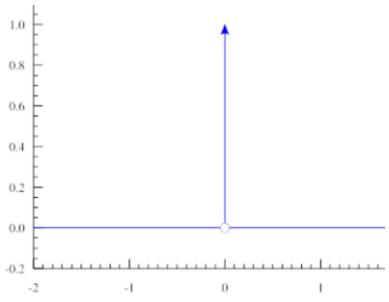


Рис. 3.2. Дельта функція

Дельта символ Кронекера

$$\delta k(t,n) = \begin{cases} 1, & t = n \\ 0, & t \neq n \end{cases}$$

Бібліотека **NumPy** має вбудовану функцію сходинки

`heaviside(x, x0)`, де x0 – зсув.

Бібліотека **SciPy** має вбудовану дельта функцію для цілочислового аргументу `unit_impulse(length<,number, dtype=>)`, де – `length` довжина масиву, `number` – номер елементу з одиничним значенням (`'mid'` – серединний елемент). За замовчанням – перший елемент.

В стандартних бібліотеках відсутня дельта функції для довільного аргументу. Описати їх, як і інші неперіодичні сигнали, можна за допомогою логічних функцій мови двома способами: оператором if та комбінацією унітарних логічних функцій або спеціальних функцій.

Улюблений програмістами спосіб if має недолік. Блок оператор не працює з масивами. Потрібно розділяти дії для скалярів та масивів.

Різновиди сигналів

Сигнали за способом представлення можна розділити на чотири типи: *аналогові* – описуються безперервними в аргументах та значеннях функціям, *дискретні* (*дискретизовані*) – описуються безперервними по значеннях функціях, мають значення на скінченій множині точок, *квантовані* – описуються безперервними в аргументах функціям та мають скінчений набір значень, *цифрові* – дискретні та квантовані одночасно (рис. 3.3).

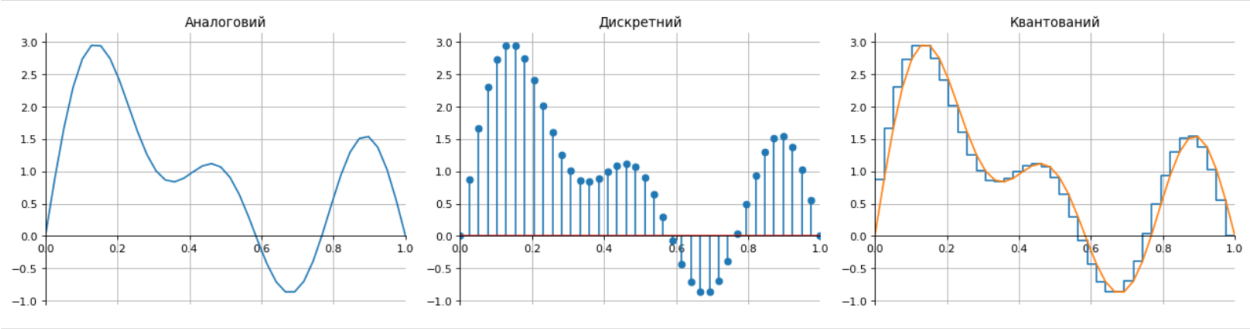


Рис. 3.3. Різновиди сигналів

Операції дискретизації та квантування сигнала

Математично операція дискретизації проводиться множенням аналогового безперервного сигнала на функцію гратки Пуассона. Гратка Пуассона є послідовністю рівновіддалених на крок дискретизації функцій Дірака (рис. 3.4).

$$Fd(t) = Fanalog(t) \sum_{n=-\infty}^{\infty} \delta(t - n\Delta t) = Fd(n\Delta t)$$

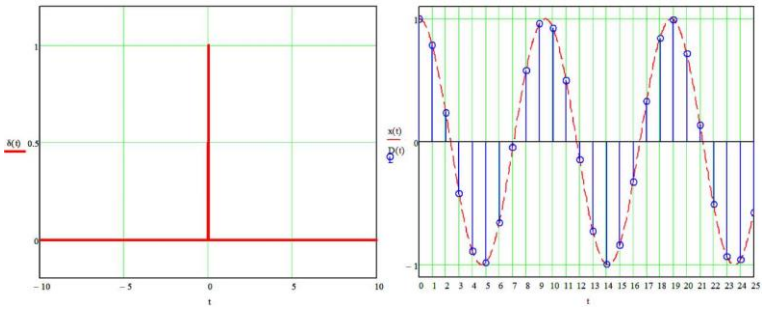


Рис. 3.4. Дискретизація сигналів

Технічно ця операція проводиться вибіркою з частотою дискретизації $fd = \frac{1}{\Delta t}$ значень з дискретизатора (АЦП в часі, фотокомірками МФП в просторі тощо). Для комп'ютерних тестових сигналів витратна теоретична операція дискретизації проводиться простим визначенням значень функції на скінченій регулярній множині аргументів.

Примітка. В комп'ютерному середовищі для опису сигналів, операцій цифрової обробки НЕ ВРАХОВУЮТЬСЯ умови дискретизації. Сигнали, системи, операції, результати описуються у вигляді масивів з аргументами у вигляді безрозмірних індексів, що відповідає нормованим аргументам. Результати ПОТРЕБУЮТЬ ДОВЕДЕННЯ ДО ФІЗИЧНИХ ЗНАЧЕНЬ.

Квантований сигнал (рис. 3.5) через функцію сходинки математично описується як

$$Fkv(t) = Fanalog(\Delta k \sum_{n=0}^{N} \sigma(t - n\Delta k),$$

де N – кількість рівнів квантування, Δк - крок квантування.

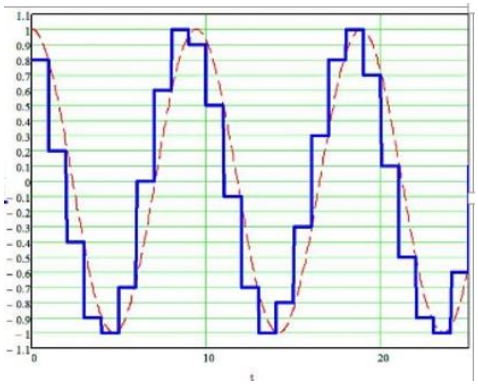


Рис. 3.5. Квантований сигнал

Технічно квантування визначається розрядністю (кроком квантування) АЦП.

Для переведення аналогового сигнала в квантований потрібно знати крок квантування Δ, мінімальне Fmin та максимальне Fmax квантоване значення АЦП. Переведення проводиться шляхом визначення піддіапазону квантування n, в який потрапляє аналогове значення та розрахунку квантованого сигнала як

$$Fkv = Fmin + n\Delta.$$

Найпростіше визначити піддіапазон логічними функціями **cell/floor**.

Імпульсні або неперіодичні сигнали

Опис імпульсних сигналів є найпростішим:

$$Fimpuls(t) =< existing\ zone\ definition >\cdot Fcontinious(t)$$

Зона існування – діапазон значень аргументу, всередині якого функція імпульсу має ненульові значення. Задається парами «початок-кінець» або «початок(зсув)-тривалість».

Визначення можливо оператором if, комбінацією унітарних логічних функцій, спеціальними функціями.

Наприклад, для зсуву x0 та тривалості T можна записати:

- ```
  if x>=x0 and x<=x0+T:
  ```
- ```
  (x>=x0)*(x<=(x0+T))
  ```
- $(\sigma(x - x0) - \sigma(x - x0 - T))$ різниця функції сходинки, зсуненої на x0 та функції сходинки, зсуненої на x0+T
- $(\sigma(x - x0) * \sigma(x0 + T - x))$ добуток функції сходинки, зсуненої на x0 та оберненої функції сходинки, зсуненої на x0+T

Періодичні сигнали

Аналогові періодичні сигнали обумовлені на всій осі значень аргументів без обмежень. Головною ознакою періодичності є рівність значень функції сигнала в будь-яких точках, віддалених на величину, кратну періоду сигнала:

$$F(t) = F(t - nT),\ \ n\ \mathrm{C}\ [-\propto..+\propto]$$

Гармонічні періодичні сигнали

Гармонічні сигнали мають за основу гармонічну тригонометричну функцію, яка може мати обмеження по амплітуді та/або по тривалості (рис. 3.6).

$$F(t) = A \cdot sincos\left(\frac{2\pi t}{T} - \varphi\right) \qquad (3.1)$$

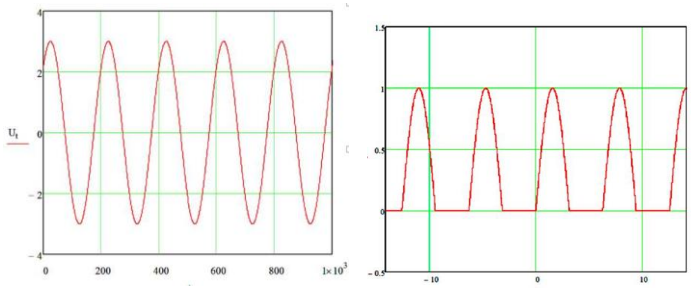


Рис. 3.6. Періодичні сигнали

Обмеження по амплітуді проводяться аналогічно врахуванню області існування імпульсу. Відмінність полягає в тому, що потрібно для перевірки використовувати не значення аргументів, а значення самої функції.

Бібліотека **NumPy** має для цього зручні функції обмежень:

$$\text{np.clip}(\text{x}, \text{x\_min}, \text{x\_max}) = \begin{cases} \text{x\_min} & x < \text{x\_min} \\ \text{x} & \text{x\_min} \le \text{x} \le \text{x\_max} \\ \text{x\_max} & x > \text{x\_max} \end{cases}$$

$$\text{np.where}(\text{condition}, [\text{x}, \text{y}]) = \begin{cases} \text{x}, & \text{condition} - \text{true} \\ \text{y}, & \text{condition} - \text{false} \end{cases}$$

Обмеження по тривалості потребують накладання умов на аргумент з урахуванням періоду функції.

Полігармонічні сигнали (рис. 3.7) складають широко поширену групу періодичних сигналів і описуються сумою гармонічних коливань:

$$F(t) = \sum_{i=1}^{N} A_i \cdot sincos(2\pi t f_i - \varphi_i)$$

Значення ff = 1/Tp, де Tp – період одного повного коливання результуючого сигнала, називають фундаментальною частотою коливань. Полігармонічні сигнали являють собою суму гармонічних складових з довільними значеннями амплітуд Ai і фаз фі, з періодами, кратними періоду фундаментальної частоти ff. Іншими словами, на періоді фундаментальної частоти Tp, яка дорівнює або

кратно менше мінімальної частоти гармонік, укладається кратне число періодів всіх гармонік, що і створює періодичність повторення сигнала.

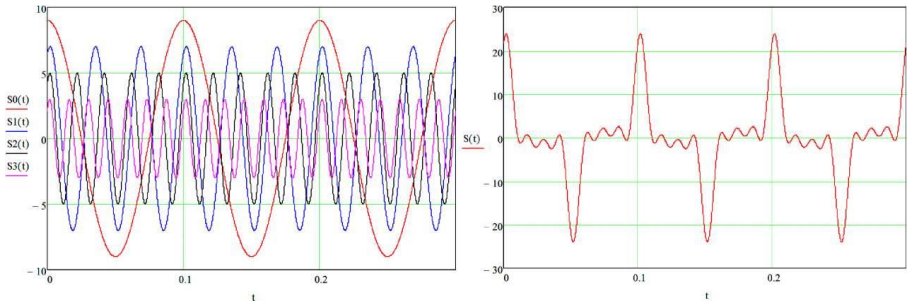


Рис. 3.7. Полігармонічні сигнали

«Квазіперіодичні» сигнали

«Квазіперіодичні» сигнали є періодичними за основною ознакою, але на періоді описуються негармонічною функцією (рис. 3.8).

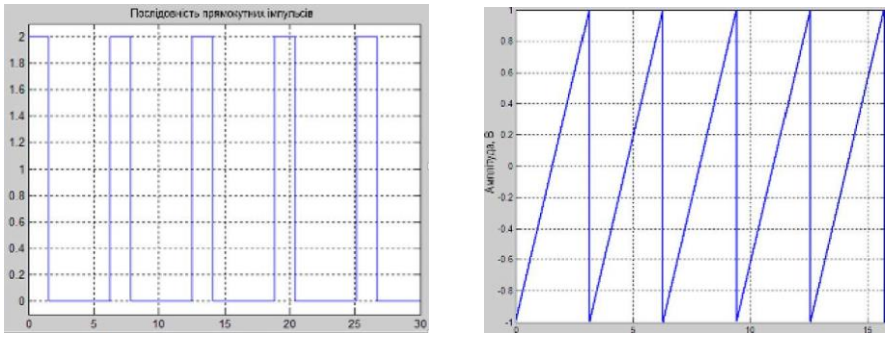


Рис. 3.8. «Квазіперіодичні» сигнали

Опис таких сигналів проводиться через функцію сигнала на першому періоді з приведенням значень аргументів зовні першого періоду до початку.

Бібліотека **SciPy** має вбудовані функції для деяких типових сигналів.

Функція `y=sc.square(t, duty=0.5)` – розраховує меандр (симетричний в діапазоні -1 … 1 періодичний сигнал з періодом 2π на інтервалі t коефіцієнтом заповнення `duty`. За замовчанням 0.5. В разі задання заповнення у вигляді масиву, моделюється широтна модуляція сигнала за шириною імпульсу.

Функція `y=sc.sawtooth(t, width)` розраховує квазіперіодичний пилкоподібний трикутний сигнал в діапазоні -1 … 1 з періодом 2π на інтервалі t, `width` – нормована ширина імпульсу [0 … 1].. Від 0 до `width*2*pi` сигнал зростає, від `width*2*pi` до `2*pi` – спадає.

Імовірнісні сигнали

Детерміновані сигнали в техніці є ідеалізацією. В будь-якому сигналі технічної системи присутні шуми – власні шуми самого явища, флуктуації зовнішніх умов, внутрішні шуми елементів технічної системи, Шуми є випадковими сигналами та описуються імовірнісними функціями.

Технічний сигнал є сукупністю детермінованої компоненти та шумової-імовірнісної (рис. 3.9).

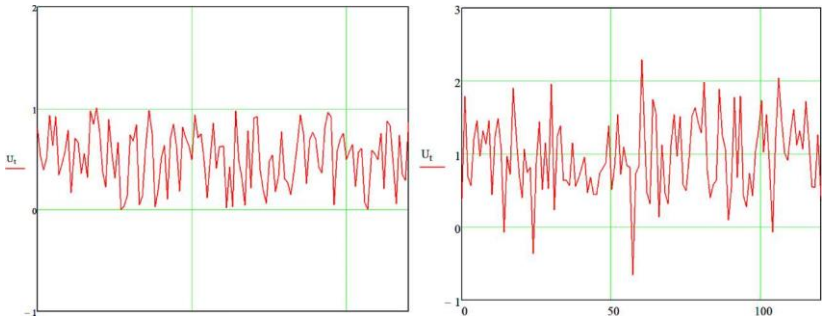


Рис. 3.9. Імовірнісні сигнали

Серед найголовних для опису випадкового сигнала є функція щільності розподілу імовірності (*probability density*), яка визначає закон зміни випадкової величини. Сигнали називають саме за назвою щільності розподілу: білий-рівномірний, експоненційний, Пуассона, Релея, нормальний, Гауссів тощо. Параметрами розподілів є перший початковий момент – середнє значення/ математичне очікування та другий центральний момент – дисперсія D, яка дорівнює квадрату середньоквадратичного відхилення СКВ σ.

Роботу з випадковими даними в **Python** дозволяють кілька бібліотек. Базовою є бібліотека **random**. Вона містить генератори випадкової змінної для типових розподілів. *З масивами не працює*.

Ширші можливості має бібліотека **numpy**.**random**. Вона реалізує всі можливості бібліотеки **random** як для скалярів, так і для масивів, та додає засоби визначення моментів та інших статистичних параметрів, побудови гістограм. Найбільш потужною є бібліотека **Scipy**.**stats**. Вона містить засоби розрахунку практично будь-яких характеристик, передбачених теорією імовірності з підвищеною точністю.

Наведемо деякі функції для генерації типових розподілів бібліотеки **numpy**.**random**.

Примітка. В версіях бібліотеки старших за 1.17 для генерації змінних введено новий клас Generator, методи якого є рекомендованими. Далі наводяться «класичні» методи, які є працездатними у всіх версіях.

Функція	Дія
`random.rand(d0, … ,dn)`	Рівномірний розподіл [0,1) $p(x) = 1$
`random.uniform(<Low=, high=, size=>)` `random.Generator.uniform`	Рівномірний розподіл [low,high). За замовчанням [0.0,1.0) $p(x) = \frac{1}{high - low}$ $m = \frac{low + high}{2}$ $D = \frac{(high - low)^2}{12}$
`random.random(<size=>)`	Рівномірний розподіл [0,1) $p(x) = 1$
`random.normal(<loc=, scale=, size=>)`	Нормальний розподіл з середнім loc, СКВ scale. За замовчанням 0, 1 $p(x) = \frac{e^{-\frac{(x-loc)^2}{2scale^2}}}{\sqrt{2\pi scale^2}}$
`random.standard_normal(<size=>)` `random.Generator.standard_normal`	Стандартний нормальний (Гауссів) розподіл.
`random.randn(<size=>)`	Стандартний нормальний (Гауссів) розподіл.
`random.poisson(<lam=, size=>)` `random.Generator.poisson`	Розподіл Пуассона. За замовчанням lam=1 $p(x) = \text{lam} \cdot e^{lam}$ $m = \text{lam}$ D $= \text{lam}$
`random.exponential(<scale=, size=>)` `random.Generator.exponential`	Експоненційний розподіл. За замовчанням scale=1 $p(x) = \frac{\exp(-\frac{x}{scale})}{scale}$ $m = \text{scale}$ D $= scale^2$

`random.rayleigh(<scale=, size=>)` `random.Generator.rayleigh`	Розподіл Релея. За замовчанням scale=1

$$p(x) = \frac{x \cdot \exp(-\frac{x^2}{2 \cdot scale^2})}{scale^2} \quad m$$

$$= \text{scale}\sqrt{\frac{\pi}{2}} \quad \text{D}$$

$$= (2 - \frac{\pi}{2})scale^2$$

Стандартні розподіли приводяться до розподілів з М та СКВ як

`М+СКВ·станд.розподіл()`

Застосування генераторів випадкових чисел є аналогічним такому в мові С++. Результатом кількох запусків генератора будуть однакові послідовності. Для скидання умов генерації слід перед запуском генератора викликати метод `numpy.random.seed()` або `numpy.random.Generator()`.

Для визначення математичного очікування масиву призначена функція

`mean(a <, axis=, dtype=, …>)`

Для визначення СКВ очікування масиву призначена функція

`std(a <, axis=, dtype=, …>)`

Гістограма

Гістограма (рис. 3.10) є наближеним експериментальним аналогом функції щільності розподілу. По осі абсцис вона містить N піддіапазонів значень функції від мінімального значення до максимального. По осі ординат вона містить кількість значень, які потрапляють в кожен піддіапазон. Кількість може бути абсолютною або нормованою відносно загальної кількості значень. Для порівняння з функцією щільності розподілу слід проводити нормування.

При побудові гістограми область значень випадкового сигнала розбивається на деяку кількість сегментів, а потім підраховується відсоток попадання даних в кожен сегмент.

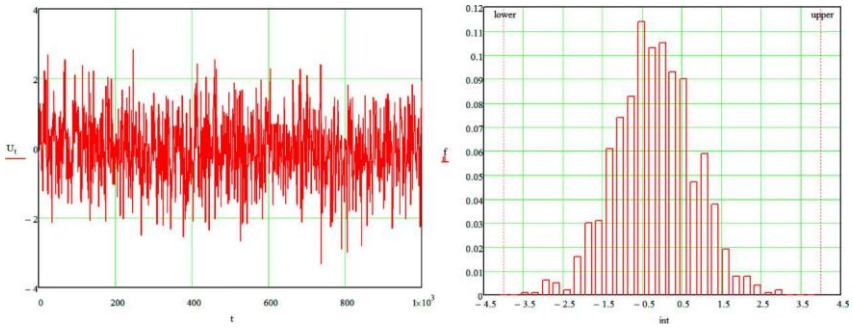


Рис. 3.10. Гістограма сигналу

Бібліотека **NumPy** має функцію розрахунку гістограми

`<val,bin=>np.histogram(data<, bins=, range=, normed=, density=>)`, де `bins` – ціле число піддіапазонів або список координат центрів піддіапазонів. За замовчанням 10; `range` – список з двох значень мінімального та максимального значень зовні яких значення відкидаються; `normed` – логічний параметр для нормування; `density` – аналогічний `normed`.

Функція повертає три масиви: перший `val` містить значення гістограми, другий `bin` – координати кінців піддіапазонів.

Відобразити гістограму можна як стовпову діаграму, графік сходинок, графік.

Проте простіше застосувати функцію бібліотеки **PyPlot.Matplotlib** `<val, xbins, lines=>pyplot.hist(x <, args>)`, яка з використанням `np.histogram()` розраховує гістограму для масиву x та виводить її на екран. Опційні результати `val, xbins, lines` є масивами значень гістограми, координат піддіапазонів стовпців, покажчиком на лінію. Опційні аргументи: `bins=` – кількість стовпців. За замовчанням 10; `range=` – двоелементний кортеж зі значеннями, поза якими значення x не враховуються; `density=True/False` – розрахунок щільності розподілу або кількості потраплянь. За замовчанням `False; cumulative=True/False` – розрахунок кумулятивного розподілу. За замовчанням `False; bottom=` – рівень нижнього значення стовпців. За замовчанням 0; `histtype='bar'/ 'barstacked'/ 'step'/ 'stepfilled'`. За замовчанням `'bar'; align='mid'/ 'left'/ 'mid'/ 'right'` – прив'язка до значень аргументів на осі; `orientation='vertical'/'horizontal'; rwidth=` – нормована ширина стовпців; `log= True/False` – логарифмічний масштаб осей. За замовчанням `False; color='Colorname'` – колір стовпців.

Завдання до виконання

Для всіх завдань:

- розробити універсальну підпрограму дії завдання. Вхідні параметри підпрограми: аргумент (час/координата) та за потреби параметри сигналу, результат – одиночне значення сигналу/масив значень. Функція має працювати для одиничного аргументу та аргументу масиву розміром M.
- побудувати графік в абсолютних координатах.

N – номер варіанта. M=N *20 для 1-5 варіантів, *10 для 6-9, *5 – для 10-15, *2 – для 16-25.

Завдання 1. Змоделювати дельта функцію блоком if, where для положення N.

Завдання 2. Змоделювати квантування функції (3.1) для A=1. Кількість рівнів квантування 5+N mod(5).

Завдання 3. Змоделювати біполярний імпульсний сигнал (рис. 3.11). Розрахувати сигнал з амплітудою N, зсувом точки симетрії N, тривалістю кожного напівімпульсу N/2.

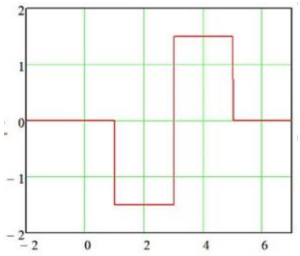


Рис. 3.11 – Біполярний імпульс

Завдання 4. Змоделювати каузальний імпульсний сигнал на інтервалі O..M, тривалістю імпульсу M/2 наступної форми:

а) x б) √x в) exp(x) г) exp(-x).

Завдання 5. Змоделювати гармонічну функцію (4.1) з обмеженнями

а) знизу значенням A унітарно б) знизу значенням A сходинкою в) згори значенням B унітарно г) згори значенням B сходинкою.

Завдання 6. Змоделювати полігармонічний сигнал, що містить 4 гармонічних складових. Параметри гармонічних складових сигнала: амплітуда гармонік $A_к$ = {9, 7, 5, 3}; частота $f_к$ = {10, 30, 50, 70}; початковий

фазовий кут коливань $\phi_к$ = {0, -0.4, -0.6, -0.8}. Вивести графік окремих складових і всього сигнала в цілому. Визначити фундаментальну частоту полігармонічного сигнала.

Завдання 7. Змоделювати періодичний сигнал з періодом M, скважністю 1. Вираз функції на періоді згідно п.4. Сигнал відобразити на інтервалі 3M.

Завдання 8. Змоделювати випадковий сигнал для 100 точок:

а) рівномірний б) Пуассона в) експоненційний г) Релея.

Математичне очікування N, СКВ* N/2.

Вивести графік сигнала, математичне очікування, СКВ.

ПРАКТИКУМ 4. СПЕКТРАЛЬНИЙ АНАЛІЗ СИГНАЛІВ

Завдання роботи: вивчення можливостей ПЗ та набуття навичок в розрахунку та побудові спектру амплітуд та фаз імпульсного сигналу, частотної фільтрація сигналу за допомогою перетворення Фур'є.

Теоретичні положення

В основі спектрального аналізу сигналів, систем та обробки сигналів лежать дві залежності Фур'є.

Для неперервних аналітичних періодичних з періодом T функцій, які вдовольняють умовам Діріхлє – розкладання в ряд Фур'є:

$$Fper(t) = \sum_{n=-\infty}^{\infty} C_n \exp(j2\pi n f_1 t), \quad f_1 = \frac{1}{T} \quad C_n = f_1 \int_0^T f(t) \exp(-j2\pi n f_1 t)\, dt$$

$$C_n = |C_n| \exp\big(j \cdot arg(C_n)\big), \qquad |C_n| = C_n C_n^* = \sqrt{C_n C_n},$$

$$\arg(C_n) = arctg\left(\frac{Im(C_n)}{Re(C_n)}\right)$$

Для неперервних аналітичних імпульсних функцій, які вдовольняють умовам Діріхлє – перетворення Фур'є:

$$Fimp(t) = \int_{-\infty}^{\infty} Fimp(t) \exp(j2\pi f t)\, df, \quad Fimp(f) = \int_{-\infty}^{\infty} Fimp(t) \exp(-j2\pi f t)\, dt$$

Фазова характеристика функції:

$$\Phi(f) = arg(C_n) = arctg\left(\frac{Im(C_n)}{Re(C_n)} = arg\big(\widetilde{F_n}\big) = arctg\left(\frac{Im(\widetilde{F_n})}{Re(\widetilde{F_n})}\right.\right.$$

Амплітудна характеристика функції:

$$A(f) = |C_n| = \sqrt{C_n C_n^*} = \big|\widetilde{F_n}\big| = \sqrt{\widetilde{F_n}\widetilde{F_n^*}}$$

Примітка. Вже для аналітичних функцій існують непорозуміння. В багатьох джерелах, зокрема в тих, на яких ґрунтуються комп'ютерні методи (і Python зокрема), використовують вирази з поміняними знаками для прямого та оберненого перетворення. В бібліотеках – вірно!

Цифрові каузальні функції визначені тільки для додатних значень аргументів, обмежені на інтервалі спостереження t_{obs}, аргумент є скінченим масивом дискретних значень.

Вказані особливості призводить для таких функцій до наступного. Функції описуються скінченими масивами чисел.

Фур'є спектр таких сигналів є періодичною функцією з періодом $fobs = \frac{1}{\Delta t}$. Якщо неперервна аналітична функція F(t) має Фур'є спектр $\tilde{F}(f)$, то дискретна функція з N точок має спектр

$$F_N(f) = \int_{-\infty}^{\infty} F(t) \sum_{k=0}^{N-1} \delta(t - k\Delta t) \exp(-j2\pi f t)\, dt$$

$$= \sum_{k=0}^{N-1} \tilde{F}\left(f - \frac{k}{\Delta t}\right) = \sum_{k=0}^{N-1} \tilde{F}(f - kfobs)$$

Для імпульсних дискретних сигналів з кроком дискретизації *Δt*, які обмежені інтервалом спостереження *tobs*:

$$N = \frac{tobs}{\Delta t} = \frac{fobs}{\Delta f}, \qquad \Delta t = \frac{1}{fobs}, \qquad \Delta f = \frac{1}{tobs}$$

$$\tilde{F}(f) = \int_{-\infty}^{\infty} Fimp(t) \exp(-j2\pi f t)\, dt = \int_{0}^{tobs} Fimp(t) \exp(-j2\pi f t)\, dt$$

$$= \sum_{n=0}^{N-1} F(n\Delta t) exp(-j2\pi n \Delta t f) \Delta t = \Delta t \sum_{n=0}^{N-1} F(n\Delta t) exp(-j2\pi n \Delta t f)$$

$$\tilde{F}(k\Delta f) = \tilde{F}_k = \Delta t \sum_{n=0}^{N-1} F_n exp(-j2\pi n \Delta t k \Delta f) = \Delta t \sum_{n=0}^{N-1} F_n exp\left(-j2\pi \frac{nk}{N}\right)$$

$$F(t) = \int_{-\infty}^{\infty} \tilde{F}(f) \exp(j2\pi f t)\, df = \int_{0}^{tobs} Fimp(t) \exp(-j2\pi f t)\, dt$$

$$= \sum_{n=0}^{N-1} \tilde{F}(n\Delta f) exp(j2\pi t n \Delta f) \Delta f = \Delta f \sum_{n=0}^{N-1} \tilde{F}(n\Delta t) exp(j2\pi t n \Delta f)$$

$$F(k\Delta t) = F_k = \Delta f \sum_{n=0}^{N-1} \tilde{F}_n exp(j2\pi k \Delta t n \Delta f) = \Delta f \sum_{n=0}^{N-1} \tilde{F}_n exp\left(j2\pi \frac{nk}{N}\right)$$

Для періодичних сигналів:

$$C_k = \frac{1}{T}\int_0^T f(t)\exp\left(-j2\pi\frac{kt}{T}\right)dt = \frac{1}{tobs}\int_0^{tobs} f(n\Delta t)\exp\left(-j2\pi\frac{k\Delta t}{tobs}\right)dt$$

$$= \frac{\Delta t}{tobs}\sum_{n=0}^{N-1} F_n exp\left(-j2\pi\frac{nk}{N}\right)$$

$$C_{k=}\frac{1}{N}\sum_{n=0}^{N-1} F_n exp\left(-j2\pi\frac{nk}{N}\right) \quad F_{k=}\sum_{n=0}^{N-1} C_n exp\left(j2\pi\frac{nk}{N}\right)$$

Неперервне та періодичне перетворення співпадають з точністю до сталого множника. Спектральна щільність неперервного перетворення імпульсу має розмірність «сигнал/аргумент», періодичного – «сигнал». Результати відрізняються на множник 1/Tobs. Номери «i» гармонік періодичного перетворення відповідають частотам $f = \frac{i}{Tobs}$ неперервного перетворення.

Слід мати на увазі особливість збереження даних цифрових перетворень. Комплексні спектри визначені не нескінченому діапазоні аргументу від -∞ до +∞. З урахування обмеженості виборки часом спостереження *tobs* – від $-\frac{fmax}{2}$ до $+\frac{fmax}{2}$, де $fmax = \frac{1}{tobs}$. Каузальні системи визначені тільки для додатних аргументів. Відповідно, спектри в масивах визначені в діапазоні від 0 до $\frac{fmax}{2}$. Перша половина значень від 0 до $\frac{N}{2}$ відповідає частотам від 0 до $\frac{fmax}{2}$, друга половина від $\frac{N}{2}+1$ до N-1 відповідає частотам від $-\frac{fmax}{2}$ до 0 (рис. 4.1).

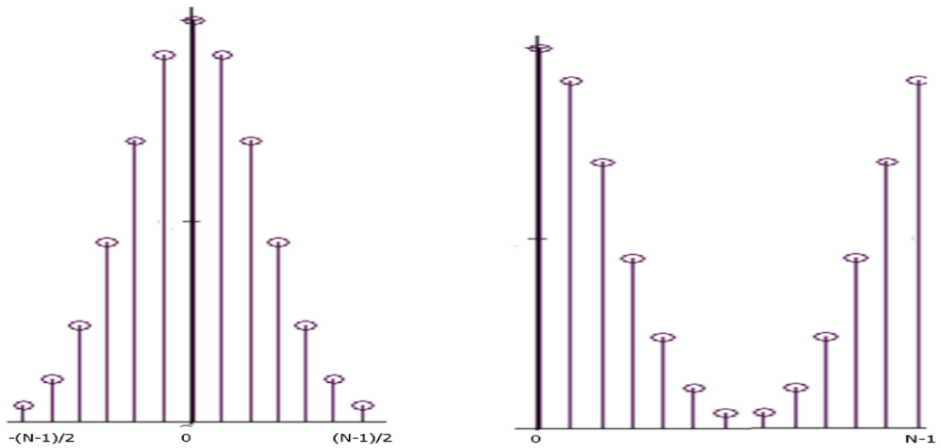


Рис. 4.1. Цифровий спектр

В обробці сигналів опис в часово-просторовій області дискретних систем ведеться для нормованого часу/простору $tnorm = \frac{t}{\Delta t},\ \Delta tnorm = 1$, де Δt – період/крок дискретизації. Розмір масивів визначається довжиною інтервалу спостереження $N = \frac{tobs}{\Delta t}$. Для нормованого часу $tobs\ norm = N.$

$$F(t)|-\propto< t <\propto= F(t)|0 \le t \le tobs = F(n\Delta t)|0 \le n \le N = F(n)$$

В обробці сигналів опис в частотний області ведеться для нормованих частот $fobs\ norm = \frac{1}{\Delta tnorm} = 1, \Delta fnorm = \frac{1}{tobsnorm} = \frac{1}{N}.$

Для нормованих координат:

$$\tilde{F}_k = \sum_{n=0}^{N-1} F_n exp\left(-j2\pi \frac{nk}{N}\right), \quad F_k = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{F}_n exp\left(j2\pi \frac{nk}{N}\right)$$

$$C_{k=}\frac{1}{N}\sum_{n=0}^{N-1} F_n exp\left(-j2\pi \frac{nk}{N}\right) \quad F_{k=}\sum_{n=0}^{N-1} C_n exp\left(j2\pi \frac{nk}{N}\right)$$

Результат неперервного перетворення (Фур'є спектр) прямокутного імпульсу амплітудою U тривалістю τ, зсувом τ0 аналітично описується наступною залежністю:

$$F(f) = U \cdot \tau \cdot e^{-j2\pi f \tau 0} \cdot \frac{\sin(\pi \cdot f \cdot \tau)}{\pi \cdot f \cdot \tau}$$

Амплітудна характеристика сигналу описується виразом:

$$A(f) = |F(f)| = U \cdot \tau \cdot |\frac{\sin(\pi \cdot f \cdot \tau)}{\pi \cdot f \cdot \tau}|$$

Фазова характеристика сигналу описується виразом:

$$\Phi(f) = -artcg(\frac{U \cdot \tau \cdot \mathrm{sinc}(\pi \cdot f \cdot \tau)\sin(2\pi \cdot f \cdot \tau o)}{U \cdot \tau \cdot \mathrm{sinc}(\pi \cdot f \cdot \tau)\cos(2\pi \cdot f \cdot \tau o)} = -arctg(2\pi \cdot f \cdot \tau o)$$
$$= -2\pi \cdot f \cdot \tau o + n\pi$$

Перший лінійний доданок відповідає експоненціальному співмножнику в виразі спектру, другий доданок враховує зміну знаку спектру F(f). Амплітудний спектр має бути знаконезмінним, мати тільки позитивні значення. Вихідний спектр є знакозмінним, існують ділянки, на яких він має від'ємний знак. Другий доданок у вигляді стрибка фази на π враховує цю особливість.

Комплексний ряд (Фур'є спектр) періодичної послідовності прямокутних імпульсів амплітудою U тривалістю τ, зсувом τ0, періодом T (прогальність К=Т/ τ) аналітично описується наступною залежністю:

$$Fp(f) = \sum_{n=-\infty}^{\infty} C_n \cdot \delta\left(f - \frac{n}{T}\right)$$

$$C_n = U\frac{\tau}{T}\frac{\sin(\pi \cdot n \cdot \frac{\tau}{T})}{\pi \cdot n \cdot \frac{\tau}{T}} e^{-j2\pi\frac{\tau 0}{T}} = \frac{U}{K}\frac{\sin(\frac{\pi \cdot n}{K})}{\pi \cdot \frac{n}{K}} e^{-j2\pi\frac{\tau 0}{T}} = \frac{U}{K} e^{-j2\pi\frac{\tau 0}{T}} sinc(\pi \cdot \frac{n}{K})$$

Амплітудна характеристика періодичного сигналу описується виразом:

$$Ap(f) = \sum_{n=-\infty}^{\infty} |C_n| \cdot \delta\left(f - \frac{n}{T}\right), \qquad Ap_n = \frac{U}{K} sinc(\pi \cdot n \cdot K)$$

Фазова характеристика періодичного сигналу описується виразом:

$$\Phi p(f) = \sum_{n=-\infty}^{\infty} \arg(C_n) \cdot \delta\left(f - \frac{n}{T}\right), \qquad \Phi_n = -2\pi \cdot \frac{n\tau 0}{T}$$

Функції швидкого дискретного перетворення Фур'є (Fast Furje Transform, FFT) містяться в бібліотеках **NumPy** та **SciPy**. Спільним для синтаксису всіх функцій є те, що для зворотного перетворення до імені функції додається префікс «i», для двомірних перетворень додається суфікс «2», багатомірних – суфікс «n», для дійсних перетворень – префікс «r» .

В бібліотеках дискретні перетворення Фур'є ЗА ЗАМОВЧАННЯМ обчислюється по наступних виразах:

$$\tilde{F}_k = \sum_{n=0}^{N-1} F_n exp\left(-j2\pi\frac{nk}{N}\right) \qquad F_k = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{F}_n exp\left(j2\pi\frac{nk}{N}\right)$$

Тобто, ВІДПОВІДАЮТЬ ТЕОРЕТИЧНИМ ПЕРЕТВОРЕННЯМ ДЛЯ НОРМОВАНИХ В ЧАСІ КООРДИНАТ.

Для масивів розміром $N=2^m$ використовується алгоритм швидкого перетворення Фур'є, для масивів з іншими розмірами – алгоритм дискретного перетворення. Розмір масивів вхідних даних та результату є ОДНАКОВИМИ.

Бібліотека **NumPy.fft** містить прямі та зворотні *комплексні* одновимірні, двовимірні та багатовимірні перетворення:

`fft()-ifft(); fft2()-ifft2(); fftn()-ifftn(),`

прямі та зворотні *дійсні* одновимірні, двовимірні та багатовимірні перетворення: `rfft()-irfft(); rfft2()-irfft2(); rfftn()-irfftn(),`

допоміжні функції зсуву масивів спектрів: `fftshift()-ifftshift(),` – розрахунку частот: `fftfreq()-rfftfreq().`

Дійсні перетворення призначені для вхідних функцій, які приймають тільки дійсні значення. Перетворення є швидшими за комплексні, бо значення спектрів для частот, більших за N/2 не розраховуються, а прирівнюються відповідним значенням з першої половини діапазону. Розмір масивів вхідних даних та результату НЕ Є ОДНАКОВИМИ. Для даних розміром N, результат має розмір $\frac{N}{2}+1$, для частот від 0 до N/2.

Бібліотека **SciPy.fft** містить крім базових перетворення для ермітових функцій (`hfft`), синусні та косинусні перетворення (`dct, dst`), додаткові допоміжні функції, графічні функції побудови спектра потужності Вінера, спектра Велча (Welch), спектра взаємної потужності (cross power spectral density), когерентності, спектрограм, періодограм тощо.

Функція `out=np.fft.fft(x<, arg>)` реалізує пряме швидке комплексне перетворення Фур'є. Опційні аргументи: `n=` - довжина вихідного масиву; `norm="backward"/"ortho"/"forward"` – тип нормування. За замовчанням `"backward".` `"backward"` – пряме перетворення без множника, зворотне – з множником `1/N, "ortho"` – пряме і зворотне перетворення з множником `1/N, "forward"` – пряме перетворення з множником `1/N`, зворотне – без множника `1/N.` Розміри масивів сигналу та спектру для комплексного перетворення є однаковими – `N.`

Розмір масиву спектру для дійсного перетворення масиву сигналу з розміром `N` для парного є `(N/2)+1`, для непарного – `(N+1)/2.`

Функція `out=np.fft.ifft(x<, arg>)` реалізує зворотне швидке комплексне перетворення Фур'є. Масив спектра має бути впорядкований за правилами перетворень. Опційні аргументи аналогічні `fft().`

Для оцифрування осі частот спектральних графіків корисною є функції `freq=np.fft.fftfreq( ), freq=np.fft.rfftfreq( ).` Аргументом є `n=` – розмір масиву. Опційний параметр: `d=` – крок дискретизації в часі Δt. За

замовчанням 1. Функції повертають дійсний масив, елементами якого є частоти, які відповідають номерам гармонік спектра.

Для комплексного перетворення повертається n додатних та від'ємних частот

`f = [0, 1, ..., n/2-1, -n/2, ..., -1] / (d*n)` для парного n;

`f = [0, 1, ..., (n-1)/2, -(n-1)/2, ..., -1] / (d*n)` для непарного n.

Для дійсного перетворення повертається n/2 додатних частот

`f = [0, 1, ..., n/2-1, n/2] / (d*n)` для парного n;

`f = [0, 1, ..., (n-1)/2-1, (n-1)/2] / (d*n)` для непарного n.

Функція `y=np.fft.fftshift(x)` застосовується для того, щоб надати масиву спектра звичний вигляд шляхом перестановки половин масиву між собою таким чином, що частоти ідуть за зростанням від мінімальної від'ємної, а гармоніка з нульовою частотою розміщується в центрі масиву.

Функція `x=np.fft.fftshift(y)` має зворотну до `fftshift(x)` дію.

Дещо скоротити кількість операцій для побудови спектральних графіків допоможуть функції бібліотеки **matplotlib** `magnitude_spectrum()` та `phase_spectrum()`.

Метод канви `<spectr, freq, linehandle=>magnitude_spectrum(x, args)` розраховує амплітудний спектр масиву x та будує його лінійний графік. Опційні аргументи: Fs= – ширина спектра ($fmax = \frac{1}{dt}$). За замовчанням 2; `window=window_hanning, window_none, numpy.blackman, numpy.hamming, numpy.bartlett` – віконна функція. За замовчанням `window_hanning; sides='onesided'/ 'twosided'` – визначає проведення дійсного перетворення `'onesided'` чи комплексного `'twosided'`. За замовчанням `'onesided'; pad_to=` – розмір масиву для розрахунку; `scale='linear'/ 'dB' (20 lg(y))` – масштаб осі ординат графіка. За замовчанням `'linear'; Fcint=` – центральна частота для x. За замовчанням 0.

Метод повертає дійсний масив спектра `spectr`, дійсний масив частот `freq` та посилання на графік `linehandle`.

Метод канви <`spectr, freq, linehandle=>phase_spectrum(x, args)` розраховує фазовий спектр масиву x та будує його лінійний графік. Аргументи та результати аналогічні `magnitude_spectrum()`. Відмінність в результаті: `spectr` – дійсний масив фазового спектра.

Частотна фільтрація

«Спектральна фільтрація реалізується фільтрами частотної селекції (ФЧС, вибіркові, трансверсальні, transversal). Такі фільтри частотно-залежну обробку сигналів, шляхом зміни співвідношення між амплітудами спектральних складових сигналу. При цьому форма сигналу теж змінюється.

Серед ФЧС залежно вигляду частотного відгуку розрізняють наступні типи фільтрів:

- фільтр нижніх частот (ФНЧ, lowpass filter). ФНЧ пропускає низькочастотні сигнали з частотою нижче верхньої граничної частоти зрізу, яку називають частотою зрізу *fc* (cutting frequency), та придушує сигнал на частотах вище граничної (рис. 4.2 *а*);
- фільтр верхніх частот (ФВЧ, highpass filters). Має дію, зворотну до дії ФНЧ. ФВЧ пропускає високочастотні сигнали з нижньої граничної частоти *fc* та придушує сигнал на частотах нижче граничної (рис. 4.2 *б*);
- смуговий фільтр (СФ, bandpass filters). СФ пропускає сигнали в визначеній смузі частот вище нижньої граничної частоти до верхньої граничної частоти. За межами смуги сигнали придушуються (рис. 4.2 *в*);
- режекторний або загороджувальний фільтр (ЗФ, РФ, bandreject, bandstop filters). Має дію, зворотну до дії СФ. ЗФ пропускає сигнали за межами визначеній смузі частот нижче нижньої граничної частоти та вище верхньої граничної частоти. Всередині смуги сигнали придушуються (рис. 4.2 *г*).» [1]

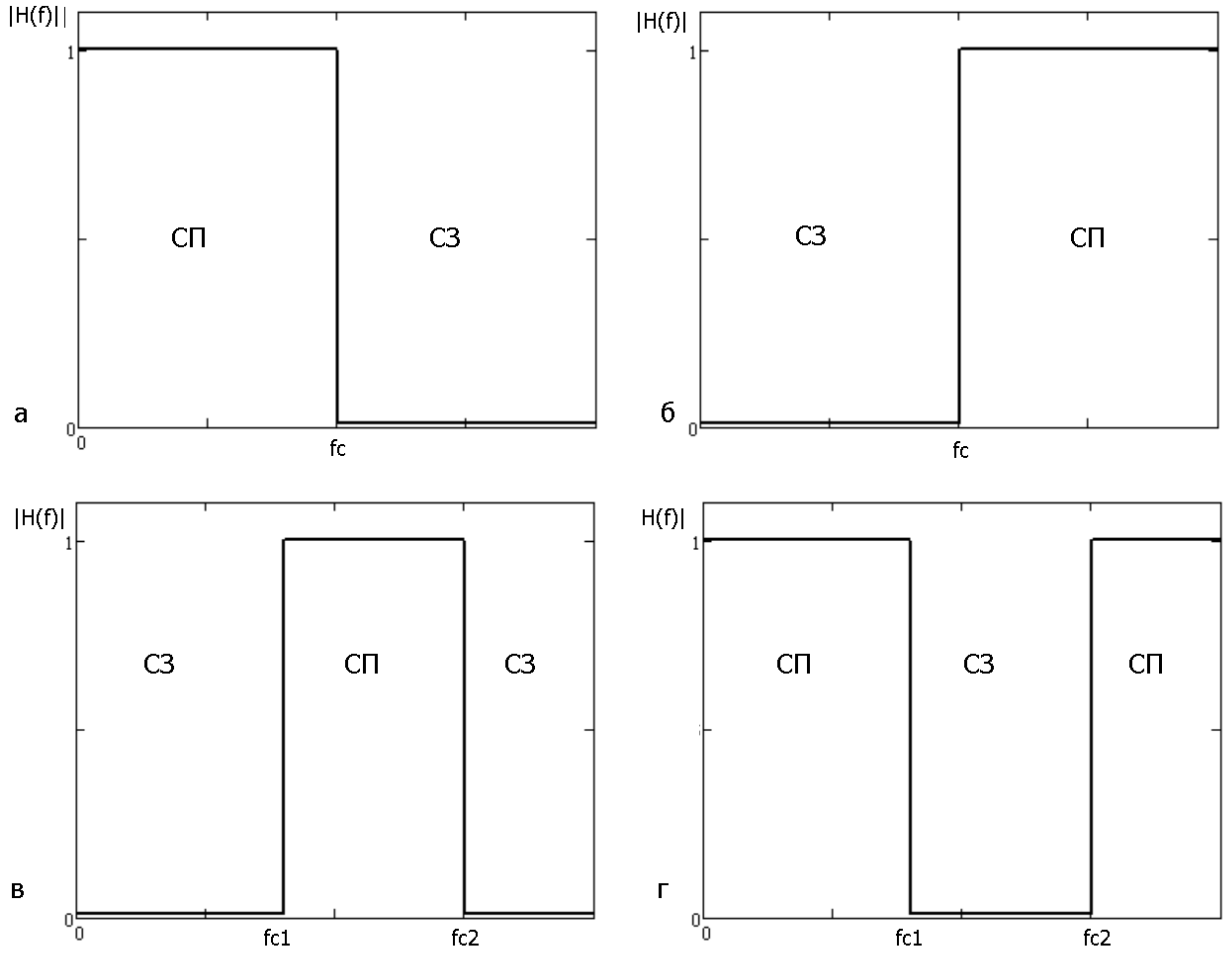


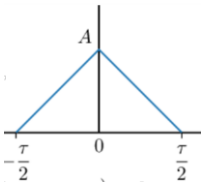
Рис. 4.2. Частотний відгук типових ідеальних ФЧС: а – ФНЧ; б – ФВЧ; в – СФ; г – ЗФ [1]

Завдання до виконання

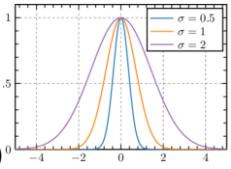
Завдання 1. Розрахувати спектр каузального сигнала (рис. 4.3) функціями бібліотек **NumPy, SciPy** та аналітично.

Побудувати графіки сигнала, дійсної, уявної частини спектра, амплітудну, фазову характеристики. Амплітудну, фазову характеристики виконати стандартними функціями ДПФ та спеціалізованими **matplotlib**. Вивести параметри сигналу. На графіках підписати та оцифрувати осі в абсолютних величинах та нормалізованих.

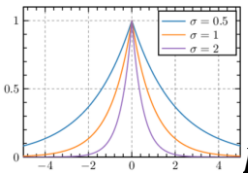
Тривалість сигналу – номер групи, амплітуда сигналу – номер варіанту «вольт», інтервал спостереження – подвійна тривалість «секунд». Кількість точок 16.

а) 

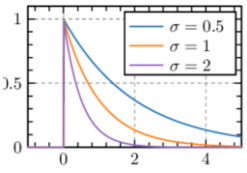
$F(x) = \begin{cases} A\left(1 - \left|\frac{2x}{\tau}\right|\right), x \le |\frac{\tau}{2}| \\ 0,\ x > |\frac{\tau}{2}| \end{cases}$, $\widetilde{F(f)} = \frac{A\tau}{2}\ sinc^2(\frac{\pi\tau}{2})$

б) 

$F(x) = A\exp(-\frac{x^2}{\sigma^2})$, $\widetilde{F(f)} = A\sigma\sqrt{\pi}exp(-\pi^2 f^2\sigma^2)$

в) 

$F(x) = A\exp(-\sigma|x|)$, $\widetilde{F(f)} = 2\frac{A\sigma}{\sigma^2+4\pi^2 f^2}$

г) 

$F(x) = \begin{cases} A\exp(-\sigma x), & x \ge 0 \\ 0, & x < 0 \end{cases}$, $\widetilde{F(f)} = \frac{A\sigma}{\sigma + j2\pi f}$

Рис. 4.3. Види сигналів до завдання 1

Примітка. Графіки та залежності рис. 4.3 а) – в) наведені для довідки для НЕКАУЗАЛЬНИХ сигналів.

Завдання 2. Проаналізувати вплив зменшення кількості гармонік на вид сигналу з п.1 за критерієм СКВ та максимальної абсолютної похибки. Кількість точок в первинному описі не менше 64.

Завдання 3. Виконати п.1 для сигнала як періодичного зі скважністю 2 на інтервалі в три тривалості імпульсу.

Завдання 4. Проаналізувати зміну часових витрат завдання п.1 для N-1, N, N+1 точок. N – найближче до умов п.1 число, кратне 2^m. Зробити висновки.

Завдання 5. До сигнала типу меандр з амплітудою – номер варіанта «вольт», скважністю 2, частотою 5 Гц на інтервалі 1 с додати білий шум з середнім 1 В, СКВ=0.5 В у 256 точках. Очистити сигнал частотною смуговою фільтрацією прямим коригуванням Фур'є спектра. Вивести сигнал та спектр до фільтрації та після фільтрації.

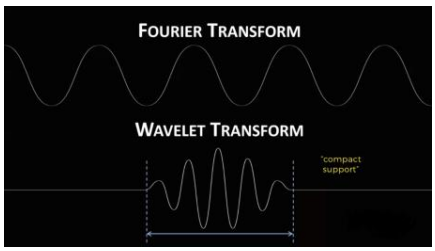
ПРАКТИКУМ 5. ЗАСТОСУВАННЯ ВЕЙВЛЕТІВ ДЛЯ ОБРОБКИ СИГНАЛІВ

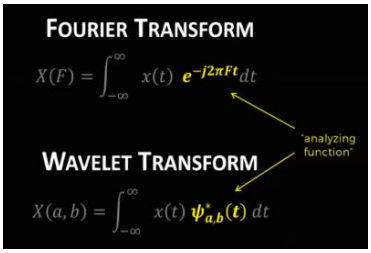
Завдання роботи: вивчення можливостей ПЗ та набуття навичок в проведенні одновимірне вейвлет-перетворення та розрахунках та аналізі вейвлетного спектру сигнала.

Теоретичні положення

Вейвлет-розкладання є одним різновидів представлення сигналу у вигляді функціонального ряду з функцій розкладання – ядра ψ та відповідних коефіцієнтів c:

$$f(t) = \sum_{n=-\infty}^{\infty} C_n \psi_n(t)$$





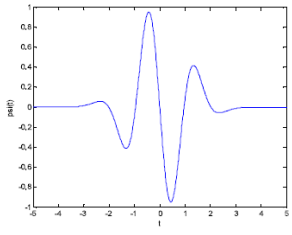
У Фур'є розкладанні ядром виступають необмежені за аргументом гармонічні тригонометричні функції. В вейвлет аналізі в якості ядра використовуються спеціальні вилоподібні «материнські, базові» функції ψ(t), які мають вдовольняти наступним умовам:

- мати нульове середнє значення $\int_{-\infty}^{\infty} \psi(t)dt = 0$, тобто не мати сталої складової;
- мати скінчену енергію $\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty$, тобто бути імпульсними, обмеженими за аргументом.

Типовими материнськими вейвлетами є

Гауссів вейвлет:

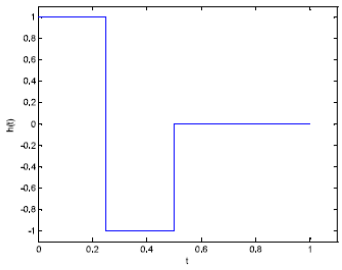
$$\Phi_a(t) = \left(\frac{2}{\pi a^2}\right)^{\frac{1}{4}} e\left(-\frac{t^2}{a^2} - jt\right), \qquad a = \frac{\omega}{\sigma}$$



a=5

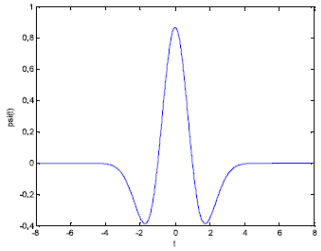
Вейвлет Хаара (Haar):

$$\Phi(t) = \begin{cases} 1, \text{ if } t \in [0; \frac{1}{2}) \\ -1, \text{ if } t \in [\frac{1}{2}; 1) \\ 0, \text{ if } t \notin [0; 1) \end{cases}$$



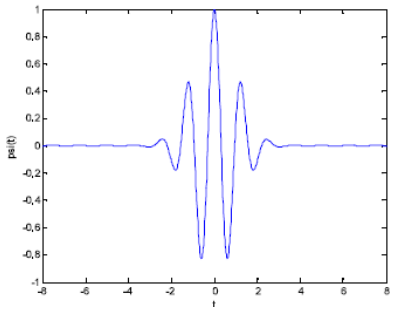
Вейвет Рікера (Ricker), «мексиканський капелюх», друга похідна від Гауссова вейвлета:

$$\Phi(t) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}\left[1 - (\frac{t}{\sigma})^2\right] e^{-\frac{t^2}{2\sigma^2}}$$

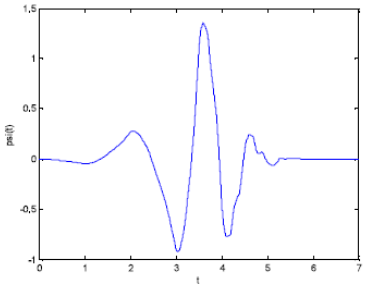


Вейвлет Морле (Morlet):

$$\Phi(t) = ke^{j\omega_0 t}e^{-\frac{t^2}{2}}$$



Вейвлет Добеши (Daubechie):



Родина вейвлетів створюється за допомогою зсувів на τ (*shift*, *transpose*) та масштабування (*scale*) з коефіцієнтом *a* базової материнської вейвлет-функції.

Масштабування вейвлета:

$$\psi_a(t) = \frac{\psi(\frac{t}{a})}{\sqrt{a}}$$

Масштаб вейвлета обернено пропорційний його частоті. Малі значення масштабу (низькі частоти, стиснуті вейвлети) фіксують повільні зміни в сигналі, великі значення масштабу (великі частоти, розтягнуті вейвлети) – швидкі зміни. Зі зменшенням масштабу вейвлет стає більш розтягнутим в часі.

Зсув вейвлета:

$$\psi_\tau(t) = \psi(t - \tau)$$

Зі зміною τ базовий вейвлет зсувається уздовж часової осі.

Перетворення вейвлета в загальному випадку записується як:

$$\psi_{a,\tau}(t) = \frac{\psi(\frac{t-\tau}{a})}{\sqrt{a}}.$$

Сама материнська вейвлет-функція є також елемент цього набору:

$$\psi_{1,0}(t) = \psi(t).$$

Деякі засоби роботи з вейвлетами для одновимірних функцій містить бібліотека **signal**.**Scipy**. Методи `daub()`, `morlet()`, `morlet2()`, `ricker()` розраховують вейвлети Добеші, Морле, Рікера відповідно. Метод `cwt()` проводить вейвлет-перетворення з використанням визначеного материнського вейвлета.

Можливості бібліотеки не можна вважати достатніми для проведення повноцінного вейвлет аналізу та фільтрації.

Метод `к=daub(p)` повертає 2p коефіцєнтів ФНЧ Добеші. Аргумент p визначає порядок нулів фільтра для нормованої частоти Найквіста від 1 до 34.

Метод `к=morlet(M<, w=, s=, complete=>)` повертає комплексний вейвлет Морлі. Цілочислений аргумент M визначає довжину вейвлета. Опційні параметри: `w` - значення центральної частоти. За замовчанням 5.0, s – масштабний коефіцієнт. За замовчанням 1.0. `complete=True/False`. За замовчанням `True`. Визначення виразу розрахунку.

Спрощений вираз має похибку для низьких частот. Застосовується для w>5.0.

$$\psi_w(t) = \frac{e^{jwt}e^{-\frac{t^2}{2}}}{\sqrt[4]{\pi}}$$

Повний вираз застосовується для всіх частот.

$$\psi_w(t) = \frac{(e^{jwt} - e^{-\frac{w^2}{2}})e^{-\frac{t^2}{2}}}{\sqrt[4]{\pi}}$$

Метод `к= morlet2(M, s<, w=>)` повертає комплексний вейвлет Морлета за повним виразом у формі, адаптовані для застосування з методом `cwt()`. В аргументах відсутній параметр `complete`. Інші аргументі є тотожними з методом `morlet()`.

Метод `к= ricker(M, s)` повертає дійсний вейвлет «мексиканський капелюх». Цілочислений аргумент M визначає довжину вейвлета. Аргумент s – масштабний коефіцієнт вейвлета виразом:

$$\psi_a(t) = \frac{2(1 - (\frac{t}{a})^2)e^{-\frac{(\frac{t}{a})^2}{2}}}{\sqrt[4]{\pi}\sqrt{3a}}$$

Для практичного застосування також можна рекомендувати окрему спеціалізовану бібліотеку **PyWavelets**.

Пряме вейвлет перетворення:

$$W_\psi(a,\tau) = \int_{-\infty}^{\infty} f(t)\psi^*_{a,\tau}(t)dt = \frac{\int_{-\infty}^{\infty} f(t)\psi^*(\frac{t-\tau}{a})dt}{\sqrt{a}}$$

Зворотне вейвлет-перетворення (формула Кальдерона [Попов]):

$$f(t) = \frac{1}{C_\psi}\int_0^{\infty}\left(\frac{\int_{-\infty}^{\infty} W_\psi(a,\tau)\psi\left(\frac{t-\tau}{a}\right)dt}{\sqrt{a}}\right)\frac{da}{a^2}dt, \text{ де } C_\psi = \int_0^{\infty}\frac{|\psi(\omega)|^2}{\omega}d\omega$$

Скалярний добуток, взаємна кореляція двох сигналів:

$$\langle a(x),\ b(x)\rangle = \int_{-\infty}^{\infty} a(x)b^*(x)dx$$

Вейвлет-перетворення є кроскореляцією сигнала та материнського вейвлета для різних параметрів масштабу та зсуву. Тобто для кожного мометну часу оцінюється взаємна кореляція сигнала та вейвлета відповідної частоти (масштабу).

Вейвлети перетворюють одновимірну функцию в двовимірну поверхню y(t)→W(t,f). W(t,f) відбиває внесок частоти f в момент часу t в сигналі.

Графічно вейвлет-спектри відображають у вигляді *скейлограм* (scalogramm) – двовимірних графіків модуля вейвлет-спектра $|W_\psi(a,\tau)|$. По одній осі відкладається час, по іншій – масштаб вейвлета.

Часткові форми скейлограми:

- *розподіл енергії за масштабом:*

$$E(a) = \frac{\int_{-\infty}^{\infty} |W_\psi(a,\tau)|^2 dt}{C_\psi}$$

Залежність показує масштаби (частоти), на яких вейвлет-складові найбільш потужно представлені в сигналі. Часова інформація втрачається.

- *розподіл енергії за часом:*

$$E(\tau) = \frac{\int_{-\infty}^{\infty} |W_\psi(a,\tau)|^2 da}{C_\psi}$$

Залежність показує розподіл енергії вейвлет-складових в часі. Частотна інформація втрачається.

Згідно з фундаментальним принципом невизначеності Гейзенберга чим точніше визначається одна характеристика, тим менш точною є друга (Time-Frequency trade-off):

$$\Delta x \Delta p \geq \frac{h}{2} \text{ або } \Delta f \Delta t \geq 1$$

В часовому просторі абсолютно точно визначений час зі сталою роздільною здатністю та відсутня інформація про частоту, в частотному просторі точно визначена частота зі сталою роздільною здатністю та відсутня інформація про час (рис. 5.1).

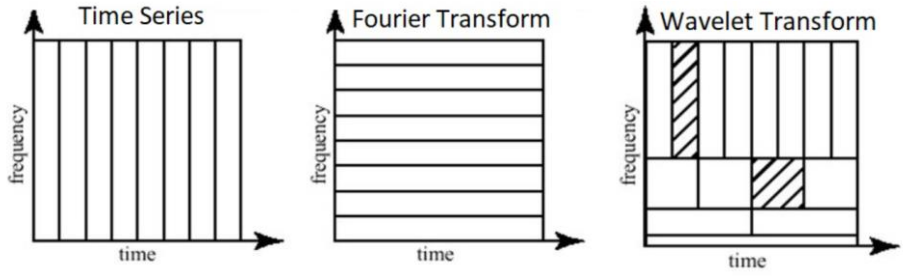


Рис. 5.1. Простори перетворень

Вейвлет-перетворення місить інформацію і про частоти, і про час одночасно. При цьому роздільна здатність є змінною в залежності від

масштабу (частоти) вейвлета. На низьких частотах маємо високе розділення по частоті та низьке по часу. На високих частотах – навпаки

Метод `к= cwt(data, wavelet, widths <, dtype=None>)` повертає двовимірний комплексний вейвлет-спектр. Аргумент `data` – одновимірний масив даних функції для перетворення. Аргумент `wavelet` визначає материнський вбудований вейвлет (`ricker, morlet2`) або вейвлет користувача. Аргумент `widths` визначає кортеж довжин вейвлета.

Перевести значення масштабу в значення псевдочастот проводиться за наступним наближеним виразом:

$$f = \frac{fo}{dt \cdot s},$$

де fo – центральна частота вейвлета. Типові значення для вейвлетів Морлі – 0.8125 Гц, Рікера – 0.252 Гц; dt – крок дискретизації сигнала [с]; s – масштаб.

Завдання до виконання

Завдання 1. Провести Фур'є (**NumPy**) та вейвлет Морлі перетворення (**SciPy**) сигналу у вигляді широтномодульованого меандру (`square()`, практикум 3). Відобразити графіки сигналу, АЧХ, ФЧХ в абсолютних координатах та скейлограми в координатах час-масштаб та час-частота.

Інтервал спостереження від 0 сек до N сек – номер варіанту. Кількість точок 512 на інтервалі. Діапазон сигналу від 0 В до 1 В. 5 імпульсів на інтервалі. Функція модуляції – додатна гармонічна (sin) від 0 до 1. Один період на інтервалі спостереження (рис. 5.2). Діапазон масштабів від 1 до 30+Nmod(5).

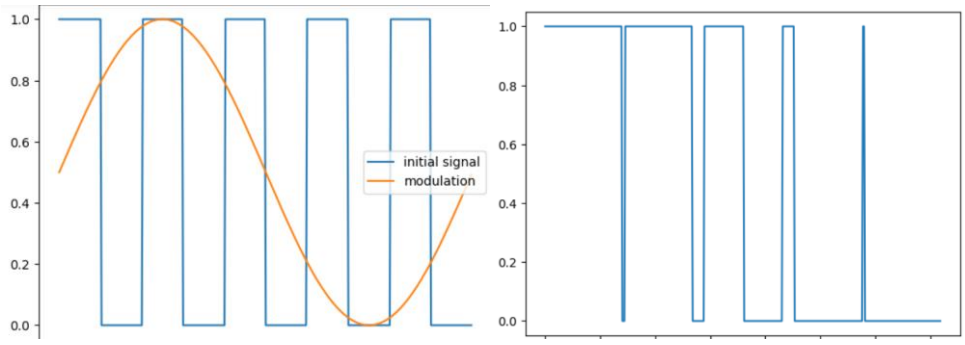


Рис. 5.2. Функція модуляції та сигнали завдання 1

ПРАКТИКУМ 6. ЦИФРОВА КОРЕЛЯЦІЯ СИГНАЛІВ

Завдання роботи: вивчення можливостей ПЗ та набуття навичок в розрахунках функцій авто та взаємної кореляції дискретних сигналів, проведенні кореляційній аналізу та узгодженої фільтрації дискретних сигналів.

Теоретичні положення

Скалярним добутком *a(x), b(x)* двох функцій називається величина:

$$\langle a(x),\ b(x)\rangle = \int_{-\infty}^{\infty} a(x)b^*(x)dx = \int_{-\infty}^{\infty} b(x)a^*(x)dx$$

Скалярний добуток – комплексний скаляр. Його можна трактувати як ступінь взаємного зв'язку однієї функції з другою. Якщо скалярний добуток дорівнює 0, говорять, що функції ортогональні.

Скалярний добуток самого сигнала повертає енергію сигнала:

$$\langle a(x),\ a(x)\rangle = \int_{-\infty}^{\infty} a(x)a^*(x)dx = E$$

Рівняння Релея: $\langle a(x),\ b(x)\rangle = \langle \widetilde{a(f)},\ \widetilde{b(f)}\rangle$

Рівність Парсеваля:

$$E = \int_{-\infty}^{\infty} a(x)a^*(x)dx = \int_{-\infty}^{\infty} |a(x)|^2 dx = \int_{-\infty}^{\infty} \widetilde{a(f)}\widetilde{a^*(f)}df = \int_{-\infty}^{\infty} |\widetilde{a(f)}|^2 df$$

Залежність $|\widetilde{a(f)}|^2$, яка є квадратом АЧХ сигнала, називають спектром потужності або спектральною щільністю енергії сигнала. Вона має одиниці «сигнал^2»· «аргумент^2» (Дж/Гц, Вт·с^2).

Спектральні щільності енергії мають сильно спадаючий за частотою характер, тому зазвичай для їхнього аналізу використовують логарифмічний масштаб в децибелах (дБ):

$$|\widetilde{a(f)}|^2 = 10\lg\left(\left|\widetilde{a(f)}\right|^2\right) = 20\lg\left(\left|\widetilde{a(f)}\right|\right), \qquad dB$$

Інтеграл **крос-кореляції** або функція **взаємної кореляції** ВКФ (cross-correlation) для неперервних сигналів зі скінченою енергією (імпульсних) (рис. 6.1):

$$Fcorr(x) = R_{12}(\mathrm{x}) = \int_{-\infty}^{\infty} F1(\alpha)F2^{*}(\alpha - x)d\alpha = \int_{-\infty}^{\infty} F1^{*}(\alpha - x)F2(\alpha)d\alpha = F1(x) * F2(x)$$

$$\tilde{F}\big(F1(x) * F2(x)\big) = \tilde{F}\big(Fcorr(x)\big) = \widetilde{F1(f)} \cdot \widetilde{F2(f)}^{*}$$

Інтеграл згортки або просто **згортка** (convolution):

$$Fconv(x) = \int_{-\infty}^{\infty} F1(\alpha)F2(x - \alpha)d\alpha = \int_{-\infty}^{\infty} F1(x - \alpha)F2(\alpha)d\alpha = F1(x) \otimes F2(x)$$

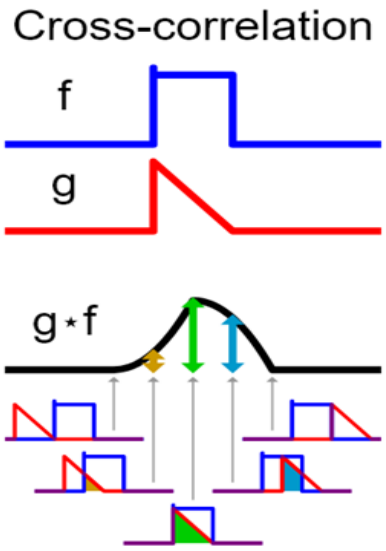


Рис. 6.1. Інтеграл кореляції (з відкритих джерел)

Інтеграл автокореляції або функція автокореляції АКФ (autocorrelation):

$$Facorr(x) = R_{1}(\mathrm{x}) = \int_{-\infty}^{\infty} F1^{*}(\alpha - x)F1(\alpha)d\alpha = \int_{-\infty}^{\infty} \left|\widetilde{F(f)}\right|^{2} \exp(j2\pi fx)\, df,$$

$$\widetilde{F(f)} \cdot \widetilde{F(f)}^{*} = |\widetilde{F(f)}|^{2} = \int_{-\infty}^{\infty} Facorr(x) \exp(-j2\pi fx)\, dx$$

Для ДІЙСНИХ каузальних імпульсних сигналів ВКФ:

$$Fcorr(x) = R_{12}(x) = \int_{0}^{x} F1(\alpha)F2(\alpha - x)d\alpha = \int_{0}^{\mathrm{x}} F1(\alpha - x)F2(\alpha)d\alpha$$

Діапазон визначення каузальної кореляційної функції для імпульсних сигналів скінченої тривалості $X_R = X_{F1} + X_{F2}$.

Розмірність кореляційної функції «сигнал^2»· «аргумент» (Дж, Вт·с).

Нормована ВКФ імпульсних сигналів:

$$B_{12}(x) = \frac{R_{12}(x)}{\int_{-\infty}^{\infty} \sqrt{F1(x)F1(x)^*}\sqrt{F2(x)F2(x)^*}dx}$$

Зв'язок між згорткою та кореляцією

$$F1(x) \otimes F2(x) = F1(x) * F2^*(-x)$$

Функція взаємної кореляції (cross-correlation) для неперервних сигналів з нескінченою енергією та скінченою потужністю (періодичних з періодом Т):

$$Rcorr(x) = \frac{1}{T}\int_0^T F1(\alpha)F2^*(\alpha - x)d\alpha = \frac{1}{T}\int_0^T F1^*(\alpha - x)F2(\alpha)d\alpha$$

Для ДІЙСНИХ періодичних сигналів ВКФ:

$$Fcorr(x) = R_{12}(x) = \frac{1}{T}\int_0^T F1(\alpha)F2(\alpha - x)d\alpha = \frac{1}{T}\int_0^T F1(\alpha - x)F2(\alpha)d\alpha$$

Розмірність кореляційної функції періодичних сигналів «сигнал^2» (Вт).

Слід відмітити розбіжності у визначеннях теорії сигналів та статистики. Описану вище функцію в статистиці називають функцією коваріації.

Нормована ВКФ періодичних сигналів:

$$B_{12}(x) = \frac{R_{12}(x)}{\frac{1}{T}\int_0^T \sqrt{F1(x)^2}\sqrt{F2(x)^2}dx}$$

Значення функцій R_{12} лежать між -∞ та +∞. Від'ємна кореляція <0 означає, шо зростання одного сигналу приводить до зменшення іншого. Додатна кореляція >0 означає, що зростання одного сигналу приводить до зростання іншого. Якщо значення ВКФ дорівнює нулю, це може значити, що зміни сигналів або не пов'язані, або механізм зв'язку між сигналами не можна виявити за допомогою ВКФ.

Значення функцій B_{12} лежать між -1 та +1. Від'ємна повна кореляція -1 означає, шо сигнали змінюються синхронно та зростання одного сигналу приводить до зменшення іншого, Додатна повна кореляція +1 означає, що сигнали змінюються абсолютно однаково. Якщо значення ВКФ дорівнює нулю, це може значити, що зміни сигналів або не пов'язані, або механізм зв'язку між сигналами не можна виявити за допомогою ВКФ.

Значення ненормованої АКФ R(0) нескінченого некаузального сигнала визначає його енергію.

Значення ненормованої АКФ R(0) періодичного некаузального сигнала визначає його середню потужність на періоді.

Вигляд результату в ЦИФРОВОМУ вигляді залежить від наданої сигналам поведінки за межами їхніх аргументів.

1. Якщо масиви сигналів F1(0...N1-1), F2(0...N2-1) МОЖУТЬ бути продовжені додаванням 0, то вони вирівнюються до розміру масиву-результату кореляції NR=N1+N2-1. Це відповідає теоретичній кореляції імпульсних сигналів. Для великих зсувів з'являється зона крайового ефекту, в якій в розрахунках зменшується кількість реально існуючих значень функцій.

2. Якщо масиви сигналів F1(0...N1-1), F2(0...N2-1) НЕ МОЖУТЬ бути продовжені додаванням 0, то розмір масиву-результату кореляції скорочується NR=max(N1,N2)-min(N1,N2)+1. Це відповідає режиму обробки зображень, коли по краях більшого за розмірами зображення вирізається рамка розмірами меншого зображення. В масиві-результаті обробляються тільки реально існуючі значення.

В цифровому вигляді для ДІЙСНИХ обмежених періодом або інтервалом спостереження сигналів нормованого часу:

$$R_{12}(i\Delta x) = R_{12}(i) = \frac{\Delta\alpha}{Tobs}\sum_{k=0}^{N-1} F1(k\Delta\alpha)F2(k\Delta\alpha - \mathrm{i}\Delta x) = [\Delta\alpha = \Delta x]$$
$$= \frac{1}{N}\sum_{k=0}^{N-1} F1(k)F2(k - \mathrm{i})$$

для ДІЙСНИХ необмежених сигналів нормованого часу:

$$R_{12}(i\Delta x) = R_{12}(i) = \Delta\alpha\sum_{k=0}^{N-1} F1(k\Delta\alpha)F2(k\Delta\alpha - \mathrm{i}\Delta x) = [\Delta\alpha = \Delta x] = \sum_{k=0}^{N-1} F1(k)F2(k - \mathrm{i})$$

В цифровому вигляді нормована ВКФ імпульсних сигналів:

$$B_{12}(i) = \frac{R_{12}(i)}{\frac{1}{N}\sqrt{\sum_{k=0}^{N-1} F1^2(k)\sum_{k=0}^{N-1} F2^2(k)}}$$

Значення цифрових ВКФ для зсуву i = 0 називають коефіцієнтами (взаємної) кореляції двох сигналів (лінійні коефіцієнти Пірсона).

В бібліотеці **NumPy** для розрахунку функції ВКФ одновимірних комплексних масивів a та b міститься метод

`res=numpy.correlate(a, b, mode=)`. Метод реалізує обчислення за виразом `res[k] = sum_n a[n+k] * conj(b[n])`. Аргумент `mode='valid'/ 'same'/ 'full'` визначає розмір масиву-результату. За замовчанням `'valid'`. `'full'` – обчислює «лінійну» кореляцію з продовженням масивів та збільшеною довжиною результату (умова 1). `'valid'` – обчислює кореляцію без продовження масивів (умова 2). `'same'` - обчислює «лінійну» кореляцію з однаковими довжинами всіх масивів.

Метод не враховує множник 1/N, тобто розраховує функцію кореляції необмежених імпульсних сигналів в нормованому часі!

Нульовий індекс (перший елемент результату) відповідає зсуву вліво на N/2-1.

В бібліотеці **SciPy** для розрахунку функції ВКФ комплексних масивів a та b містяться універсальний метод розрахунку ВКФ багатомірних масивів `correlate` та спеціалізований метод розрахунку ВКФ двовимірних масивів `correlate2d`, орієнтований на зображення.

Метод `correlate` реалізує обчислення за виразом `res[k] = sum[i] a[i] * conj(b[i - k])`.

Аргумент `mode=` є аналогічним з методом NumPy. `Method='auto'/ 'direct'/ 'fft'` – визначає спосіб обчислення. За замовчанням `'fft'` . `'direct'` – обчислення за виразами кореляції, `'fft'` – обчислення через перетворення Фур'є.

Для отримання інформації про застосовані зсуви призначено метод `correlation_lags()`, який повертає масив значень зсувів-затримок (lag) для операції кореляції одновимірних масивів. Аргументами є два числа – розміри масивів, що корелюються та режим обрахунку `Method=`.

Фур'є спектр трикутного імпульсу амплітудою 1 тривалістю 1, зсувом 0 аналітично описується наступною залежністю:

$$F(f) = \frac{e^{-j2\pi f}(1 + j2\pi f) - 1}{\pi^2 \cdot f^2 \cdot 4}$$

Енергетичний спектр трикутного імпульсу:

$$|F(f)|^2 = \frac{\pi^2 \cdot f^2 \cdot 4 - 4\pi f sin(2\pi f) + 4sin^2(\pi f)}{\pi^4 \cdot f^4 \cdot 16}$$

Ненормована автокореляційна функція трикутного сигналу:

$$R(x) = \begin{cases} \frac{x^2(3-x)}{6}, & 0 \le x < 1 \\ \frac{(x-2)^2(x+1)}{6}, & 1 < x \le 2 \end{cases}$$

Енергія сигналу:

$$\int_0^1 x^2 dx = \frac{1}{3}, \quad Rmax = R(1) = \frac{1}{3}, \quad \int_{-\infty}^{\infty} |F(f)|^2 df = \frac{1}{3}$$

Для експрес аналізу доцільно використовувати вбудовані функції **Matplotlib.Axis** розрахунку та відображення кореляційних функцій та енергетичного спектру.

Метод `<lags, corr, lineCol, line2=>Axes.acorr(x <,arg>)` повертає масив автокореляційної функції `corr`, масив зсувів `lags`, покажчики на лінії `lineCol`, `line2` та виводить лінійну `stem` діаграму НЕКАУЗАЛЬНОЇ функції. Аргументом є одновимірний масив сигналу x, для якого розраховується кореляційна функція в режимі `full`. Опційні аргументи: `normed=True/False`. За замовчанням `True`. Нормування значення функції до 1; `usevlines=True/False`. За замовчанням `True`. Керує видимістю вертикальних ліній графіка; `maxlags` – кількість зсувів в один бік для виведення графіка. За замовчанням – всі `2 * len(x) - 1`; `linestyle=` - формат вертикальних ліній; `marker=` – формат маркерів за умови вимкнення вертикальних ліній. За замовчанням замальована точка 'o'.

Метод для ВКФ відрізняється наявністю імені другого масиву в аргументах.

Метод `<psd, freq,line2=>Axes.psd(x <,arg>)` повертає масив енергетичного спектра `psd`, масив частот `freq`, покажчик на лінію `line2` та виводить графік енергетичного спектра в одиницях `10lg(PSD)` «дБ/Гц» в логарифмічному масштабі. Спектр розраховується через періодограму Велча (Welch). Аргументи є аналогічними методу амплітудного спектра цієї ж бібліотеки. Аргументом є одновимірний масив сигналу x, для якого

розраховується спектр. Опційні аргументи: Fs= – ширина спектра ($fmax = \frac{1}{dt}$). За замовчанням 2; `window=window_hanning, window_none, numpy.blackman, numpy.hamming, numpy.bartlett` – віконна функція. За замовчанням `window_hanning; sides='onesided'/ 'twosided'` – визначає проведення дійсного перетворення ‘`onesided'` чи комплексного `'twosided'`. За замовчанням `'onesided'; pad_to=` – розмір масиву для розрахунку; `scale_by_freq=True/False` – масштаб осі частот графіка. За замовчанням `'True'-` в абсолютних одиницях; `Fcint=` – центральна частота для x. За замовчанням 0. `Return_line=True/False` – наявність серед результатів на третьому місці покажчика на лінію графіка. За замовчанням `'True'; detrend='none'/ 'mean'/ 'linear'` – фільтрувальна дія над даними перед обчисленням спектра. За замовчанням `'none'. NFFT=` – кількість точок швидкого перетворення Фур’є. За замовчанням 256.

Кореляційний аналіз

Кореляційний аналіз допомагає визначити ступінь незалежності одного процесу від іншого, встановити подібність одного набору даних до іншого, виявити гармонічні складові в процесі, вилучити корисний сигнал з суміші сигнала та шуму.

Узгоджена фільтрація

Типовим завданням обробки сигналів при їхньому прийомі є виявлення корисного сигнала на тлі шумів за умов апріорних відомостей шодо характеристик корисного сигнала та завад.

$$s_{in}(t) = Ks(t - t_0) + n(t)$$

Якщо корисний сигнал $s\ (t)$ є повністю детермінованим, то фільтр повинен за час спостереження T забезпечити максимізацію відношення

$$SNR = \frac{\frac{1}{T}\int_0^T s(t)g(x-t)dt}{D_n} \to max$$

Згідно з нерівністю Коші-Буняковського імпульсний відгук фільтра, який забезпечує максимум SNR

$$g(t) = As(t_0 - t).$$

Імпульсний відгук такого фільтра повторює собою корисний сигнал, тому називається узгодженим.

Вихідний сигнал після фільтрації обчислюється через інтеграл кореляції, тому фільтр називають ще кореляційним

$$Sout(x) = \int_0^T s(t)g(x-t)dt = \int_0^T s(t)s(x-t)dt$$

Узгоджений фільтр є оптимальним для обробки адитивної суміші детермінованого корисного сигнала та білого шуму та забезпечує максимальне можливе відношення «сигнал/шум».

Кореляційний аналіз імовірнісних сигналів

Для візуального аналізу наявності та виду зв'язку між даними двох вибірок спостережень використовують кореляційне поле або поле кореляції. Кореляційне поле – це графік у вигляді сукупності точок у прямокутній системі координат. Кількість точок на полі відповідає кількості пар даних у вибірках. По осі ординат відкладаються значення однієї вибірки, вибраної в якості результативної ознаки Y, по осі абсцис – значення іншої вибірки – факторної ознаки X (рис. 6.2).

Візерунок з точок кореляційного поля вказує на наявність або відсутність зв'язку між вибірками.

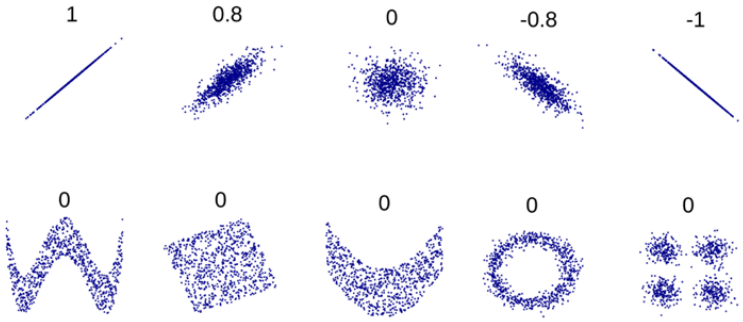


Рис. 6.2. Кореляційні поля

Для кількісного статистичного аналізу використовують статистичні моменти.

Для одновимірних залежностей жодних непорозумінь не виникає:

Початковий $E(x^к)$

	безперервний	дискретний
k=1: перший, середнє, математичне очікування	$\nu 1 = \int xp(x)dx$	$M = \frac{\sum x_i}{N}$
k=2: другий, середньоквадратичне	$\nu 2 = \int x^2 p(x)dx$	$\overline{M^2} = \frac{\sum x_i^2}{N}$

Центральний $E((x-E(x))^к$

k=1: перший	$\mu 1 = \int (x - \nu 1)p(x)dx$	$M1 = \frac{\sum(x_i - M)}{N-1}$
k=2: другий, дисперсія	$\mu 2 = \int (x - \nu 1)^2 p(x)dx$	$D = \frac{\sum(x_i - M)^2}{N-1}$

Для багатовимірних залежностей існують розбіжності в назвах змішаних моментів. Перший центральний та початковий моменти називають і коваріацією, і кореляцією. Ґрунтуючись на англомовних джерелах та назвах функцій мови Python будемо називати початковий момент кореляцією, а центральний – коваріацією.

Початковий змішаний $E(x^к)$

	безперервний	дискретний
k=1: перший	$\nu 1 = \int xyp(x,y)dx$	$Mxy = \frac{\sum x_i y_i}{N}$

Центральний $E((x-E(x))^к$

k=1: перший	$\mu 1 = \int (x - \nu 1x)(y - \nu 1y)p(x,y)dx$	$\overline{Mxy} = \frac{\sum(x_i - Mx)(y_i - My)}{N-1}$

Матриця коваріації C слугує кількісною оцінкою лінійного взаємозв'язку між двома процесами. Для двох процесів, які не мають взаємного зв'язку, коваріація дорівнює 0. Залежні процеси мають ненульові значення коваріації. Для отримання універсальної оцінки використовують нормоване значення коваріації, яке називають коефіцієнтом кореляції

$$C = \begin{bmatrix} Dxx & Dxy \\ Dyx & Dyy \end{bmatrix} \quad r_{12} = \frac{\overline{Mxy}}{\sqrt{D_x D_y}} = \begin{bmatrix} 1 & r_{12} \\ r_{21} & 1 \end{bmatrix}.$$

Значення коефіцієнта кореляції знаходиться в діапазоні від -1 до +1.

Для кореляційного порівняння послідовностей в **NumPy** призначені методи `cov`(), `corrcoef`().

Метод `mat=np.cov(x <arg>)` розраховує коваріаційну матрицю С масиву x. Опційні аргументи y – другий масив факторних ознак для розрахунку взаємної коваріації; `rowvar=True/False` – вигляд масиву x. За замовчанням `True` – пари розподілені по рядках в одному стопці, `False` – пари розподілені по стовпцях в одному рядку; `bias=True/False` – нормувальний коефіцієнт N-1 або N. За замовчанням `False` (N-1). Аналогічну дію виконує `ddof=0/1 - N-ddof`.

Примітка. Для теоретичних оцінок дисперсії, СКВ, коваріації застосовується множник N. Для статистичних оцінок виборок даних використовується множник N-1 та говорять про «частинні» (sample) дисперсію тощо.

Метод `mat=np.corrcoef(x <arg>)` розраховує матрицю нормованих в діапазон від -1 до 1 коефіцієнтів кореляції Пірсона R масиву x. Аргументи аналогічні методу `cov`.

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}} = \frac{C_{ij}}{\sqrt{D_x D_y}}$$

За відомої АКФ крок дискретизації для забезпечення потрібної дисперсії похибки можна визначити як

$$R_1(\Delta t) = \frac{D - 2R_1(0)}{2}$$

Завдання до виконання

Завдання 1. Розрахувати АКФ, енергетичний спектр, енергію/потужність засобами **NumPy**/**SciPy**. АКФ – з сигналу та через енергетичний спектр. спектр – з АКФ, з сигналу через ДПФ та спеціалізованою вбудованою функцією **Matplotlib** Відобразити графічно в абсолютних координатах сигнал, АКФ, АЧХ сигналу, енергетичний спектр для

- імпульсного каузального прямокутного сигналу з амплітудою «номер групи» Вольт, тривалістю «поточне число місяця» секунд на інтервалі спостереження від 0 до подвійній тривалості. Кількість виборок - «номер варіанту+10».

- періодичного сигналу прогальності 2 на інтервалі спостереження від 0 до шести тривалостей. Інші параметри сигналу незмінні.

Порівняти результати, зробити висновки.

Завдання 2. Визначити ступінь ортогональність на інтервалі ортогональності, рівному періоду базової частоти 1 с, двох гармонічних сигналів з частотами 2/T та 3/T з однаковими початковими фазами та амплітудами «номер варіанту» Вольт та «номер варіанту/2» Вольт. Зміною фази сигналу 3/T домогтися ортогональності сигналів. Вивести первинні та змінені сигнали, ступінь ортогональності.

Завдання 3. Визначити статистичну кореляційну залежність за значеннями коефіцієнта кореляції двох послідовностей довжиною 1000 елементів: перша x1=f(x) – відповідно завданню 8 практикуму 3, друга $x2 = kx1 + \sqrt{1-k^2}f(x), k = 0.0001, 0.5, 0.9, 1.0.$
Вивести відповідні кореляційні поля, коефіцієнти кореляції.

Завдання 4. Провести кореляційну фільтрацію узгодженим фільтром суміші гармонічного сигналу та шуму. Вивести реалізації сигналу, шуму, суміші, результату фільтрації, енергетичний спектр до та після фільтрації. Шум – з нульовим середнім, інше за умовами завдання 8 практикуму 3. Сигнал має амплітуду «вольт» та період «сек» – номер варіанту. Дослідити, як впливає інтервал спостереження на результати.

ПРАКТИКУМ 7. ЦИФРОВА ЗГОРТКА СИГНАЛІВ

Завдання роботи: вивчення можливостей ПЗ та набуття навичок в розрахунках згортки дискретних сигналів та лінійної фільтрації сигналів, моделювання фільтрів низьких та високих частот і дослідженні результатів їхнього застосування.

Теоретичні положення

Інтеграл згортки або просто згортка (convolution):

$$Fconv(x) = \int_{-\infty}^{\infty} F1(\alpha)F2(x-\alpha)d\alpha = \int_{0}^{x} F1(x-\alpha)F2(\alpha)d\alpha = F1(x) \otimes F2(x)$$

$$\tilde{F}(F1(x) \otimes F2(x)) = \widetilde{F1(f)} \cdot \widetilde{F2(f)}$$

Для періодичних сигналів $F_T(x)$ періодом T:

$$Fconv(x) = \frac{1}{T}\int_{0}^{T} F_T(\alpha)F2(x-\alpha)d\alpha$$

Властивості згортки.

Комутативність F1(t) ⊗ F2(t) = F2(t)⊗ F1(t)

Дистрибутивність F1(t) ⊗ (F2(t) + F3(t)) = F1(t) ⊗ F2(t) + F1(t) ⊗ F3(t)

Асоціативність F1(t) ⊗ (F2(t) ⊗ F3(t)) = (F1(t) ⊗ F2(t)) ⊗ F3(t)

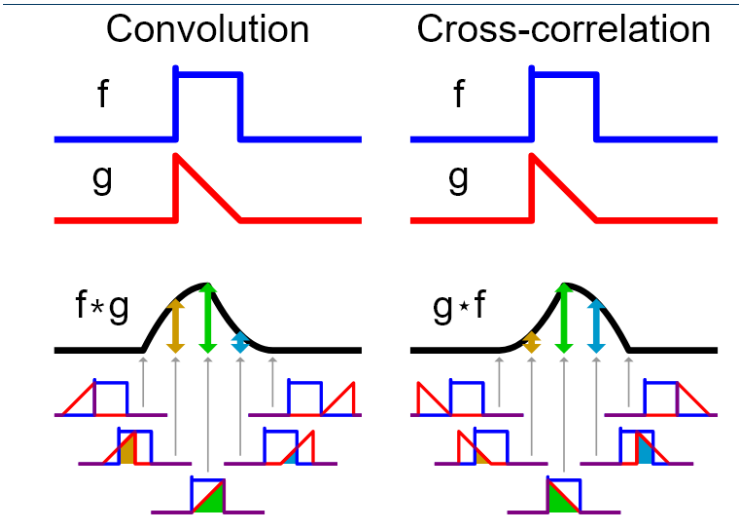


Рис. 7.1. Порівняння кореляції та згортки (з відкритих джерел)

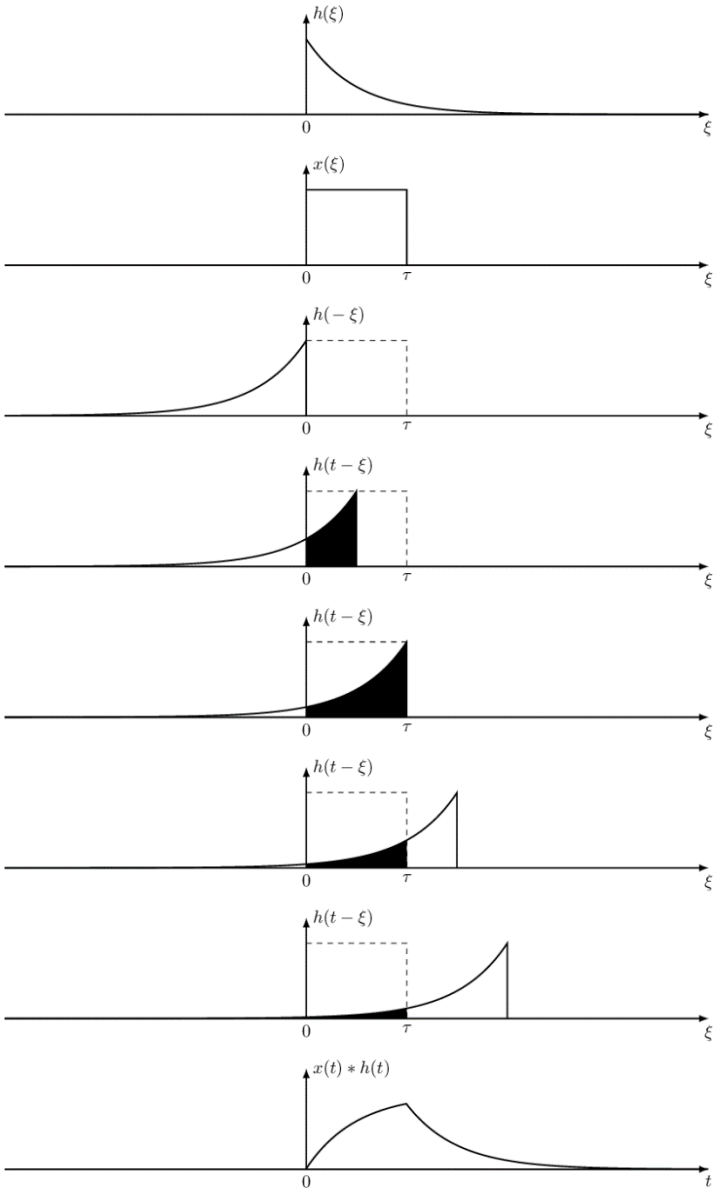


Рис. 7.2. Операція згортки

Згортка описує як вихід системи визначається взаємодією входу з самою імпульсним відгуком системою.

В дискретному вигляді для нескінчених послідовностей:

$$Fconv(i\Delta x) = \int_{-\infty}^{\infty} F1(\alpha)F2(x-\alpha)d\alpha \approx \Delta\alpha \sum_{k=-\infty}^{\infty} F1(k\Delta\alpha)F2(i\Delta x - k\Delta\alpha)$$

$$Fconv_i = \Delta\alpha \sum_{k=-\infty}^{\infty} F1(k)F2(i-k) \sim \sum_{k=-\infty}^{\infty} F1(k)F2(i-k), \qquad \Delta x = \Delta\alpha$$

Лінійна згортка дискретних масивів

Для скінчених масивів *F1(n), n=0…M-1, F2(n), n=0…N-1* розміром M та N відповідно результуючий масив лінійної згортки матиме розмір *N+M-1*.

$$Fconv_i = \sum_{k=0}^{M-1} F1(k)F2(i-k), \qquad i = 0..M+N-1, \qquad 0 < i-k < N-1$$

Для обчислення дискретної згортки *Fconv* зазвичай масив однієї з функцій F1 (довшої) розташовується за зростанням індексів. Масив другої функції F2 (коротшої) розвертається у зворотному порядку за зменшенням індексів. Результат дорівнює сумі добутків значень першої функції на відповідні значення оберненої другої функції.

Вихідний сигнал можна розглядати як суму добутків значень імпульсної характеристики СІВ фільтра з відповідними значеннями оберненого вхідного сигнала.

В бібліотеці **NumPy** для розрахунку дискретної згортки одновимірних масивів a та b міститься метод `res=np.convolve(a, b<, mode=>)`. Аргумент `mode='valid'/ 'same'/ 'full'` визначає розмір масиву-результату. За замовчанням `'full'`. `'full'` – обчислює лінійну згортку зі збільшеною довжиною результату. `'valid'` – обчислює згортку без продовження масивів (довжина масиву-результату: `max(N,M)-min(N,M)+1`). `'same'` - обчислює згортку з однаковими довжинами всіх масивів `max(N,M)`.

Метод НЕ ВРАХОВУЄ множник dα, тобто розраховує функцію згортки необмежених імпульсних сигналів в нормованому часі!

Циклічна згортка (circular convolution)

Циклічною згорткою двох масивів ОДНАКОВОЇ довжини є масив такої ж довжини.

$$FconvC_i = \sum_{k=0}^{N-1} F1(k)F2(i-k) mod\ N, \qquad i,k = 0..N-1$$

Позначення $(\bullet)_{\bmod N}$ означає, що різниця (i-k) береться по модулю N, тобто береться залишок від ділення (i-k) на N:

$$(i-k)mod\ N = \begin{cases} (i-k)mod\ N, & i \geq k \\ N - \big((k-i)mod\ N\big), & i < k \end{cases}$$

Наприклад, для N=4 масив довжиною 8 повинен формуватися наступним чином. Перші 4 елементи з індексами 0…3 залишаються незмінними. На місце п'ятого елемента ставиться елемент з індексом (4)mod 4=0. На місце шостого – з індексом (5)mod 4=1, сьомого – 2, восьмого – 3.

Розрахунок по модулю означає, що значень масиву з індексами більше за N беруться переносом на їхнє місце передніх значень зі зсувом на модуль N.

Алгоритм циклічної згортки застосовується для розрахунку згортки періодичних функцій з періодом N.

Для цифрових сигналів саме між циклічною згорткою та ДПФ виконується рівняння

$$FFT(F1 \otimes F2) = FFT(F1) \cdot FFT(F2)$$

$$FconvC = iFFT(FFT(F1) \cdot FFT(F2))$$

Розрахунок лінійної згортки через циклічну

Лінійна дискретна згортка має довжину N+M-1. За визначенням довжини всіх масивів циклічної згортки мають бути однаковими.

Щоб застосувати властивості ДПФ слід вирівняти розміри всіх масивів до значення N+M-1. Масиви функцій F1 та F2 слід справа доповнити нулями. Для цього можна застосувати функції доповнення масиву `np.append()` або поєднання масивів `np.concatenate()` . Масив F1 довжиною M доповнюється N-1 нулями. Масив F2 довжиною N доповнюється M-1 нулями.

В бібліотеці **SciPy** для розрахунку згортки містяться декілька методів: `convolve`() – розраховує дискретну згортку N-вимірних масивів; `fftconvolve`() – розраховує дискретну згортку N-вимірних масивів методом Фур'є перетворення; `oaconvolve`() – розраховує згортку методом перекриття; `convolve2d`() – метод розрахунку згортки двовимірних масивів; `sepfir2d`() – метод розрахунку згортки сигнала та двовимірного СІВ фільтру; `choose_conv_method`() – метод пошуку ефективного способу розрахунку згортки.

Метод `res=sc.signal.fftconvolve (F1, F2<, mode=, axes=>)` проводить згортку багатовимірних масивів F1, F2 перетворенням Фур'є з опціонально визначеними режимом `mode='valid'/ 'same'/ 'full'` (за замовчанням `'full')` та віссю `axes=0/1/…`

Метод `res=sc.signal.convolve(F1, F2<, mode=, method=>)` проводить згортку багатовимірних масивів F1, F2 опціонально визначеним режимом `method='auto'/'direct'/'fft'` (за замовчанням `'auto')`та `mode='valid'/ 'same'/ 'full'` (за замовчанням `'full')`.

Метод пропонує найшвидший метод режим згортки. Точність оцінюється як 99% для двовимірних масивів та 85% для одновимірних.

Ідентифікація систем, обернення згортки

Терміном ідентифікація позначають визначення однієї з функцій (імпульсної характеристики h(n) або сигналу x(n)) за відомими результатом згортки та другої функції.

$$h_0 = \frac{y_0}{x_0}, \qquad h_n = \frac{y_n - \sum_{m=0}^{n-1} h_m x_{n-m}}{x_0}, \qquad n \geq 1, x_0 \neq 0$$

$$x_0 = \frac{y_0}{h_0}, \qquad x_n = \frac{y_n - \sum_{m=0}^{n-1} h_m x_{n-m}}{h_0}, \qquad n \geq 1, h_0 \neq 0$$

Зв'язок між згорткою та кореляцією.

Згортка еквівалентна функції взаємної кореляції двох сигналів, в якій одна з початкових послідовностей обернена у часі. Це означає, що згортку і кореляцію можна обчислювати за одним і тим самим алгоритмом, просто обертаючи у часі одну з двох послідовностей.

Завдання до виконання

Завдання 1. Провести лінійну фільтрацію імпульсного сигнала відповідно варіанту з п. 4 практикуму 3 СІВ НЧ фільтром порядку 10+N mod(5) як вікон з бібліотеки **windows SciPy**.

а) Барлета (bartlett) б) Блекмана (blackmann) в) Хамінга (hamming) г)Хана (hann)

дискретною лінійною згорткою та Фур'є перетворенням безпосередньо.

засобами **NumPy** та **SciPy**. Побудувати та порівняти графіки сигнала (на інтервалі не менше 3М), імпульсного відгуку фільтра, згортки та їхніх амплітудних спектрів в абсолютних координатах.

Завдання 2. Виконати п. 1 для сигнала як періодичного зі скважністю 2 циклічною згорткою та Фур'є перетворенням. Порівняти результати п.1 та п.2.

Література

1. Цифрова обробка сигналів та зображень. Рекомендації до виконання розрахунково-графічної роботи : навчальний посібник для здобувачів ступеня магістра за освітньою програмою «Комп’ютерно-інтегровані системи та технології в приладобудуванні» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» / І. В. Кравченко, М. С. Мамута ; КПІ ім. Ігоря Сікорського. - Київ : КПІ ім. Ігоря Сікорського, 2023. - 90 с.
2. Цифрова обробка сигналів та зображень : навч. посіб. для здобувачів ступеня магістра за освіт. програмою «Комп’ютерно-інтегровані системи та технології в приладобудуванні» спец. 174 Автоматизація, комп’ютерно-інтегровані технології та робототехніка / КПІ ім. Ігоря Сікорського ; уклад.: В. І. Микитенко, Г. С. Тимчик. – Київ : КПІ ім. Ігоря Сікорського, 2025. – 181 с.
3. NumPy reference. Release: 2.2, Режим доступу: https://numpy.org/doc/stable/reference/index.html
4. John Hunter, Darren Dale, Eric Firing, Michael Droettboom, Matplotlib Release 3.1.1. Режим доступу: https://matplotlib.org/3.1.1/Matplotlib.pdf
5. SciPy Reference Guide. Release 1.7.1, Режим доступу: https://docs.scipy.org/doc/scipy/
6. Allen B. Downey, Stranneby Think DSP: Digital Signal Processing in Python. – O'Reilly Media. – 2016. – 165 p.
7. Andreas Antoniou, Digital Signal Processing. – McGraw-Hill. – 2006. – 991 p.
8. Jonathan Blackledge, Digital Signal Processing. Second Edition. – Horwood Publishing. – 2006. – 840 p.
9. Dag Stranneby, Digital Signal Processing. DSP and Application. – Butterworth-Heinemann. – 2001. – 239 p.