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

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ ТА СИСТЕМ

ПРАКТИКУМ

Для студентів спеціальністі 151 «Автоматизація та комп'ютерно-інтегровані технології» освітньої програми "Комп'ютерно інтегровані системи та технології в приладобудуванні"

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензенти: *Аврутов Вадим Вікторович, доктор техн. наук, проф*
Поздняков Дмитро Вікторович, канд. техн. наук

Відповідальний редактор *Колобродов Валентин Георгійович, доктор техн. наук, проф.*

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол №х від хх.хх.хххх р.) за поданням Вченої ради приладобудівного факультету (протокол № х/хх від хх.хх.хххх р.)

Електронне мережне навчальне видання

Кравченко Ігор Володимирович

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ ТА СИСТЕМ

ПРАКТИКУМ

Математичне моделювання процесів та систем: Практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності «Автоматизація та комп'ютерно-інтегровані технології» / Уклад. І. В. Кравченко; КПІ ім. Ігоря Сікорського . – Електронні текстові дані (1 файл: 4,35 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 133с.

Розглянуто технологію, методики, особливості застосування системи комп'ютерної математики (СКМ) «MATLAB» для виконання технічних розрахунків та набуття практичних навичок в моделюванні автоматизованих систем. Містить теоретичний матеріал, приклади та комплекс завдань для комп'ютерного практикуму та самостійної роботи. Розглянуто питання проведення коммп'ютених експериментів із використанням імітаційних моделей, чисельного спектрального аналізу, методи обробки цифрових зображень, оптимізації, моделювання динамічних систем Може бути корисний студентам та фахівцям технічних спеціальностей для набуття навичок застосування інформаційних технологій у вигляді сучасних СКМ в учбовій та науково-технічній практиці.

Зміст

Вступ

Практичні заняття з дисципліни "Математичне моделювання процесів та систем систем" є формою практичної роботи студентів та проводиться згідно учбового плану підготовки студентів освітньої програми "Комп'ютерно-інтегровані системи та технології в приладобудуванні" кваліфікаційного рівня «магістр».

Мета практикума – здобуття студентами навичок у застосуванні комп'ютерного пакета Матлаб для розв’язання обчислювальних завдань та моделювання функціонування приладів із використанням чисельних методів та комп’ютерної графіки.

Завдання практикума:

- закріплення теоретичних знань щодо інтерпретації, оцінки похибок результатів, застосування спектральних, вірогідносних, оптимізаційних методів для моделювання характеристик приладів;
- набуття навичок у розробці середовища та проведенні компютерного експеримента в СКМ Матлаб.

Практикум виконується студентами індивідуально у вигляді прикладів та завдань, що наведені в даному посібику.

Оцінювання роботи студентів проводиться відповідно до вимог силабусу з дисципліни "Математичне моделювання процесів та систем".

Загальні відомості

СКМ Матлаб широко поширена в інженерних та університетських колах. Назва Матлаб є скороченням від "Matrix Laboratory". В СКМ Матлаб реалізовані класичні чисельні алгоритми розв'язання рівнянь, задач лінійної алгебри, знаходження значень визначених інтегралів, апроксимації, розвязання систем, окремих дифференціальних рівнянь тощо.

Система відрізняється наявністю великої кількості розширень і доповнень, які дозволяють використовувати її не тільки як розрахунковий математичний пакет, а й застосовувати для проведення вимірювань, обробки результатів, розвязання спеціалізованих інженерних задач моделювання . Спектр завдань, дослідження яких може бути здійснено за допомогою СКМ Матлаб і його розширень «Toolbox», охоплює: матричний аналіз, обробку сигналів та зображень, задачі математичної фізики, оптимізаційні задачі, фінансові завдання, роботу з картографічними зображеннями, нейронні мережі, нечітку логіку і багато іншого. Близько сорока спеціалізованих Toolbox можуть бути вибірково встановлені разом з Матлабом.

Наявність опціонального модуля Simulink дозволяє проводити візуальне моделювання частотних і часових характеристик на системному рівні. Спеціалізовані розширення модуля Simulink доповнюють вбудовані засоби, наприклад:

- Simscape (раніше SimPowerSystems/Fixed-Point Blockset, SimMechanics) призначено для моделювання електроенергетичних, механічних систем;
- Real-Time Windows Target – призначено для проведення експериментів із введенням даних через пристрої введення/виведення;
- Stateflow – призначено для моделювання кінцевих автоматів;
- Aerospace Blockset – бібліотека елементів аерокосмічних систем;
- Embedded Target for the TI TMS320C6000TM DSP Platform, MATLAB Link for Code Composer StudioTM – бібліотеки компонентів контроллерів Texas InstrumentsTM;
- CDMA Reference Blockset – бібліотека елементів бездротового зв'язку стандарту CDMA IS-95A;
- DSP Blockset – бібліотека елементів систем цифрової обробки сигналів;
- Communications Blockset – бібліотека елементів систем зв'язку тощо.

Роботи комп'ютерного практикуму орієнтовані на застосування пакету версії 8. Можливо використання будь яких версій не нижче 6. Можливі розбіжності стосуються групування дій в пунктах меню та вигляді вікон пакета.

Практикум 1. Цифрові спектральні моделі

Мета заняття

Метою заняття є вивчення принципів та особливостей методів цифрових спектральних перетворень системи Матлаб. Набуття навичок в підготовці даних, оформленні та аналізі результатів перетворення Фур'є.

Теоретичні положення

Для вивчення особливостей цифрового спектрального аналізу засобами системи Матлаб в якості базового сигналу використаємо сигнал типа "прямокутник".

$$\mathbf{rect(U,\tau,\tau 0,x)} = \begin{cases} \mathbf{U,\ \tau 0 - \frac{\tau}{2} < x < \tau 0 + \frac{\tau}{2}} \\ \mathbf{0,\ \tau 0 - \frac{\tau}{2} \geq x \geq \tau 0 + \frac{\tau}{2}} \end{cases} , \qquad (1.1)$$

де U – амплітуда імпульсу; τ – тривалість імпульсу; τо – зсув імпульсу; x – поточна координата.

Результат безперервного перетворення (Фур'є спектр) прямокутного імпульсу аналітично описується наступною залежністю:

$$F(\nu) = U \cdot \tau \cdot e^{-j \cdot 2 \cdot \pi \cdot \nu \cdot \tau o} \cdot \frac{\sin(\pi \cdot \nu \cdot \tau)}{\pi \cdot \nu \cdot \tau} , \qquad (1.2)$$

де ν – частота, U- амплітуда, τ – тривалість, τ_0 – зсув.

Фазова характеристика сигналу описується виразом (1.3), амплітудна – (1.4).

$$\phi(\nu) = arctg(\frac{\mathrm{Im}(F(\nu))}{\mathrm{Re}(F(\nu))}) \qquad (1.3)$$

$$\phi(\nu) = |F(\nu)| \qquad (1.4)$$

В загальному випадку дискретне перетворення Фур'є розраховується згідно виразу (1.5). Дії проводяться над масивом вихідних даних f . Результат зберігається в масиві F. Довжини масивів повинні співпадати**.** Вираз (1.5) враховує, що початковий індекс масивів дорівнює 1.

$$F_k = dx \cdot \sum_{i=1}^{N} f_i \cdot \exp(-\frac{2 \cdot \pi \cdot j \cdot (k-1) \cdot (i-1)}{N}) , \qquad (1.5)$$

де F_k – масив результатів спектральної щільності; dx – крок зміни аргумента x (відстань між виборками); f_i – опис вихідної просторово-часової функціїї;

N – довжина масивів F, f (чисельний аналог інтервала T існування вихідної функції).

Для виконання швидкого Фур'є перетворення в пакет вбудована функція **FFT(X,<N>),**

де X – масив вихідних даних; <N> – опціональне значення кількості точок перетворення. Якщо опція не задана, пакет додає/обрізає вихідні дані до значення 2^K.

Розрахункова формула функції:

$$X(k) = \sum_{n=1}^{N} x(n) \cdot \exp\left(\frac{-j \cdot 2 \cdot \pi \cdot (k-1) \cdot (n-1)}{N}\right)$$

$$1 <= k <= N \qquad (1.6)$$

Завдання 1.1.

Описати функцію розрахунку аналітичного перетворення Фур'є «прямокутного» імпульсу (1.2). Аргументи функції: тривалість імпульсу – τ, зсув імпульсу – τ_0, амплітуда імпульсу – U, поточна частота – ν. Результат: аналітичне значення спектральної щільності (Фур'є спектра) в заданій точці. Розробити скрипт - файл для побудови графіків дійсної та уявної частин спектра, амплітудної та фазової характеристики для U =2, t_0 =0, τ =1 в діапазоні ν=0 ... $к \cdot \nu_м$, к=5. Значення граничної частоти (частоти Котельникова - Найквіста) $\nu_м = 1/2\tau$ (рис.1.1). Для розрахунку фазової характеристики застосувати вираз (1.3).

Пояснення.

Фазова характеристика (фазовий спектр) «прямокутного» сигналу описується виразом (1.7).

$$\phi(\nu) = \mathrm{arctg}\left(\frac{U \cdot \tau \cdot \mathrm{Sinc}(\nu \cdot \tau) \cdot \sin(2 \cdot \pi \cdot \nu \cdot \tau_0)}{U \cdot \tau \cdot \mathrm{Sinc}(\nu \cdot \tau) \cdot \cos(2 \cdot \pi \cdot \nu \cdot \tau_0)}\right) =$$

$$= \mathrm{arctg}(\mathrm{tg}(2 \cdot \pi \cdot \nu \cdot \tau_0)) == 2 \cdot \pi \cdot \nu \cdot \tau_0 + n \cdot \pi \qquad (1.7)$$

Вираз (1.7) є результатом аналітичних перетворень виразу (1.3) для «прямокутного» сигналу. Перший лінійний додаток відповідає експоненціальному співмножнику в виразі (1.2), другий доданок враховує зміну знаку спектру F(ν). Амплітудний спектр є знаконезмінним, він має 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.

Безпосереднє використання будь якої вбудованої функції Матлабу (angle, atan) не компенсує вищенаведеної особливості та потребує додаткових дій.

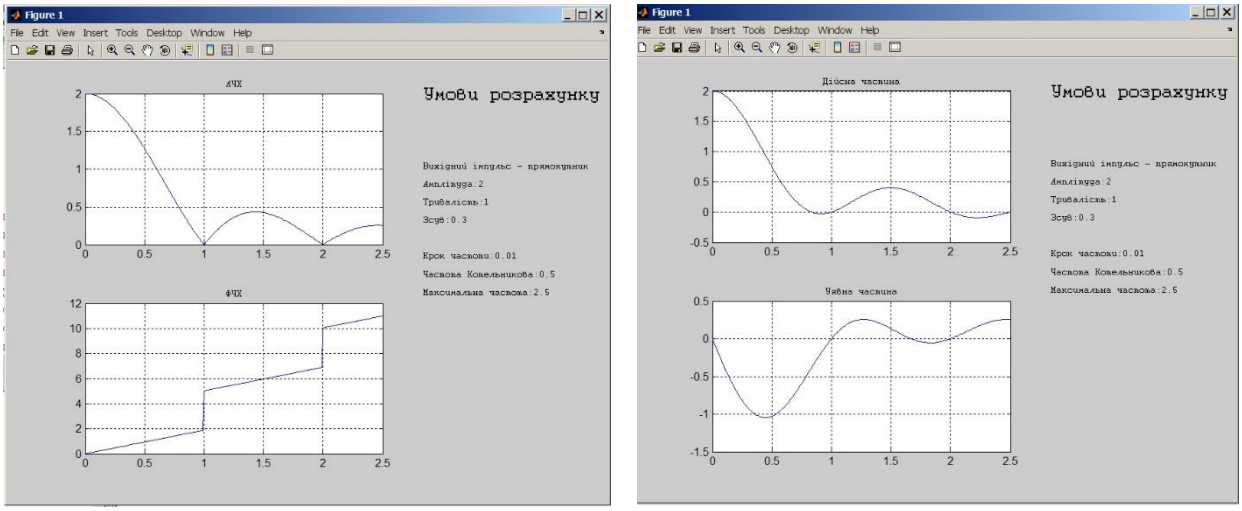


Рис.1.3 - Вигляд екранів спектрів «прямокутного» сигналу

Завдання 1.2.

а) Розробити функцію дискретного перетворення Фур'є (1.5). Вихідні дані: вектор сигналу, довжиною N, результат – комплексний дискретний спектр: вектор розміром N. Порівняти графіки розробленої функції, вбудованої функції та аналітичного розрахунку (див. завдання 1.1) для «прямокутного» сигналу.

б) Дослідити швидкість обчислення спектру ДПФ за допомогою вбудованої функції БПФ при 1000-кратному розрахунку вихідного «прямокутного» сигналу розміром 2048 та 2040 точок.

Практикум 2. Імовірнісні моделі

Мета заняття

Метою заняття є вивчення принципів побудови імовірнісних моделей, можливостей вбудованих та інтегрованих діалогових функцій Матлабу та бібліотеки Statistics Toolbox. Набуття навичок аналізу імовірнісних та експериментальних даних, генерації імовірнісних величин із завданим розподілом.

Теоретичні відомості

Імовірність дискретного явища в теорії імовірності визначається як відношення частоти успішних подій до числа всіх можливих результатів. Для безперервних подій така імовірність дорівнює нулю. В якості характеристик безперервних стохастичних процесів використовуються функція імовірності та функція щільності розподілу імовірності (функція щільності імовірності, функція розподілу імовірності).

Функція імовірності та функція щільності імовірності пов'язані між собою співвідношенням (1.1).

$$F(x) = P(X \leq x) = \int_{-\propto}^{x} f(t)dt, \qquad (2.1)$$

де *F(x)* – функція імовірності; *f(t)* – функція щільності імовірності; *P* – імовірність того, що можливі значення X менше визначеного порогу x.

Зворотна функція імовірності визначає «квантиль», тобто поріг, який забезпечує завдане значення функції імовірності. Наприклад. Якщо значення функції розподілу для рівня 0.1 є 0.539, то зворотна функція імовірності для аргументу 0.539 має значення 0.1.

Нижче наведено приклади найбільш розповсюджених законів розподілу ймовірностей.

Експоненціальний розподіл

Є частковим випадком гамма розподілу при a = 1.

Функція щільності розподілу з параметром розподілу μ (середнє значення 1/ μ) визначається виразом $y = f(x, \mu) = \frac{1}{\mu} e^{-\frac{x}{\mu}}$

Нормальний розподіл

Є аналогом біноміального розподілу для безперервних функцій.

Параметри розподілу: μ – середнє значення, σ – середнє квадратичне відхилення (СКО).

Функція щільності розподілу визначається виразом

$$y = f(x, \mu, \sigma) = \frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

Логнормальний розподіл

Якщо x має нормальний розподіл з параметрами μ и σ, то ln(X) має нормальний розподіл з такими ж параметрами.

Функція щільності розподілу визначається виразом

$$y = f(x, \mu, \sigma) = \frac{e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}}{x\sigma\sqrt{2\pi}}$$

Розподіл Пуассона

Є дискретним аналогом експоненціального розподілу.

Функція щільності розподілу визначається виразом

$$y = f(x, \lambda) = \frac{\lambda^x}{x!} e^{-\lambda} I_{(0,1,\dots)}(x)$$

Параметр розподілу: λ – середнє.

Розподіл Стьюдента

Функція щільності розподілу визначається виразом

$$y = f(x, \nu) = \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2})} \frac{1}{\sqrt{\nu\pi}} \frac{1}{(1+\frac{x^2}{\nu})^{\frac{\nu+1}{2}}},$$

де $\Gamma(\cdot)$ – гамма функція.

Параметри розподілу: ν – кількість ступенів свободи. При збільшенні ν розподіл Стьюдента наближується до нормального.

В стандартному комплекті Матлаб для аналізу імовірнісних процесів мається деяка кількість вбудованих функцій.

Функція **rand(<n><,m><,'state',k'<,type>)** генерує імовірнісні числа з рівномірним розподілом в діапазоні (0..1). Числа є псевдо імовірнісними послідовності генеруються детермінованим алгоритмом та повторюються при наступних викликах з незмінними параметрами. Число **n** визначає розмір квадратної матриці результату, пара чисел **n,m** – розміри матриці результату, За потреби параметр-рядок **type** може визначити тип значень результатц – **'single'/'double'**.

Для генерування випадкового числа зі зменшеною імовірністю повторів можна зв'язати стан генератора з вбудованим годинником комп'ютера з використанням наступних конструкцій через параметр **'state'.**

»**rand('state',sum(100*clock**)),

або

»**rand ('state', time(6)**)

При визначенні параметра **state** у вигляді числа рекомендується використовувати непарні числа в діапазоні 10000-20000.

В версіях СКМ після 7 рекомендується керувати станом генератора функцією **rng**.

Функція **RANDN** генерує числа з нормальним розподілом. Аргументи функції аналогічні аргументам функції **RAND**.

SPRANDN, SPRAND, SPRANDSYM – функції генерування матриць з рівномірним та нормальним розподілом, відповідно.

Для виведення графічних результатів імовірнісного моделювання можна застосовувати функції **errorbar** для малювання графіка з граничними відхиленнями та **hist** для малювання гістограми.

Розширення **Statistics Toolbox** (пакет статистичних обчислень) розширює можливості для роботи з імовірнісними величинами. Він являє собою набір програм для виконання статистичних розрахунків в рамках СКМ Матлаб. Пакет орієнтовано на широке коло задач: від генерації імовірних чисел та підбору кривих експериментальних даних до планування експериментів та задач промислового статистичного контролю. Інструментальні засоби пакета дозволяють використовувати його як систему команд в режимі командного рядка, так і як набір графічних інтерактивних

програм з діалоговим інтерфейсом користувача. **Statistics Toolbox** вміщує понад 200 функцій. Деякі з них представлені в табл. 2.1).

В бібліотеці реалізовано наступні типи розподілів ймовірностей: бета; хи-квадрат, біноміальне, експоненціальне, рівномірне, гамма розподіл, геометричне, логнормальне, гіпергеометричне, нормальне, від'ємне геометричне, Релея, Пуассона, Вейбула, Фішера (f розподіл), Стьюдента (t розподіл) тощо.

Імовірнісні функції різних розподілів однакові «групові» назви, які відрізняються префіксом – назвою закону розподілу:

- функція щільності імовірності (probability density function, *pdf) ;
- інтегральна функція розподілу (cumulative distribution function, *cdf);
- функція, зворотна до інтегральної функції розподілу (*inv);
- функція генерації випадкових чисел із завданим законом розподілу (*rnd);
- функції оцінки середнього значення та дисперсії (*stat);
- функції оцінки параметрів закону розподілу (*fit, *like).

Функції щільності ймовірностей повертають значення функції щільності розподілу імовірності.

Функції ймовірностей повертають значення, які пов'язані з обраним законом розподілу імовірності. Імена та синтаксис функцій цієї підгрупи ідентичні іменам функцій щільності розподілу та відрізняються тільки закінченням "**cdf**".

Зворотні інтегральні функції. повертають квантиль розподілу, тобто значення аргументу функції вірогідності, який забезпечує завдане її значення. Ім'я функції утворюється з імені закону розподілу та закінчення "**inv**".

Функції генерації випадкових чисел повертають імовірнісні числа з завданим законом розподілу. Ім'я функції створено шляхом злиття перших букв назви закону та закінчення "**rnd**".

Опис імовірнісних функцій наведено в додатку А.

Таблиця 2.1. Групи функцій статистичного аналізу **Statistics Toolbox**

Розподіли ймовірностей	Для кожного закону розподілу: функції щільності імовірності, інтегральні функції розподілу, зворотні інтегральні функції розподілу, функції генерації імовірнісних чисел із завданим законом розподілу, функції оцінки середнього значення та дисперсії, функції оцінки параметрів закону розподілу.
Дескрипторна статистика	Функції кореляційного аналізу, статистичного аналізу даних.
Виробничий контроль	Функції статистичної обробки даних.
Лінійне моделювання	Функції одно- та двофакторного дисперсійного аналізу та лінійного по параметрах моделі регресійного анализу.
Нелінійна регресія	Функції регресійного анализу в нелінійному наближені.
Статистичні графіки, функції планування, функції екстраполяції, функції файлового обміну, функції кластерного аналізу тощо.	

Функції оцінок функції розподілу.

Функція **MEAN** обчислення середнього, **STD** – обчислення СКВ. **VAR** – обчислення дисперсії.

Імена функцій, що повертають математичне очікування та дисперсію розподілу в залежності від значення параметрів закону, створюються шляхом злиття назви закону на закінчення "**stat**". Результатом функцій є вектор, який складається з двох елементів: значень середнього та дисперсії. Наприклад: **[M, V] = *stat(A,B).**

Функції визначення параметрів закону розподілу повертають оцінки параметрів законів розподілу за експериментальними даними. Імена функцій мають закінчення «**fit**». Аргументи: вектор (матриця) даних, опціонально – відносна похибка визначення. Без завдання похибки приймається оцінка 0.05 (95%). Результат: вектор параметрів.

Наприклад, функція **[m,u] = expfit(r,0.01)** оцінює параметри експоненціального розподілу даних з вектора **r** з точністю 99%.

Функції статистичних графіків дозволяють графічно відображати різноманітні імовірнісні характеристики. Нижче наведено деякі з таких функцій.

Функція **qqplot(x<,y>)** будує графік порівняння розподілів даних з вкторів **x, y**. При співпадінні розподілів дані будуть лежати на одній прямій. На графіку хрестиками позначені точки, які відповідають однаковим процентілям обох вибірок. Суцільною лінією з'єднані 25-відсоткові та 75-відсоткові процентілі. Якщо другий аргумент не задано, порівняння проходить з нормальним розподілом.

Функція **[h,stats]=cdfplot(X)** будує графік функції вірогідності по даним з вектора X. Функція повертає наступні значення:

h – покажчик на графіка,

stats – структура даних, яка містить

stats.min	мінімальне значення
stats.max	максимальне значення
stats.mean	середнє
stats.median	медіана
stats.std	СКВ

Ітеративна графічна команда **disttool** призначена для вивчення впливу параметрів розподілу на інтегральну функцію розподілу та функцію щільності розподілу вбудованих законів (рис. 2.1).

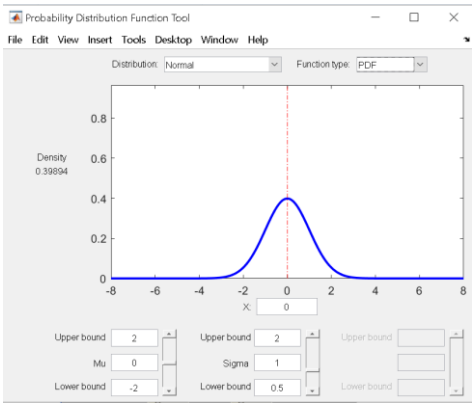


Рис. 2.1 – Вікно функції **disttool**

Ітеративна графічна команда **`randtool('output')`** призначена для генерації змінних з визначеним розподілом. Функція повертає вектор імовірнісних змінних (рис. 2.2).

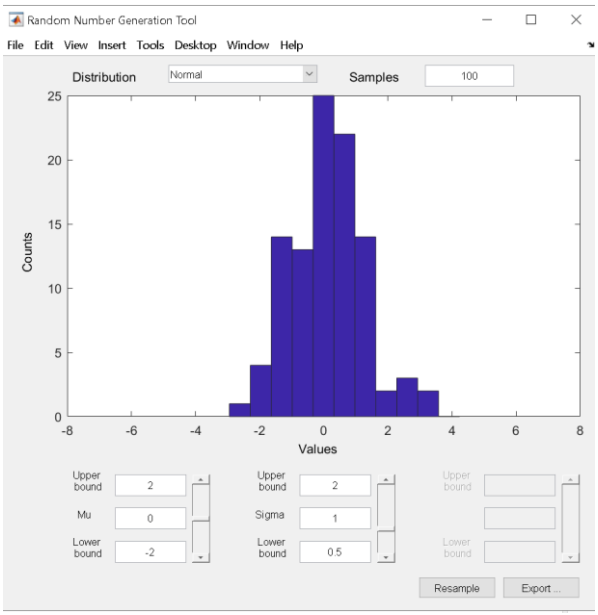


Рис. 2.2 – Вікно команди **`randtool`**.

Інтерактивна графічна команда **`cftool`** призначена для регресійного аналізу (рис. 2.3).

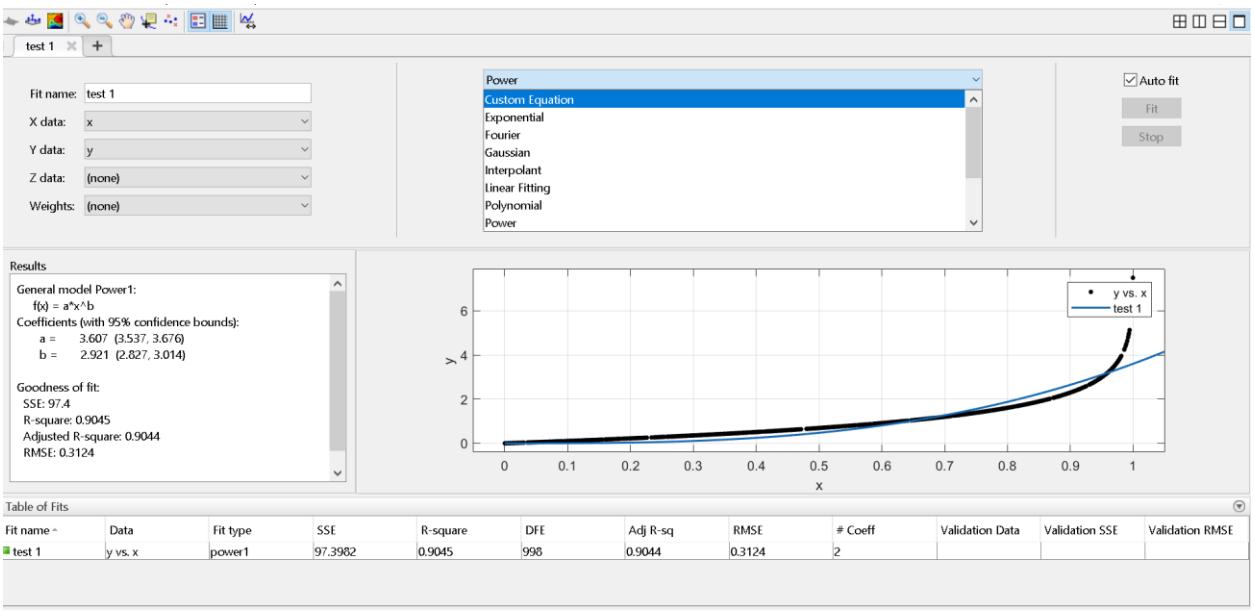


Рис. 2.3. – Вікно команди **cftool**

Команда оперує з даними з **`workspace`**, тобто з даними, які попередньо визначені розрахунками в командному вікні або скрипт-файлами.

Дані для аналізу обираються в списках **`x/y/z-data`** в лівому верхньому квадранті вікна.

Тип наближення обирається зі списку в верхній уентральній частині вікна. Доступні наступні залежності: Поліноміальна до 9-го степеню (**Polynomial**), Екпоненційна (**Exponential**), рядом Фур'є (**Fourier**), гауссоідою (**Gaussian**), степенева (**Power**), раціональним поліномом (**Rational**), сумою тригонометричних функцій (**Sum of Sine**), Функцєю Вейбулла (**Weibull**), інтерполяцією (**Interpolation**), сплайнами (**Spline**), функцією користувача (**Custom Equation**).

В центральній частині виводяться вихідні дані та графік наближаючої кривої. Ліворуч від графіка розташовуються кількісні параметри наближення: коефіцієнти наближаючої кривої, середньоквадратичні похибки наближення тощо.

Генерація змінних із завданим розподілом методом інверсії

Вихідною умовою для отримання імовірнісної величини "y" із щільністю розподілу $f_y(y)$ за умови наявності відомої імовірнісної величини "x" з щільністю розподілу $f_x(x)$ є рівняння їх функцій вірогідності (2.2)

$$F_y(y0) = \int_a^{y0} f_y(y)dy = F_x(x0) = \int_b^{x0} f_x(x)dx \qquad (2.2)$$

Приклад 2.1.

Визначити вираз для генерації імовірнісної величини з експоненціальним розподілом через змінну, яка рівномірно розподілена в діапазоні (0,1). Розрахувати параметри розподілу. Порівняти отримані результати з результатами вбудованої функції **`exprnd`**.

Розв’язок

Рівномірний розподіл має щільність розподілу

$$f_x(x) = \frac{1}{(b-a)},$$

де a, b –початок та кінець діапазону, відповідно.

Для експоненціального та рівномірного законів розподілу вираз (2.2) може бути записано в наступному вигляді:

$$\int_0^{y0} \frac{1}{m} e^{-\frac{y}{m}} dy = \int_0^{x0} dx = 1 - e^{-\frac{y0}{m}} = x0$$

Звідки

y0=-m·ln(1-x0),

де y0 – шукана змінна, яка має експоненціальний закон розподілу з параметром (середнім) “m”; x0 – змінна з рівномірним законом розподілу від 0 до 1.

Текст команд:

```
>>N=1000;            % Завдання разміру вектора даних
>>x=rand(1,N);       % Генерування змінної з рівномірним розподілом
>>y=-log(1-x);       % Розрахунок вектора з експоненціальним розподілом
>>y1=exprnd(1,1,N);  %Створення вектора за допомогою вбудованої
                     %функції
>>[ye xe]=hist(y);   % Отримання векторів даних розрахованої функції
>>[ye1  xe1]=hist(y1);%Отримання векторів даних вбудованої
                     % функції
>>ye=ye/N;           % Центрування даних
>>ye1=ye1/N;
```

Порівняння результатів

1. Оцінка параметрів розподілу.

Середнє повинно дорівнювати 1

```
>> mean(y)        % Розрахунок середнього
   ans =1.036934636471254% Для розрахованої залежності
>> mean(y1)
   ans =1.049265248971211  % Для вбудованої залежності
```

Оцінка параметрів розподілу

```
>> expfit(y)
   ans =1.036934636471254%Для розрахованої залежності
>> expfit(y1)
   ans =1.049265248971211% Для вбудованої залежності
```

Результати показують, що

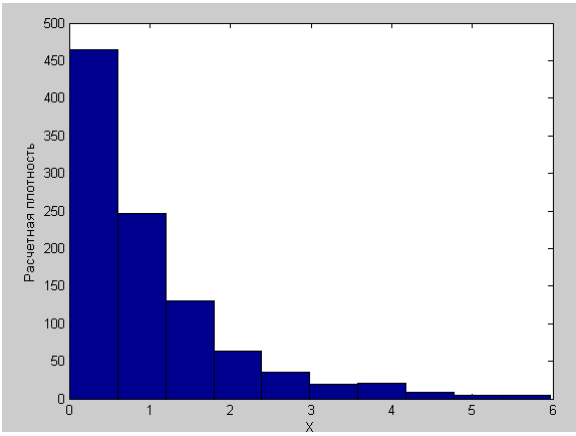
- середні значення співпадають з параметром розподілу, що і повинно відбуватися для експоненціального розподілу,
- результати розрахованої та вбудованої залежностей подібні, розрізняються на 5%, при чому розраховані значення ближче до істинного.

2. Оцінка функції розподілу.

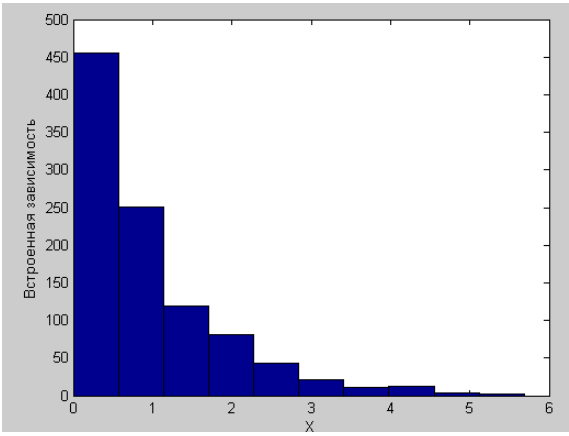
Візуальне порівняння вигляду функції розподілу.

Вигляд функції щільності розподілення можна оцінити по гістограмі розподілення (рис. 2.4).

```
>> figure(1)
>> hist(y,xe)
>> figure(2)
>> hist(y1,xe1)
```



Розрахована залежність



Вбудована залежність

Рис. 2.4. Функції розподілення векторів y та y1

Оцінка по значенням функціональної регресії.

Для знаходження регресійних коефіцієнтів імовірнісних векторів за розрахованою та вбудованою залежностями, похибок апроксимації використаємо функцію **cftool**.

Результати розрахованої залежності (рис. 2.5).

```
General model Exp1:
     f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
```

```
        a =       0.7358   (0.7229, 0.7486)
        b =      -0.9573   (-0.9798, -0.9348)
Goodness of fit:
  SSE: 6.091e-05
  R-square: 0.9998
  Adjusted R-square: 0.9997
  RMSE: 0.002759
```

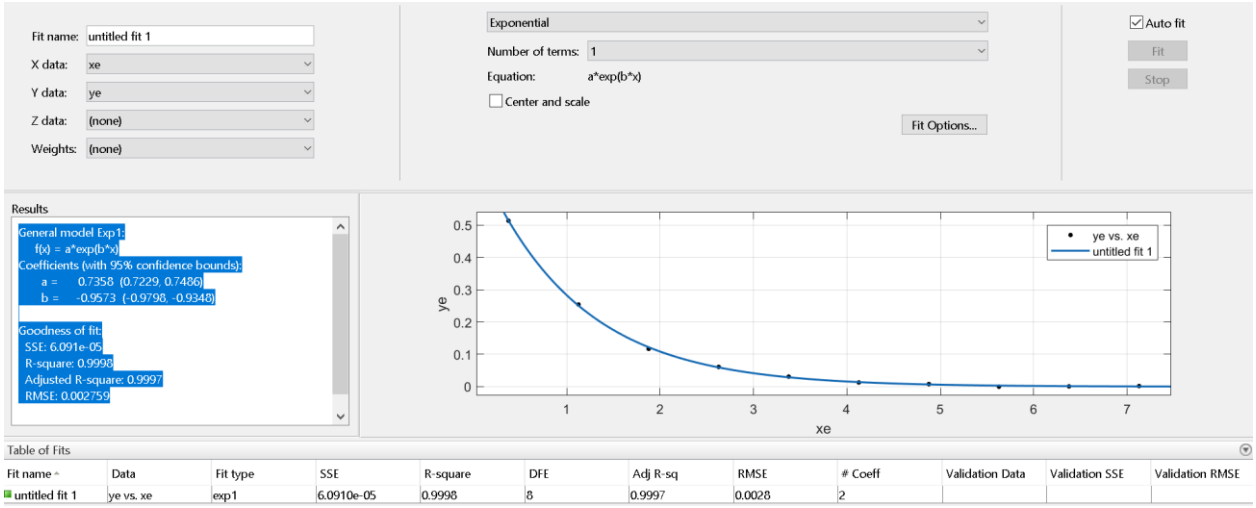


Рис.2.5 – Вигляд вікна функції **cftool** для розрахованої залежності

Результати вбудованої залежності (рис. 2.6).

```
General model Exp1:
     f(x) = a*exp(b*x)
Coefficients (with 95% confidence bounds):
        a =       0.867   (0.8428, 0.8912)
        b =     -0.9376   (-0.9722, -0.9031)
Goodness of fit:
  SSE: 0.0001622
  R-square: 0.9995
  Adjusted R-square: 0.9994
  RMSE: 0.004503
```

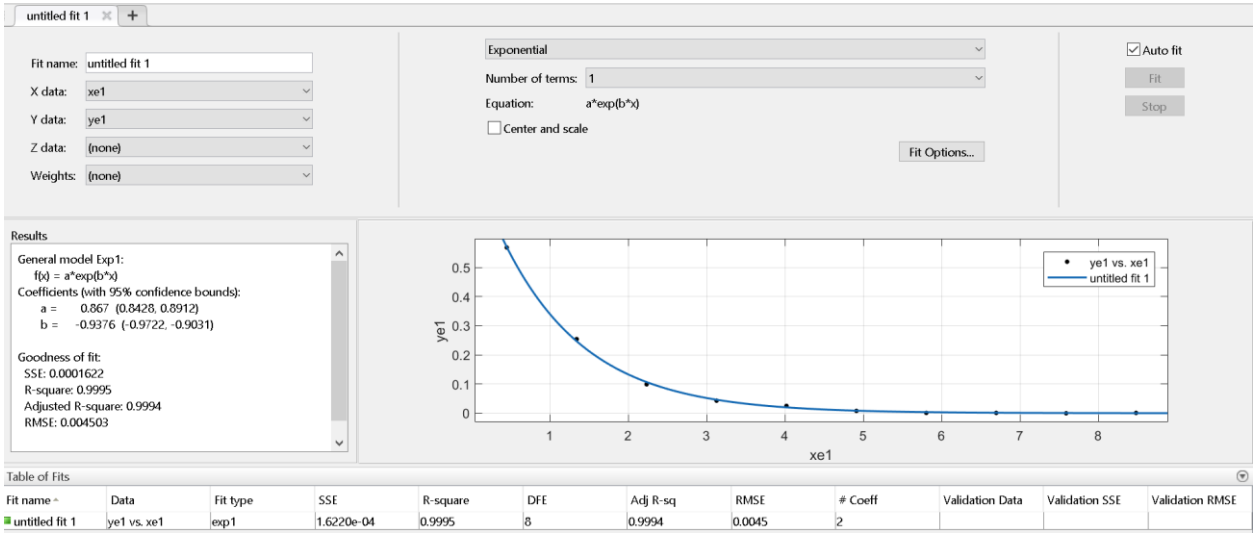


Рис. 2.6 – Вигляд вікна функції **`cftool`** для вбудованої залежності

Результати показують:

- розраховані дані точніші за вбудовані (див. похибки RMSE, SSE),
- значення коефіцієнта «b» близько до вірного, коефіцієнт «a» визначається з суттєвою похибкою.

Приклад 2.2.

Провести моделювання методом Монте-Карло розподілення теплової швидкості частинок по закону Максвела.

Розв’язок

Для реалізації метода Монте-Карло треба провести розрахунок для достатньої кількості частинок (number=500). В циклі проводиться генерація імовірнісної величини швидкості за законом Максвела доки генероване значення не перебільшить випадкового порогу. Поріг задається «рулеткою Монте-Карло» у вигляді числа з рівномірним розподілом. Знайдена швидкість присвоюється частинці. Розподіл частинок виводиться командою hist.

Текст програми:

```
m=3.32e-27; k=1.38e-23; T=250;
step=1;                 % Крок швидкості
number=500;             % Кількість частинок
np=20;                  % Розбиття гістограми
d=m/2/k/T;a=4/sqrt(pi)*d^1.5;%параметри закону Максвела
```

```
nv=zeros;      % заготовка вектора розподілу частин
for mv=1:number      % Запуск циклу по частинкам
  y=rand;%генерація порогу порівняння Монте-Карло;
   s=0;  v=0;  f=0;
   while s<y
     v=v+step; f=a*exp(2*log(v)-d*v^2); s=s+f*step;
   end
 nv(mv)=v; % заповнення масиву швидкостей частинок
   end
hist(nv,np);   %форматування та виведення щільності
              %  розподілу
title ('Розподіл частинок (закон Максвела')
xlabel('v(м/сек)'); ylabel('f(v)')
s=sprintf('Температура   %g\n   Кількість   частинок:…
%g',T,number);
legend(s)
```

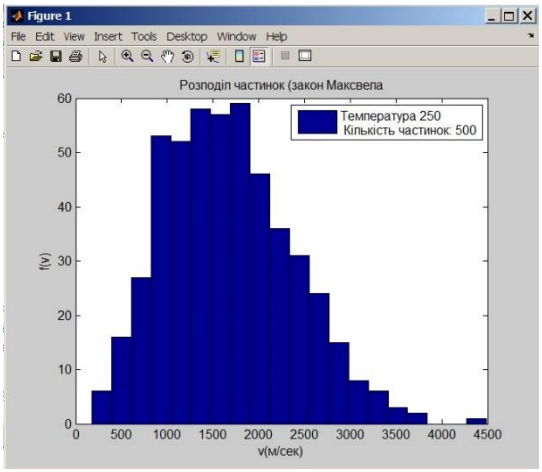


Рис. 2.7 – Розподіл частинок (закон Максвела).

Завдання 2.1.

Згенерувати вектор випадкових чисел, які розподілені за законом Релея

$$f(x,\sigma) = \frac{x}{\sigma}\exp\left(-\frac{x^2}{2\sigma^2}\right), x \geq 0, \sigma \geq 0,$$

за допомогою вбудованої функції, та функцією користувача через рівномірний розподіл. Оцінити вплив розміру вихідних даних на результати (N=100, 1000, 50000). Знайти параметри розподілу, функції щільності. Порівняти результати.

Завдання 2.2.

Перевірити чисельно центральну граничну теорему. В якості даних використати вектори з рівномірним, експоненціальним розподілами.

Дослідити:

- вплив розміру вихідних векторів на вигляд результуючого розподілу (10,100,1000,10000),
- вплив кількості векторів на результуючий розподіл (2,4,10),
- вплив типу розподілів векторів на результуючий розподіл.

Практикум 3. Оптимізаціні моделі

Мета заняття

Метою заняття є вивчення можливостей вбудованих функцій СКМ Матлаб та бібліотеки **Optimization Toolbox**. Набуття навичок розв'язання оптимізаційних задач.

Теоретичні положення

Задача параметричної оптимізації полягає в знаходженні значення параметрів $x = (x_1, x_2, x_n)$ які є найкращими (оптимальними) в сенсі якогось критерію. В найпростішому випадку вирішується задача пошуку мінімуму скалярної цільової функції f(x) без будь-яких обмежень. В більш складному, в мінімізації функції, на яку накладені явні $x_{мін}<x< x_{макс}$ або функціональні $g(x) = 0$ $(i = 1, 2, \tau_e)$, $g(x) < 0$ $(i = \tau_e+1,m)$ обмеження.

При лінійній цільовій функції оптимізація проводиться засобами лінійного програмування, при квадратичній – квадратичного програмування, в загальному випадку – нелінійного програмування.

Безумовна оптимізація

Алгоритми безумовної оптимізації можуть бути поділені за порядком похідної від цільової функції: алгоритми, що базуються на використанні похідних (1-го порядку – градієнтні, 2-го порядку – ньютонівські) та ті, що використовують тільки значення цільової функції (0-го порядку – безградієнтні або прямі).

Одним з найкращих прямих методів мінімізації функції декількох змінних, що не вимагає обчислення похідних функції, вважається симплекс метод (Нельдер – Мід). Він зводиться до побудови симплекса в n-вимірному просторі, заданого n+1 вершиною. У двовимірному просторі симплекс є трикутником, в тривимірному – пірамідою. На кожному кроці ітерацій вибирається нова точка рішення всередині або поблизу симплекса. Вона порівнюється з однією з вершин симплекса. Найближча до цієї точки вершина симплекса зазвичай замінюється цією точкою. Таким чином, симплекс перебудовується і зазвичай дозволяє знайти нове, більш точне положення точки рішення. Рішення повторюється, поки розміри симплекса по всім змінним не стануть менше заданої похибки рішення.

Градієнтні методи використовують інформацію про нахил функції для вибору напрямку пошуку екстремуму. В методі найшвидшого спуску на кожній ітерації рух до точки мінімуму здійснюється в напрямку -g(x) (де **g(x)=Vf(x)** – вектор-градієнт цільової функції f(x)).

Ньютонівські методи використовують квадратичну апроксимацію цільової функції, при цьому на кожній ітерації вирішується завдання локальної мінімізації

f(x)=H·x+g=0, (3.1)

де H – симетрична і позитивно обумовлена матриця других похідних (матриця Гессе, або Гесіан), g – вектор градієнта. Ньютонівські алгоритми безпосередньо обчислюють H і здійснюють рух в розрахованому на черговій ітерації напрямку зменшення цільової функції до досягнення мінімуму з використанням методів одновимірного пошуку.

Квазіньютоновскі алгоритми використовують деяку апроксимацію. Серед подібних алгоритмів є так званий BFGS-алгоритм, який отримав свою назву за прізвищами що запропонували його авторів (Бройдо, Флетчер, Гольдфарб, Шанно), в якому апроксимація H розраховується ітераційно, за формулою (3.2)

$$H_{k+1} = H_k + \frac{q_k q_k^T}{q_k^T s_k} - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k} \qquad (3.2)$$

$$s_k = x_{k+1} - x_k \qquad q_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

та метод DFP (Давидсон—Флетчер—Пауэл), в якому апроксимація застосовується для зворотньої матриці H-1 .

Для випадків, коли кількість параметрів не збігається з кількістю функцій оптимізації використовується метод найменших квадратів МНК (Ньютон—Гаус, Левенберг—Марквардт) за формулою (3.3).

$$min \frac{\|F(x)\|_2^2}{2} = \frac{\sum_i F_i^2(x)}{2} \qquad (3.3)$$

Оптимізація за наявності обмежень

В задачах умовної оптимізації шлях рішення зазвичай складається із заміни вихідної задачі з обмеженнями на оптимізацію без обмежень. Така

заміна провадиться з допомогою умов Куна-Такера, які мають наступний вигляд:

$$\begin{aligned}
&\nabla f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{x}^*) = \mathbf{0}, \\
&\nabla g_i(\mathbf{x}^*) = \mathbf{0}, \quad i = 1, 2, \ldots, m_e, \\
&\lambda_i^* = 0, \quad i = m_e+1, \ldots, m,
\end{aligned} \tag{3.4}$$

де λ — множники Лагранжа.

Для розв'язання системи (3.4) зазвичай використовують алгоритм послідовного квадратичного програмування (Sequential Quadratic Programming чи SQP), який є різновидом квазіньютонівського методу з квадратичною апроксимацією функції Лагранжа (для врахування обмежень)

$$\begin{aligned}
&\min_{d \in R^n} \frac{1}{2} \mathbf{d}^T \mathbf{H}_k \mathbf{d} + \nabla f^T(\mathbf{x}_k)\mathbf{d}, \\
&\nabla g_i^T(\mathbf{x}_k)\mathbf{d} + g_i(\mathbf{x}_k) = 0, \quad i = 1, 2, \ldots, m_e, \\
&\nabla g_i^T(\mathbf{x}_k)\mathbf{d} + g_i(\mathbf{x}_k) \le 0, \quad i = m_e+1, \ldots, m.
\end{aligned} \tag{3.5}$$

Вбудовані функції (див. табл. 4.1) СКМ Матлаб дозволяють знаходити мінімум функцій однієї чи кількох змінних. Результатом є локальний мінімум.

Для пошуку локального мінімуму функції однієї змінної на заданому відрізку застосовується функція **fminbnd**, яка використовує метод квадратичної чи кубічної інтерполяції.

Для безумовної мінімізації функцій декількох змінних застосовуються функції **fminsearch** (симплекс метод), **fminunc** (метод Ньютона, квазіньютонівський метод DFP), **lsqnonlin** (метод НК, Ньютона або Левенберга) **.**

Для умовної мінімізації функцій декількох змінних застосовуються функції **quadprog** (метод SQP для квадратичних функцій), **fmincon** (метод SQP) , **lsqnonlin** (метод Лагранжа) **.**

Таблиця 4.1. Базові функції оптимизації СКМ Матлаб

Тип задачі	Математичний запис	Функція Матлаб
Безумовна оптимізація	**min(f(x))**	**fminbnd, fminunc, fminsearch**
Умовна мінімізація квадратичних функцій	**min(xT·H·x/2 +fT·x)** **A·x<b, Aeq·x=beq, xL<x<xи**	**quadprog**
Умовна мінімізація	**min (f(x))** **A·x< b, Aeq x=beq, xL<x< xu**	**fmincon**
Умовна та безумовна мінімізація	**min (∑f(x)2)** **xL<x< xu**	**lsqnonlin**

Позначення: **f(x)** – скалярні функції; **x, y** – векторні аргументи; **A, Aeq, C, H** – матриці; **b, beq, d, f, w, goal, xdata, ydata** – вектори; **xL, xu** – нижня та верхня межі області існування аргументу.

Синтаксис виклику функції оптимізації має наступний вигляд:

```
[X,<fval>,<exitflag>,<output>,..]=
MLABfun(fun,xl,x2<,options,p1,p2,...>)   ,
```

де **X** – шуканий вектор точки мінімуму; **<fval>** – значення функції в точці мінімуму; **<exitflag>** – ціле число, що показує причину закінчення розрахунку: 1 Значення градієнта менше точності; 2 Крок X менше точності; 3 Крок зміни цільової функції менше точності; 0 Досягнуто максимальну кількість ітерацій; -1 Алгоритм завершено по значенню функції; -2 Одномірний метод не має коректної початкової точки, - **<output>** – структура даних, яка містить додаткову інформацію:

output .algorithm – використаний алгоритм;

output .funcCount – кількість оцінок функції;

output.iterations – кількість ітерацій.

MLABfun – ім'я функції пакету;

fun – функція, що оптимізується (цільова).

xl, <x2> – початкова точка для функції кількох змінних або діапазон пошуку для функції однієї змінної;

<options> – структура, яка керує виглядом результатів обчислень та процесом оптимізації. Значення структури слід попередньо задати за допомогою функції **optimset**.

<p1,p2> – додаткові аргументи функції **fun**.

Аргументи **optimset** задаються попарно:

options=optimset(...,вид контролю, значення,...).

Можливі значення параметрів **options** наведено в таблиці 3.2.

Таблиця 3.2. Основні параметри функції **optimset**

Вид контролю	Значення	Результат
'Display'	'off	Інформація про процес не виводиться
	'iter'	Виводиться інформація кожного кроку процесу
	'final'	Виводиться заключна інформація (використовується за замовчанням)
'MaxFunEvals'	Ціле число	Максимальна кількість викликів функції **fminbnd, fmincon, fminimax, fminsearch, fminunc, fseminf, lsqnonlin**
'Maxiter'	Ціле число	Максимальна кількість ітерацій
'TolFun'	Дійсне число	Точність обрахунку функції
'TolX'	Дійсне число	Точність по аргументу для останова обчислень
'TolCon'	Дійсне число	Точность на порушення обмежень для останова обчислень

Таблиця 3.2. Продовження

Вид контролю	Значення	Результат

‘DerivativeCheck’	[on/ {off}]	Включення вибору способа розрахунку похідних: чисельний або аналітичний (в функції fun) **fmincon, fminimax, fminunc, fseminf, lsqnonlin**
‘Diagnostics’	[on/ {off}]	Виведенння інформації про поведінку функції
‘DiffMaxChange’	0.1	Максимальний крок сітки кінечної різниці **fmincon, fminimax, fminunc, fseminf, lsqnonlin**
’DiffMinChange’	10^{-8}	Мінімальний крок сітки кінечної різниці **fmincon, fminimax, fminunc, fseminf, lsqnonlin**
’GradConstr ‘	[on/ {off}]	Задання градієнта обмежень користувачем **fmincon, fminimax**
’GradObj ‘	[on/ {off}]	Задання градієнта функції користувачем **fmincon, fminimax, fminunc, fseminf**
’Hessian’	[function | {[]}]	Задання гесіана функції користувачем **fmincon, fminunc**
’HessUpdate’	[{bfgs} | dfp | gillmurray | steepdesc]	Спосіб розрахунку гесіана в квазиньютоновських алгоритмах: Бройдена, Флетчера, Мюррея, градієнтного спуску **fminunc**

Таблиця 3.2. Закінчення

Вид контролю	Значення	Результат
’Jacobian’	[on | {off}]	Задання якобіана користувачем **lsqnonlin**

’LargeScale’	[on | {off}]	Включення алгоритмів великої розмірності **fmincon, fminunc, lsqnonlin, quadprog**
’LevenbergMarquardt’	[on | {off}]	Перемикання між методами Ньютона та Левенберга **lsqnonlin**
’LineSearchType’	[cubicpoly | {quadcubic}]	Вибор типу інтерполяції для одномірної оптимізації **lsqnonlin**

Наприклад, рядки

```
options = optimset('Display', 'iter','TolX',1.0e-09);
x2 = fminbnd('ftest',  0.7, 2.0, options)
```

призводять до появи в командному вікні окрім результату ще й таблиці з інформацією про хід процесу. Кожен рядок таблиці містить інформацію про номер кроку, поточні значення аргументу та функції, використаний метод.

```
Func-count   x        f(x)       Procedure
1            1.19656   -0.290321    initial
2            1.50344  0.222246   golden
.   .    .
11           1.15545   -0.313158  parabolic
```

Функція **optimset,** яка викликана з іменем функції оптимізації **MLABfun** Матлаб в якості аргументу, виводить в командне вікно перелік можливих установок для алгоритмів, які реалізовані в функції.

Склад установок залежить від значення параметра **LargeScale.** Значення '**On**' дозволяє використання **Large-scale**-алгоритма (Ньютона за допомогою функцій **fminunc, lsqnonlin**).

Якщо необхідно задати аргументи, які розташовані не послідовно, треба на місце незаданих параметрів поставити знак [ ] (пустий масив).

Ім'я цільової функції може задаватися:

1) у вигляді рядка з описом функції, наприклад

```
x=fminbnd('cos(x)',0,1);
```

Формула, яка обрана в апострофи, може містити тільки змінну з іменем "x". Використання інших імен викликає помилку.

2) покажчиком на в m-файл, в якому описана функція, наприклад

```
function ff = test(v)
ff = sin(pi*v); % опис функції виклик функції:
x=fminbnd(@test,0,1) %@test- покажчик на функцію
```

3) рядком з іменем функції з m-файлу, наприклад

```
x=fminbnd('test',0,1)  або
x=fminbnd('test(x)',0,1)
```

3) за допомогою «inline» функції, наприклад

```
f = inline('1./((x - 0.3)^2 + 0.01) - 6'); % опис
x=fminbnd(f,0,1);  % застосування
```

Безумовна оптимізація функції однієї змінної

Для безумовної оптимізації функції однієї змінної призначена функція **fminbnd**:

```
[X,<fval>,<exitflag>,<output>] =
        fminbnd(fun,x1,x2<,options, p1,p2,...>)
```

Функція використовує методи "золотого перетину" чи інтерполяції. Опції функції описано вище.

Для безумовної оптимізації функції однієї змінної можна використовувати функцію **fminsearch**.

Приклад 3.1. Визначити локальні мінімуми функції

$$f(x) = e^{x^2} + \sin(3 \cdot \pi \cdot x) \qquad (3.6)$$

РОЗВ'ЯЗОК

Для вивчення поведінки функції намалюємо її графік послідовністю команд або скріпт – файла:

```
x=-1.5:0.05:1.5;      % задання діапазону "x"
y=fun(x);
plot (x,y)
hold on;
grid;
xlabel('x');
ylabel('y');
title('Графік функції exp(x^2)+sin3pix')
```

З графика (рис. 3.1), видно, що функція має чотири локальних мінімума на відрізку [-1.5,1.5].

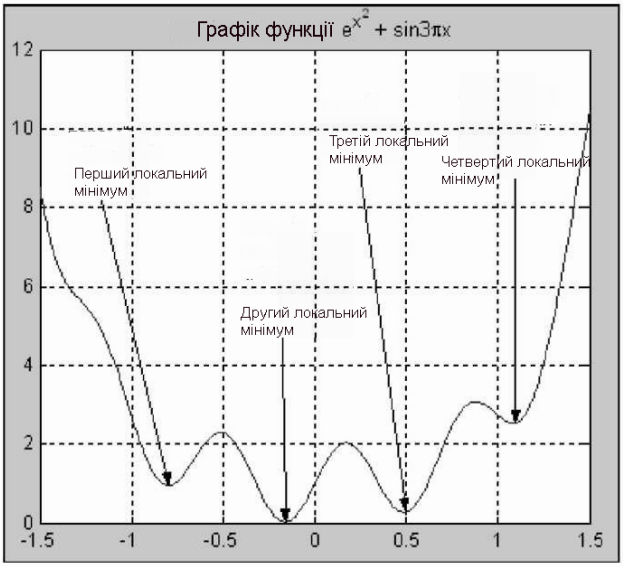


Рис.3.1 – Розташування локальних мінімумів функції (3.6)

Для простого визначення значення мінімума достатньо викликати функцію **fminbnd** з відповідним аргументом:

```
fminbnd('exp(x^2)+sin(3*x*pi)',-1,0)
ans =    -0.1629
```

Пакет знайшов другий локальний мінімум (рис.3.1).

Для отримання додаткової інформації треба змінити виклик функції **fminbnd:**

```
[x f a b]=fminbnd('exp(x^2)+sin(3*x*pi)',-1,0)
```

Результат:

```
x =  -0.1629
```

```
f = 0.0275
a =  1
b =iterations: 8
   funcCount: 9
   algorithm: 'golden section search, parabolic interpolation'
   message: [1x112 char]
```

Для илюстрації процеса оптимізації можна на графік функції накласти зображення знайденої оптимальної точки.

Опис функції оптимізації:

```
function y=fun71(x)
y=exp(x.^2)+sin(3*x.*pi);
```

Знаходження точки оптимума та її зображення у вигляді червоного кола:

```
[x,f]=fminbnd('fun71',-1,0);
plot(x,f,'ro')
```

Завдання.

Знайдить всі точки оптимума на відрізку [-1.5 1.5].

Безумовна оптимізація функції кількох змінних

Для безумовної оптимізації функції кількох змінних передбачені функції **fminsearch, fminunc , lsqnonlin.**

Функція **fminsearch** може застосовуватися для оптимізації функцій як однієї, так і багатьох змінних, вона використовує для пошуку оптимуму симплекс – метод. Функція потребує завдання початкового наближення (точки), яке в разі функції однієї змінної повинно бути числом, а для функції кількох змінних – вектор -рядок (точку).

Наприклад, початкове наближення для приклада 3.1 дасть наступну відповідь:

```
» fminsearch(@fun71, -0.5)
ans =-0.1629
```

В разі, якщо функція f(x) є гладкою можна використати для оптимізації функцію **fminunc**. Функция за допомогою формули Тейлора будує локальну квадратичну апроксимацію цільової функціїв області поточної точки. Вона

застосовує ньютонівські, квазіньютонівські або градієнтні методи в залежності від значення структури **optimset.**

Параметри структури **optimset** функції: **Display, TolX, TolFun, DerivativeCheck, Diagnostics, GradObj** (як другий результат обрахунку f(x)), **HessPattern, LineSearchType, Hessian** (як третій результат обрахунку функції f(x)), **HessMult, HessUpdate, MaxFunEvals, MaxIter, DiffMinChange and DiffMaxChange, LargeScale, MaxPCGIter, PrecondBandWidth, TolPCG, TypicalX.**

В простому випадку її аргументами є показчик на функцію f(x) та початкова точка. Застосування аналітичних формул для обрахунку градієнта та гесіана цільової функції дозволяє скоротити час оптимізації.

Для оптимізації цільових функцій, які записано в вигляді суми квадратів функцій оптимізації призначено функцію **Isqnonlin**. Функція використовує метод найменших квадратів (МНК):

$$\min F(x) = f_1^2(x) + f_2^2(x) + \ldots = \min \frac{1}{2}\left\|F(x)\right\|^2 = \frac{1}{2}\sum_i f_i^2(x) \qquad (3.8)$$

Синтаксис функції:

[X,<Rn,Rs,E,O,L,J>]=lsqnonlin(fun,X0,<Lb,Ub,Op>) , **де X** – знайдена точка оптимуму; **Rn** – значення цільової функції: **sum(fun(X).^2)**; **Rs** – вектор функцій оптимізації **fun(X)**; **E** – код завершення алгоритму: 1 – успішне завершення; 2 – крок X менше точності; 3 – значення цільової функції точності; 4 – градієнт менше точності; 0 – кількість викликів функції більше заданого; -1 – алгоритм завершено функцією оптимізації; -2 – межі порушено; -4 - одномірний пошук невдалий.

O – Умови розрахунку: Display, TolX, TolFun, Jacobian, DerivativeCheck, Diagnostics, FunValCheck, JacobMult, JacobPattern, LineSearchType, LevenbergMarquardt, MaxFunEvals, MaxIter, DiffMinChange, DiffMaxChange, LargeScale, MaxPCGIter, PrecondBandWidth, TolPCG, TypicalX, PlotFcns, OutputFcn

L – коефіцієнти Лагранжа; **J** – матриця Якобі; **fun** – вектор функцій оптимізації. В описі елементів вектору може знаходитись додатково Якобіан; **X0** – вектор початкової точки; **Lb,Ub** – вектори меж обмежень

аргументу X: **Lb <= X <= Ub**. За відсутності обмежень використовують пусті вектори; **Op** – вектор умов виклику функції.

Приклад 3.2. Визначити точки локальних мінімумів функції

$$f(x, y) = \sin(\pi \cdot x) \cdot \sin(\pi \cdot y) \qquad (3.9)$$

симплекс методом в області x ∈(0..2), y ∈ (0..2).

РОЗВ'ЯЗРК

Для вивчення поведінку функції намалюємо її графік послідовністю команд або скріпт – файла (рис. 3.2):

```
% побудова графика функції
» [X  Y] = meshgrid(0:0.01:2);
» Z = sin(pi*X).*sin(pi*Y);
»[CM            h]=contour(X,Y,Z,[-0.96,-0.9,-0.8,-0.5,…
-0.1,0.1,0.5,0.8, 0.9,0.96]);
» clabel (CM, h)
» colormap(gray)
» xlabel( 'x')
» ylabel('y')
```

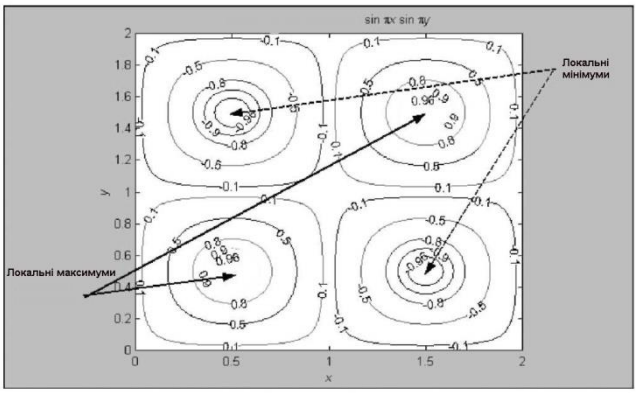


Рис. 3.2 – Розташування локальних екстремумів функції (3.9)

Для визначення функції можна використати будь який з наведених 3-х способів:

```
1)function f = fun(v)
    f = sin(pi*v(1)).*sin(pi*v(2));
2)f='sin(pi*x).*sin(pi*y)'
```

```
3)ff=inline('sin(pi*x).*sin(pi*y) ')
```

Виклик функції **fminsearch**:

```
1) M = fminsearch('fun72(x)',[1.4 0.6])
або
1.1)  » M = fminsearch(@fun72,[1.4 0.6])
або
2)M  =  fminsearch('sin(pi*x(1))*sin(pi*x(2))',[1.4
0.6])
або
2.1) » M = fminsearch(f,[1.4 0.6])
або
3) » M = fminsearch(ff,[1.4 0.6])
```

Результатом виклику буде один з мінімумів

```
m =   1.500018797676021   0.499987787366408
```

Завдання. Визначити значення другого мінімума самостійно.

Приклад 3.3. Дослідити пошук оптимуму функції

$f(x) = x_1^3 + x_2^3 - 3 \cdot x_1 \cdot x_2$ (декартов лист) (3.10)

методом Ньютона, квазіньютонівським методом із чисельним та аналітичним розрахунком градієнта цільової функції.

РОЗВ'ЯЗОК

Для вивчення поведінку функції намалюємо її графік послідовністю команд або скріпт-файла (рис. 3.3):

```
x0=-1.5:0.01:2.5;      % діапазон графіка
[x y]= meshgrid(x0);   % сітка графіка
z=x.^3+y.^3-3*x.*y;
v=-2.8:0.2:1;          % лінії рівня графіка
[c h]=contour(x,y,z,v)% контурний графік
clabel(c,h)            % лінії рівня
colormap(gray)
```

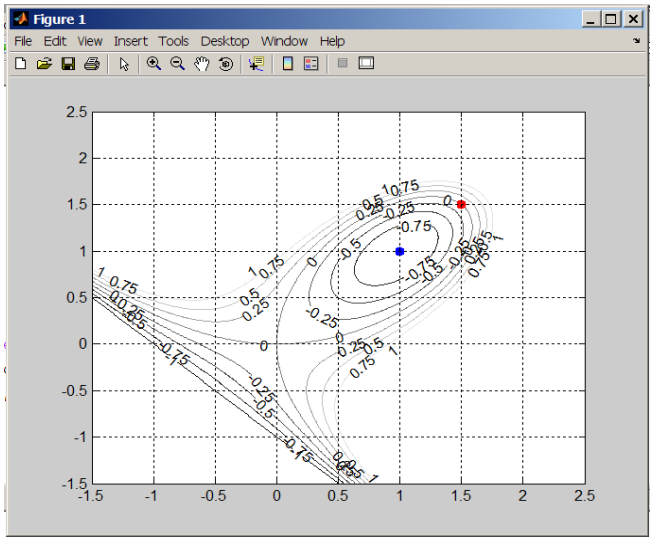


Рис. 3.3 – Графік функції 3.10

Опис функції декартового листа:

```
function [f,g]=fun73(x)
f=x(1)^3+x(2)^3-3*x(1)*x(2);
if nargout>1
   g=[3*x(1)^2-3*x(2)    3*x(2)^2-3*x(1)];
end
```

Пояснення:

Якщо функція викликається звичайним способом:

```
f = fun73(x),
```

то вона повертає обчислене значення функції (4.8), якщо викликається із зазначенням двох результатів

```
[ a b] = fun73(x),
```

то вона повертає вектор з двома елементами. Перший елемент – значення функції (3.10), другий – вектор часткових похідних функції (3.10)

Для застосування метода Ньютона використаємо функцію **fminunc**. Метод викликається за умови вмикання алгоритму високої розмірності та аналітичного опису градієнта. Тому в функції **fminunc** потрібно задати параметри **'Large scale','On'** та **'Gradobj','on'**в структурі **optimset.**

Проведення оптимізації методом Ньютона та малювання початкової точки червоним та оптимальної точки синім:

```
x0=[1.5 1.5];    % початкова точка
line(x0(1),x0(2),'Marker','.','Color','red',…
'MarkerSize',20);
[x  f  e  o]  =  fminunc('fun73',x0,optimset('Gradobj',…
'on','LargeScale','On'))
line(x(1),x(2),'Marker','.','MarkerSize',20);
```

Відповідь:

```
Optimization terminated: first-order optimality less
than OPTIONS.TolFun, and no negative/zero curvature
detected in trust region model.
    x =    1.0000    1.0000
    f =   -1.0000
    e =      1
    o =     iterations: 4
     funcCount: 5
     cgiterations: 4
     firstorderopt: 6.9707e-008
       algorithm: 'large-scale: trust-region Newton'
      message: [1x137 char]
```

Пояснення.

Значення **x=[1.0 1.0]** - визначена точка оптимума. Цільова функція в точці оптимума **f(x)=-1.0.** Значення **e=1 (**exitflag) дає інформаціію про те, що розрахунок завершено тому, що досягнуто значення градієнта менше точності.

Додаткова інформація: кількість ітерацій – 4, кількість проб функції – 5. ступінь оптимальності (firstorderopt – норма вектора-градієнта в знайденій точці: **6.9707e-008**, використано алгоритм Ньютона великої розмірності **'large-scale: trust-region Newton'**.

Проведення оптимізації квазіньютонівським методом з чисельним розрахунком градієнта.

Метод викликається за умови вмикання алгоритму середньої розмірності та чисельного розрахунку. Тому в функції **fminunc** потрібно

задати параметри **'Large scale', 'Off' та 'Gradobj', 'off'** в структурі **optimset**.

```
x0=[1.5 1.5];          % початкова точки
[x  f  e  o]  =  fminunc('fun73',x0,optimset('Gradobj',…
'off','LargeScale','Off'))
```

Відповідь:

```
Optimization terminated: relative infinity-norm of
gradient less than options.TolFun.
x =     1.0000    1.0000
f =   -1.0000
e =      1
o =     iterations: 3
funcCount: 15
stepsize: 0.3333
firstorderopt: 8.9407e-008
algorithm: 'medium-scale: Quasi-Newton line
search'
message: [1x85 char]
```

Пояснення:

Значення **x=[1.0 1.0]** - визначена точка оптимума. Цільова функція в точці оптимума **f(x)=-1.0.** Значення **e=1 (exitflag**) дає інформаціію про те, що розрахунок завершено тому, що досягнуто значення градієнта менше точності.

Додаткова інформація: кількість ітерацій – 3, кількість проб функції – 15, ступінь оптимальності (**firstorderopt,** норма вектора-градієнта в знайденій точці: **8.9407e-008**, використано квазіньютонівський алгоритм **'medium-scale: Quasi-Newton line search'**.

Проведення оптимізації квазіньютонівським методом з чисельним розрахунком градієнта.

Метод викликається за умови вмикання алгоритму середньої розмірності та аналітичного розрахунку. Тому в функції **fminunc** потрібно задати параметри **'Large scale', 'Off' та 'Gradobj', 'on'** в структурі **optimset**

```
[x f e o]=fminunc('fun73',x0,optimset('Gradobj','on'))
```

Відповідь:

```
Optimization terminated: relative infinity-norm of gradient less than options.TolFun.
x =     1     1
f =    -1
e =     1
o =     iterations: 3
 funcCount: 5
 stepsize: 0.3333
firstorderopt: 0
algorithm: 'medium-scale: Quasi-Newton line   search'
message: [1x85 char]
```

Пояснення:

Значення **x=[1.0 1.0]** - визначена точка оптимума. Цільова функція в точці оптимума **f(x)=-1.0.** Значення **e=1 (**exitflag) дає інформаціію про те, що розрахунок завершено тому, що досягнуто значення градієнта менше точності.

Додаткова інформація: кількість ітерацій – 3, кількість проб функції – 5, ступінь оптимальності (firstorderopt - норма вектора-градієнта в знайденій точці: 0, використано квазіньютонівський алгоритм **'medium-scale: Quasi-Newton line search'**.

Висновки

Всі методи другого порядку функції для заданих вихідних даних дозволяють знайти точку оптимуму та їхні розв’язання співпадають. Серед методів даних більш ефективним є квазіньютонівський з аналітичним описом градієнта. Він забезпечує найменшу кількість ітерацій та проб функції, найменше значення градієнта (табл. 3.3).

Таблиця 3.3. Порівняння методів функції **fminunc**

Метод	Ітерації	Проби	Градієнт
Ньютона	4	5	6.9707e-008

Квазіньютона чисельний	3	15	8.9407e-008
Квазіньютона аналітичний	3	5	0

Приклад 3.4. Знайти мінімум функції

$$f(x) = \sum_{k=1}^{10} (2 + 2 \cdot k - e^{k \cdot x(1)} - e^{k \cdot x(2)})^2 \qquad (3.11)$$

при початковій точці x0 = [0.3 0.4].

РОЗВ'ЯЗОК

Цільова функція записана в вигляді суми квадратів функцій оптимізації, тому можна використати функцію **Isqnonlin.**

Опис цільової функції

```
function f=f74(x)
к = 1:10;
f = 2 + 2*к - exp(к*x(1)) - exp(к*x(2));
```

Виклик функції оптимізації

```
» xO = [0.3 0.4];
» lsqnonlin('f74',x0)
```

Результати розрахунку:

```
Optimization terminated successfully:  Norm of the current step is less than OPTIONS.TolX
ans =     0.2578    0.2578
```

Запуск з опціями:

```
[x f q q w]=lsqnonlin('f75',x0)
```

Результати розрахунку:

```
Optimization terminated successfully:
 Norm of the current step is less than OPTIONS.TolX
x =     0.2578    0.2578
```

```
f =  124.3622
q =     1
q =     1
w =   firstorderopt: 2.3893e-004
          iterations: 24
          funcCount: 73
cgiterations: 14
algorithm: 'large-scale: trust-region reflective Newton'
```

Результати показують, що використано метод Ньютона великої розмірності, проведено 24 ітерації, 73 викликів функції, в точці оптимуму **`x = (0.2578 0.2578)`** значення функції складає 124.3622.

Завдання 3.1.

Дослідити ефективність вбудованих методів безумовної оптимізації для пошуку мінімуму функції Розенброка

$$f(x) = 5 \cdot (x_1 - x_2^2)^2 + (1 - x_1)^2 \qquad (3.12)$$

із початкової точки [-1.9 2].

Використати:

а) симплекс метод (**`fminsearch`**);

б) метода Ньютона середньої розмірності (**`lsqnonlin`**);

в) метода Левенберга середньої розмірності (**`lsqnonlin`**);

д) метода Ньютона великої розмірності (**`lsqnonlin`**);

е) квазиньютонівський метод Флетчера DFP (**`fminunc`**);

ж) квазиньютонівський метод Бройдена BFGS (**`fminunc`**);

з) метод градієнтного спуску (**`fminunc`**).

Практикум 4. Актуалізація цифрових зображень

Мета роботи – здобуття студентами навичок з витягнення з цифрових зображень інформації про геометричні характеристики зображень, навичок з базових перетворень бітових зображень.

Завдання роботи:

Набуття навичок із зчитування графічного файлу зображення, керування сіткою на зображенні, виведення назви та розміру файлу зображення,

отримання з файлу та виведення на екран інформації про параметри комірок фотоприймача, поля зору, фокусної відстані тощо, з проведення "resampling" зображення.

Теоретичні положення

В Матлаб, як і у більшості систем, операції аналізу та перетворення зображень проводяться над статичними зображеннями – кадрами. В разі роботи з відеопослідовностями обов'язковим є етап виділення з потоку статичного зображення (**data aquision**).

Пакет може обробляти значну кількість стандартних графічних бінарних файлів: CUR (Cursor File), GIF (Graphics Interchange Format), HDF (Hierarchical Data Format), ICO (Icon File), BMP(Windows Bitmap), JPEG (Joint Photographic Experts Group), JPEG 2000 (Joint Photographic Experts Group 2000), PBM (Portable Bitmap), PCX (Windows Paintbrush), PGM (Portable Graymap), PNG (Portable Network Graphics), XWD (X Window Dump), PPM (Portable Pixmap), RAS (Sun Raster), TIFF (Tagged Image File Format).

Кількість інформації в графічному файлі обумовлюється кількістю точок зображення (пікселів, pixels) та обсягом даних для кожної точки. Мірилом обсягу даних точки виступає бітова глибина (**`Bitdepth`**). Вона розраховується множенням значення бітів та канал (**`bits-per-sample`**) на значення каналів на точку (**`samples-per-pixel`**). Наприклад, для чорно-білого зображення типу GIF маємо один канал на 8 біт. Бітова глибина – 8. Для кольорового зображення підвищеної ємності PNG маємо 3 канали по 16 біт на кожен канал – колір. Бітова глибина – 48.

Для технічного вжитку найбільш перспективними можна вважати TIFF PNG, розширені версії JPEG – JPEG XT, JPEG XL та JPEG 2000 формати. Саме ці формати файлів забезпечують підвищену точність від 10 – 12 до 16 біт на піксель замість стандартних 8 біт на піксель.

За замовчанням чорно-білі зображення з R рядками та C стовпцями представляються в системі матрицею RxC, кольорові – структурою 3xRxC. Тип даних пакет визначає за внутрішнім описом, який містять графічні файли. Звичайні файли зчитуються з даними типу **`uinit8`** (ціле без знаку, 1 байт). Якщо файл має підвищену точність даних, то пакет автоматично переводить дані до типу **`uinit16`** (ціле без знаку, 2 байти).

Зчитування графічних зображень проводиться функцією

```
A = imread('СПФ', <FMT>),
```

де **СПФ** – специфікація файлу з вказанням повного шляху до нього: **FMT** - опціональний параметр. Текстовий рядок з розширенням файлу. При визначенні цього параметру, вказувати розширення в специфікації файлу не потрібно.

Для окремих типів файлів можуть застосовуватись додаткові параметри функції. Повний опис функції наведено в довіднику системи.

Обов'язковим при проведенні технічного моделювання є "актуалізація" зображення. Тобто, опис зображення в абсолютних координатах. Індекс і пікселя цифрового зображення пов'язується з відповідною абсолютною координатою x_i через розмір **dx** комірки фотоприймача:

$$x_i = dx \cdot (i+0.5).$$

Додатково потрібні значення фокусної відстані оптичної системи, поля зору, розмірів комірок фотоперетворювача.

Можна вводити ці значення в діалоговому режимі, автоматизувати цей процес можна за допомогою додаткової інформації (**exif**), яка зберігається в кожному графічному файлі.

EXIF (*Exchangeable Image File Format*) – стандарт, який формалізує можливість додавати до зображень та інших медіафайлів додаткову інформацію – метадані, які коментують файл, описують умови його створення, використання та розповсюдження.

Переважна більшість сучасних фотокамер записує метадані під час знімання.

В метадані входить інформація про: виробника камери, модель камери, витримку, діафрагму, спалах, фокусну відстань, параметри кадра, фотографічну чутливість, дату, координати, баланс білого, координати тощо.

Якщо фотоапарат автоматично не записує метадані, потрібно використовувати "фірмове" програмне забезпечення камери.

Розробником стандарту є Japan Electronics and Information Technology Industries Association (JEITA).

EXIF є частиною більш широкого стандарту **DCF** (Design Rule for Camera File System) – промислового стандарту структури даних у пристроях отримання, зберігання та відображення цифрових зображень, який прийнятий у 1998 році JEIDA (Japan Electronics Industry Development Association).

Для актуалізації зображення застосовуються поля **Image Width, Image Height** – розміри зображення в пікселях, **FocalLength** – фокальна відстань в міліметрах, **FocalPlane XResolution**, **FocalPlane YResolution** - щільність сенсорів на фотоприймачі на одиницю вимірювання **FocalPlane ResolutionUnit**.

Лінійний метричний розмір зображення/фотоприймача уздовж координати X визначається через **FocalPlane ResolutionUnit** як

Image Width/ FocalPlane XResolution.

Наприклад, для **FocalPlane ResolutionUnit** = 2 (що означає дюйми) та **FocalPlane XResolution** = 2048000/595, це становить 3442 пікселів на дюйм, Для зображення з **Image Width**=2048 ширина зображення/фотоприймача 25,4x2048/3442 = 15,11 мм мм, відповідно ширина комірки – 25.4x1000/3442=7.4мкм.

Для отримання інформації про зображення в Матлаб слугує функція **imfinfo('filename').** Функція повертає структуру з параметрами зображення. Для файлів JPG, TIFF структура має кілька розділів.

Наприклад, для файлу [https://the-python-challenge-solutions.hackingnote.com/level-12.html] виклик функції **a= imfinfo('f:evil.jpg')** дасть відповідь

```
a =
            Filename: 'f:evil1.jpg'
         FileModDate: '18-Aug-2018 20:24:27'
            FileSize: 86490
              Format: 'jpg'
       FormatVersion: ''
               Width: 640
              Height: 480
            BitDepth: 24
           ColorType: 'truecolor'
     FormatSignature: ''
     NumberOfSamples: 3
        CodingMethod: 'Huffman'
       CodingProcess: 'Sequential'
             Comment: {}
                Make: 'Canon'
```

```
                Model: 'Canon PowerShot S400'
          Orientation: 1
          XResolution: 180
          YResolution: 180
       ResolutionUnit: 'Inch'
             Software: 'Adobe Photoshop 7.0'
             DateTime: '2003:05:25 11:11:41'
     YCbCrPositioning: 'Centered'
        DigitalCamera: [1x1 struct]
        ExifThumbnail: [1x1 struct]
```

До значень ширини та висоти зображення в пікселах можливо дістатися безпосередньо через поля **Width**, **Height** структури. Додаткова інформація знаходиться в підстуктурі **DigitalCamera** .

```
a.DigitalCamera
ans =
ExposureTime: 0.1250
FNumber: 2.8000
ExifVersion: [4x1 double]
DateTimeOriginal: '2003:05:24 16:40:33'
DateTimeDigitized: '2003:05:24 16:40:33'
ComponentsConfiguration: 'YCbCr'
CompressedBitsPerPixel: 3
ShutterSpeedValue: 3
ApertureValue: 2.9688
ExposureBiasValue: 0
MaxApertureValue: 2.9688
MeteringMode: 'Pattern'
Flash: [1x148 char]
FocalLength: 7.4063
UserComment: [1x264 double]
FlashpixVersion: [4x1 double]
ColorSpace: 'sRGB'
CPixelXDimension: 400
CPixelYDimension: 300
FocalPlaneXResolution: 8.1143e+003
FocalPlaneYResolution: 8.1143e+003
```

```
FocalPlaneResolutionUnit: 2
SensingMethod: 'One-chip color area sensor'
FileSource: 'DSC'
CustomRendered: 'Normal process'
ExposureMode: 'Auto exposure'
WhiteBalance: 'Auto white balance'
DigitalZoomRatio: 1
SceneCaptureType: 'Standard'
```

Властивість **'Bitdepth'** може приймати наступні значення:

- jpg – 8, 12, 16 для чорно-білих, 8, 12 – для кольорових;
- png – 1, 2, 4, 8, 16 для чорно-білих, 8, 16 – для кольорових, 8, 16 – для чорно-білих з каналом прозорості, 1, 2, 4, 8 – для кольорових індексованих;

Для виведення масиву **C** з зображенням на екран слугує функція **image**.

Функція має два формати виконання. Виклик високого рівня застосовує для автоматичного форматування осей вбудовану функцію **newplot**. При цьому автоматично визначаються межі осей **XLim**, **YLim**, напрям виведення зображення **YDir**. Виклики

```
<h>=image(C)
image(x,y,C)або
image(x,y,C,'PropertyName',PropertyValue,...)
```

є викликами високого рівня. Двоелементні вектори **x, y** визначають межі значень написів на осях. Використовується при необхідності цифрувати осі в одиницях, відмінних від пікселів. В разі **x(1)>x(2)** чи **y(1)>y(2)**, зображення віддзеркалюється.

Виклик низького рівня "накладає" зображення на поточні осі без виконання функції **newplot**. Аргументами функції можуть бути тільки пари "Властивість"-"Значення".

```
image('PropertyName',PropertyValue,...)
```

Основними властивостями об’єкту **image** є:

AlphaData – прозорість,
AlphaDataMapping – метод визначення прозорості,
ButtonDownFcn – дія при натисканні миші над зображенням,

CData – матриця точок зображення,

CDataMapping – спосіб визначення кольорів,

Clipping – режим обрізання зображення,

CreateFcn – функція, яка виконується перед виведенням зображення,

DeleteFcn - функція, яка виконується перед закриттям зображення,

DisplayName – назва,

EraseMode – режим виведення зображення,

HandleVisibility – видимість в програмах ,

UIContextMenu – покажчик на меню,

UserData – масив даних користувача,

Visible – видимість на екрані,

XData/ YData двоелементий вектор [1 size(CData,2)]. Слугує для визначення розміру елемента зображення на екрані

(XData(2)-XData(1))/(size(CData,2)-1).

Додатковими можливостями володіє функція **imshow** з розширення **Image Processing Toolbox**. Вона дозволяє керувати в діалоговому режимі способом виведення зображення, визначити діапазон інтенсивностей для "сірих" зображень, рамку, збільшення, задати зображення назвою файлу.

Розширені можливості забезпечує розширення **Image Processing** Toolbox:

- статистичний, морфологічний та інші аналізи зображення;
- фільтрація зображення;
- геометричні перетворення та перетворення яскравості;
- спектральні перетворення;
- обробка великих, первинних, багатокадрових зображень;
- багатопроцесорна обробка даних.

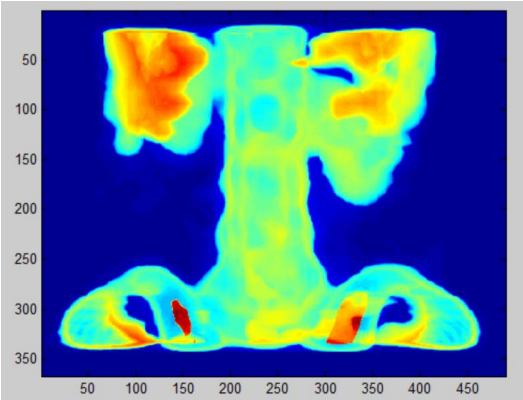
Для збереження зображення у файлі призначена функція

imwrite(A,<map>,filename,<fmt>,Property Name>,<Value>).

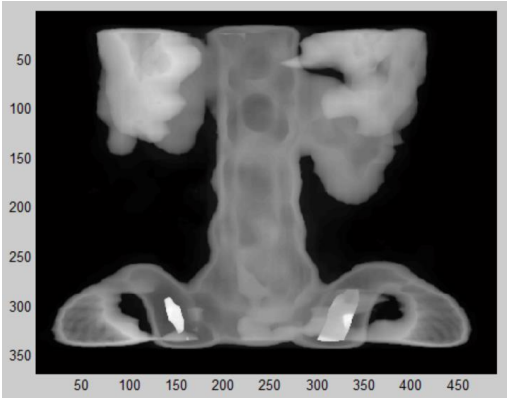
Функція зберігає масив **A** або індексований масив **A, map** у файлі **filename** у форматі <**fmt**>. За необхідності можна записати у файл дані **<Property Name>,<Value>** об’єкту **image**.

Вигляд зображення на екрані не завжди очікуваний. За замовчанням кольорові зображення виводяться на екран без корекції каналів RGB. Чорно-біле зображення на екрані не є однозначним результатом того, що виведена інформація від монохромного джерела. Графічні редактори перетворюють кольорове зображення в чорно-біле трансформацією кожної з трьох матриць кольорів. При цьому матриця зображення залишається структурою RxCx3. Навпаки, відображення монохромного джерела за замовчанням є кольоровим.

Наприклад, в пакеті мається файл даних spine.mat, який є записом кадра [367x490] з рентген апарату. При візуалізації даних зображення є кольоровим (рис. 4.1а), хоча даними не є "кольорова" матриця [367x490x3]. Для того, щоб зробити вигляд чорно-білого зображення для матриці RxCx1 більш реалістичним слід скорегувати карту кольорів (Рис. 4.1б).



а



б

Рис. 4.1 – Зображення від монохромного джерела

а – за замовчанням, б – з відкоригованою картою кольорів

Для коригування використовується матриця кольорів. Матриця повинна мати 3 стовпця. Кількість рядків довільна. Елементами рядка матриці є дійсні числа від 0.0 до 1.0, які визначають інтенсивність червоного, зеленого та синього кольору, відповідно.

Працює з матрицею кольорів функція **`colormap`**.

Виклик **`colormap('default')`** встановлює матрицю за замовчанням.

Виклик **`cmap = colormap`** зчитує в змінну значення поточної матриці.

Визначення нової стандартизованої матриці проводиться викликом

`colormap('name')` або **`colormap(name(N))`**,

де **name** – стандартизовані типи палітр (рис. 4.2): **N** – натуральне число до 65536. Визначає кількість градацій кольору.

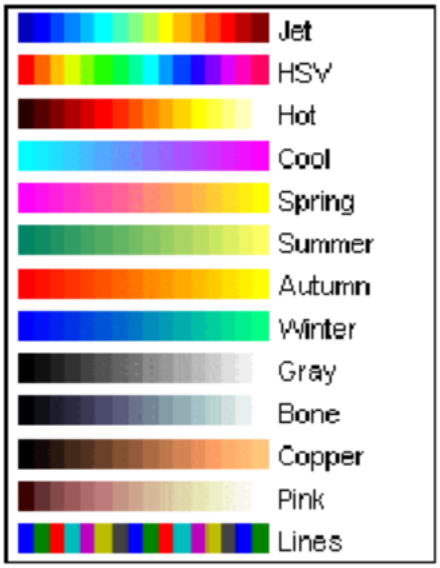


Рис. 4.2 – Стандартизовані палітри

Функція змінює матрицю кольорів для всього вікна. Керувати нею палітрами кількох зображень незалежно неможливо. Функція не впливає на кольорові зображення з матрицями RxCx3.

Трансформувати кольорову матрицю RxCx3 в чорно-білу RxCx1 мржливо двома способами: простим усередненням значень відповідних пікселів всіх трьох підматриць або зваженим усередненням з урахуванням чутливості людського ока.

При роботі з матрицями зображень слід мати на увазі, що стандартні графічні файли представляють дані в форматі цілих чисел без знаку. Матлаб працює з матрицями зображень з даними в форматі цілих чисел без знаку та дійсних чисел.

При математичних діях з елементами таких матриць слід брати до уваги особливості обробки даних різних типів.

Для переведення типів даних можна використовувати наступні вбудовані функції: **double, single, int8, int16, int32, int64, uint8, uint16 int32 uint64**.

"Ресамплінг" цифрових зображень

При обробці зображень шляхом геометричних перетворень часто розв'язується задача перерахунку координат комірок зображення. Типовим прикладом можна вважати зміну мірила зображення. В разі збільшення зображення пікселі вихідного зображення «розсуваються». Для того, щоб заповнити проміжки між розсунутими комірками застосовують інтерполяційні методи.

Вбудовані функції інтерполяції в пакеті зосереджено в папці **`polyfun`** Матлабу. Всі вбудовані функції орієнтовані на роботу з векторами та матрицями. Тобто вихідна функція повинна задаватися у вигляді вектора (матриці) в точках аргументу - вектора стовбця. В разі, коли аргумент не задано, вважається, що аргументом є вектор x=[1 .. N], де N співпадає з довжиною вектора вихідної функції.

Для одномірної інтерполяції призначено функції **`interp1q`**, **`interp1`**, **`interpft`**, **`spline`**, **`polyfit`**, **`polyval`**, для багатомірної - **`interpn`**, **`interp2`**, **`interp3`**.

Функції **`polyfit`**, **`polyval`** проводять регресійний аналіз по всіх N точках вихідних даних. В разі, коли степінь поліному N на одиницю менша розміру векторів аргументів, результатом є інтерполяційний поліном. В інших випадках – апроксимаційний.

Всі інші інтерполяційні функції Матлабу реалізовують кускову ("ковзаючи") інтерполяцію. Методи **`'nearest'`**, **`'linear'`** використовують дві найближчі точки, **`'spline'`**, **`'pchip'`**,**`'cubic'`** – чотири.

Одномірна інтерполяція

Функція **`[P<,S,M>] = polyfit(X,Y,N)`** призначена для знаходження коефіцієнтів апроксимаційного ступеневого поліному P ступеню N (4.1) по вихідних векторах X,Y

$$F(z)=P(1)\cdot z^{N} + P(2)\cdot z^{N-1} +...+ P(N)\cdot z^{1} + P(N+1)\cdot z^{0} \qquad (4.1)$$

Коефіцієнти заносяться в результуючий вектор **P** в порядку зниження ступеню. Опціональний параметр **S** є структурою, яка містить наступні поля: коефіцієнт розкладання матриці Вандермонде **R**, ступінь свободи **df**, норму нев'язки **normr**. Структура використовується функцією **`polyval`**. Опціональний параметр **M** є двоелементним вектором, що містить середнє значення та СКВ аргументу X..

Функція **[Y<,DELTA>] = polyval(P,X<,S>)** розраховує значення ступеневого поліному в точці (точках) **X**. В разі наявності опційного аргументу **S**, результуючий параметр **DELTA** – СКВ результату.

Функція **yi = interp1q(<x>,Y,xi)** повертає лінійно інтерпольовані значення вектора одновимірної функції **Y** з аргументами, які визначено вектором **x**, в точках, які визначені у векторі стовбці **xi**.

Функція

interp1(<x>,Y,xi,<method>,<'extrap'>,<extrapval>,<'pp'>)

є більш універсальною. В ній можна визначити спосіб інтерполяції: **'nearest'** – найближчий, **'linear'** – лінійний, **'spline'** – кубічний сплайн, **'pchip','cubic'-** кусковий кубічний поліном Ерміта. Для методів, окрім **'nearest'** , **'linear'**, можливо провести екстраполяцію за межами інтервалу x опцією **'extrap'** та задати фіксовані значення екстраполяції **extrapval**. Опція **'spline'** викликає функцію **spline**, опції **'pchip'**, **'cubic'** – **pchip**.

Функція **y = spline(x,y,xi)** виконує одновимірну кубічну сплайн інтерполяцію функції **y** з аргументами, які визначено вектором **x**, в точках, які визначені у векторі стовбці **xi**.

Функція **y = interpft(x,n)** виконує одновимірну інтерполяцію з використанням перетворення Фур'є. Вона повертає вектор **y** довжиною **n**, по значенням періодичної функції **x** довжиною **m (m<=n).**

Двомірна інтерполяція

Відносно зображень задача інтерполяції [1] (resampling) полягає в знаходженні кольору та інтенсивності в точці P(x,y) за даними кольору та інтенсивностей точок I вихідного зображення (рис. 4.3).

Слід пам'ятати, що порядок запису параметрів для точок зображення та елементів матриць відрізняється. Загально прийнято, що координати точки записуються в порядку x,y – P(x,y). Індекси матриць прийнято визначати в зворотному порядку: спочатку записується індекс рядка – координата y, потім записується індекс стовпця – координата x.

x_0 x x_1

y_0	I_{00}	I_{01}
y	P	
y_1	I_{10}	I_{11}

Рис. 4.3 – Схема інтерполяції точок зображення

Метод "найближчих" елементів є найбільш простим та швидким засобом інтерполяції (resampling) зображень. Параметри точки P(x,y) прирівнюються до параметрів комірки зображення I_{ji}, яка знаходиться найближче до точки. Якщо координати точки визначені в пікселях, то критерієм є округлення за арифметичними правилами результату ділення відповідної координати на відповідний індекс. В разі визначення абсолютних координат точки, критерієм слугує виділена за арифметичними правилами ціла частина від ділення відповідної координати на добуток індексу комірки та кроку між комірками.

Метод білінійної інтерполяції є розширенням лінійної інтерполяції на двомірні масиви, якими є зображення. Для розрахунку потребується чотири навколишні точки [2x2]. Білінійна інтерполяція є результатом послідовного застосування двох лінійних одномірних інтерполяцій, спочатку по одній координаті, потім по другій. Хоча метод базується на лінійних одномірних перетвореннях, він не є лінійним (4.2).

$$P(y,x) \approx a_0 + a_1 x + a_2 y + a_3 xy \qquad (4.2)$$

Для знаходження коефіцієнтів інтерполяції необхідно розв’язати систему лінійних рівнянь (4.3).

$$\begin{bmatrix} 1 & x_0 & y_0 & x_0 \cdot y_o \\ 1 & x_0 & y_1 & x_0 \cdot y_1 \\ 1 & x_1 & y_0 & x_1 \cdot y_0 \\ 1 & x_1 & y_1 & x_1 \cdot y_1 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} I_{00} \\ I_{01} \\ I_{10} \\ I_{11} \end{bmatrix} \qquad (4.3)$$

Інтерполяція виконується в два етапи. На першому етапі знаходяться комірки зображення I_{00}, I_{01}, I_{10}, I_{11}, між якими розташовується потрібна точка P(y,x) (рис. 4.3). На другому етапі проводиться інтерполяція або шляхом розв’язання системи лінійних рівнянь (4.3), або послідовно проводиться одномірна інтерполяція по одній координаті, потім з використанням знайденого значення – по другій.

В разі, коли координати точки P визначають відстань від точки P до вузла I_{00}, система рівнянь спрощується та можливо отримати обчислювальний вираз (4.4)

$$P = I_{00}(1-x)(1-y) + I_{01}(1-y)x + I_{10}(1-x)y + I_{11}xy \qquad (4.4)$$

Для біквадратичної інтерполяції (4.5) потребуються дев'ять навколишніх точок [3x3]. Розв'язання потребує знаходження 9-ти коефіцієнтів $a_{к.}$ системи (4.6). В Матлабі відсутня вбудована функція біквадратичної інтерполяції.

$$P(y,x) \approx a_0 + a_1x + a_2y + a_3xy + a_4x^2 + a_5y^2 + a_6x^2y + a_7xy^2 + a_8x^2y^2$$

$$P(y,x) \approx \sum_{i=0}^{2}\sum_{j=0}^{2} a_{ij}x^i y^j \qquad (4.5)$$

$$\begin{bmatrix} 1 & x_0 & y_0 & x_0y_0 & x_0^2 & y_0^2 & x_0^2y_0 & x_0y_0^2 & x_0^2y_0^2 \\ 1 & x_0 & y_1 & x_0y_1 & x_0^2 & y_1^2 & x_0^2y_1 & x_0y_1^2 & x_0^2y_1^2 \\ 1 & x_0 & y_2 & x_0y_2 & x_0^2 & y_2^2 & x_0^2y_2 & x_0y_2^2 & x_0^2y_2^2 \\ 1 & x_1 & y_0 & x_1y_0 & x_1^2 & y_0^2 & x_1^2y_0 & x_1y_0^2 & x_1^2y_0^2 \\ 1 & x_1 & y_1 & x_1y_1 & x_1^2 & y_1^2 & x_1^2y_1 & x_1y_1^2 & x_1^2y_1^2 \\ 1 & x_1 & y_2 & x_1y_2 & x_1^2 & y_2^2 & x_1^2y_2 & x_1y_2^2 & x_1^2y_2^2 \\ 1 & x_2 & y_0 & x_2y_0 & x_2^2 & y_0^2 & x_2^2y_0 & x_2y_0^2 & x_2^2y_0^2 \\ 1 & x_2 & y_1 & x_2y_1 & x_2^2 & y_1^2 & x_2^2y_1 & x_2y_1^2 & x_2^2y_1^2 \\ 1 & x_2 & y_2 & x_2y_2 & x_2^2 & y_2^2 & x_2^2y_2 & x_2y_2^2 & x_2^2y_2^2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \\ a_8 \end{bmatrix} = \begin{bmatrix} I_{00} \\ I_{01} \\ I_{02} \\ I_{10} \\ I_{11} \\ I_{12} \\ I_{20} \\ I_{21} \\ I_{22} \end{bmatrix} \qquad (4.6)$$

Бікубічна інтерполяція (4.7) є також результатом послідовного застосування двох одновимірних кубічних інтерполяцій, спочатку по одній координаті, потім по другій .

$$P(y,x) \approx \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j \qquad (4.6)$$

Розв'язання потребує знаходження 16-ти коефіцієнтів $a_{ij.}$

Функція Матлаб

```
interpn(<X1,X2,...,>V,<Y1,Y2,...>,<ntime>,<method>, ...
<extval>)
```

призначена для багатомірної інтерполяції Вона обчислює значення багатомірної функції **V**, що визначена на аргументах **<X1,X2,X3,...,>,** для нових аргументів - **<Y1,Y2,Y3,...>.** Інші аргументи аналогічні аргументам **interp1** функції. Можливі способи: **'nearest'**, **'linear'**, **'spline'**, **'cubic'**. Для нерівномірної сітки спосіб **'cubic'** є аналогічним до **'spline'**. Опція **ntime** веде до рекурсивного повторення інтерполяції **ntime** разів.

Для двомірної інтерполяції призначено функцію

interp2(<X>,<Y>,Z,<XI,YI>, <ntime>, <'method'>, ... <extval>)

Для пришвидшення дій рекомендується визначати метод інтерполяції у наступному вигляді: **'linear', 'cubic', 'spline', 'nearest',** **XI** – як вектор рядок або скаляр, **YI** – як вектор стовпець або скаляр.

Для створення двомірної сітки аргументів призначено функції **ndgrid** та **meshgrid**. Синтаксис та дія функцій подібні. Функції копіюють вектори аргументи в матриці результатів. Різниця полягає тільки в порядку створення матриць. В функції **meshgrid** перша матриця результат складається з однакових рядків, що є копіями першого вектора аргументу. Друга – складається з однакових стовпців, що є копіями другого вектора аргументу. В функції **ndgrid** перша матриця результат складається з однакових стовпців, що є копіями першого вектора аргументу. Друга – складається з однакових рядків, що є копіями другого вектора аргументу.

Приклади та завдання для самостійного виконання

Приклад 4.1.

Завантажити зображення сузір'я "Котяче око", яке зроблене телескопом Хабл з файлу "ngc6543a.jpg". В одному вікні вивести оригінальне зображення, чорно-біле зображення, яке отримано простим усередненням з палітрою за замовчанням та чорно-біле зображення, яке отримано простим усередненням з "гарячою" (hot) палітрою (Рис. 4.3).

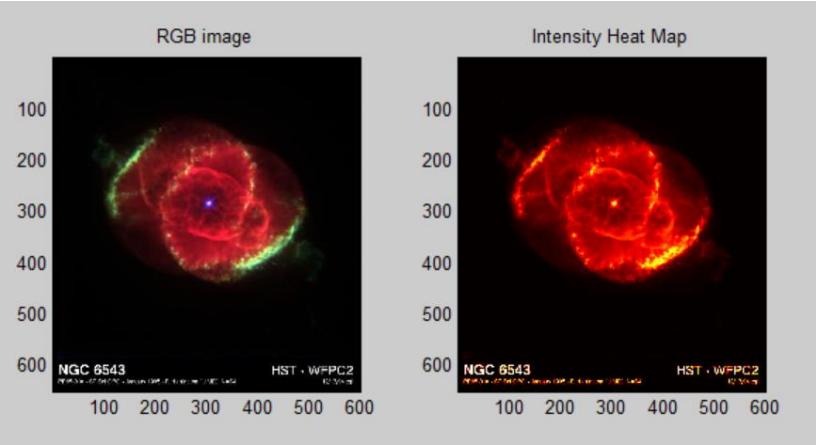


Рис. 4.3 – Зображення сузір’я з прикладу 4.1

РОЗВ’ЯЗАННЯ

```
figure
ax(1) = subplot(1,2,1);
rgb = imread('ngc6543a.jpg');
image(rgb); title('RGB image')
ax(2) = subplot(1,2,2);
im = mean(rgb,3);
image(im); title('Intensity Heat MAP’)
colormap(‘hot’)
```

Завдання 4.1.

Перетворити кольорове зображення з прикладу 4.1 на чорно-біле простим усередненням та зваженням з коефіцієнтами 0.2989, 0.5870, 0.1140 відповідно для червоного, зеленого та синього кольорів [ITU-BT.R.601 (CCIR 601)]. Вивести в одному вікні вихідне, два чорно-білі зображення, та матрицю різницю чорно-білих зображень.

Порівняти отримані "зваженим" усередненням результати з результатом вбудованої функції **`rgb2gray`** у вигляді зображення різниці та кількісно.

Приклад 4.2.

Із застосуванням середовища **GUIDE** розробити застосунок (Рис. 4.4), який:

- вводить назву графічного файлу через поле введення або кнопку **Browse** з стандартним файловим діалогом;
- зчитує графічний файл;
- виводить зображення на екран;
- для зображення виводить інформацію про повну назву файлу, розміри зображення в пікселях, розміри комірок фотоприймача;
- радіокнопками керує виглядом написів розмірів на осях: без написів, в пікселях, в мікрометрах;
- полями вибору керує видимістю сітки та її типом: центрована чи ні.

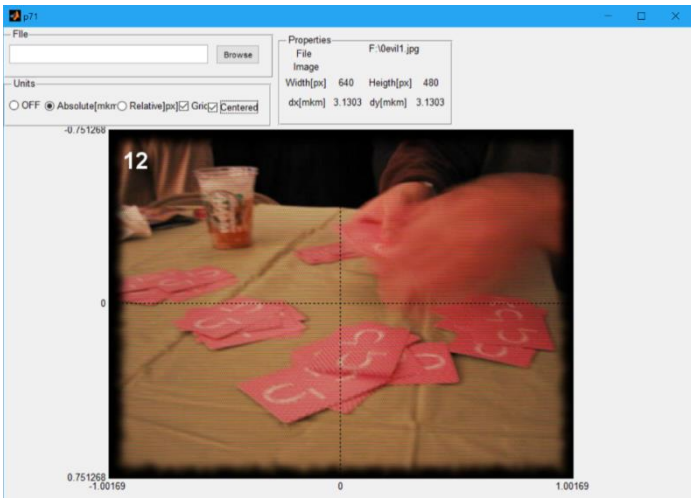


Рис. 4.4 – Застосунок прикладу 4.2

РОЗВ’ЯЗАННЯ

В середовищі **GUIDE** створити нове вікно. В ньому розмістити дві панелі **`uipanel1`**, **`uipanel2`**, групу вибору радіокнопок **`uipanel12`**, осі **`axes1`** (Рис. 4.5).

В першу панель **`uipanel1`** **File** вставити поле введення **`fileedit`**, кнопку **`browsebutton`** "**Browse**".

В другу панель **`uipanel2`** "**Properties**" вставити **`text1`** "**File**", **`text2`** "**Image**", **`text3`** "**Width(pix)**", **`text4`** "**Height(pix)**", **`text5`** "**dx(mkm)**",

text6 "**dy(mkm)**", **widthtext, heighttext, dxtext, dytext, filenametext.**

В групу радіокнопок розмістити радіокнопки **offrb** "**None**", **absoluterb** "**Mkm**", **relativerb** "**Pixels**", поля вибору **gridcheckbox** "**Grid**", **centeredcheckbox** "**Centered**". Активною поставити радіокопку **offrb.** Поля вибору та радіокнопку **absoluterb** зробити недоступними наданням значення '**off'** властивості **enable**, групу радіокнопок **uipanel12**зробити невидимою наданням значення '**off'** властивості **visible**.

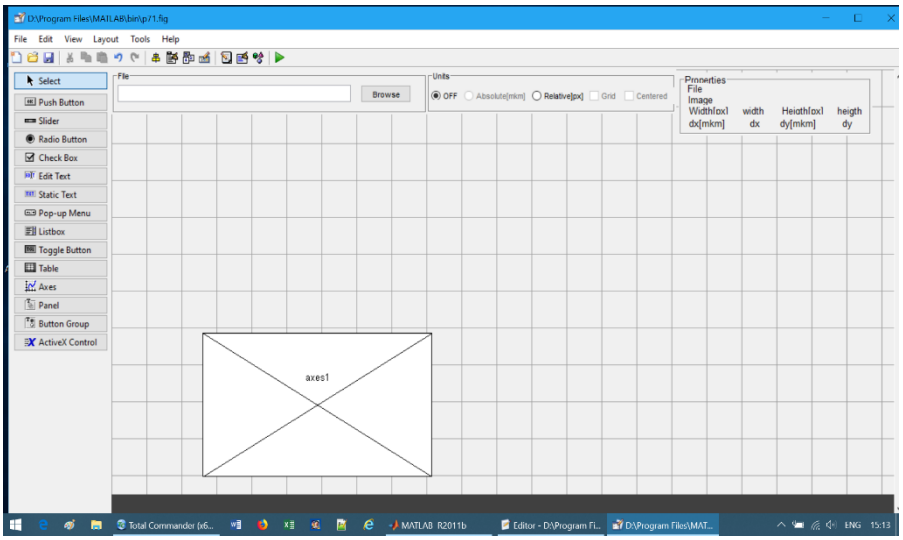


Рис. 4.5 – Заготовка вікна для прикладу 3.2.

Призначення елементів видно з їхніх назв та екранних написів.

Заготовку зберегти з назвою **p1**. Відразу після зберігання Матлаб створить відповідний програмний файл функції **p1.m**. Для кожного елемента керування автоматично створені відповідні зворотні **callback** функції.

Застосунок готовий для запуску, але ніяких дій, крім виведення самих елементів на екран, він не виконує.

Результати дії поля введення назви файлу та кнопки **Browse** однакові. Має бути зчитаний файл. Для цього потрібна повна специфікація файлу. З файлу повинно бути витягнуто зображення та **exif** інформація. Для цього доцільно створити окрему функцію **filedata**. Аргументами функції є повна специфікація файлу та структура **handles**, яку створює **GUIDE**. Структура **handles** є зручною для отримання інформації про покажчики на графічні

об’єкти та інформації користувача, яка має використовуватися різними функціями. Це робить непотрібним введення глобальних змінних.

Для отримання покажчиків на об’єкти вікна можна використати функції **findobj**, **guihandles**, але ці дії потрібно виконувати в кожній зворотній функції, що призводить до зменшення ефективності програми.

В застосунку потрібно програмно змінювати написи на осях зображення в залежності від режиму виведення інформації. Це можна зробити безпосереднім визначенням значень властивостей **XTickLAbel,'YTickLAbel','XTick','YTick'**осей. Для компактності в структуру введено поле-структура **modestring** . Поле містить значення цих полів. Для даних користувача про параметри зображення в структурі зарезервовано порожнє векторне поле **imdata**. Ці дії додаються в зворотну функцію **p1_OpeningFcn** (Рис. 4.6).

```
function p1_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
handles.modestring=struct('XTickLAbel','','YTickLAbel',
'','XTick',[],'YTick',[]);
handles.imdata=[];
guidata(hObject, handles);
initialize_gui(hObject, handles, false);
```

Рис. 4.6 – Дані користувача в функції **p1_OpeningFcn**

Зворотна функція кнопки **Browsebutton** (Рис. 4.7) формує специфікацію файлу **fullfilename** поєднанням двох рядків: шляху до файлу **path** та імені файлу **filename**. Ці рядки є результатом вбудованої функції зчитування файлів **uigetfile**. Запис формує рядок шаблону типу файлів для відображення у вікні. Вибір передбачає відображення файлів **'*.jpg'**або **'*.tif'** або **'*.png'** чи **'*.jpg;*.tif;*.png;*.gif'.** В разі відмови від вибору або помилки **uigetfile** повертає 0. Це значення є критерієм для проведення подальших дій функцією **filedata**.

```
function Browsebutton_Callback(hObject, eventdata, handles)
[filename,path]=uigetfile({'*.jpg','jpg files';…
'*.tif','tiff files';'*.png','png files';...
```

```
'*.jpg;*.tif;*.png;*.gif','All Image Files'},...
'mytitle','f:\');
if name ~=0
      fullfilename=strcat(path, filename);
      [imag]=filedata(fullfilename, handles);
end
```

Рис. 4.7 – Лістинг функції **Browsebutton_Callback**

Зворотна функція поля **fileedit** (Рис. 4.8) формує специфікацію файлу **fullfilename** зчитуванням атрибуту **String** поля введення. Перевірка існування файлу проводиться шляхом його відкриття функцією **fopen**. В разі помилки ця функція повертає від'ємне число, яке є критерієм для відмови від подальших дій. В цьому випадку функцією **set** в полі введення витирається введений користувачем рядок та «витирається» значення змінної **fullfilename**.

```
function fileedit_Callback(hObject, eventdata, handles)
fullfilename=get(hObject,'String');
fid=fopen(fullfilename);
if fid<0
    set(hObject,'String','');
    fullfilename='';
else
    fclose(fid);
        [imag ]=filedata(fullfilename, handles);
    end
```

Рис. 4.8 – Лістинг функції **fileedit_Callback**

Функція **filedata** виконує основні дії по створенню даних для виведення. Функція (Рис. 4.9):

- зчитує зображення **imag** в матрицю з файлу функцією **imread** ;
- зчитує інформацію exif в структуру **a** функцією **imfinfo**;
- «витирає» дані з відповідних текстових полів шляхом зміни властивості **String** функцією **set**. Для доступу до полів використовуються дані з структури **handles** , в якій зберігаються назви '**tag**' об'єктів;

- виводить дані про назву файлу та розміри зображення в пікселях;
- зчитує з структури **a** дані про розміри зображення в пікселях. Виводить їхні значення у відповідні тестові поля.
- зчитує поточні дані про координати та розміри осей **axes1** з властивості **Position** стуктури **handles** в вектор **tmp** функцією **get.**
- встановлює нові значення розмірів осей **axes1** у властивість **Position** та значення написів (без написів) з поля **modestring** структури **handles** функцією **set;**
- вмикає доступність поля вибору **absolutecheckbox** зміною властивості **'Enable',** видимість групи радіокнопок **uipanel12** зміною властивості **'Enable'**.
- перевіряє наявність полів **FocalPlaneXResolution, FocalPlaneYResolution, FocalPlaneResolutionUnit** функцією **isfield**. В разі наявності
- зчитує їхні значення в змінні **XR, YR, unt**;
- в залежності від одиниці вимірювання значень (міліметри – 1, дюйми - 2), визначає фізичні розміри комірок фотоприймача та виводить ці значення в мкм у відповідні текстові поля;
- дописує в структуру **handles** поле **imdata** з даними про розміри фотоприймача та зображення у вигляді вектора з чотирьох елементів функцією **guidata;**

```
function [imag ] =filedata(fullfilename,handles)
 imag=imread(fullfilename);
    set(handles.dxtext,'String','')
    set(handles.dytext,'String','')
    set(handles.widthtext,'String','')
    set(handles.heighttext,'String','')
    a=imfinfo(fullfilename);
        im(1)=a.Width;
        im(2)=a.Height;
        set(handles.widthtext,'String',a.Width)
        set(handles.heighttext,'String',a.Height)
        set(handles.uipanel12,'Visible','on')
```

```
            if isfield(a,'DigitalCamera')
                    if
isfield(a.DigitalCamera,'FocalPlaneXResolution')
        XR=a.DigitalCamera.FocalPlaneXResolution;
        YR=a.DigitalCamera.FocalPlaneYResolution;
        unt=a.DigitalCamera.FocalPlaneResolutionUnit;
                    if unt == 2
                         im(3)=25.4/XR*1000;
                         im(4)=25.4/YR*1000;
                     elseif unt== 1
                         im(3) =1/XR*1000;
                         im(4)=1/YR*1000;
                     end
    set(handles.dxtext,'String',num2str(im(3)))
    set(handles.dytext,'String',num2str(im(4)))
    set(handles.absoluterb,'Enable','on')
                end
            end
     handles.imdata=im;
     guidata(handles.figure1, handles)
     tmp=get(handles.axes1,'Position');
     tmp(3)=handles.imdata(1);
     tmp(4)=handles.imdata(2);
     set(handles.axes1,'Position',tmp)
     image(imag);
    set(handles.uipanel12,'Visible','on')
     set(handles.axes1,handles.modestring)
     set(handles.filenametext,'String',fullfilename)
```

Рис. 4.9 - Лістинг функції **filedata**

Функція **gridcheckbox_Callback** (Рис. 4.10) описує дії при зміні стану поля вибору **gridcheckbox** . Функція по покажчику з аргументу **hObject** зчитує поточний стан (1/0) поля з властивості **Value** в змінну **tmp**. В разі вимкнення елемента на осях **axes1** вимикається зображення сітки, та навпаки.

```
function        gridcheckbox_Callback(hObject,eventdata,…
handles)
tmp= get(hObject,'Value');
```

```
if tmp   grid(handles.axes1,'on')
else grid(handles.axes1,'off') end
```

Рис. 4. 10 – Лістинг функції **gridcheckbox_Callback**

Функція **Centeredcheckbox_Callback** (Рис. 4.11) описує дії при зміні стану поля вибору **Centeredcheckbox**. Функція по даних про розміри зображення та фотоприймача з масиву **imdata** структури **handles** розраховує граничні значення по координатах X та У. По покажчику з аргументу **hObject** зчитує поточний стан (1/0) поля з властивості **Value** в змінну **tmp**. В залежності від обраного режиму зміною властивостей **YTickLabel, XTickLabel** функцією **set** формує потрібні написи на осях **axes1**.

```
    function        Centeredcheckbox_Callback(hObject,…
eventdata, handles)
    tmp= get(hObject,'Value');
       st=handles.imdata(3)*handles.imdata(1)/1000/2;
       st1=handles.imdata(4)*handles.imdata(2)/1000/2;
    if tmp
        set(handles.axes1,'YTickLabel',[-st1;0.0;st1])
        set(handles.axes1,'XTickLabel',[-st;0.0;st])
    else
      set(handles.axes1,'YTickLabel',[0.0;st1;2*st1])
      set(handles.axes1,'XTickLabel',[0.0;st;2*st])
    end
```

Рис. 4. 11 – Лістинг функції **Centeredcheckbox_Callback**

Замість написання зворотних функцій для кожної радіокнопки використовується функція **uipanel12_SelectionChangeFcn** (Рис. 4.12) групи радіокнопок.

Функція отримує аргумент **eventdata**. В властивість **Tag** поля **NewValue** цього аргументу містить назву радіокнопки, яку обрано в даний час.

Функція множинного вибору **switch** по ключах цих назв виконує відповідні дії.

В разі вибору кнопки вимкнення написів **offrb** зміною властивості **Enable** об’єктів робляться недоступними поля **Centeredcheckbox,**

gridcheckbox . Написи на осях витираються в один рядок з використанням поля **modestring** структури **handles**.

В разі вибору кнопки вимкнення написів **relativerb** зміною властивості **Enable** об'єктів робляться доступними поля **Centeredcheckbox, gridcheckbox**. Змінюються властивості **xtick ytick** осей з використанням перших двох значень поля **imdata** структури **handles**, де зберігаються розміри в пікселях. Кількість елементів в векторах визначає кількість елементів векторів властивостей **xticklabel, xticklabel.** Для зменшення розміру коду, замість безпосереднього ручного визначення цих векторів, вмикається автоматичний режим **xticklabelmode**, **xticklabelmode**.

В разі вибору кнопки вимкнення написів **absoluterb** зміною властивості **Enable** об'єктів робляться доступними поля **Centeredcheckbox, gridcheckbox**. Змінюються властивості **xtick ytick** осей з використанням перших двох значень поля **imdata** структури **handles**, де зберігаються розміри в пікселях. З використання всіх елементів поля **imdata** розраховуються фізичні розміри зображення та змінюються властивості **xticklabel, xticklabel.**

```
    function      uipanel12_SelectionChangeFcn(hObject,
eventdata, handles)
    switch get(eventdata.NewValue,'Tag'

    case 'absoluterb'
            set(handles.Centeredcheckbox,'Enable','on')
            set(handles.gridcheckbox,'Enable','on')
    set(handles.axes1,'xtick',[1handles.imdata(1)/2…
handles.imdata(1)],ytick',[1   handles.imdata(2)/2   ...
handles.imdata(2)])

tmp=handles.imdata(3)*handles.imdata(1)/2000;
set(handles.axes1,'xticklabel',[0;tmp;2*tmp])
tmp=handles.imdata(4)*handles.imdata(2)/2000;
set(handles.axes1,'yticklabel',[0; tmp;2*tmp])
```

```
    case 'relativerb'
            set(handles.Centeredcheckbox,'Enable','on')
            set(handles.gridcheckbox,'Enable','on')
set(handles.axes1,'xtick',[1   handles.imdata(1)/2   ...
handles.imdata(1)],'ytick',[1   handles.imdata(2)/2   ...
handles.imdata(2)])
set(handles.axes1,'xticklabelmode','auto',           ...
'yticklabelmode','auto')

    case 'offrb'
      set(handles.Centeredcheckbox,'Enable','off')
      set(handles.gridcheckbox,'Enable','off')
          set(handles.axes1,handles.modestring)
    end
```

Рис. 4.12 – Лістинг функції **uipanel12_SelectionChangeFcn**

Завдання 4.2.

Модифікувати застосунок прикладу 4.2 для виведення інформації про глибину кольору, тип: grayscale/color, фокусну відстань, поле зору.

Приклад 4.3.

Розробити комп’ютерну модель для дослідження методів двомірної інтерполяції. Модель повинна зчитувати функцію для формування даних, значення чотирьох точок аргументів по координатах x, y відповідно, координат x, y точки інтерполяції. Розраховувати та виводити на екран значення функції в точках аргументів, інтерполяційні значення за вбудованими лінійним та кубічним методами, лінійним та квадратичним методами із розв’язанням системи лінійних рівнянь та безпосереднім розрахунком. Для всіх методів розраховувати та виводити час розрахунку та абсолютну похибку інтерполяції (рис. 4.13).

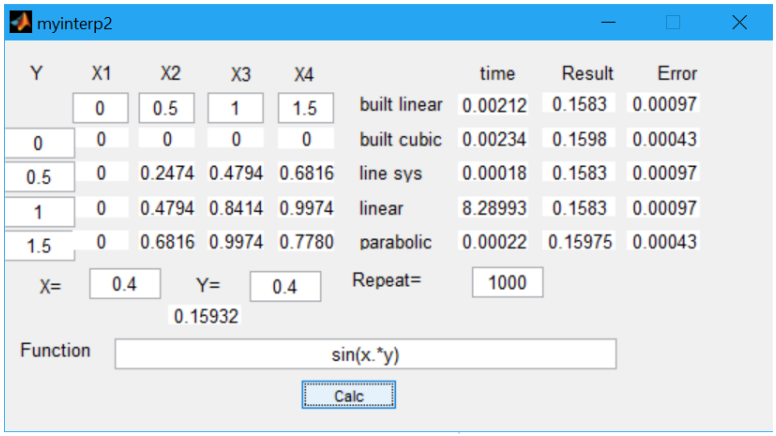


Рис. 4.13 – Вигляд вікна моделі

РОЗВ'ЯЗАННЯ

Початковими даними моделі є аргументи функції – вектори **x [0 0.5 1 1.5]** та **y [0 0.5 1 1.5]**, функція **sin(x·y)**, вектор точки інтерполяції – [**0.3 0.3**], кількість повторів – **1000**. Передача цих та інших даних між підпрограмами в структурі введено поля **x, y** – вектори точок аргументів, **z** – матриця значень функції, **point** – вектор точки інтерполяції, **q** – базова функція, **n** - кількість повторів.

Вбудована білінійна інтерполяція виконується функцією **interp2** з ключем **linear**. Вбудована бікубічна інтерполяція – функцією **interp2** з ключем **cubic.**

Для білінійної інтерполяції використано вирази (4.2) та (4.3), для біпараболічної – (4.5).

Для скорочення витрат оформлення елементів проведемо в середовищі **GUIDE**.

У вікні розташовані: поля введення опорних значень аргументів **x1, x2, x3, x4, y1, y2, y3, y4**, точки інтерполяції p, функції інтерполяції, кількості повторень, поля виведення результатів значень функції в шістнадцяти опорних точках, точці інтерполяції, значення вбудованої білінійної, бікубічної інтерполяції, білінійної та біпараболічної інтерполяції з розв'язанням систем рівнянь (4.2), (4.5), білінійної послідовної інтерполяції, часу розрахунку та абсолютної похибки кожного виду інтерполяції.

Дії компонентів

Лістинг функції керування наведено на рис. 4.14.

Поля введення даних зчитують текстовий рядок, перетворюють його на чисельне значення та записують у відповідне поле структури **handles**. Для зміни первинної функції використовується **inline** функція Матлабу, яка виконує дію, прописану в текстовому рядку свого аргументу.

Для того, щоб не повторювати однакові рядки коду в зворотних функціях обробки дій полів введення, застосовано окрему функцію користувача **myinit**. Функція виводить результати значень функції в базових точках у відповідні поля вікна після кожної зміни введених даних. Викликається в кожній зворотній функції полів даних.

Розрахунки по виразах (4.2), (4.3), (4.5) проводяться в зворотній функції кнопки **Calcbutton**. Для кожного методу запускається секундомір **tic**, **n** разів в циклі проводиться інтерполяція, результати переводяться в текстовий формат та виводяться у відповідне поле виведення.

Аналіз результатів

Результати моделювання показують, що всі методи одномірної інтерполяції забезпечують абсолютну похибку менше ніж 0.001 (0.1%). Найменшу похибку для гладких функцій серед досліджуваних методів дають біпараболічна та бікубічна інтерполяції.

Послідовне застосування одномірної інтерполяції потребує більше часу, ніж безпосереднє розв’язання системи лінійних рівнянь.

Слід відмітити, що для двомірної інтерполяції прямі методи теж є більш ефективними. Біквадратична інтерполяція має таку ж похибку, як і вбудована бікубічна інтерполяція, але потребує майже на порядок менше часу для проведення дій.

```
function varargout = myinterp2(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
           'gui_OpeningFcn', @myinterp2_OpeningFcn, ...
'gui_OutputFcn',  @myinterp2_OutputFcn, ...
                    'gui_LayoutFcn',  [] , ...
                       'gui_Callback',   []);
```

```
    if nargin && ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
    end

    if nargout
[varargout{1:nargout}]=gui_mainfcn(gui_State,        ...
varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
    % End initialization code - DO NOT EDIT

    function  myinterp2_OpeningFcn(hObject,  eventdata,
handles, varargin)
    handles.output = hObject;
    handles.q=inline('sin(x.*y)');
    handles.x=[0 0.5 1 1.5];handles.y=[0 0.5 1 1.5]
    handles.n=10;handles.point=[0.3 0.3];
    guidata(hObject, handles);
    myinit(hObject, eventdata, handles);


    function  varargout  =  myinterp2_OutputFcn(hObject,
eventdata, handles)
    varargout{1} = handles.output;

    function  pushbutton1_Callback(hObject,  eventdata,
handles)
    d=handles.q(handles.point(1),handles.point(2));
    tic;
    [X Y]=meshgrid(handles.x,handles.y);
    for i=1:handles.n
    tmp=interp2(X,Y,handles.z,handles.point(1),...
    handles.point(2),'linear');
    end
    t=toc;
    set(handles.blintimetext,'String',t);
    set(handles.blinrestext,'String',num2str(tmp,4));
```

```
    set(handles.blinerrortext,'String',                ...
num2str(abs(tmp-d),4));
    tic;
    for i=1:handles.n
    tmp=interp2(X,Y,handles.z,handles.point(1),...
    handles.point(2),'cubic');
    end
    t=toc;
    set(handles.bqtimetext,'String',t);
    set(handles.bqrestext,'String',num2str(tmp,4));
    set(handles.bqerrortext,'String',                  ...
num2str(abs(tmp-d),4));
    tic;
    for i=1:handles.n
    A=[1            handles.x(1)            handles.y(1)
handles.x(1)*handles.y(1);...
       1            handles.x(1)            handles.y(2)
handles.x(1)*handles.y(2);...
       1            handles.x(2)            handles.y(1)
handles.x(2)*handles.y(1);...
       1            handles.x(2)            handles.y(2)
handles.x(2)*handles.y(2)];
    c=linsolve(A,[handles.z(1,1);
handles.z(1,2);handles.z(2,1);handles.z(2,2)]);
tmp=c(1)+c(2)*handles.point(1)+c(3)*handles.point(2)+…
    c(4)*handles.point(1)*handles.point(2);
    %classical
    end
    t=toc;
    set(handles.lintimetext,'String',t);
    set(handles.linrestext,'String',num2str(tmp,4));
    set(handles.linerrortext,'String',                 ...
num2str(abs(tmp-d),4));
    tic;



    for i=1:handles.n
```

```
    Iz(1)=handles.z(1,1)+(handles.z(1,2)-...
    handles.z(1,1))*(handles.point(1)-...
    handles.x(1))/(handles.x(2)-handles.x(1));
    Iz(2)=handles.z(2,1)+(handles.z(2,2)-...
    handles.z(2,1))*(handles.point(1)-...
    handles.x(1))/(handles.x(2)-handles.x(1));
    tmp=Iz(1)+(Iz(2)-Iz(1))*(handles.point(2)-...
    handles.y(1))/(handles.y(2)-handles.y(1));
    end
    t=toc;
    set(handles.lin2timetext,'String',t);
    set(handles.lin2restext,'String',num2str(tmp,4));
    set(handles.lin2errortext,'String',              ...
num2str(abs(tmp-d),4));
    tic;
    for i=1:handles.n
    tmp=quadinterp2(handles.x,handles.y,handles.z,...
handles.point);
    end
    t=toc;
    set(handles.partimetext,'String',t);
    set(handles.parrestext,'String',num2str(tmp,4));
    set(handles.quaderrortext,'String',              ...
num2str(abs(tmp-d),4_);

function Fedit_Callback(hObject, eventdata, handles)
    tmp=get(hObject,'String');
    handles.q=inline(tmp);
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles);

function x1edit_Callback(hObject, eventdata,  handles)
    handles.x(1)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles);


function X2edit_Callback(hObject, eventdata, handles)
```

```
    handles.x(2)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles)';

function X3edit_Callback(hObject, eventdata, handles)
    handles.x(3)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles);

function Xedit_Callback(hObject, eventdata, handles)
    handles.point(1)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles);

function Nedit_Callback(hObject, eventdata, handles)
    handles.n=str2int(get(hObject,'String'));
    guidata(hObject,handles);

function myinit(hObject, eventdata, handles)
    [X Y]=meshgrid(handles.x,handles.y);
    handles.z=handles.q(X,Y);
    guidata(hObject,handles);
    set(handles.realtext,'string',num2str(...
    handles.q(handles.point(1),handles.point(2))));
set(handles.z11text,'string',num2str(handles.z(1,1)));
set(handles.z12text,'string',num2str(handles.z(1,2)));
set(handles.z13text,'string',num2str(handles.z(1,3)));
set(handles.z14text,'string',num2str(handles.z(1,4)));
set(handles.z21text,'string',num2str(handles.z(2,1)));
set(handles.z22text,'string',num2str(handles.z(2,2)));
set(handles.z23text,'string',num2str(handles.z(2,3)));
set(handles.z24text,'string',num2str(handles.z(2,4)));
set(handles.z31text,'string',num2str(handles.z(3,1)));
set(handles.z32text,'string',num2str(handles.z(3,2)));
set(handles.z33text,'string',num2str(handles.z(3,3)));
set(handles.z34text,'string',num2str(handles.z(3,4)));
set(handles.z41text,'string',num2str(handles.z(4,1)));
set(handles.z42text,'string',num2str(handles.z(4,2)));
```

```
set(handles.z43text,'string',num2str(handles.z(4,3)));
set(handles.z44text,'string',num2str(handles.z(4,4)));

function Yedit_Callback(hObject, eventdata, handles)
    handles.point(2)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles);

function Y1edit_Callback(hObject, eventdata, handles)
    handles.y(1)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles)';

function Y2edit_Callback(hObject, eventdata, handles)
    handles.y(2)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles)';

function Y3edit_Callback(hObject, eventdata, handles)
    handles.y(2)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles)';

function Y4edit_Callback(hObject, eventdata, handles)
    handles.y(4)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles)';

function X4edit_Callback(hObject, eventdata, handles)
    handles.x(4)=str2num(get(hObject,'String'));
    guidata(hObject,handles);
    myinit(hObject, eventdata, handles)';

function p1=quadinterp2(x,y,I,z)
A=[1   x(1)   y(1)   x(1)*y(1)   x(1)^2   y(1)^2   ...
x(1)^2*y(1) x(1)*y(1)^2 x(1)^2*y(1)^2 ;...
1   x(1)   y(2)   x(1)*y(2)   x(1)^2   y(2)^2   ...
 x(1)^2*y(2) x(1)*y(2)^2 x(1)^2*y(2)^2;...
```

```
1    x(1)    y(3)    x(1)*y(3)    x(1)^2    y(3)^2    ...
 x(1)^2*y(3) x(1)*y(3)^2 x(1)^2*y(3)^2;...
1    x(2)    y(1)    x(2)*y(1)    x(2)^2    y(1)^2    ...
 x(2)^2*y(1) x(2)*y(1)^2 x(2)^2*y(1)^2;...
1    x(2)    y(2)    x(2)*y(2)    x(2)^2    y(2)^2    ...
 x(2)^2*y(2) x(2)*y(2)^2 x(2)^2*y(2)^2;...
1    x(2)    y(3)    x(2)*y(3)    x(2)^2    y(3)^2    ...
 x(2)^2*y(3) x(2)*y(3)^2 x(2)^2*y(3)^2;...
1    x(3)    y(1)    x(3)*y(1)    x(3)^2    y(1)^2    ...
 x(3)^2*y(1) x(3)*y(1)^2 x(3)^2*y(1)^2;...
1    x(3)    y(2)    x(3)*y(2)    x(3)^2    y(3)^2    ...
 x(3)^2*y(2) x(3)*y(2)^2 x(3)^2*y(2)^2;...
1    x(3)    y(3)    x(3)*y(3)    x(3)^2    y(3)^2    ...
 x(3)^2*y(3) x(3)*y(3)^2 x(3)^2*y(3)^2];
d=[I(1,1);I(1,2);I(1,3);I(2,1);I(2,2);I(2,3);I(3,1);...
I(3,2);I(3,3)];
    c=linsolve(A,d);
p1=c(1)+c(2)*z(1)+c(3)*z(2)+c(4)*z(1)*z(2)+          ...
c(5)*z(1)^2+c(6)*z(2)^2+...
c(7)*z(1)^2*z(2)+c(8)*z(1)*z(2)^2+c(9)*z(1)^2*z(2)^2;
```

Рис. 4.14 – Лістинг моделі прикладу 4.3

Завдання 4.3. Додати до розробленого в завданні 4.2 застосунку поле зміни мірила та групу радіокнопок методу інтерполяції: найближча, лінійна, параболічна, кубічна. При введенні значення мірила зображення перемальовується з інтерполяцією відповідним методом.

Пояснення.

Вихідне зображення міститься в матриці E розміром RxC. Сусідні елементи матриці відповідають сусіднім пікселям відображення на екрані. При цьому відповідні сусідні точки зображення P знаходяться одна від одної на відстані dx по горизонталі та dy по вертикалі. В найпростішому випадку dx=dy=1. При введенні мірила «к» відстань між точками зображення змінюється на dx/k та dy/k відповідно. Значення елементів матриці перестають співпадати зі значеннями координат точок зображення.

Перерахунок параметрів зображення проводиться відносно базової точки, від якої розраховуються відстані. Базовою точкою може бути точка

середини зображення або кутова точка зображення. Наприклад, при значення мірила к=1.5 для базової точки лівого верхнього кута зображення (елемент матриці з індексами 1, 1) використати безпосереднім перенесенням значень з вихідної матриці в масштабовані можна тільки третину елементів (рис. 4.15). Для отримання значень інших елементів масштабованої матриці слід застосувати інтерполяцію.

$E_{1,1}$=P(0,0)	$E_{1,2}$=P(0,dx)	$E_{1,3}$=P(0,2dx)	$E_{1,4}$=P(0,3dx)
$E_{2,1}$=P(dy,0)	$E_{2,2}$=P(dy,dx)	$E_{2,3}$=P(dy,2dx)	$E_{2,4}$=P(dy,3dx)
$E_{3,1}$=P(2dy,0)	$E_{3,2}$=P(2dy,dx)	$E_{3,3}$=P(2dy,2dx)	$E_{3,4}$=P(2dy,3dx)
$E_{4,1}$=P(3dy,0)	$E_{4,2=}$P(3dy,dx)	$E_{4,3}$=P(3dy,2dx)	$E_{4,4}$=P(3dy,3dx)

а)

$Q_{1,1} = P(0,0) = E_{1,1}$	$Q_{1,2} = P(0,\frac{dx}{1.5})$	$Q_{1,3} = P(0,\frac{2dx}{1.5})$	$Q_{1,4} = P(0,2dx)$
$Q_{2,1} = P(\frac{dy}{1.5},0)$	$Q_{2,2} = P(\frac{dy}{1.5},\frac{dx}{1.5})$	$Q_{2,3} = P(\frac{dy}{1.5},\frac{2dy}{1.5})$	$Q_{2,4} = P(\frac{dy}{1.5},2dx)$
$Q_{3,1} = P(\frac{2dy}{1.5},0)$	$Q_{3,2} = P(\frac{2dy}{1.5},\frac{dx}{1.5})$	$Q_{3,3} = P(\frac{2dy}{1.5},\frac{2dx}{1.5})$	$Q_{3,4} = P(\frac{2dy}{1.5},2dx)$
$Q_{4,1} = P(2dy,0) = E_{3,1}$	$Q_{4,2} = P(2dy,\frac{dx}{1.5})$	$Q_{4,3} = P(2dy,\frac{2dx}{1.5})$	$Q_{4,4} = P(2dy,2dx) = E_{3,3}$

б)

Рис. 4.15 – Матриці зображень: а – вихідна, б – масштабована з к=1.5

Аргументами вбудованої функції двомірної інтерполяції є дві матриці координат вихідних точок X, У, двомірна матриця значень функції, та дві матриці або вектори нової відмасштабованої координатної сітки. Результатом є матриця значень функції в вузлах відмасштабованої сітки. В разі проведення масштабування зображення (resampling) матрицею функції є канал зображення.

Послідовність дій для максимального використання вбудованих можливостей:

- виділення з матриці зображень трьох підматриць каналів кольору.

- створення матриць вихідних точок функцією **meshgrid**. Для бази у верхньому лівому куті вектор - аргумент функції - це значення в діапазоні від 0 до С. С – величина на 1 менша за кількість пікселів зображення по відповідній координаті.
- створення матриць відмасштабованої сітки функцією **meshgrid**. Для бази у верхньому лівому куті вектор аргумент функції - це значення в діапазоні від 0 до С/К. С – величина на 1 менша за кількість пікселів зображення по відповідній координаті, К – мірило.
- виклик функції інтерполяції для кожної виділеної підматриці.
- поєднання підматриць в матрицю зображення.

Вбудована функція двомірної інтерполяції не реалізує біквадратичну інтерполяцію, тому для цього методу слід розробити окрему процедуру.

Аргументи функції рекомендується зробити подібними до аргументів вбудованої функції **interp2**. Послідовність дій функції може виглядати наступним чином:

- Для кожного елемента матриці Е обрати по співпадінню індексів відповідну точку відмасштабованих координат (хн, ун).
- Серед елементів вихідних координат знайти індекси рядка «r» та стовпця «c» точки, яка є найближчою точкою, більшою до (хн, ун).
- З матриць вихідних координат вирізати триелементні вектори по знайдених індексах рядка «r» та стовпця «c».
- Сформувати матрицю функції розміром 3x3 вирізанням з матриці зображення Е елементи праворуч індексу «c» та нижче індексу «r».
- Викликати функцію біпараболічної інтерполяції **quadinterp2**.

Практикум 5. Моделювання зображуючих оптико-електронних систем

Мета роботи – вивчення моделей дисторсійних оптичних систем та на здобуття студентами навичок в комп'ютерному моделюванні властивостей оптичних систем.

Завдання роботи:

Набуття навичок із розрахунках параметрів оптичних систем по бітовим зображенням, перетворенню зображень по різним моделям систем.

Теоретичні положення

"Геометричні" моделі оптичних систем

Ширококутні оптичні системи "риб'яче око" знаходять широкого вжитку в застосуваннях, де потрібен дуже широкий кут огляду, близький до 180^0. Зображення таких систем супроводжується суттєвими абераціями дисторсії.

Для опису систем типу "риб'яче око" використовують радіально симетричні проекційні моделі. В цих моделях враховується залежність радіуса відстані від оптичної осі системи r в залежності від кута зору Θ та фокальної відстані f ' (рис. 5.1).

Модельні вирази [1] визначають значення радіуса точки зображення за значеннями фокальної відстані та кута зору точки предмета в системі координат, яка є центрованою відносно оптичної вісі.

Перспективна (стандартна) модель:

$$r = f' \cdot tg(\vartheta) \tag{5.1}$$

Стереографічна модель:

$$r = 2 \cdot f` \cdot \mathrm{tg}(\frac{\vartheta}{2}) \tag{5.2}$$

Ортогональна модель:

$$r = f` \cdot \sin(\vartheta) \tag{5.3}$$

Еквідістантна модель:

$$r = f` \cdot \vartheta \tag{5.4}$$

Еквісолідна модель:

$$r = 2 \cdot f` \cdot \sin(\frac{\vartheta}{2}) \qquad (5.5)$$

Декартові координати в центрованій системі координат визначаються як

$$x = r \cdot \cos(\varphi) \quad y = r \cdot \sin(\varphi) \qquad (5.6)$$

в абсолютній системі координат –

$$x = x_c + r \cdot \cos(\varphi) \quad y = y_c + r \cdot \sin(\varphi) \qquad (5.6')$$

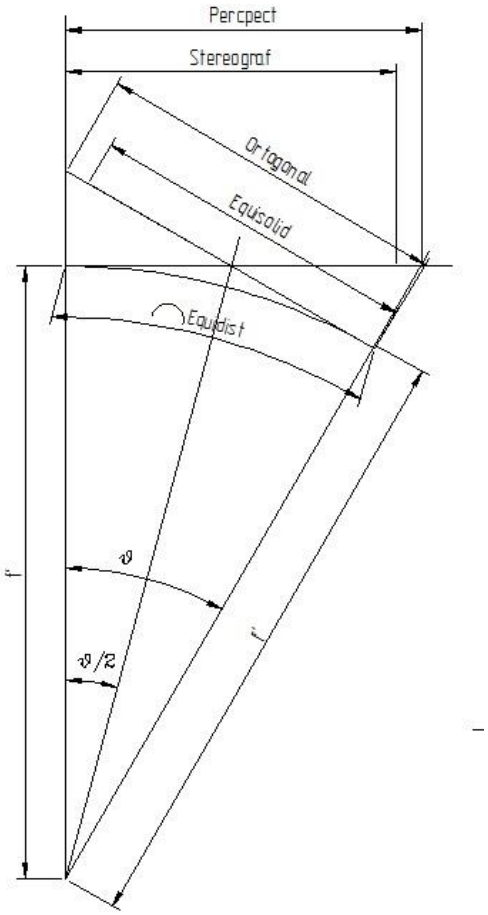


Рис. 5.1 – Моделі дисторсійної оптичної системи

Для перерахунку зображень різних моделей використовується властивість, за якою точки зображення знаходяться на одній прямій незалежно від типу моделі. Тобто *поточний кут зору є сталим для всіх моделей.*

Вихідними даними для перерахунків моделей є абсолютні значення фокусної відстані f, повного горизонтального кутового поля зору Θ, яке спирається на граничне коло зображення, значення радіуса кола поля зору R_{pix}, координат центра кола поля зору (x_{cpix}, y_{cpix}) в точках зображення для первинної моделі.

Послідовність дій для перерахунку зображень моделей без зміни кількості точок зображення може виглядати наступним чином:

1. Визначення абсолютного радіуса поля зору r по виразах (5.1) – (5.5) для моделі результату з урахування того, що повне поле зору дорівнює подвійному значенню максимального кута зору.
2. Визначення фізичного періоду комірок моделі результату

$$dxres=r/R_{pixx} \quad dyres=r/R_{pixy}$$

3. Для кожної i-ї точки моделі результату (x_i, y_i) в пікселях розрахунок фізичних значень поточного азимутального кута

$$\varphi_i = arctg(\frac{dy(y_{cpix}-y_i)}{dx(x_i-x_{cpix})}) \tag{5.7}$$

та поточного радіуса

$$r_i = \sqrt{dx^2(x_i - x_{cpix})^2 + dy^2(y_i - y_{cpix})^2} \tag{5.8}$$

4. Розрахунок поточного кута зору θi для моделі результату за відомими f та r_i по виразах (5.1 – 5.5).
5. По значенню кута зору θi та азимутального кута φi для моделі результату в центрованій системі розрахунок фізичних значень поточного радіуса rk по виразах (5.1 - 5.5) та декартових фізичних координат точки по виразах (5.6) в площині початкової моделі в центрованій системі.
6. Перерахунок фізичних декартових координат точки моделі результату з центрованої системи в пікселі в абсолютній системі.
7. Знаходження освітленості точки з розрахованими координатами методами двомірної інтерполяції (робота 4).

Для отримання координат центру зображення та радіуса поля зору для дисторсійних зображень системами "риб'яче око", поле зору яких повністю вписано в зображення хоча б по одній координаті, слід використати факт того, що точки (x_i, y_i) межі поля зору знаходяться на колі з радіусом R та координатами центру (x_c, y_c) (5.9)

$$(x_i - x_c)^2 + (y_i - y_c)^2 - R^2 = 0 \tag{5.9}$$

В такому разі визначення проводиться як апроксимаційна задача знаходження трьох параметрів моделі функціональної регресії, в якій спочатку

проводиться усереднення по всім вихідним точкам (x_i, y_i) і усередненні дані використовуються для однократного знаходження параметрів кола.

$$f(x_c, y_c, R) = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_c)^2 + (y_i - y_c)^2 - R^2 = 0 \qquad (5.10)$$

Варіант 1. Пряме визначення потребує розв'язання системи нелінійних рівнянь (5.11).

$$\begin{cases} \dfrac{\partial f(x_c, y_c, R)^2}{\partial x_c} = \sum_i ((x_i - x_c)^2 + (y_i - y_c)^2 - R^2)\cdot 2\sum_i (x_c - x_i) = 0 \\ \dfrac{\partial f(x_c, y_c, R)^2}{\partial y_c} = \sum_i ((x_i - x_c)^2 + (y_i - y_c)^2 - R^2)\cdot 2\sum_i (y_c - y_i) = 0 \\ \dfrac{\partial f(x_c, y_c, R)^2}{\partial R} = \sum_i ((x_i - x_c)^2 + (y_i - y_c)^2 - R^2)\cdot(-2R) = 0 \end{cases} \qquad (5.11)$$

Варіант 2а. Лінеаризація моделі (5.10) [2] введенням проміжних змінних $z_i^2 = x_i^2 + y_i^2$, $c = R^2 - (x_c^2 + y_c^2)$ спрощує модель:

$$fl(x_c, y_c, c) = \sum_{i=1}^{N}(x_i^2 - 2x_c x_i + x_c^2 + y_i^2 - 2y_c y_i + y_c^2 - R^2) =$$

$$= \sum_{i=1}^{N}(z_i^2 - c - 2x_c x_i - 2y_c y_i) = 0$$

Врахування того, що

$$\begin{cases} \dfrac{\partial fl(x_c, y_c, c)^2}{\partial x_c} = -2\sum_{i=1}^{N} x_i \\ \dfrac{\partial fl(x_c, y_c, c)^2}{\partial y_c} = -2\sum_{i=1}^{N} y_i \\ \dfrac{\partial fl(x_c, y_c, c)^2}{\partial c} = N + 1 \end{cases} \qquad (5.12)$$

дозволяє звести систему нелінійних рівнянь до системи лінійних рівнянь (5.13), що дозволяє значно спростити обчислення та зменшити обчислювальні похибки.

$$2 \cdot x_c \cdot \sum_{i=1}^{N} x_i^2 + 2 \cdot y_c \cdot \sum_{i=1}^{N} x_i \cdot y_i + c \cdot \sum_{i=1}^{N} x_i = \sum_{i=1}^{N} x_i \cdot z_i^2$$

$$2 \cdot y_c \cdot \sum_{i=1}^{N} y_i^2 + 2 \cdot x_c \cdot \sum_{i=1}^{N} x_i \cdot y_i + c \cdot \sum_{i=1}^{N} y_i = \sum_{i=1}^{N} y_i \cdot z_i^2 \qquad (5.13)$$

$$2 \cdot x_c \cdot \sum_{i=1}^{N} x_i + 2 \cdot y_c \cdot \sum_{i=1}^{N} y_i + c \cdot N = \sum_{i=1}^{N} z_i^2$$

Варіант 2б. Аналогічного результату можна досягти використанням канонічної форми рівняння кола (5.14).

$$f(D,E,F) = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_c)^2 + (y_i - y_c)^2 - R^2 = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

(5.14)

Для кола A=C=1, B=0, D=-2x_c , E=-2y_c ,F=x_c^2+y_c^2-R^2. З урахуванням того, що

$$\begin{cases} \dfrac{\partial f(D,E,F)^2}{\partial D} = \sum_{i=1}^{N} x_i \\ \dfrac{\partial fl(D,E,F)^2}{\partial E} = \sum_{i=1}^{N} y_i \\ \dfrac{\partial fl(D,E,F)^2}{\partial F} = N + 1 \end{cases} \qquad (5.15)$$

система нелінійних рівнянь зводиться до системи лінійних рівнянь (5.16).

$$D \cdot \sum_{i=1}^{N} x_i^2 + E \cdot \sum_{i=1}^{N} x_i \cdot y_i + F \cdot \sum_{i=1}^{N} x_i = -\sum_{i=1}^{N} (x_i^2 + y_i^2) x_i$$

$$D \cdot \sum_{i=1}^{N} x_i y_i + E \sum_{i=1}^{N} y_i^2 + F \cdot \sum_{i=1}^{N} y_i = -\sum_{i=1}^{N} y_i \cdot (x_i^2 + y_i^2) \qquad (5.16)$$

$$D \sum_{i=1}^{N} x_i + E \cdot \sum_{i=1}^{N} y_i + F \cdot N = -\sum_{i=1}^{N} x_i^2 + y_i^2$$

Варіант 3. Багаторазове проведення розрахунку параметрів кола по трьох точках (x_A, y_A) , (x_B, y_B) . (x_C, y_C) з усього набору вихідних даних з наступним усередненням результатів.

Координати центра кола можуть бути обраховані по виразах (5.17), (5.18)

$$x_c = \frac{1}{D}\begin{bmatrix} x_A^2 + y_A^2 & y_A & 1 \\ x_B^2 + y_B^2 & y_B & 1 \\ x_C^2 + y_c^2 & y_C & 1 \end{bmatrix} \quad y_c = \frac{1}{D}\begin{bmatrix} x_A^2 + y_A^2 & x_A & 1 \\ x_B^2 + y_B^2 & x_B & 1 \\ x_C^2 + y_c^2 & x_C & 1 \end{bmatrix} \quad D = \begin{bmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{bmatrix} \tag{5.17}$$

$$E = x_B^2 - x_c^2 + y_B^2 - y_c^2 \quad F = x_C^2 - x_A^2 + y_C^2 - y_A^2 \quad H = x_A^2 - x_B^2 + y_A^2 - y_B^2$$

$$D = x_A(y_B - y_C) + x_B(y_C - y_A) + x_C(y_A - y_B)$$

$$x_0 = -\frac{y_A E + y_B F + y_C H}{2D} \quad y_0 = \frac{x_A E + x_B F + x_C H}{2D} \quad R = \frac{abc}{S}, \tag{5.18}$$

де a, b, c – довжини сторін трикутника, S – площа трикутника.

Засоби Матлаб для розв'язання систем рівнянь

Системи лінійних рівнянь, яка записана в матричному вигляді $A\vec{x} = \vec{b}$ в Матлаб розв’язуються за допомогою оператора «\» x= A\b.

Безпосередньо для розв’язання системи нелінійних рівнянь $F(\vec{x}) = 0$ призначено функцію **fsolve**. В разі запису цільової функції оптимізації у вигляді системи нелінійних рівнянь для знаходження коренів системи можуть застосовуватись функції **fminsearch** (симплекс метод), **fminunc** (метод Ньютона, квазіньютонівський метод DFP), **lsqnonlin** (метод НК, Ньютона або Левенберга) з розширення **Optimization toolbox**.

Синтаксис всіх наведених функцій має однаковий виглід:

```
x,<fval>,<exitflag>,<output>,<jacobian>]=
MLABFUN(fun,x0,<options>,...),
```

де **x** - вектор коренів системи, **fun** – функція опису системи (цільової функції). Приймає аргумент-вектор значень x, повертає результат-вектор значень рівнянь системи. В разі застсування методів розв’язання з використанням аналітично визначеного Якобіану, опис функції повинен містити розрахунок останнього. Використання аналітично визначеного Якобіану проводиться записом опції в параметрі

options=optimset('Jacobian','on'). В такому випадку функція повинна мати результат – вектор з двох елементів. Перший елемент – вектор значень рівнянь, другий – матриця значень Якобіану; **x0** – вектор початкових значень. Точка початку пошуку; **options** – опціональна структура опису умов роботи функції **fsolve**; **fval** – опціональний результат. Вектор значень рівнянь системи в точці x; **exitflag** – опціональний результат. Код ознаки закінчення дії функції; **output** – опціональний результат. Структура з описом ходу розв'язання системи; **<options>** - опціональний результат. Структура, яка керує виглядом результатів обчислень та процесом обчислень. Значення структури слід попередньо задати за допомогою функції **optimset**; **jacobian** – опціональний результат. Матриця Якобіан перших похідних рівнянь системи.

Ім'я функції **fun** при викликах може задаватися у вигляді рядка з описом функції, покажчиком на в m-файл, в якому описана функція, рядком з іменем функції з m-файлу, за допомогою «**inline**» функції. Найбільш універсальним вважаються виклики за покажчиком та іменем функції. Особливістю таких записів є те, що обробляється тільки один аргумент. В разі параметризації функції кількома аргументами рекомендується застосування вкладених чи анонімних функцій.

Наприклад, розв'язання системи

$$\begin{cases} 2x_1 - x_2 - e^{-x_1} = 0 \\ 2x_2 - x_1 - e^{-x_2} = 0 \end{cases}$$

з початковою точкою (-5, 5) може виглядати наступним чином:

```
function F=myfun(x)
F=[2*x(1)-x(2)-exp(-x(1));   -x(1)+2*x(2)-exp(-x(2))];
options=optimset('Display','iter');
%Option to display output
[x,fval]=fsolve(@myfun, [-5; 5], options)
```

```
Iteration Func-count    f(x)     Norm of  step   First-
order optimality    Trust-region radius
    0        3        23535.6  1      2.29e+004    1
    1         6  6001.72   1  5.75e+003            1
    2         9  1573.51   1  1.47e+003            1
    3        12  427.226   1  388                   1
```

```
    4        15  119.763   1  107                    1
    5        18  33.5206   1  30.8                   1
    6        21  8.35208   1  9.05                   1
    7        24  1.21394   1  2.26                   1
    8        27 0.016329   0.759511   0.206   2.5
    9        30 3.51575e-006 0.111927 0.00294  2.5
   10        33 1.64763e-013 0.00169132 6.36e-007  2.5
```

Equation solved.fsolve completed because the vector of function values is near zeroas measured by the default value of the function tolerance, and the problem appears regular as measured by the gradient.

```
x =     0.5671     0.5671

fval =   1.0e-006 *        -0.4059       -0.4059
```

Розв’язання системи з додатковим параметром "c"

$$\begin{cases} 2x_1 - x_2 - e^{-Cx_1} = 0 \\ 2x_2 - x_1 - e^{-Cx_2} = 0 \end{cases}$$

може виглядати наступним чином:

а) з використанням анонімних функцій.

```
function F = myfun(x,c)
F=[2*x(1)-x(2)-exp(c*x(1)); x(1)+2*x(2)-exp(c*x(2))];

c = -1; % define parameter first
x = fsolve(@(x) myfun(x,c),[-5;-5])
```

б) з використання вкладених функцій

```
function y = myfun(x,c,x0)
options=optimset('Display',  'off');  %  Turn  off
Display
 y = fsolve(@(x) myfun11(x,c),x0);


function F = myfun11(x,c)
```

```
F=[2*x(1)-x(2)-exp(c*x(1));x(1)+2*x(2)-exp(c*x(2))];
    end % end myfun11
     end  % end myfun
```

Для використання оптимізуючих засобів Матлаб вихідна функція рівняння має бути записана у вигляді цільової функції. Наприклад, для рінняння кола (5.10) :

```
function F = myfun11(x,y,z)
F=sum(((x-z(1)).^2+(y-z(2)).^2-z(3)^2).^2);
end
```

Розв'язання симплекс методом проводиться функцією **fminsearch**.

Для застосування метода Ньютона в функції **fminunc** потрібно задати параметри **'Large scale','On'** та **'Gradobj','on'**в структурі **optimse** та передбачити аналітичний розрахунок градієнта в описі функції **fun.**

Квазіньютонівський метод з чисельним розрахунком градієнта викликається за умови вмикання алгоритму середньої розмірності та чисельного розрахунку. Тому в функції **fminunc** потрібно задати параметри **'Large scale'='Off'** та **'Gradobj'='off'** в структурі **optimset**.

Квазіньютонівський метод з аналітичним розрахунком градієнта викликається за умови вмикання алгоритму середньої розмірності та аналітичного розрахунку. Тому в функції **fminunc** потрібно задати параметри **'Large scale'='Off'** та **'Gradobj'='on'** в структурі **optimset**

Завдання 5.1. Розробити застосунок та провести комп'ютерний експеримент з дослідження методів оцінки параметрів кола (5.10). Вихідні параметри: координати центра кола x_c, y_c, радіус R, кількість точок на колі N, кількість повторів циклу розрахунків, амплітуда "білого" шуму noise.

Проаналізувати час та похибку знаходження точки центру кола та радіуса методом трикутників (5.17), розв'язанням лінійних систем (5.13),

(5.16), розв'язанням нелінійних систем (5.11) функцією fsolve, симлекс методом, ньютонівським та квазіньютоніським методами для

Можливий вигляд вікна застосунку наведено на рис. 5.2.

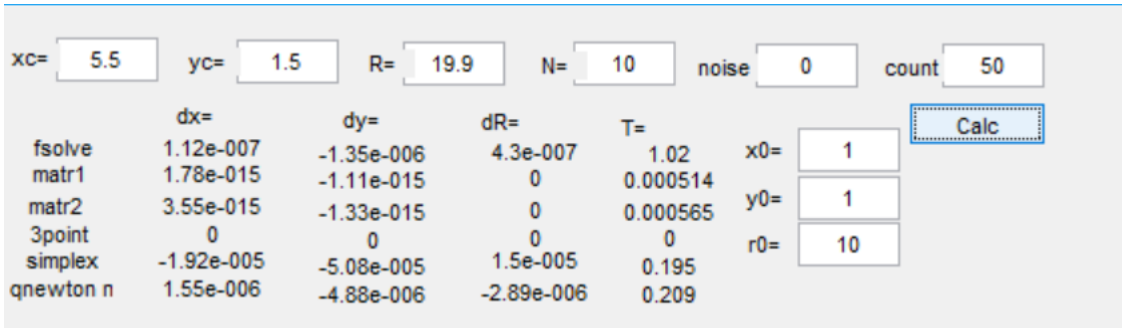


Рис. 5.2 – Вікно застосунку завдання 5.1

Пряме знаходження кола поля зору

Аналіз геометричних елементів зображення може проводитися послідовно над R, G, B матрицями кольорового зображення з наступним усередненням результатів, або однократно після перетворення кольорового зображення в чорно-біле.

Пряме знаходження з похибкою не більше Err контуру кола поля зору в матриці зображення M у вигляді масивів xcirc ycirc, які містять індекси точок контура, складається з наступних дій:

1. Задати відносний поточний поріг освітленості $T_п$=0.1.
2. Визначити поточну порогову освітленість через мінімальну $E_{мін}$, максимальну $E_{макс}$ освітленості зображення M, поріг $T_п$

$$E_п = E_{мін} + (E_{мін} + E_{макс}) \cdot T_п$$

3. Задати значення кількості точок count=1.
4. В циклі по всіх рядках виконати дії п. 5 – 10.
5. Задати початкове значення індексу стовпців 1.
6. В умовному циклі з умовою, що $M_{рс}<T_п$ та індекс стовпця менший ширини зображення, переглянути стовпці зліва направо.
7. В разі, коли індекс стовпця менший ширини зображення, збільшити кількість точок count та заповнити елементи масивів ycirc(count), xcirc(count) значеннями індексів рядка та стовпця відповідно.
8. Задати початкове значення індексу стовпців рівним ширині зображення

9. В умовному циклі з умовою, що $M_{pc}<T_п$ та індекс стовпця більший за 0, переглянути стовпці зправа наліво.
10. В разі, коли індекс стовпця більший за 0, збільшити кількість точок count та заповнити елементи масивів ycirc(count), xcirc(count) значеннями індексів рядка та стовпця відповідно.
11. По значенням координат точок ycirc, xcirc розрахувати радіус та координати центра кола в пікселах.
12. Розрахувати вектор радуса кола для всіх точок ycirc, xcirc. Розрахувати поточну середньоквадратичну помилку.
13. В разі, коли помилка більше визначеної Err та поточний поріг менше 0.9, повернутися на п.2. В разі, коли помилка більше визначеної, видати вікно підтвердження подальших дій з інформацією про значення помилки.

Визначення кола поля зору засобами Матлабу

Вбудовані засоби знаходження контурів працюють з бінарним зображенням.

Функція **im2bw(M, threshold)** виконує бінаризацію чорно-білого або кольорового зображення **M** по порогу **threshold**. Поріг є дійсним значенням в діапазні від 0 до 1.

Оптимальне значення порогу **threshold** бінаризації зображення M методом Отсу виконує функція **graythresh (M)**. Але вона працює з чорно-білим зображенням M, тому потрібна або перетворення кольорового зображення в чорно-біле функцією **rgb2gray**, або послідовна обробка R, G, B матриць кольорів зображення.

Координати контура розраховує функція

bwtraceboundary(BW, P, fstep <,conn, N>), де **P** – двоелементний вектор з номерами рядка, стовпця точки початку пошуку, **fstep** – рядок, який визначає початковий напрям пошуку (рис. 5.3).

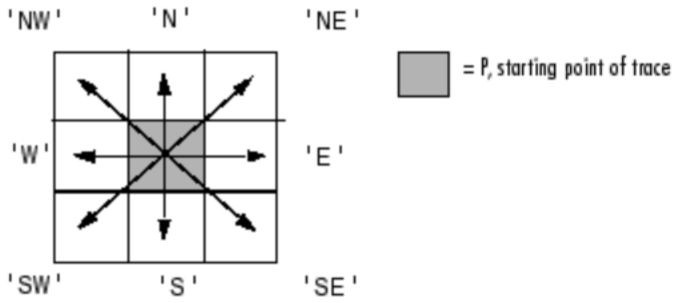


Рис. 5.3 - Значення аргументу **`fstep`**

Функція повертає матрицю розміром Qx2, де Q – кількість точок на контурі. Елемент містить індекси рядків та стовпців відповідних точок.

conn – опціональний аргумент, який визначає кількість напрямів, в яких потребується безперервність контуру. Можливі значення: 4, 8 (діє за замовчанням).

Так, як вихідне зображення M для функції є бінарним, то визначити індекси початкової точки можна послідовним перебором точок по рядках та стовпцях, з перериванням циклів при виконанні умови Mij >0. Наприклад,

```
while M(i,j) break end
```

N – опціональний аргумент, який визначає кількість точок в матриці результату.

Завдання 5.2. Розробити програму та провести дослідження точності та швидкості способів визначення контурів кола вбудованими функціями та прямим методом по кількох зображеннях.

Завдання 5.3. Розробити програму, яка зчитує обраний графічний файл. та переводить зображення з однієї моделі дисторсії в іншу за виразами 5.1 – 5.5.

Варіанти індивідуальних завдань

№	Тип джерела	Тип результату	Тип інтероляції
1	Перспективний	Ортографічний	Найближчий

2	Перспективний	Стереографічний	Білінійний
3	Перспективний	Еквідістантний	Біквадратичний
4	Перспективний	Еквісолідний	Найближчий
5	Ортографічний	Перспективний	Білінійний
6	Ортографічний	Стереографічний	Біквадратичний
7	Ортографічний	Еквідістантний	Найближчий
8	Ортографічний	Еквісолідний	Білінійний
9	Стереографічний	Перспективний	Біквадратичний
10	Стереографічний	Ортографічний	Найближчий
11	Стереографічний	Еквідістантний	Білінійний
12	Стереографічний	Еквісолідний	Біквадратичний
13	Еквідістантний	Перспективний	Найближчий
14	Еквідістантний	Ортографічний	Білінійний
15	Еквідістантний	Стереографічний	Біквадратичний
16	Еквідістантний	Еквісолідний	Найближчий
17	Еквісолідний	Перспективний	Білінійний
18	Еквісолідний	Ортографічний	Біквадратичний
19	Еквісолідний	Стереографічний	Найближчий
20	Еквісолідний	Еквідістантний	Білінійний

Література

1. **T. Nathan Mundhenk** Techniques for Fisheye Lens Calibration using a Minimal Number of Measurements / T. Nathan Mundhenk, Michael J. Rivett, Xiaoqun Liao, Ernest L. Hall // Proc. SPIE.- 2001.- vol.4197.- Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision.- DOI: 10.1117/12.403762
2. **Juho Kannala** A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses / Juho Kannala, Sami S. Brandt // IEEE Transactions on Pattern Analysis and Machine Intelligence.- 2006.- Volume:28.- Issue: 8.- p.1335–1340.- DOI:10.1109/TPAMI.2006.153
3. MATLAB. Image Processing Toolbox. User guide [Елекронний ресурс] / The MathWorks. Inc. - 2016.- 800р.- Режим доступу: http://mathworks.com/ help/pdf_doc/ matlab/index.html.- Назва з екрану
4. MATLAB. Image Processing Toolbox. User guide [Елекронний ресурс] / The MathWorks. Inc. - 2016.- 800р.- Режим доступу: http://mathworks.com/ help/pdf_doc/ matlab/index.html.- Назва з екрану

Додаток А. Основні функції Матлаб

Спеціальні функції

besselj(n, Z) – функція Бесселя першого роду n порядку;
bessely(n, Z) – функція Бесселя другого роду n порядку;
besseli(n, Z) – модифікована функція Бесселя першого роду n порядку;
besselk(n, Z) – модифікована функція Бесселя другого роду n порядку.
beta(Z,W) – бета-функція;
betainc(X,Z,W) – неповна бета-функція;
gamma(Z) – гамма-функція;
gammainc(Z,A) – неповна гамма-функція;
gammaln (Z,A) – логарифм гамма-функція;
legendre(n,X) – узагальнена функція Лежандра;
ellipj(U,M) – еліптична функція Якоби;
ellipke(M) – повний еліптичний інтеграл;
erf(X) – функція похибки (гаусов розподіл);
erfc(X) – додаткова функція похибок;
erfcx(X) – нормована додаткова функція похибок;
erfinv(Y) – зворотна функція похибок.

Інші функції

tic – запуск таймера;
toc – зупинка таймера, повернення часу з моменту запуску в секундах,
clock – поточний час. Результат – вектор з 6 чисел: рік, місяць, день, година, хвилина, секунда.
etime(t1,t2) – час між моментами, які завдано векторами t2, t1. Вектори формату **clock**.
cputime – час роботы процесора в мілісекундах зі старту пакета.

Матричні функції

rand(M,N) – створення матриці розміром (M*N) з випадкових чисел, які рівномірно розподілені в діапазоні від 0 до 1,
randn(M,N) – створює матрицю розміром (M*N) з випадкаових чисел, які розподілені по гаусову закону с нульовим середнім,
mean(a) – середнє значення елементів вектору,

Функції введення – виведення та перетворення

num2str(X) – перетворення числа в рядок. Форма символьного представлення визначається режимом виведення чисел на екран (Numeric Format).
inttostr(X) – перетворення цілого числа в рядок.
mat2str(A,[n]) – перетворення матриці в рядок. Результат – рядок в квадратних дужках. Опціональний параметр «n», визначає кількість знаків в результаті.
eval(['string']) – перетворення рядка в матрицю.

Імовірносні функції

RAND – генерує псевдовипадкові числа з рівномірним розподілом в діапазоні (0..1) мультиплікативним конгруентним методом.
RAND(N) – генерує матрицю N x N
S = RAND(metod) – зчитує 35-ти елементний вектор стану генератора.
RAND(metod,S) – установлює генератор в стан S.
RAND(metod,0) – установлює вихідний стан генератора.

Параметр «**metod**» може приймати значення:

'twister' – використовує метод Mersenne Twister (Nishimura, Matsumoto). Є основним в версіях вище 7.4. Забезпечує період повторення $(2^{19937}-1)/2$.

'state' – використовує метод Marsaglia's Subtract-with-Borrow. Був основним в версіях з 5.0 до 7.3. Забезпечує період повторення 2^{1492}.

'seed' –- використовує конгруентний метод Був основним в версії 4.0. Забезпечує повторення період $2^{31}-1$.

Функції розрахунку щільності імовірності

binopdf(X,N,P) – розрахунок функції щільності розподілу за біноміальним законом. Параметри: X - аргумент, N, P – параметри розподілення (N – натуральне, P – повинно знаходитись в діапазоні [0 1]).
betapdf(X,A,B) – розрахунок функції щільності розподілу за Бета законом. Параметри: X – аргумент в діапазоні [0 1], A, B – параметри розподілення (A, B > 0).
chi2pdf(X,V) – розрахунок функції щільності розподілу за законом Хі-квадрат. Параметри: X – аргумент в діапазоні [0 1], V – параметр розподілення (V > 0).
exppdf(X,M) – розрахунок функції щільності розподілу за експоненціальним законом. Параметри: X - аргумент, M – параметр розподілення (M > 0).

fpdf(X,V1,V2) – розрахунок функції щільності розподілу за законом Фішера. Параметри: X – аргумент (X>0), V1, V2 – параметри розподілення (натуральні числа).

gampdf(X,A,B) – розрахунок функції щільності розподілу за гамма законом. Параметри: X – аргумент (X>0), A, B – параметри розподілення (натуральні числа).

geopdf(X,P) – розрахунок функції щільності розподілу за геометричним. Параметри: X – аргумент, P – параметр розподілення.

hygepdf(X,M,K,N) – розрахунок функції щільності розподілу за гіпергеометричним законом. Параметри: X – аргумент (X>0), M, K, N (N<M) – параметри розпроділу, натуральні числа.

lognpdf(X,M,S) – розрахунок функції щільності розподілу за логнормальним законом. Параметри: X – аргумент, M (середнє), S (СКВ) – параметри розподілу.

ncfpdf(X,N1,N2,D) – розрахунок функції щільності розподілу за зсунутим законом Фішера: X - аргумент, N1 (кількість ступенів свободи), N2 , D (зсув) – параметри розподілу.

nctpdf(X,A,B) – розрахунок функції щільності розподілу за зсунутим законом Стьюдента. Параметри: X – аргумент, A (кількість ступенів свободи), B (зсув) – параметри розподілу.

ncx2pdf(X,A,B) – розрахунок функції щільності розподілу за зсунутим Хі-квадрат законом. Параметри: X – аргумент, A – кількість ступенів свободи, B (зсув) – параметри розподілу.

normpdf(X,M,A) – розрахунок функції щільності розподілу за нормальним законом. Параметри: X – аргумент, , M – середнє, A – СКВ. За замовчанням M=0, A=1 (розподіл Гауса).

pdf('n',X,A,B,C) – розрахунок функції щільності розподілу за заданим '**n**' законом. Параметри: X – аргумент, A, B, C – параметри розподілення, «n» – ім'я розподілу: **'beta', 'bino', 'chi2', 'exp', 'f', 'gam', 'geo','hyge', 'logn', 'nbin', 'ncf', 'nct', 'ncx2', 'norm', 'poiss', 'rayl', 't', 'unif', 'unid', 'weib'**.

poisspdf(X,A) – розрахунок функції щільності розподілу за законом Пуасона. Параметри: X – аргумент, A – параметр розподілення (A>0).

raylpdf(X,A) – розрахунок функції щільності розподілу за законом Релея. Параметри: X – аргумент, A – параметр розподілення

tcdf(X,A) – розрахунок функції щільності розподілу за законом Стьюдена. Параметри: X – аргумент, A – параметр розподілення.

unidpdf(X,A) – розрахунок функції щільності розподілу за дискретним рівномірним законом. Параметри: X – аргумент, A – параметр розподілення (A – натуральне).

unifpdf(X,A,B) – розрахунок функції щільності розподілу за рівномірним законом в діапазоні [A … B]. Параметри: X – аргумент, A, B – параметри розподілення.

weibpdf(X,A,B) – розрахунок функції щільності розподілу за законом Вейбула. Параметри: X – аргумент, A, B – параметри розподілення (A, B>0).

Функції розрахунку імовірності

Функціїї повертають значення функції імовірності обраного закону.

Імена та синтаксис функцій практично ідентичні іменам функцій щільності розподілу та відрізняються тільки закінченням не **pdf**, а **cdf**.

cdf('n',X,A,B,C) – розрахунок імовірності за заданим **'n'** законом. Параметри: X – аргумент, A, B, C – параметри розподілення, «n» – ім'я розподілу: **'beta' , 'bino', 'chi2', 'exp', 'f', 'gam', 'geo','hyge', 'logn', 'nbin', 'ncf', 'nct', 'ncx2', 'norm', 'poiss', 'rayl', 't', 'unif', 'unid', 'weib'**.

Функції розрахунку зворотних інтегральних функцій розподілу.

Функції повертають значення квантилю розподілу, тобто значення аргументу функції імовірності, який забезпечує обрану імовірність.

Імена та синтаксис функцій практично ідентичні іменам функцій щільності розподілу та відрізняються тільки закінченням **inv**.

Першим полем аргументів задається потрібне значення імовірності, далі задаються параметри розподілу.

icdf('n',P,A,B,C) – розрахунок квантилю розподілу за заданим **'n'** законом. Параметри: P – аргумент, A, B, C – параметри розподілення, «n» – ім'я розподілу: **'beta' , 'bino', 'chi2', 'exp', 'f', 'gam', 'geo','hyge', 'logn', 'nbin', 'ncf', 'nct', 'ncx2', 'norm', 'poiss', 'rayl', 't', 'unif', 'unid', 'weib'**.

Функції генерації випадкових чисел

Функції повертають згенероване за обраним законом вірогідносне число.

Імена та синтаксис функцій практично ідентичні іменам функцій щільності розподілу та відрізняються тільки закінченням **rnd**.

Аргументами є параметри розподілу та розміри матриці результату.

Наприклад:

*(A,B) генерує (матрицю 1x1) розподілу з параметрами A, B.

*(A, B, c) генерує матрицю c x c розподілу з параметрами A, B.

*(A, B, c, e) генерує матрицю c x e розподілу з параметрами A, B.

random('n',A,B,C,m,e)– генерація імовірносного числа за заданим **'n'** законом. Параметри: A, B, C – параметри розподілення, «**m,e**» -розміри матриці результату, «n» – ім'я розподілу: **'beta' , 'bino', 'chi2', 'exp', 'f', 'gam', 'geo','hyge', 'logn', 'nbin', 'ncf', 'nct', 'ncx2', 'norm', 'poiss', 'rayl', 't', 'unif', 'unid', 'weib'**.

Функції розрахунку моментів розподілу

MEAN(X<,M>) – розрахунок середнього. **X** – аргумент–вектор (матриця) даних, **M** - опціональний аргумент, номер стовбця матриці даних X.

STD(X,FLAG,M) – обчислення СКВ. X – аргумент-вектор (матриця) даних, M – опціональний аргумент, номер стовбця матриці даних X, FLAG – ознака нормалізації СКВ. FLAG=0 – нормалізація (N-1), FLAG <>0 – N.

VAR – обчислення дисперсії. Синтаксис аналогічний функції **STD**.

Назви функцій, які повертають середнє та дисперсію обраного розподілу відрізняються закінченням **stat**. Аргументами функцій є парметри розподілу. Результатом – вектор з двох елементів: середнього та дисперсії. Наприклад: **[M, V] = *stat(A,B)**

Функції оцінки параметрів закону розподілу

Функції повертають значення параметрів законів розподілу по відомим даним. Імена функцій відрізняються закінченням **fit**.

Аргументами є вектор (матриця) даних та опціонально – відносна похибка оцінки. За замовчанням похибка 0.05 (5%), результат – вектор параметрів.

Наприклад, функція **`[m,u] = expfit(r,0.01)`** поверне параметри експоненціального розподілу даних з вектора r з точністю 99%.

mle('n',r,alpha) – оцінка параметрів розподілу даних з вектора **«r»** за заданим **'n'** законом з точністю «**alpha**», «n» – ім'я розподілу: **`'beta'`**, **`'bino'`**, **`'chi2'`**, **`'exp'`**, **`'f'`**, **`'gam'`**, **`'geo'`**,**`'hyge'`**, **`'logn'`**, **`'nbin'`**, **`'ncf'`**, **`'nct'`**, **`'ncx2'`**, **`'norm'`**, **`'poiss'`**, **`'rayl'`**, **`'t'`**, **`'unif'`**, **`'unid'`**, **`'weib'`**.