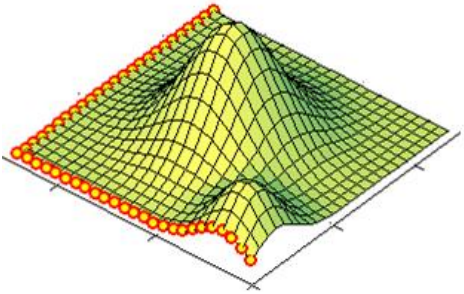
КОМП'ЮТЕРНІ СИСТЕМИ



Міністерство освіти і науки України
Чернівецький національний університет
імені Юрія Федьковича

Підлягає поверненню на кафедру

КОМП’ЮТЕРНІ СИСТЕМИ

Методичні вказівки до лабораторних робіт

Чернівці
Чернівецький національний університет ім. Ю. Федьковича
2021

УДК 004.7 (076.5)
К 637

Друкується за ухвалою редакційно-видавничої ради
Чернівецького національного університету
імені Юрія Федьковича

К 637 **Комп'ютерні системи** : методичні вказівки до лабораторних робіт / укл.: Баловсяк С. В., Одайська Х. С. Чернівці : Чернівецький національний університет ім. Ю. Федьковича, 2021. 72 с.

Дисципліна "Комп'ютерні системи" читається студентам спеціальності 123 "Комп'ютерна інженерія" ОПП "Комп'ютерна інженерія" та ОПП "Програмування мобільних і вбудованих комп'ютерних систем та засобів інтернету речей". Методичні вказівки сприятимуть підготовці до виконання п'яти лабораторних робіт, які дозволять практично засвоїти теоретичний матеріал дисципліни. Лабораторні роботи адаптовані до кредитно-модульної системи навчання. Середовище виконання лабораторних робіт – система MATLAB.

Для студентів вищих навчальних закладів.

УДК 004.7 (076.5)

ВСТУП

Завдання навчальної дисципліни „Комп'ютерні системи" – на основі отриманих теоретичних знань виробити у студентів уміння користуватися існуючими, а також створювати власні комп'ютерні системи, зокрема кластерні; розробляти й аналізувати топології комп'ютерних систем, розробляти програмне забезпечення для паралельних і розподілених комп'ютерних систем.

Мета навчальної дисципліни – надати студентам систематизовані знання про завдання та принципи роботи комп'ютерних систем, відомості про структуру паралельних і розподілених комп'ютерних систем, зокрема про векторні та векторно-конвеєрні комп'ютерні системи, матричні обчислювальні системи, організацію пам'яті та введення-виведення в комп'ютерних системах, надійність та експлуатацію комп'ютерних систем, сучасні обчислювальні системи та їх топології, зокрема комп'ютерні системи класу MIMD, симетричні мультипроцесорні системи, кластерні обчислювальні системи, системи з масовою паралельною обробкою, обчислювальні системи з неоднорідним доступом до пам'яті.

У результаті вивчення дисципліни студенти повинні:

- **ЗНАТИ** основні архітектури комп'ютерних систем, зокрема векторних, векторно-конвеєрних і матричних систем, систем класу MIMD, симетричних мультипроцесорних систем, кластерних системи та систем з масовою паралельною обробкою; топології комп'ютерних систем і способи забезпечення їх відмовостійкості.
- **ВМІТИ** аналізувати та проектувати топології комп'ютерних систем, створювати програмне забезпечення для кластерних комп'ютерних систем, розробляти програми для виконання паралельних обчислень, підвищувати надійність і відмовостійкість комп'ютерних систем, зокрема за допомогою відмовостійкої дискової пам'яті.

Кожна лабораторна робота містить: тему, мету та завдання роботи; опис необхідного обладнання та програмного забезпечення; короткі теоретичні відомості, необхідні для виконання лабораторної роботи; порядок виконання роботи; контрольні запитання для самоперевірки; список літератури.

Звіт по виконаній лабораторній роботі має містити:

– номер і тему лабораторної роботи;

– мету роботи;

– опис використаних апаратно-програмних засобів;

– порядок виконання лабораторної роботи з коротким аналізом виконаних етапів роботи;

– результати виконання роботи у вигляді тексту, схем, графіків, таблиць, схем алгоритмів.

Середовище виконання лабораторних робіт – система MATLAB.

Мета **роботи № 1** „Векторизація циклів у системі MATLAB" полягає у вивченні принципів ефективного послідовного програмування в системі MATLAB, зокрема принципів векторизації циклів; під час виконання роботи студенти навчаються створювати власні швидкодіючі програми.

У **роботі № 2** „Паралельне програмування в системі MATLAB у режимі pmode" вивчаються принципи паралельного програмування в системі MATLAB, зокрема програмування в режимі pmode; студенти навчаються керувати планувальником задач і робочими процесами, виконувати паралельні обчислення.

У **роботі № 3** „Паралельне програмування в системі MATLAB із використанням m-файлів" розглядаються принципи паралельного програмування у системі MATLAB, зокрема програмування з використанням m-файлів.

Мета **роботи № 4** „Розпаралелювання згортки зображень у системі MATLAB" – вивчення принципів паралельного програмування в системі MATLAB, зокрема згортки цифрових зображень за допомогою паралельних робочих процесів.

Робота № 5 „Моделювання решітчастих топологій обчислювальних систем" передбачає вивчення основних характеристик решітчастих топологій обчислювальних систем, зокрема плоскої решітчастої топології.

Вищевказані лабораторні роботи входять до складу двох змістових модулів: змістовий модуль 1 містить роботи № 1 та № 2, змістовий модуль 2 – роботи № 3, № 4, № 5. Рекомендується оцінювати знання та вміння студентів за результатами виконання лабораторних робіт так: робота № 1 – максимальна оцінка 6 балів, № 2 – 6 балів, № 3 – 6 балів, № 4 – 6 балів, № 5 – 6 балів.

ЛАБОРАТОРНА РОБОТА № 1

ВЕКТОРИЗАЦІЯ ЦИКЛІВ У СИСТЕМІ MATLAB

Мета: вивчити принципи ефективного послідовного програмування в системі MATLAB, зокрема принципи векторизації циклів; навчитися створювати власні швидкодіючі програми в системі MATLAB.

Завдання: створити програму, призначену для обчислення значень функції, у вигляді модуля в системі MATLAB; дослідити швидкодію створеної програми при обчисленні значень функції без векторизації циклів та з векторизацією циклів.

Обладнання: персональний комп'ютер (ПК).

Програмне забезпечення: система MATLAB.

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Основні відомості про систему MATLAB

MATLAB (скорочення від англ. «Matrix Laboratory») – це система (пакет прикладних програм), розроблена для обробки масивів і виконання матричних операцій [1, 2]. MATLAB – це також мова програмування, що застосовується в однойменній системі.

Головним вікном системи MATLAB є командне вікно «Command Windows». У цьому вікні відображаються команди, які набираються користувачем із клавіатури, а також результати виконання команд. Ознакою того, що система готова до сприйняття та виконання наступної команди, є поява в останньому рядку текстового поля командного вікна знака запрошення «>>».

В арифметичних виразах мови MATLAB використовуються типові позначення для операцій, наприклад: «+» – додавання, «-» – віднімання, «*» – множення, «/» – ділення, «^» – піднесення до степеня.

Система MATLAB має кілька зарезервованих імен змінних:

- *j* – уявна одиниця;
- *pi* – число π;
- *inf* – позначення машинної безкінечності;
- *NaN* – позначення невизначеного результату (наприклад, 0/0);
- *eps* – похибка операцій над числами із плаваючою комою;
- *ans* – результат останньої операції без знака присвоювання;
- *realmax* – максимальна величина числа;
- *realmin* – мінімальна величина числа.

До математичних функцій MATLAB належать, зокрема, *sin* (*x*), *cos* (*x*), *tan* (*x*), *exp* (*x*), *log* (*x*) – натуральний логарифм, *sqrt* (*x*) – корінь квадратний числа *x*, *abs* (*x*) – модуль числа *x*, *round* (*x*) – округлення числа *x* до найближчого цілого.

Створення програми в системі MATLAB здійснюється за допомогою вбудованого редактора. Код програми зберігається в m-файлах (тестових файлах з розширенням *.m). Новий m-файл можна створити через меню «File/New». Відкриття і збереження існуючих m-файлів відбувається через меню «File».

Текст програми оформлюється згідно з такими правилами:

1. Кожен рядок програми закінчується символом «;».
2. Якщо в одному рядку розміщено кілька команд, то вони відділяються символами «;» або «,».
3. Довгу команду можна записувати у кілька рядків, при цьому попередній рядок команди повинен завершуватися трьома крапками «...».
4. Рядок програми, який починається із символу «%», не виконується і сприймається як коментар.
5. Ім’я змінної може містити до 30 символів і не повинно збігатися з іменами функцій системи MATLAB і системних змінних; система розрізнює ВЕЛИКІ і малі. В іменах змінних можна використовувати тільки символи латинського алфавіту та цифри, назва змінної повинна починатися з букви.
6. Змінні не оголошуються, але перед використанням змінної їй потрібно присвоїти певне значення.

Під вектором в MATLAB розуміється одновимірний масив чисел, а під матрицею – багатовимірний. У випадку поелементних операцій перед знаком «*», «/» або «^» потрібно вставити крапку (наприклад, поелементне множення векторів *a* та *b* записується у вигляді *a.*b*). В іншому випадку виконуються векторні операції.

Графіки будуються функцією *plot* і виводяться в окреме графічне вікно (*figure*). Функція *loglog* аналогічна функції *plot*, але графіки будуються в логарифмічному масштабі.

Розглянемо приклад (лістинг 1.1) програми MATLAB (файл «p_CS_Lab_1_V0.m»), призначеної для обчислення значень функції *z* (*x*), заданої в *Q* точках. На початку програми встановлюються параметри функції *z* (*x*): мінімальне і максимальне значення *x*, кількість точок *Q*. Залежно від варіанта функція *z* (*x*) обчислюється через функції *exp* (*x*), *sin* (*x*) або *cos* (*x*).

Лістинг 1.1. Фрагмент коду програми «p_CS_Lab_1_V0.m»

```
  % p_CS_Lab_1_V0; Назва модуля програми
clear all; close all; clc;    % видалення змінних з оперативної пам'яті
disp('   ---- p_CS_Lab_1_V0 ---- '); % повідомлення про початок
% роботи програми в командному вікні (Command Windows)
  % Set z(x) parameter; Встановити параметри функції z(x) :
x_min=1;   % мінімальне значення x записується у змінну x_min
x_max=4.9E3;  % максимальне значення x, 4.9E3 = 4.9 · 10^3
Q=1E4; % Кількість точок, в яких потрібно обчислити значення x та z
x_step=(x_max-x_min)/(Q-1);       % обчислення кроку зміни x
i0=1;  % origin i for plot (visualization), i - number x(i)
% i0 – значення i, починаючи з якого потрібно відобразити графік z(x)
Qp=100;  % Quantity i for plot, Qp - кількість точок на графіку z(x)
Tx=50;   % Період функцій sin або cos, які описують функцію z(x)
SigmaG=(Qp/Q)*(x_max-x_min)/6;  % SigmaG – середнє квадратичне
  % відхилення (СКВ) для розподілу Гауса, який описує функцію z(x)
 mG=x_min+SigmaG*3;              % mG  – центр розподілу Гауса
 disp('SigmaG='); disp(SigmaG);
  % показати SigmaG в командному вікні

% Calculation z(x):  обчислення значень x, z  в циклі (Cycle)
Mode_Calc_c=1;    % 0- Not Calc Cycle; 1- Calc Cycle
  % Змінна Mode_Calc_c дозволяє (1) або забороняє (0)
  % обчислення значень x, z в циклі (без векторизації)
if Mode_Calc_c==1  %  обчислення значень x, z дозволено
  time_begin=clock;  % time_begin : час початку розрахунків

% Calculation z(x), обчислення значень масивів
  vx0c=zeros(0); % ініціалізація масиву (створення порожнього масиву)
  vz0c=zeros(0);
for i=1:1:Q              % цикл по i від 1 до Q  з кроком 1
 vx0c(i)=x_min+(i-1)*x_step; % обчислити елемент матриці значень x
 vz0c(i)=sin(2*pi*vx0c(i)/Tx); % обчислити елемент матриці vz0c
 %vz0c(i)=exp(-(vx0c(i)-mG).^2/(2*SigmaG.^2)); % елемент z /exp/
end;                     % кінець циклу по i
   vxc=zeros(0);    % vxc – фрагмент vx0c, на основі якого потрібно
                    % побудувати графік з кількістю точок Qp
   vzc=zeros(0);
for i=1:1:Qp
  vxc(i+i0-1)=vx0c(i);  % matrix x /Cycle/ for plot
  vzc(i+i0-1)=vz0c(i);  % matrix z /Cycle/ for plot
end; % i
```

```
  % clock - [year,month,day,hour,minute,second]
  time_end_c = clock;         % Time end (Cycle)
  mdtime=time_end_c-time_begin;
  tc=(((mdtime(3)*24+mdtime(4))*60+mdtime(5))*60)+mdtime(6); % sec
  disp(' tc,s = ');  disp(tc);  % час обчислення значень  z(x) в циклі
end; % Mode_Calc_c=1; завершення обчислення z(x) в циклі

% Plot (visualization) z(x) / Cycle / побудова графіку z(x)
    % встановити розмір шрифту на графіку
set(0,'DefaultAxesFontSize',13,'DefaultAxesFontName','Arial Cyr');
    % опис кольорів ліній і маркерів на графіку
    % в також типів ліній
    %    Various line types, plot symbols and colors may be obtained with

  %   PLOT(X,Z,S) where S is a character string made from one element
  %   from any or all the following 3 columns:
  %         b     blue          .     point              -     solid
  %         g     green         o     circle             :     dotted
  %         r     red           x     x-mark             -.    dashdot
  %         c     cyan          +     plus               --    dashed
  %         m     magenta       *     star             (none)  no line
  %         y     yellow        s     square
  %         k     black         d     diamond
  %         w     white         v     triangle (down)
  %                             ^     triangle (up)    <    triangle (left)
  %                             >     triangle (right)
  %                             p     pentagram      h     hexagram
   figure('Color','w'); % створити вікно для відображення графіку
   plot(vxc,vzc,'bo-','LineWidth',2,...
  'MarkerEdgeColor','b', 'MarkerFaceColor','c','MarkerSize',6); % plot z(x)

% 'bo-': колір лінії – синій (b), тип маркера – коло (o)
% тип лінії – суцільна(-)
  axis on;       % показати осі
  xlabel('x');   ylabel('z');  % показати підписи осей

  st='z(x) /c/';  % початок назви графіку
  st=strcat(st,'; i0= ');  % до рядкової змінної st додається фрагмент '; i0= '
  st=strcat(st,num2str(i0)) % num2str(i0) – перетворити ціле число в рядок
  st=strcat(st,'; tc,s = ');     st=strcat(st,num2str(tc,'%6.3f'));
% num2str(tc,'%6.3f') – перетворити дійсне число в рядок
   title(st); % показати заголовок графіка
```

Обчислені значення $z(x)$ відображаються на графіку в Q_p точках, починаючи з точки i_0. Для функцій $sin(x)$ та $cos(x)$ встановлюється значення періоду T_x. Для функції $exp(x)$, яка описує розподіл Гауса, встановлюється значення *SigmaG* /σ_G/ (середнього квадратичного відхилення розподілу) і m_G (центру розподілу Гауса).

Обчислення значень *x* та *z* в циклі (*Cycle*) виконується, якщо значення змінної *Mode_Calc_c* = 1. Час початку розрахунків записується в змінну *time_begin*. Значення *x* та *z* обчислюються поелементно (в циклі по *i* від 1 до *Q* з кроком 1) і записуються в масиви *vx0c* та *vz0c* розміром *Q* елементів. Формула, за якою обчислюються значення *vz0c*, залежить від варіанта роботи.

З масивів *vx0c* та *vz0c* (розміром *Q* елементів) зчитуються Q_p значень, які записуються у масиви *vxc* та *vzc* (розміром Q_p елементів). Далі на графіках відображаються значення функції $z(x)$, записані в масиви *vxc* та *vzc*. Це пояснюється тим, що кількість елементів *Q* надто велика для коректного відображення графіка, тому візуалізується тільки частина елементів $Q_p \sim 100$. Час обчислення значень функції $z(x)$ в циклі записується у змінну t_c (в секундах).

Графік функції $z(x)$ будується за допомогою функції *plot* (*X, Z, S*), де *X* – вектор координат точок *x*, *Z* – вектор координат точок *z*, параметр *S* описує колір, тип і товщину лінії графіка, розмір, тип і колір маркерів.

1.2. Векторизація циклів у системі MATLAB

Швидкодію комп'ютерних програм можна збільшити навіть без використання паралельних обчислень. У системі MATLAB для підвищення швидкодії послідовного коду використовується дві техніки послідовного програмування: попереднє розміщення (preallocation) і векторизація (vectorization). У цій роботі основну увагу звернемо на техніку векторизації (векторизацію циклів).

MATLAB виконує інтерпретацію команд в машинний код і послідовно їх виконує. Процес інтерпретації займає багато часу, якщо код програми містить цикли, оскільки кожен рядок циклу інтерпретується стільки разів, скільки разів виконується цикл. При попередньому розміщенні виконується ініціалізація масиву кінцевого розміру ще до виконання обчислень елементів масиву. Попереднє розміщення дозволяє уникнути багаторазових динамічних змін розміру масиву під час виконання циклів, оскільки кожна зміна розміру масиву потребує процесорного часу.

Векторизація циклів – це така зміна програмного коду, після якої послідовні поелементні обчислення в циклах замінюються на еквівалентні матричні або векторні операції. Ідея векторизації циклів полягає в тому, що замість багатьох поелементних операцій в циклі виконується одна операція з векторами. Основне призначення векторизації циклів – підвищення швидкодії програм.

Розглянемо приклад (лістинг 1.2) векторизації циклів у програмі MATLAB, призначеної для обчислення значень функції $z\,(x)$, заданої в Q точках. Значення x та z обчислюються як вектори і записуються в масиви *vx0v* та *vz0v* (розміром Q елементів). З масивів *vx0v* та *vz0v* зчитується Q_p значень, які записуються у масиви *vxv* та *vzv* (розміром Q_p елементів). Далі на графіках відображаються значення функції $z\,(x)$, записані в масиви *vxv* та *vzv*.

Лістинг 1.2. Фрагмент програми «p_CS_Lab_1_V0.m» із векторизацією циклів

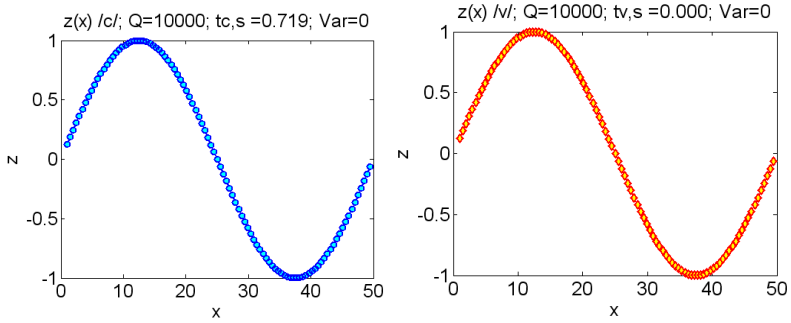
```
% Calculation z(x): vector
   vx0v=x_min:x_step:x_max;  % формування вектору матриці x
   vz0v=sin(2*pi*vx0v/Tx);       % векторне обчислення z /sin/
% vz0v=1.0*exp(-(vx0v-mG).^2/(2*SigmaG.^2)); %  обчислення z /exp/
    % vxv - фрагмент vx0v
 vxv=vx0v(i0:i0+Qp-1);     % matrix x for plot (visualization)
 vzv=vz0v(i0:i0+Qp-1);     % matrix z for plot (visualization)
```

Для наочності покажемо тільки циклічне (без векторизації) та векторне обчислення матриць у випадку, коли функція $z\,(x)$ описується синусоїдою (лістинг 1.3):

Лістинг 1.3. Фрагмент програми «p_CS_Lab_1_V0.m» із циклічним та векторним обчисленням матриць

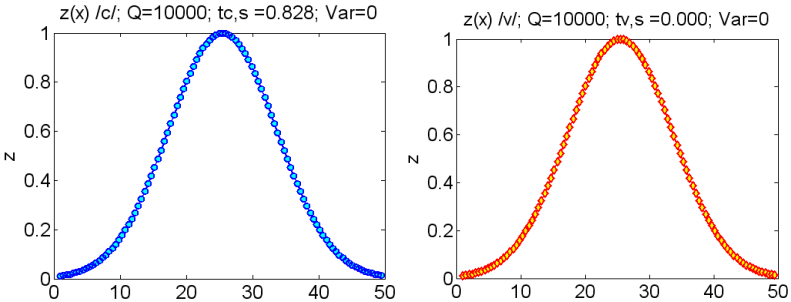
```
% Обчислення матриць в циклі (без векторизації)
 for i=1:1:Q                          % цикл по і від 1 до Q  з кроком 1
    vx0c(i)=x_min+(i-1)*x_step;   % обчислити значення x
    vz0c(i)=sin(2*pi* vx0c(i)/Tx);  % обчислити значення z
 end;                                 % кінець циклу по і
% Обчислення з векторизацією циклів (еквівалентне до циклічного)
  vx0v=x_min:x_step:x_max;     % формування вектора матриці x
  vz0v=sin(2*pi*vx0v/Tx);          % векторне обчислення z /sin/
```

Результати виконання програми «p_CS_Lab_1_V0.m» у випадку, коли функція $z\,(x)$ описується синусоїдою, показані на рис. 1.1 (варіант $V = 0$). Результати обчислення значень функції $z\,(x)$, яка описується розподілом Гауса, зображені на рис. 1.2.



а *б*

Рис. 1.1. Фрагмент обчислених значень функції $z\,(x)$, яка описується синусоїдою: а – обчислення $z\,(x)$ в циклі; б – з векторизацією циклів



а *б*

Рис. 1.2. Фрагмент обчислених значень функції $z\,(x)$, яка описується розподілом Гауса: *а* – обчислення в циклі; *б* – з векторизацією циклів

В обох випадках (рис. 1.1, рис. 1.2) час t_c циклічного обчислення значно перевищує час t_v векторного обчислення значень функції.

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання для всіх варіантів *V* 1...*V* 100

1. Створити програму «p_CS_Lab_1_V.m» в системі MATLAB, де *V* – номер варіанта. Програму створити на основі прикладу «p_CS_Lab_1_V0.m». Згідно з варіантом V вибрати формулу, яка описує функцію $z\,(x)$ (табл. 1.1). Отриману формулу запрограмувати в коді програми «p_CS_Lab_1_V.m» для випадку циклічного обчислення (час t_c) та для векторного обчислення (час t_v) значень функції. Мінімальне та максимальне значення x використати ті ж самі, що й в прикладі.

Таблиця 1.1

Формули, які описують функцію *z* (*x*)

*V*0	*z* (*x*)
0	$z = 1.0 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.03}}{0.1 \cdot T_x}\right)^{1.03}, m_G = x_{\min} + 3\sigma_G$
1	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.01}}{0.2 \cdot T_x}\right)^{1.01}, m_G = x_{\min} + 3\sigma_G$
2	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \sin\left(\frac{2\pi \cdot x^{1.01}}{0.1 \cdot T_x}\right)^{1.02}, m_G = x_{\min} + 2\sigma_G$
3	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.25 \cdot \sin\left(\frac{2\pi \cdot x^{1.01}}{0.2 \cdot T_x}\right)^{1.03}, m_G = x_{\min} + 4\sigma_G$
4	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \sin\left(\frac{2\pi \cdot x^{1.01}}{0.15 \cdot T_x}\right)^{1.04}, m_G = x_{\min} + \sigma_G$
5	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.01}}{0.25 \cdot T_x}\right)^{1.05}, m_G = x_{\min} + 5\sigma_G$
6	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.05}}{0.25 \cdot T_x}\right)^{1.06}, m_G = x_{\min} + 3\sigma_G$
7	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \sin\left(\frac{2\pi \cdot x^{1.05}}{0.1 \cdot T_x}\right)^{1.07}, m_G = x_{\min} + 2\sigma_G$
8	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.25 \cdot \sin\left(\frac{2\pi \cdot x^{1.05}}{0.2 \cdot T_x}\right)^{1.08}, m_G = x_{\min} + 4\sigma_G$
9	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \sin\left(\frac{2\pi \cdot x^{1.05}}{0.15 \cdot T_x}\right)^{1.09}, m_G = x_{\min} + 4\sigma_G$
10	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.05}}{0.25 \cdot T_x}\right)^{1.10}, m_G = x_{\min} + 4\sigma_G$

Продовження таблиці 1.1

*V*0	*z* (*x*)
11	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \cos\left(\frac{2\pi \cdot x^{1.10}}{0.15 \cdot T_x}\right)^{1.11}$, $m_G = x_{\min} + 3\sigma_G$
12	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \cos\left(\frac{2\pi \cdot x^{1.10}}{0.08 \cdot T_x}\right)^{1.12}$, $m_G = x_{\min} + 2\sigma_G$
13	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.25 \cdot \cos\left(\frac{2\pi \cdot x^{1.10}}{0.15 \cdot T_x}\right)^{1.13}$, $m_G = x_{\min} + 4\sigma_G$
14	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.08 \cdot \cos\left(\frac{2\pi \cdot x^{1.10}}{0.1 \cdot T_x}\right)^{1.14}$, $m_G = x_{\min} + \sigma_G$
15	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.15 \cdot \cos\left(\frac{2\pi \cdot x^{1.10}}{0.15 \cdot T_x}\right)^{1.15}$, $m_G = x_{\min} + 5\sigma_G$
16	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \cos\left(\frac{2\pi \cdot x^{1.15}}{0.08 \cdot T_x}\right)^{1.16}$, $m_G = x_{\min} + 3\sigma_G$
17	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.15 \cdot \cos\left(\frac{2\pi \cdot x^{1.15}}{0.15 \cdot T_x}\right)^{1.17}$, $m_G = x_{\min} + 2\sigma_G$
18	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.25 \cdot \cos\left(\frac{2\pi \cdot x^{1.15}}{0.2 \cdot T_x}\right)^{1.18}$, $m_G = x_{\min} + 4\sigma_G$
19	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.2 \cdot \cos\left(\frac{2\pi \cdot x^{1.15}}{0.15 \cdot T_x}\right)^{1.19}$, $m_G = x_{\min} + \sigma_G$
20	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \cos\left(\frac{2\pi \cdot x^{1.15}}{0.25 \cdot T_x}\right)^{1.20}$, $m_G = x_{\min} + 5\sigma_G$

Продовження таблиці 1.1

*V*0	*z* (*x*)
21	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.20}}{0.2 \cdot T_x}\right)^{1.21}$, $m_G = x_{\min} + 3\sigma_G$
22	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \sin\left(\frac{2\pi \cdot x^{1.20}}{0.1 \cdot T_x}\right)^{1.22}$, $m_G = x_{\min} + 2\sigma_G$
23	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.25 \cdot \sin\left(\frac{2\pi \cdot x^{1.20}}{0.2 \cdot T_x}\right)^{1.23}$, $m_G = x_{\min} + 4\sigma_G$
24	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \sin\left(\frac{2\pi \cdot x^{1.20}}{0.15 \cdot T_x}\right)^{1.24}$, $m_G = x_{\min} + \sigma_G$
25	$z = 0.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.20}}{0.25 \cdot T_x}\right)^{1.25}$, $m_G = x_{\min} + 5\sigma_G$
26	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.25}}{0.25 \cdot T_x}\right)^{1.26}$, $m_G = x_{\min} + 3\sigma_G$
27	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \sin\left(\frac{2\pi \cdot x^{1.25}}{0.1 \cdot T_x}\right)^{1.27}$, $m_G = x_{\min} + 2\sigma_G$
28	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.25 \cdot \sin\left(\frac{2\pi \cdot x^{1.25}}{0.2 \cdot T_x}\right)^{1.28}$, $m_G = x_{\min} + 4\sigma_G$
29	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) + 0.1 \cdot \sin\left(\frac{2\pi \cdot x^{1.25}}{0.15 \cdot T_x}\right)^{1.29}$, $m_G = x_{\min} + 4\sigma_G$
30	$z = 1.5 \cdot \exp\left(-\frac{(x-m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.25}}{0.25 \cdot T_x}\right)^{1.30}$, $m_G = x_{\min} + 4\sigma_G$

Номер рядка в таблиці *V*0 обчислюється на основі номеру варіанта *V* так: якщо $V \le 30$, то $V0 = V$; якщо $30 < V \le 60$, то $V0 = V - 30$; якщо $60 < V \le 90$, то $V0 = V - 60$; якщо $V > 90$, то $V0 = V - 90$. Для варіантів $V > 30$ показник степеня для функції *sin* або *cos* обчислюється за формулою: $k_S = 1 + V \cdot 0.01$.

2. Обчислити значення функції $z(x)$ для кількості точок $Q = 10^4$, отримані графіки $z(x)$ для циклічного та векторного випадку оформити згідно з даними таблиці 1.2 і додати до звіту (з вказанням варіанта). Для циклічного обчислення значень функції $z(x)$ встановити значення змінної Mode_Calc_c=1.

Таблиця 1.2

Параметри оформлення для графіків функції $z(x)$

V1	Циклічне обчислення $z(x)$			Векторне обчислення $z(x)$		
	Тип маркера	Колір маркера	Колір лінії	Тип маркера	Колір маркера	Колір лінії
0	o	голубий	синій	d	жовтий	червоний
1	s	жовтий	зелений	o	голубий	синій
2	d	жовтий	чорний	v	зелений	синій
3	v	синій	чорний	<	фіолетовий	синій
4	<	жовтий	синій	>	червоний	фіолетовий
5	>	зелений	синій	p	синій	червоний
6	p	фіолетовий	синій	h	синій	чорний
7	h	червоний	фіолетовий	*	жовтий	синій
8	*	жовтий	червоний	+	жовтий	зелений
9	+	синій	червоний	.	жовтий	чорний

Номер рядка таблиці V1 – молодша цифра номеру варіанта V.

3. Визначити час обчислення значень функції $z(x)$ (циклічний і векторний випадок) для кількості точок $Q = 5 \cdot 10^3$. Кількість точок Q збільшувати з кроком $5 \cdot 10^3$, для кожного Q зберігати отримані значення часу t_c та t_v. Якщо час t_c циклічного обчислення перевищує 10 с, то для більших Q циклічне обчислення не проводити (встановити значення змінної Mode_Calc_c=0). Отримані залежності $t_c(Q)$ та $t_v(Q)$ показати у вигляді таблиці та у вигляді графіка. Для візуалізації залежностей $t_c(Q)$ та $t_v(Q)$ створити модуль «p_CS_Lab_1_V_Time.m». Визначити, в скільки разів швидкодія векторного обчислення значень функції перевищує швидкодію циклічного обчислення значень $z(x)$.

4. У модулі «p_CS_Lab_1_V_Time.m» провести апроксимацію залежностей $t_c(Q)$ та $t_v(Q)$ за допомогою поліномів степеня P (для парних варіантів $P = 2$, для непарних $P = 3$). Коефіцієнти поліномів для циклічного випадку визначити функцією

kpc = polyfit (mQc, mtc, P),

де mQc – вектор значень Q, mtc – вектор значень часу $t_c(Q)$.

У випадку поліному степеня $P = 1$ апроксимовані значення часу обчислюються за формулою

mtcA(t) = kpc(1) · mQc(t) + kpc(2),

де t – номер значення Q, $t = 1,..., Q_{tc}$.

Коефіцієнти поліномів kpv та апроксимовані значення часу mtvA для векторного випадку визначити аналогічно. Отримані апроксимовані залежності часу mtcA та mtvA у вигляді графіків додати до звіту, коефіцієнти поліномів kpc, kpv додати до звіту.

5. Лістинги програм «p_CS_Lab_1_V.m» та «p_CS_Lab_1_V_Time.m» з коментарями додати до звіту.
6. Додаткове завдання. Обчислені значення $z(x)$ для циклічного і векторного випадку показати на одному графіку, візуалізація залежностей $t_c(Q)$ та $t_v(Q)$ – в логарифмічному масштабі.

3. ПРИКЛАД ВИКОНАННЯ РОБОТИ (ВАРІАНТ $V = 0$)

1. Створив програму «p_CS_Lab_1_0.m» в системі MATLAB, де $V = 0$ – номер варіанта. Згідно з варіантом V вибрав формулу, яка описує функцію $z(x)$ (табл. 1.1):

$$z = 1.0 \cdot \exp\left(-\frac{(x - m_G)^2}{2\sigma_G^2}\right) \cdot 0.2 \cdot \sin\left(\frac{2\pi \cdot x^{1.03}}{0.1 \cdot T_x}\right)^{1.03}, \quad m_G = x_{\min} + 3\sigma_G .$$

Отриману формулу запрограмував у коді програми «p_CS_Lab_1_0.m» для випадку циклічного обчислення (час t_c) та для векторного обчислення (час t_v) значень функції. Мінімальне і максимальне значення x використав ті ж самі, що й в прикладі (x_min=1; x_max=4.9E3).

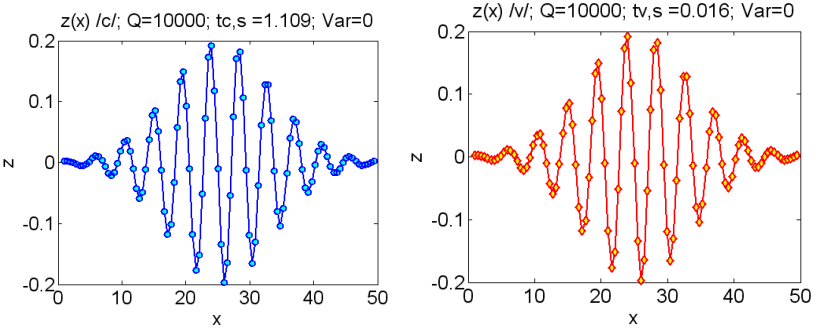
2. Обчислив значення функції $z(x)$ для кількості точок $Q = 10^4$, отримані графіки $z(x)$ для циклічного і векторного випадку оформив згідно з даними таблиці 1.2 (рис. 1.3).
3. Визначив час обчислення значень функції $z(x)$ (циклічний і векторний випадок) для кількості точок $Q = 5 \cdot 10^3$. Кількість точок Q збільшував з кроком $5 \cdot 10^3$, для кожного Q зберігав отримані значення часу t_c та t_v (табл. 1.3, рис. 1.4).

Для візуалізації залежностей $t_c(Q)$ та $t_v(Q)$ створив модуль «p_CS_Lab_1_0_Time.m», в якому значення часу t_c і кількості точок Q для циклічного випадку записав у масиви mtc та mQc відповідно:

```
mtc=zeros(0);  mQc=zeros(0); % time Cycle, створення масивів
mQc(1)=5E3;    mtc(1)=0.296;  ....
```

а значення часу t_v і кількості точок Q для векторного випадку записав у масиви mtv та mQv відповідно:

```
mtv=zeros(0); mQv=zeros(0);  % time Vector, створення масивів
mQv(1)=5E3;   mtv(1)=0.014;   ...
```



а *б*

Рис. 1.3. Фрагменти обчислених значень функції $z(x)$ (варіант $V = 0$): *а* – обчислення значень функції в циклі; *б* – з векторизацією циклів

Таблиця 1.3

Залежності часу t_c та t_v від кількості точок Q (варіант $V = 0$)

Q	$5 \cdot 10^3$	$1 \cdot 10^4$	$1.5 \cdot 10^4$	$2 \cdot 10^4$	$1 \cdot 10^5$	$5\ 10^5$	$1 \cdot 10^6$
t_c, *c*	0.296	1.109	4.484	7.797			
t_v, *c*	0.014	0.016	0.031	0.032	0.125	0.563	1.141

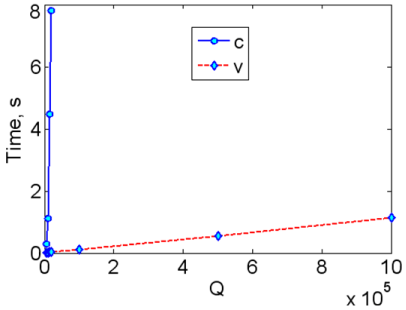


Рис. 1.4. Залежності часу t_c та t_v від кількості точок Q (варіант $V = 0$): c – обчислення значень функції в циклі, v – з векторизацією циклів

4. В модулі «p_CS_Lab_1_0_Time.m» провів апроксимацію залежностей $t_c(Q)$ та $t_v(Q)$ за допомогою поліномів степеня $P = 2$. Коефіцієнти поліному для циклічного випадку визначив функцією

kpc = polyfit (mQc, mtc, P),

де mQc – вектор значень Q, mtc – вектор значень часу t_c (див. завдання 3).

Значення коефіцієнтів kpc = 0.0000 -0.0001 0.0770.

Коефіцієнти поліному для векторного випадку визначив функцією

kpv = polyfit (mQv, mtv, P),

де mQv – вектор значень Q, mtv – вектор значень часу t_v (див. завдання 3).

Значення коефіцієнтів kpv = 0.0000 0.0000 0.0106.

Апроксимовані значення часу для циклічного випадку обчислив за формулою

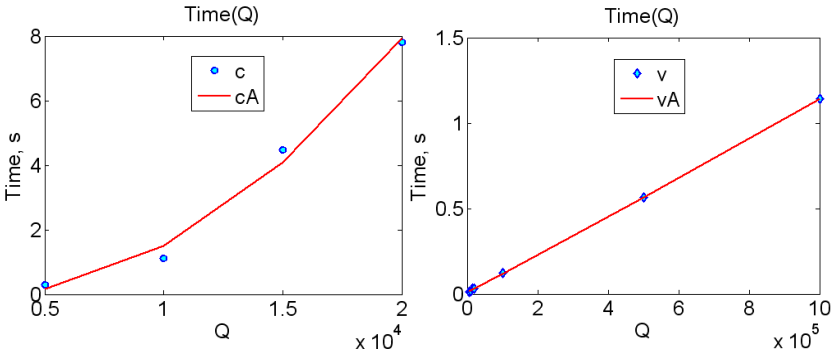
mtcA(t) = kpc(1) · mQc(t)^2 + kpc(2) · mQc(t) + kpc(3),

де t – номер значення Q, t = 1,..., 4.

Апроксимовані значення часу для векторного випадку обчислив за формулою

mtvA(t) = kpv(1) · mQv(t)^2 + kpv(2) · mQv(t) + kpv(3),

де t – номер значення Q, t = 1,..., 7.



а *б*

Рис. 1.5. Апроксимовані залежності часу t_c та t_v від кількості точок Q: *а* – обчислення значень функції в циклі; *б* – з векторизацією циклів; c, v – обчислені значення часу, cA, vA – апроксимовані значення

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. За допомогою яких технік можна підвищити швидкодію послідовного коду?
2. Яке головне призначення системи MATLAB?
3. Назвіть імена основних зарезервованих змінних системи MATLAB.
4. Що таке m-файл?
5. Які правила оформлення тесту програми в системі MATLAB?
6. Чим відрізняється поелементне та векторне множення матриць?
7. Як програмно реалізується поелементне множення матриць?
8. Поясніть код програми в лістингу 1.1.
9. Як у MATLAB програмно реалізуються цикли та умовні оператори?
10. Як в MATLAB визначити час виконання програми?
11. Якими засобами можна побудувати графік? Які параметри графіка можна змінити?
12. Що таке векторизація циклів? Поясніть, як можна програмно реалізувати векторизацію циклів на прикладі лістингів 1.2 та 1.3.
13. Покажіть в коді програми, як згідно з варіантом запрограмовано функцію $z\,(x)$ при її циклічному та векторному обчисленні. Як при цьому враховано параметр m_G?
14. Як змінено графіки функції $z\,(x)$ згідно з даними варіанта роботи?
15. Поясніть отримані залежності часу t_c та t_v від кількості точок Q, в яких проводилося обчислення значень функції.
16. Покажіть в коді програми, як проведено апроксимацію залежностей часу t_c та t_v від кількості точок Q.
17. Як впливає векторизація циклів на швидкодію програми? Поясніть причини.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Лазарєв Ю. Ф. Довідник з MATLAB. Електронний навчальний посібник з курсового і дипломного проектування / Київ: НТУУ «КПІ», 2013. 132 с.
2. Дьяконов В. Matlab 6 : Учебный курс / Санкт-Петербург: Питер, 2001. 592 с.

Довідка

Для зміни розміру шрифту в командному вікні (Command Window) системи MATLAB потрібно відкрити меню «File/ Preferences/ Fonts» і вибрати потрібний розмір шрифту.

ЛАБОРАТОРНА РОБОТА № 2
ПАРАЛЕЛЬНЕ ПРОГРАМУВАННЯ В СИСТЕМІ MATLAB У РЕЖИМІ PMODE

Мета: вивчити принципи паралельного програмування в системі MATLAB, зокрема програмування в режимі pmode; навчитися керувати планувальником задач і робочими процесами, виконувати паралельні обчислення в системі MATLAB.

Завдання: вивчити можливості пакетів розширень MATLAB Distributed Computing Toolbox та MATLAB Distributed Computing Engine, встановити параметри планувальника задач Job Manager і робочих процесів Worker, виконати паралельне обчислення визначеного інтегралу в режимі pmode за допомогою двох процесів.

Обладнання: персональний комп'ютер (ПК).

Програмне забезпечення: система MATLAB (R2007b-R2011b).

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Основи паралельного програмування в системі MATLAB

Технологія паралельних обчислень реалізована в системі MATLAB [1] за допомогою двох взаємопов'язаних пакетів розширень: MATLAB Distributed Computing Toolbox і MATLAB Distributed Computing Engine в MATLAB R2007b та пакета MATLAB Distributed Computing Server в MATLAB R2011b.

На цей час фактичним стандартом паралельного програмування служить досить складний для початкового вивчення інтерфейс передачі повідомлень MPI (Message Passing Interface), який є бібліотекою передачі повідомлень, набором функцій для комунікації (обміну даними і синхронізації задач) між процесами паралельної програми з розподіленою пам'яттю. Якісні реалізації MPI забезпечують асинхронну комунікацію, ефективне керування буфером повідомлень, інші функціональні можливості. MPI містить великий набір колективних операцій комунікації, віртуальних топологій і підтримує неоднорідні мережі.

Додатки, розроблені з використанням бібліотеки MPI і реалізовані в системі MATLAB [2-3], спрощують практичне використання паралельних обчислень на багатоядерних комп'ютерах, кластерах і GRID-системах. В подальшому, якщо не буде вказано окремо, під терміном *розподілені обчислення* будуть розумітися як паралельні, так і розподілені обчислення.

Пакети розширень Distributed Computing Toolbox (**DCT**) та MATLAB Distributed Computing Engine (**MDCE**), призначені для паралельного програмування в MATLAB R2007b, використовуються тільки разом. Пакети розширень призначені для збільшення продуктивності комп'ютерних систем і використовують такі терміни:

Client – сесія MATLAB, в якій визначаються та з якої надсилаються задачі. Це сесія MATLAB, в якій звичайно працює розробник-програміст. Розповсюджена також назва MATLAB-клієнт.

Cluster – набір комп'ютерів, які об'єднані в мережу з метою вирішувати спільну обчислювальну задачу.

Head Node – звичайно це вузол кластера, призначений для запуску процесів планувальника задач **job manager** і компонента сервера ліцензій **license manager**. Всі сервісні процеси для розподілених обчислень можуть бути запущені на одному комп'ютері.

Coarse-grained Application – додатки, для яких час виконання (обчислення) значно більше, ніж час створення, надсилання та отримання даних виконуваної задачі.

Distributed Application – додатки, аналогічні Coarse-grained Application, які виконуються незалежно на різних вузлах. Не виконують обмін повідомленнями, не мають спільних даних або точок синхронізації між вузлами. Розподілені обчислення можуть бути як **coarse-grained**, так і **fine-grained** (розгалужені задачі).

Distributed computing – обчислення, які виконуються за допомогою розподілених додатків, що виконуються на кількох вузлах одночасно.

Job – описана обчислювальна операція великої розмірності для виконання в MATLAB, яка складається з набору підзадач.

License Manager – компонент сервера ліцензій, який відповідає за перевірку і підтримку існуючих ліцензій на ядро/пакети розширень.

Task – підзадача, деякий сегмент основної задачі. Саме підзадача надсилається для обчислення у системний процес **worker.**

MDCE (MATLAB Distributed Computing Engine) – служба, яка відповідає пакету **MDCE**. Служба **MDCE** повинна бути запущеною на всіх комп'ютерах перед запуском процесів **job manager** або **worker**. Це основне середовище для запуску інших процесів.

Job Manager – фірмовий планувальник задач The Mathworks; це процес, який утворює систему черг із задач (job) і розподіляє задачі для системних процесів workers.

Scheduler – планувальник задач; основні функції полягають у плануванні та керуванні чергами задач; як планувальник може виступати Mathworks **Job Manager** або продукт сторонніх розробників.

Worker – системний робочий процес MATLAB, який виконує обчислення підзадач (**Task**). Поширені також назви MATLAB worker або worker-процес.

Розглянемо схему розподілених обчислень, яка лежить в основі Distributing Computing Toolbox (рис. 2.1). При використанні кластера як MATLAB client може виступати будь-який комп'ютер, який під'єднаний до мережі та має доступ до кластера; для планувальника scheduler звичайно виділяють один із вузлів кластера (на тому ж вузлі можна запустити також процес worker), на всіх інших вузлах кластера можна запустити по одному процесу. Запускати на одному вузлі кілька процесів доцільно у випадках, якщо вузол багатопроцесорний або багатоядерний (або з навчальною метою).

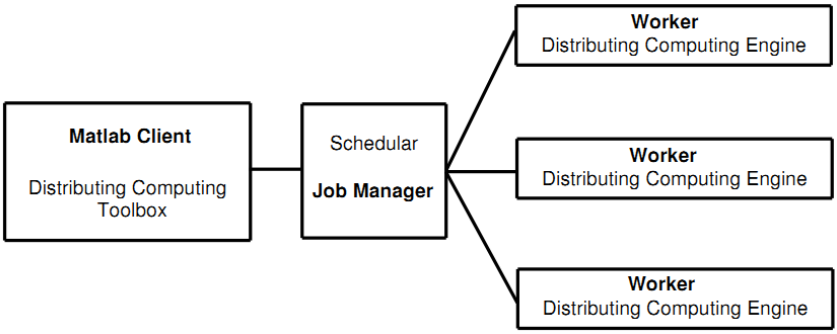


Рис. 2.1. Конфігурація, яка реалізує механізм розподілених обчислень MATLAB

Для інсталяції та роботи Distributed Computing Toolbox (**DCT**) комп'ютер повинен задовольняти кілька умов. По-перше, на комп'ютері повинна бути встановлена система MATLAB. По-друге, комп'ютер повинен бути під'єднаний до мережі. Мінімальні вимоги: до розміру дискового простору – 460 Мб, до розміру оперативної пам'яті – 512 Мб. Вимоги до мережі такі: розподілений обчислювальний процес повинен визначати інші процеси за іменами хостів, для чого в мережі потрібні служби перетворення імен хостів (наприклад, DNS).

1.2. Інсталяція і запуск MDCE

Служба **MDCE** (MATLAB Distributed Computing Engine) призначена для запуску процесу job manager. Розглянемо роботу зі службою MDCE в ОС Windows. Для перевірки коректного встановлення пакетів розширення (toolbox) в командному вікні потрібно ввести команду >>ver. У списку проінстальованих toolbox обов'язково повинні бути такі пакети :

- Distributed Computing Toolbox, MATLAB Distributed Computing Engine для MATLAB R2007b.
- MATLAB Distributed Computing Server для MATLAB R2011b.

Шлях до папки (каталогу), в яку інстальовано MATLAB, будемо позначати path (наприклад, path = 'C:\MATLABR2007B'). Кореневий каталог установки MATLAB можна визначити, наприклад, такою командою в командному вікні :

```
>>command1=matlabroot
```

В папці path\toolbox\distcomp\bin містяться файли, які запускають службові сервіси та процеси. Для інсталяції служби MDCE потрібно в командному вікні MATLAB виконати команду

```
>>!path\toolbox\distcomp\bin\mdce install
```

Знак «!» в команді вказує MATLAB, що команда системна; у результаті успішного виконання команди в списку служб Windows (Панель управления/ Администрирование/Службы) повинна з'явитися нова служба – MATLAB Distributed Computing Engine (рис. 2.2). Після цього службу MDCE треба запустити.

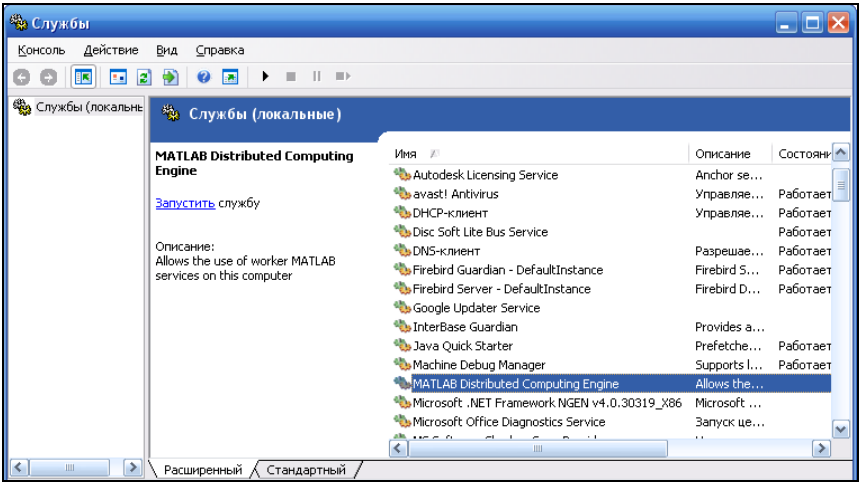


Рис. 2.1. Служба MDCE серед інших служб Windows

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання для всіх варіантів *V* 1...*V* 100

1. На комп'ютері з встановленою системою MATLAB запустити службу **MDCE**; якщо служба відсутня, то її потрібно проінсталювати (див. теоретичні відомості).

2. Виконати запуск планувальника задач **Job Manager**. Системний процес планувальника задач **Job Manager** виконується за допомогою файла startjobmanager.bat, який знаходиться в папці path\toolbox\distcomp\bin. Запуск планувальника задач виконати в командному вікні MATLAB командою

```
>>!path\toolbox\distcomp\bin\startjobmanager
-name JobManagerNV -remotehost hostN -v,          (2.1)
```

де

- ключ «-name» встановлює назву планувальника задач JobManagerNV для варіанту *V* (наприклад, для варіанта *V* = 0 назва планувальника «JobManagerN0»);
- ключ «-remotehost» встановлює назву хоста hostN, на якому буде виконано запуск процесу планувальника; у нашому випадку як hostN потрібно вказати ім'я комп'ютера (Панель управления/Система/Имя компьютера); як hostN також можна вказувати IP адреси комп'ютера.
- ключ «-v» дозволяє показувати системні повідомлення при виконанні команди.

Зупинка планувальника задач (після завершення роботи з MDCE) виконується командою

```
>>!path\toolbox\distcomp\bin\stopjobmanager
-name JobManagerNV -remotehost hostN -v.          (2.2)
```

Якщо в системі відбувся збій, то запуск планувальника потрібно виконувати з ключем «-clean».

3. Виконати запуск робочих процесів **worker**. Системні робочі процеси worker, які ще називаються віддаленими сесіями MATLAB, забезпечують розв'язання підзадач task. Запуск worker аналогічний до запуску процесу JobManager. Параметри процесу worker передаються за допомогою ключів, а запуск процесу виконується командою

```
>>!path\toolbox\distcomp\bin\startworker
-name workerNV1 -remotehost hostN -jobmanager JobManagerNV, (2.3)
```

де

- ключ «-name» визначає ім’я першого процесу workerNV1 для варіанту *V* (наприклад, для варіанта *V* = 0 назва робочого процесу «workerN01»);
- ключ «-remotehost» встановлює назву хоста hostN (аналогічно як в команді 2.1);
- ключ «-jobmanager» встановлює назву планувальника задач JobManagerNV для варіанта *V* (аналогічно як в команді 2.1), з цим планувальником JobManagerNV буде асоційований робочий процес workerNV1 (буде отримувати від нього підзадачі).

Запуск другого робочого процесу виконати командою

>>!path\toolbox\distcomp\bin\startworker
-name workerNV2 -remotehost hostN -jobmanager JobManagerNV, (2.4)

Отже, у кластері для планувальника задач JobManagerNV буде запущено два робочих процеси workerNV1 та workerNV2. Інформацію про запущений планувальник задач та робочі процеси (в кластері) для вузла (node) можна отримати командою nodestatus

>>!path\toolbox\distcomp\bin\nodestatus
-remotehost hostN -infolevel 3, (2.5)

де

- ключ «-remotehost» встановлює назву хоста hostN (аналогічно як у команді 2.1);
- ключ «-infolevel» вказує рівень інформативності системних повідомлень (1 – мінімальна, 3 – детальна).

При правильному виконанні команд (2.1-2.5) статус планувальника задач JobManagerNV повинен бути «Running» (з номером порта Port), а до планувальника під’єднано (Connected) два робочі процеси workerNV1 та workerNV2. Лістинг повідомлень, отриманих командою (2.5), вставити у звіт.

4. Виконати перехід у режим програмування **pmode**. Режим **pmode** в системі MATLAB є одним з двох підходів до розв’язання паралельних задач.

Перший підхід засновано на виконанні m-файлів з інструкціями для планувальника JobManager, у яких можуть використовуватися функції MPI для комунікації між процесами.

Другий підхід засновано на роботі в режимі **pmode**, в якому можливе звернення до робочих процесів worker з командного вікна MATLAB. У режимі **pmode** можлива робота з локальними змінними всіх процесів. При цьому команди, які вводяться в робочому вікні MATLAB, виконуються всіма робочими процесами, асоційованими з планувальником JobManager.

У даній лабораторній роботі використовується режим **pmode** з метою знайомства з елементами паралельного програмування, для розуміння парадигми паралельного програмування та налагодження паралельних програм.

Перехід у режим **pmode** виконати командою

>>pmode start local numlabs, (2.6)

де **numlabs** – загальна кількість робочих процесів, асоційованих із планувальником задач; у нашому випадку numlabs = 2.

Для кожного робочого процесу існує його порядковий номер (ID), який приймає значення від 1 до numlabs і визначається через ідентифікатор **labindex** (рис. 2.2).

У режимі **pmode** створити змінну *a*, яка буде визначена в сесіях (робочих просторах) усіх процесів. Змінна *a* повинна приймати значення номеру процесу

a = labindex, (2.7)

а вираз для обчислення змінної *a* потрібно ввести в командному рядку вікна для паралельної обробки команд (рис. 2.2), який починається з символів «P>>» (в подальшому команди для паралельної обробки вводяться в цьому рядку). Обчислені значення змінної *a* будуть показані у вікні для паралельної обробки команд (рис. 2.2).

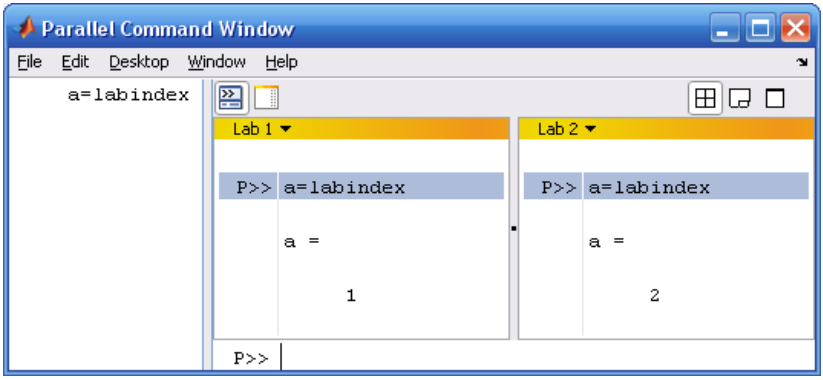


Рис. 2.2. Вікно для паралельної обробки команд зі значенням змінної *a*; вікно «lab1» відповідає процесу №1, «lab2» – процесу №2

Кілька робочих процесів можуть бути запущені навіть на однопроцесорному одноядерному комп’ютері, проте приріст продуктивності буде тільки на багатопроцесорних або багатоядерних комп’ютерах.

5. За допомогою двох паралельних робочих процесів обчислити значення визначеного інтегралу для функції f(x)

$$I_V = \int_{x1}^{x2} f(x)dx, \qquad (2.8)$$

де підінтегральна функція $f(x)$ обчислюється за формулою

$$f(x) = (V + 0.5) \cdot x, \qquad (2.9)$$

V – номер варіанта, а межі інтегрування обчислюються за формулами:

$$x_1 = 0.1 \cdot V, \; x_2 = 1 + 0.1 \cdot V. \qquad (2.10)$$

Функцію $f(x)$ показати у вигляді графіка з межами інтегрування (рис. 2.3). Для розпаралелювання обчислень слід скористатися тією властивістю, що кожний робочий процес може інтегрувати функцію $f(x)$ на своїй ділянці відрізка $[x_1, x_2]$ (рис. 2.3). Процес №1 інтегрує функцію на відрізку $[x_1, x_3]$, а процес №2 – на відрізку $[x_3, x_2]$, де x_3 – середина відрізка $[x_1, x_2]$. Відповідно інтеграл на всьому відрізку $[x_1, x_2]$ дорівнює сумі інтегралів Ip1 та Ip2.

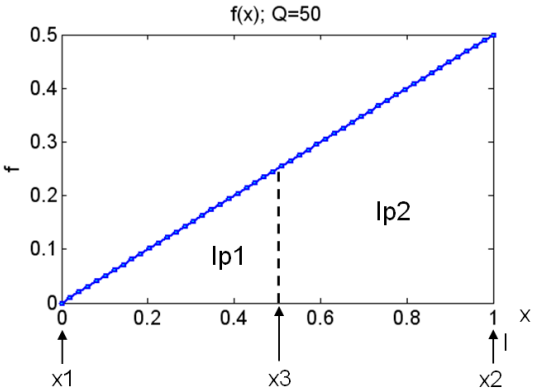


Рис. 2.3. Межі інтегрування для функції $f(x)$; Ip1 – інтеграл функції $f(x)$ на відрізку $[x_1, x_3]$, Ip2 – інтеграл функції $f(x)$ на відрізку $[x_3, x_2]$

Розповсюдження функції $f(x)$ (2.9) серед всіх робочих процесів виконати командою

P>>f=@(x) ((V+0.5).*x). (2.11)

Після цього функція $f(x)$ стає визначеною в усіх робочих процесах. Межі інтегрування (2.10) для всього відрізка $[x_1, x_2]$ ввести командами

P>>x1=V.*0.1, (2.12)

P>>x2=(V.*0.1)+1. (2.13)

Далі потрібно встановити межі інтегрування $[a_1, a_2]$ для кожного робочого процесу. Для цього виконати команди

P>>a1=((labindex-1)/numlabs).*(x2-x1)+x1. (2.14)

P>>a2=(labindex/numlabs).*(x2-x1)+x1. (2.15)

де labindex – номер робочого процесу,

numlabs – загальна кількість робочих процесів.

Скориставшись функцією quadl(f,a1,a2), параметрами якої є підінтегральна функція f(x) і межі інтегрування [a1, a2], обчислити значення визначеного інтеграла числовим методом для кожного робочого процесу командою

P>>IntegralP=quadl(f,a1,a2). (2.16)

В результаті для кожного процесу (сесії) буде обчислено значення змінних IntegralP, в яких записано значення визначеного інтеграла для відповідних відрізків $[x_1, x_3]$ та $[x_3, x_2]$. Глобальне додавання значень змінної IntegralP для всіх процесів виконати функцією gplus(IntegralP), результат записати у змінну IntegralG .

6. Значення визначеного інтеграла (2.8) обчислити аналітично через його первісну і записати у змінну I_V. Порівняти точне значення інтегралу I_V зі значенням IntegralG, обчисленим методом паралельного програмування.

3. ПРИКЛАД ВИКОНАННЯ РОБОТИ (ВАРІАНТ *V* = 0)

1. На комп'ютері з встановленою системою MATLAB запустив службу **MDCE.**

2. Виконав запуск планувальника задач **JobManagerN0**.

3. Виконав запуск робочих процесів **workerN01** та **workerN02**. За допомогою команди nodestatus отримав такий лістинг повідомлень:

```
Job manager lookup process:
    Status                Running
Job manager:
    Name                  JobManagerN0
    Running on host       hostN
    Number of workers     2
```

... (у звіті навести повний текст лістингу).

4. Виконав перехід в режим програмування **pmode**, в якому запустив два робочі процеси **workerN01** та **workerN02**. Для кожного процесу створив змінну *a* (рис. 2.2).

5. За допомогою двох паралельних робочих процесів обчислив значення визначеного інтегралу для функції f(x) = 0.5 x (2.9) і меж інтегрування x1= 0, x2 = 1 (рис. 2.3). Виконав розповсюдження функції *f* (*x*) серед всіх робочих процесів (рис. 2.4).

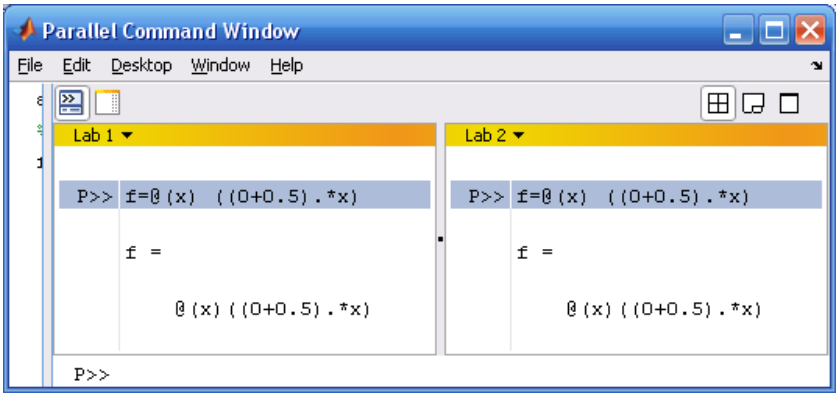


Рис. 2.4. Вікно для паралельної обробки команд зі значенням функції *f* (*x*) (варіант *V* = 0)

Далі встановив для робочих процесів значення меж інтегрування x1 та x2 (рис. 2.5).

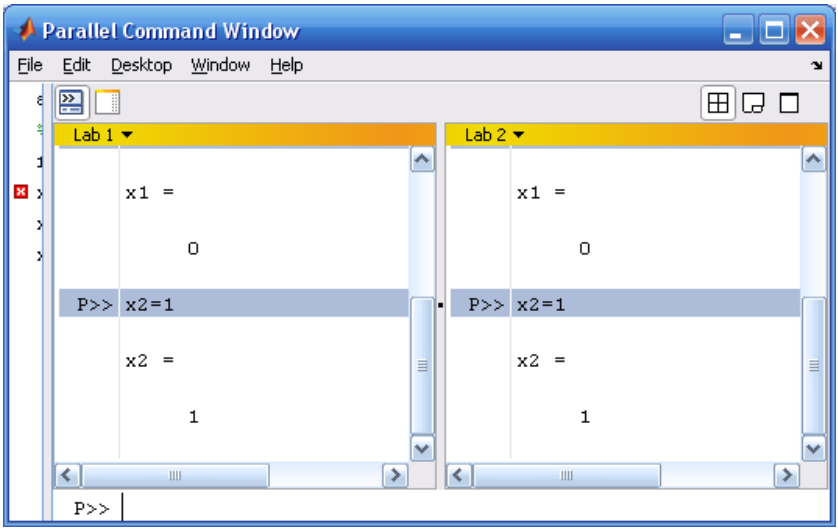


Рис. 2.5. Вікно для паралельної обробки команд зі значенням меж інтегрування x1 та x2 (варіант *V* = 0)

Після цього встановив межі інтегрування $[a_1, a_2]$ для кожного робочого процесу (рис. 2.6).

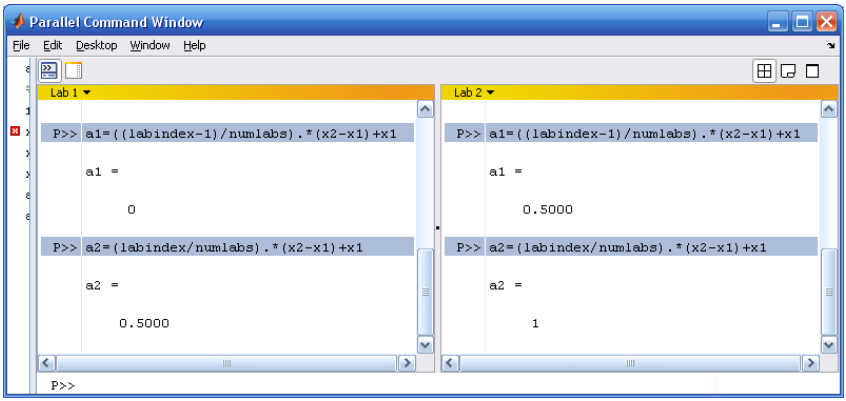


Рис. 2.6. Вікно для паралельної обробки команд зі значенням меж інтегрування $[a_1, a_2]$ для кожного робочого процесу (варіант $V = 0$)

За допомогою функції quadl(f,a1,a2) обчислив значення визначеного інтеграла IntegralP для кожного робочого процесу (рис. 2.7).

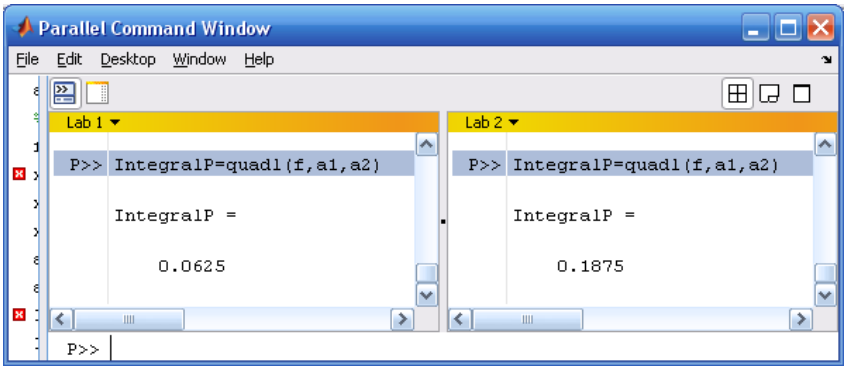


Рис. 2.7. Вікно для паралельної обробки команд зі значенням визначеного інтеграла IntegralP для кожного робочого процесу (варіант $V = 0$)

Глобальне додавання значень змінної IntegralP для всіх процесів виконав функцією gplus(IntegralP), у результаті чого методом паралельного програмування отримав значення визначеного інтеграла IntegralG (рис. 2.8).

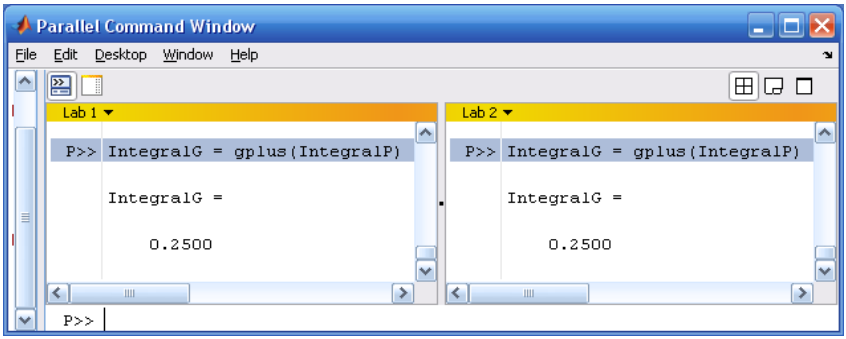


Рис. 2.8. Вікно для паралельної обробки команд із сумарним значенням визначеного інтеграла IntegralG (варіант $V = 0$)

6. Значення визначеного інтеграла (2.8) обчислив через його первісну та записав у змінну I_V:

$$I_V = \int_0^1 0.5x\,dx = 0.5\frac{x^2}{2}\Big|_0^1 = \frac{x^2}{4}\Big|_0^1 = \frac{1}{4}.$$

Порівняв точне значення інтеграла $I_V = 0.25$ зі значенням IntegralG = 0.2500, обчисленим методом паралельного програмування. Значення визначеного інтеграла обчислено правильно з точністю до 4 значущих цифр.

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. За допомогою яких пакетів реалізована технологія паралельних обчислень у системі MATLAB?
2. Яке призначення MPI?
3. Поясніть значення термінів «Client», «Cluster», «Job» і «Task», які використовуються при паралельному програмуванні в системі MATLAB.
4. Проаналізуйте, як взаємодіють планувальник і робочі процеси при паралельних обчисленнях.
5. Яке призначення служби MDCE?
6. Якою командою виконується запуск планувальника задач Job Manager? Поясність призначення ключів команди.
7. Якою командою виконується запуск робочих процесів worker? Поясність призначення ключів команди.
8. Яке призначення команди «nodestatus»?
9. Дайте оцінку двох підходів до розв’язання паралельних задач, які використовуються в системі MATLAB.
10. Як виконується перехід в режим паралельного програмування pmode?

11. Поясність, як у режимі pmode можна визначити номер робочого процесу?
12. Поясність, як розпаралелюється процес обчислення визначеного інтегралу в режимі pmode?
13. Як у режимі pmode можна обробляти значення змінних із різних процесів?
14. Проаналізуйте, чим відрізняється обчислення визначеного інтеграла аналітичним методом і числовим методом при паралельному програмуванні.
15. Обґрунтуйте переваги та недоліки паралельного програмування.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Оленёв Н. Н., Печёнкин Р. В., Чернецов А. М. Параллельное программирование в MATLAB и его приложения / Москва: Вычислительный центр им. А. А. Дородницына РАН, 2007. 120 с.
2. Лазарєв Ю. Ф., Лазарєв Ю. Ф. Довідник з MATLAB. Електронний навчальний посібник з курсового і дипломного проектування / Київ: НТУУ «КПІ», 2013. 132 с.
3. Дьяконов В. Matlab 6 : Учебный курс / В. Дьяконов. – Санкт-Петербург: Питер, 2001. 592 с.

Довідка

Процес планувальника JobManager і робочі процеси worker існують в системі окремо від клієнтської сесії MATLAB (навіть при зупинці клієнтського додатка MATLAB). Якщо запущено зайві планувальники та робочі процеси, то інформацію про них потрібно отримати командою nodestatus. Після цього слід зупинити спочатку зайві робочі процеси, а потім планувальники.

Служба MDCE, планувальник JobManager і робочі процеси worker генерують робочі файли, які бувають двох типів: файли журналу (log) і файли контрольних точок (checkpoints). Файли журналу зберігаються в папці «TEMP\MDCE\log», а файли контрольних точок – в папці «TEMP\MDCE\Checkpoints». Якщо розміри файлів журналу і контрольних точок займають значний об’єм, то їх потрібно очистити командами destroy(job), destroy(Checkpoint).

При виникненні проблем зі службою MDCE брандмауер операційної системи потрібно або вимкнути, або виконати його спеціальне налаштування. Іноді потрібно відключити антивірусні програми.

ЛАБОРАТОРНА РОБОТА № 3
ПАРАЛЕЛЬНЕ ПРОГРАМУВАННЯ В СИСТЕМІ MATLAB ІЗ ВИКОРИСТАННЯМ М-ФАЙЛІВ

Мета: вивчити принципи паралельного програмування в системі MATLAB, зокрема програмування з використанням m-файлів; навчитися керувати планувальником задач і робочими процесами, здійснювати обмін даними з робочими процесами, виконувати паралельні обчислення в системі MATLAB.

Завдання: вивчити можливості пакетів розширень MATLAB Distributed Computing Toolbox та MATLAB Distributed Computing Engine при роботі з m-файлами, встановити параметри планувальника задач JobManager і робочих процесів Worker, виконати паралельне обчислення визначеного інтеграла з використанням m-файлів за допомогою двох процесів.

Обладнання: персональний комп'ютер (ПК).

Програмне забезпечення: система MATLAB (R2007b-R2011b).

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

Паралельне програмування в системі MATLAB із використанням m-файлів

Керування паралельними процесами в системі MATLAB із використанням **m-файлів** ефективніше, ніж у режимі **pmode** [1]. Для керування паралельними процесами з m-файлів потрібно отримати посилання на ці процеси з клієнтської сесії MATLAB [2, 3]. У такому випадку в робочій області MATLAB отримується об'єкт розподілених обчислень.

При написанні m-файлу паралельного завдання слід пам'ятати, що цей m-файл буде виконуватися кількома процесами. Змінні, які визначаються в цьому файлі, будуть локальними для кожної сесії процесу, до них не буде відкрито доступ від інших процесів без використання відповідних функцій MATLAB для передачі повідомлень між процесами.

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання для всіх варіантів *V* 1...*V* 100

В пунктах 1-6 даної лабораторної роботи виконуються ті ж завдання, що й в лабораторній роботі № 2, але не в режимі **pmode**, а за допомогою **m-файлів**.

1. На комп'ютері з встановленою системою MATLAB запустити службу **MDCE**; якщо служба відсутня, то її потрібно проінсталювати (див. лабораторна робота № 2). Інформацію про запущений планувальник задач та робочі процеси (у кластері) для вузла (node) отримати командою **nodestatus**

>>!path\toolbox\distcomp\bin\nodestatus
-remotehost hostN -infolevel 3, (3.1)

де

- path – шлях до папки (каталогу), в яку інстальовано MATLAB (наприклад, path = 'C:\MATLABR2007B'). Кореневий каталог установки MATLAB можна визначити, наприклад, наступною командою в командному вікні: >> path = matlabroot.
- ключ «-remotehost» встановлює назву хоста hostN, на якому буде виконано запуск процесу планувальника (наприклад, на віддаленій робочій станції); у нашому випадку як hostN потрібно вказати ім'я комп'ютера (Панель управления/Система/Имя компьютера).
- ключ «-infolevel» вказує рівень інформативності системних повідомлень (1 – мінімальна, 3 – детальна).

Закрити всі запущені робочі процеси командою «stopworker» і всі планувальники задач командою «stopjobmanager».

2. Виконати запуск планувальника задач **JobManagerNV** в командному вікні MATLAB командою

>>!path\toolbox\distcomp\bin\startjobmanager
-name JobManagerNV -remotehost hostN -v, (3.2)

де

- ключ «-name» встановлює назву планувальника задач JobManagerNV для варіанта *V* (наприклад, для варіанту *V* = 0 назва планувальника «JobManagerN0»);
- hostN – ім'я комп'ютера (аналогічно як в команді 3.1);
- ключ «-v» дозволяє показувати системні повідомлення при виконанні команди.

Виконати запуск робочого процесу **workerNV1** командою

>>!path\toolbox\distcomp\bin\startworker
-name workerNV1 -remotehost hostN -jobmanager JobManagerNV, (3.3)

де

- ключ «-name» визначає ім'я першого процесу workerNV1 для варіанта *V* (наприклад, для варіанта *V* = 0 назва робочого процесу «workerN01»);

- ключ «-remotehost» встановлює назву хоста hostN (аналогічно як у команді 3.1);
- ключ «-jobmanager» встановлює назву планувальника задач JobManagerNV для варіанта *V* (аналогічно як у команді 3.2), з цим планувальником JobManagerNV буде асоційований робочий процес workerNV1 (буде отримувати від нього підзадачі).

Аналогічно запустити другий робочий процес workerNV2.

3. Створити головний **m-файл «p_CS_Lab_3_V1.m»** (*V* – номер варіанта), в якому записувати команди наступних пунктів 4-6. Всі m-файли лабораторної роботи повинні знаходитися **в одній папці.**

4. **Посилання pJmV** (*V* – номер варіанта) **на процес планувальника «JobManagerNV»** отримати за допомогою функції «findResource»:

```
pJmV = findResource('scheduler', 'type', 'jobmanager',...
                    'Name','JobManagerNV','LookupURL','hostN'),    (3.4)
```

якій як параметри передати тип процесу ('jobmanager'), назву процесу ('JobManagerNV'), ім'я вузла ('hostN'); інші параметри встановлюються за замовчуванням. Функція «findResource» належить пакету MATLAB Distributed Computing Toolbox (для MATLAB R2007b) або пакету MATLAB Distributed Computing Server (для MATLAB R2011b). Результатом виконання даної команди є об'єкт pJmV:

```
pJmV = distcomp.jobmanager,
```

який є посиланням на процес планувальника jobmanager.

Вивести список властивостей об'єкта pJmV командою «get(pJmV)», а методів об'єкта – командою «methods(pJmV)» .

5. Для створення **паралельної задачі JobV** (*V* – номер варіанта) використати команду

```
pJobV = pJmV.createParallelJob.                    (3.5)
```

Результатом виконання даної команди є посилання **pJobV** на паралельну задачу:

```
pJobV = distcomp.paralleljob,
```

яка є посиланням на процес планувальника паралельної задачі.

Мінімальну Qw_min = 2 і максимальну Qw_max = 2 кількість робочих процесів для паралельної задачі встановити командами:

```
set(pJobV,'MinimumNumberOfWorkers',Qw_min);
set(pJobV,'MaximumNumberOfWorkers',Qw_max).        (3.6)
```

6. Створити **паралельну підзадачу TaskV** у вигляді **m-файлу «fint*V*1.m»** (*V* – номер варіанта), який буде виконуватися всіма робочими процесами одночасно, а запускатися паралельна підзадача повинна з файлу **«p_CS_Lab_3_V1.m»**. Зв'язок паралельної задачі (посилання pJobV) з m-файлом «fint*V*1.m» встановити командою:

set(pJobV,'FileDependencies',{'fint*V*1.m'});

За допомогою паралельної задачі обчислити значення інтеграла (аналогічно як в лабораторній роботі № 2) для функції f(x)

$$I_V = \int_{x1}^{x2} f(x)dx, \qquad (3.7)$$

де підінтегральна функція $f(x)$ обчислюється за формулою

$$f(x) = (V + 0.5) \cdot x, \qquad (3.8)$$

V – номер варіанта, а межі інтегрування обчислюються за формулами:

$$x_1 = 0.1 \cdot V, \; x_2 = 1 + 0.1 \cdot V. \qquad (3.9)$$

Функцію $f(x)$ та межі інтегрування x_1, x_2 задати у файлі **«p_CS_Lab_3_V1.m»:**

f=@(x) ((V+0.5).*x); % function for Integral

Функцію $f(x)$ показати у вигляді графіка з межами інтегрування для кількості точок Q = 50 (рис. 3.1). Процес №1 інтегрує функцію на відрізку $[x_1, x_3]$, а процес №2 – на відрізку $[x_3, x_2]$, де x_3 – середина відрізка $[x_1, x_2]$. Відповідно інтеграл на всьому відрізку $[x_1, x_2]$ дорівнює сумі інтегралів Ip1 та Ip2.

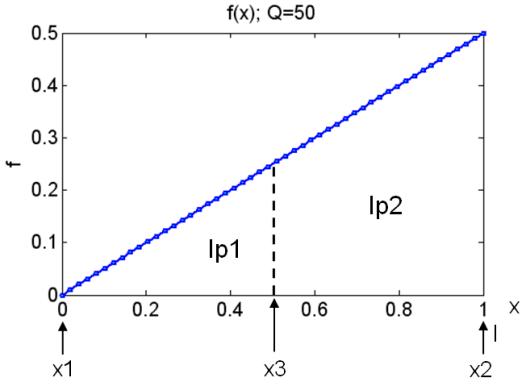


Рис. 3.1. Межі інтегрування для функції $f(x)$; Ip1 – інтеграл функції $f(x)$ на відрізку $[x_1, x_3]$, Ip2 – інтеграл функції $f(x)$ на відрізку $[x_3, x_2]$; Q – кількість точок на графіку

У такому випадку m-файл **«fint*V*1.m»** паралельної підзадачі потрібно створити у вигляді такої функції MATLAB:

```
function pint V1=fint V1(f,x1,x2);
 xw1=((labindex-1)/numlabs).*(x2-x1)+x1; % Ліва межа інтеграла
 xw2=(labindex/numlabs).*(x2-x1)+x1;      % Права межа інтеграла
  Ip=quadl(f,xw1,xw2);  % Обчислення інтеграла Ip
pint01=gplus(Ip);           % Обчислення суми для всіх процесів
```

Назва функції «fint*V*1» повинна збігатися з назвою файлу «fint*V*1», а межі відрізка інтегрування $[x_{w1}, x_{w2}]$ обчислюються окремо для кожного процесу.

Для передачі m-файлу **«fint*V*1.m»** до робочого процесу використовується об'єкт **pTaskV** (посилання на паралельну підзадачу). Об'єкт **pTaskV**, який є дочірнім об'єктом для об'єкта pJobV, створити методом createTask:

pTaskV = createTask (pJobV,'fintV1',1,{f,x1,x2}). (3.10)

В об'єкт pTaskV через параметри передається кількість вихідних аргументів (в нашому випадку – 1) і вхідні аргументи (f,x1,x2). Результатом виконання даної команди є об'єкт:

pTaskV=distcomp.task.

Після цього задачу з посиланням pJobV потрібно направити планувальнику за допомогою команди:

submit(pJobV);

Задача починає виконуватися робочими процесами не одразу, а коли вільними виявляться стільки процесів, скільки попередньо було вказано у властивості «MinimumNumberOfWorkers». До цього часу задача знаходиться в черзі. Якщо кількість вільних робочих процесів стає достатньою, то всі робочі процеси починають обробляти файл **«fint*V*1.m»**. У цей момент властивість State об'єкта pJobV набуває значення running, а після виконання задачі властивість State об'єкта pjobV приймає значення finished. Момент завершення виконання задачі визначається командою

waitForState(pJobV);

а результати виконання задачі потрібно повернути командою

results=getAllOutputArguments(pJobV).

7. Виконати паралельні обчислення значення визначеного інтеграла (див. пункт 6) і підібрати таке значення x_3, при якому інтеграл Ip1 на відрізку $[x_1, x_3]$ дорівнює $(0.5+V \cdot 0.005)$ від інтеграла Ip на всьому відрізку $[x_1, x_2]$, тобто $Ip1 = (0.5+V \cdot 0.005) \cdot Ip$. У першому наближенні значення x_3

вибрати як середину відрізка [x_1, x_2], а далі в циклі з лічильником i_t значення x_3 уточнювати (для 5-ти ітерацій). Створити головний файл «p_CS_Lab_3_V2.m», а для паралельної підзадачі – файл «fintV2.m». При цьому потрібно організувати обмін даними між робочими процесами, як це показано в лістингу 1 та лістингу 2 (див. приклад виконання роботи).

На основі наведених лістингів 1, 2 потрібно створити власні файли «p_CS_Lab_3_V2.m» та «fintV2.m» (згідно з варіантом V). Отримані значення лівої mPartIL і правої mPartIR частин інтеграла для кожної ітерації показати у вигляді графіка.

3. ПРИКЛАД ВИКОНАННЯ РОБОТИ (ВАРІАНТ *V* = 0)

1. На комп'ютері з встановленою системою MATLAB запустив службу **MDCE**. Інформацію про запущений планувальник задач і робочі процеси (в кластері) отримав командою **nodestatus**. Закрив всі запущені робочі процеси та всі планувальники задач.

2. Виконав запуск планувальника задач **JobManagerN0**, виконав запуск робочих процесів **workerN01** та **workerN02**.

3. Створив головний m-файл «**p_CS_Lab_3_01.m**».

4. **Посилання pJm0 на процес планувальника «JobManagerN0»** отримав за допомогою функції «findResource»:

```
pJm0=findResource('scheduler', 'type', 'jobmanager',...
    'Name','JobManagerN0','LookupURL','hostN');
```

Вивів список властивостей і методів об'єкта **pJm0**.

5. Створив **паралельну задачу** з посиланням **pJobV** командою

```
pJobV = pJm0.createParallelJob.
```

Встановив мінімальну (2) і максимальну (2) кількість робочих процесів для паралельної задачі.

6. Створив **паралельне завдання** у вигляді m-файлу **«fint01.m»:**

```
function pint01=fint01(f,x1,x2);
 xw1=((labindex-1)/numlabs).*(x2-x1)+x1;
 xw2=(labindex/numlabs).*(x2-x1)+x1;
   Ip=quadl(f,xw1,xw2);
pint01=gplus(Ip);
```

Записав програмний код файлу **«p_CS_Lab_3_01.m»**:

```
%% Parallel program
pJm0=findResource('scheduler', 'type', 'jobmanager',...
'Name','JobManagerN0','LookupURL','microsof-140e50.');
```

```
 disp('pJm0='); disp(pJm0);
%% createParallelJob % JobManager
pJob0=pJm0.createParallelJob;
 disp('pJob0='); disp(pJob0);
%%
Qw=2;   % Quantity of Workers
 set(pJob0,'MinimumNumberOfWorkers',Qw);
 set(pJob0,'MaximumNumberOfWorkers',Qw);
%%
set(pJob0,'FileDependencies',{'fint01.m'});
%%
f=@(x) ((0.5).*x);  % function for Integral
 x1=0;   % Left Limit of Integral (Global)
 x2=1;   % Right Limit of Integral (Global)
%% createTask
pTask0=createTask(pJob0,'fint01',1,{f,x1,x2});
 disp('pTask0='); disp(pTask0);
%%
 submit(pJob0);  % Send pJob0 to JobManager
 waitForState(pJob0); % wait State(pJob0) == finished
%%
 results=getAllOutputArguments(pJob0);
 disp('results='); disp(results);
```


В результаті виконання файлу **«p_CS_Lab_3_01.m»**, який викликає **паралельну підзадачу** у вигляді m-файлу **«fint01.m»,** обчислив значення інтеграла (results = 0.2500). Отриманий результат співпадає з результатом лабораторної роботи № 2 з точністю до 4 значущих цифр.

7. Виконав паралельні обчислення значення визначеного інтеграла та підібрав таке значення x_3, при якому інтеграл Ip1 на відрізку $[x_1, x_3]$ дорівнює (0.5) від інтеграла Ip на всьому відрізку $[x_1, x_2]$, тобто Ip1 = (0.5+V · 0.005) · Ip. Створив головний файл «p_CS_Lab_3_02.m», а для паралельної підзадачі – файл «fint02.m». Отримані значення лівої mPartIL і правої mPartIR частин інтеграла для кожної ітерації показав у вигляді графіка (рис. 3.2). Для уточненого значення x3 = 0.7031 значення лівої та правої частин інтеграла дорівнюють 0.1236 та 0.1264 відповідно.

Лістинг 1. Головний файл «p_CS_Lab_3_V2.m»

```
pJm0=findResource('scheduler', 'type', 'jobmanager',...
'Name','JobManagerN0','LookupURL','microsof...');
```

```
% Initial Parameter ----------------------------------------------------
  mPartI=zeros(0);
  mPartI(1)=0.5;  % Початкове значення лівої частини інтеграла Ip1
  mPartI(2)=0.5; % Початкове значення правої частини інтеграла Ip2
  Delta_x=0.25;    % крок зміни x3
  x3=0.5;              % початкове значення x3
   mPartIL=zeros(0);    mPartIR=zeros(0);
  % it begin ------------------------------------------------------------
  Qit=5;       % Кількість ітерацій з лічильником it
 for it=1:1:Qit
    %% createParallelJob
pJob0=pJm0.createParallelJob;
    Qw=2;   % Quantity of Workers
 set(pJob0,'MinimumNumberOfWorkers',Qw);
 set(pJob0,'MaximumNumberOfWorkers',Qw);
    %%
 set(pJob0,'FileDependencies',{'fint02.m'});
   %%
f=@(x) ((0.5).*x);  % function for Integral
 x1=0;   % Left Limit of Integral (Global)
 x2=1;   % Right Limit of Integral (Global)
 %% createTask
pTask0=createTask(pJob0,'fint02',1,{f,x1,x2,mPartI,x3,Delta_x});
 % інформація між робочими процесами передається через
 % змінні mPartI, x3
submit(pJob0);  % Send pJob0 to JobManager
 waitForState(pJob0); % wait State(pJob0) == finished
results =getAllOutputArguments(pJob0);
   res1=results';           % транспонування змінної results результату
   res1=cell2mat(res1); % перетворення типу res1 до дійсного
  mPartI=zeros(0);
   mPartI(1)=res1(1);    % значення лівої частини інтеграла Ip1
   mPartI(2)=res1(3);    % значення правої частини інтеграла Ip2
x3=res1(2);                % уточнене значення x3
  Delta_x=Delta_x/2;    % зменшення кроку зміни x3
   mPartIL(it)=mPartI(1);  % Left Part of Integral (it)
   mPartIR(it)=mPartI(2);  % Right Part of Integral (it)
%%
  destroy(pTask0); % очищення підзадачі
  destroy(pJob0);   % очищення задачі
end; % it
```

Лістинг 2. Файл «fint02.m» паралельної підзадачі

```
function pint02 =fint02(f,x1,x2,mPartI,x3,Delta_x);
  % у файлі реалізовано наближення значення x3 з кроком Delta_x
  % отже, щоб отримати
  % потрібне відношення частин інтеграла mPartI(1) та mPartI(2)

 xw1=((labindex-1)/numlabs).*(x2-x1)+x1; % Left Limit of Integral
 xw2=(labindex/numlabs).*(x2-x1)+x1;       % Right Limit of Integral
   if labindex==1
      if mPartI(1)<mPartI(2)
        xw2=x3+Delta_x;
      end;
      if mPartI(1)>mPartI(2)
        xw2=x3-Delta_x;
      end;
   end; % labindex==1
   if labindex==2
      if mPartI(1)<mPartI(2)
        xw1=x3+Delta_x;
      end;
      if mPartI(1)>mPartI(2)
        xw1=x3-Delta_x;
      end;
   end; % labindex==1
    if labindex==1
      x3=xw2;
   end;
   if labindex==2
      x3=xw1;
   end;
   myInt=quadl(f,xw1,xw2);
 pint02(1)=myInt;  % Integral for Worker
 pint02(2)=x3;        % new x3
```

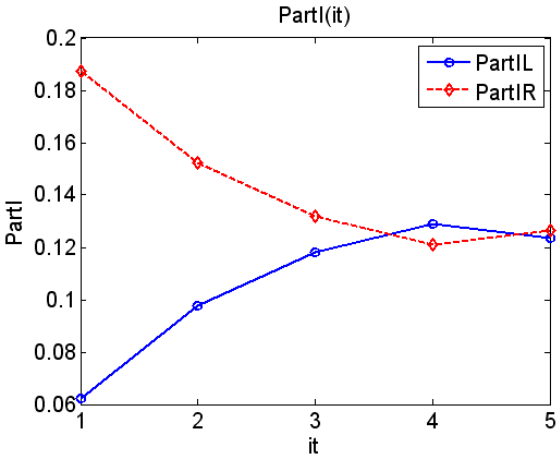


Рис. 3.2. Отримані значення лівої mPartIL і правої mPartIR частин інтеграла для кожної ітерації it

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Які є способи керування паралельними процесами в системі MATLAB?
2. Які дії потрібно виконати перед запуском планувальника задач?
3. Які m-файли використано при виконанні пунктів 1-6 роботи?
4. Як створити посилання на процес планувальника?
5. Як вивести список властивостей і методів об’єкта-посилання на процес планувальника?
6. Опишіть процес створення паралельної задачі та встановлення її параметрів.
7. Опишіть процес створення паралельної підзадачі.
8. Як виконується зв’язок паралельної задачі з m-файлом підзадачі?
9. Опишіть процес виконання паралельної задачі. Як визначається момент її завершення?
10. Як повернути результати розрахунків із паралельної задачі?
11. Поясніть, як реалізовано обмін даними між робочими процесами та прикладі лістингів 1 і 2.
12. Проаналізуйте, як взаємодіють планувальник і робочі процеси при паралельних обчисленнях.
13. Поясність, як можна визначити номер робочого процесу?
14. Поясність, як розпаралелюється процес обчислення визначеного інтеграла з використанням m-файлів?

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Оленёв Н. Н., Печёнкин Р. В., Чернецов А. М. Параллельное программирование в MATLAB и его приложения / Москва: Вычислительный центр им. А. А. Дородницына РАН, 2007. 120 с.
2. Лазарєв Ю. Ф., Лазарєв Ю. Ф. Довідник з MATLAB. Електронний навчальний посібник з курсового і дипломного проектування / Київ: НТУУ «КПІ», 2013. 132 с.
3. Дьяконов В. Matlab 6 : Учебный курс / В. Дьяконов. – Санкт-Петербург: Питер, 2001. 592 с.

Довідка

Перелік термінів і позначень

- JobManagerNV – планувальник паралельних задач.
- WorkerNV1 – робочий процес.
- JobV – паралельна задача.
- TaskV – паралельна підзадача.
- m-файл «p_CS_Lab_3_V1.m» – головний файл (для пунктів 1-6).
- m-файл «fintV1.m» – файл паралельної підзадачі TaskV, викликається з файлу «p_CS_Lab_3_V1.m».
- pJmV – посилання (об’єкт) на процес планувальника паралельних задач «JobManagerNV».
- pJobV – посилання (об’єкт) на паралельну задачу JobV (pJobV=pJmV.createParallelJob).
- pTaskV – посилання (об’єкт) на паралельну підзадачу TaskV (pTaskV=createTask(pJobV,'fintV1',1,{f,x1,x2})).

ЛАБОРАТОРНА РОБОТА № 4
РОЗПАРАЛЕЛЮВАННЯ ЗГОРТКИ ЗОБРАЖЕНЬ У СИСТЕМІ MATLAB

Мета: вивчити принципи паралельного програмування в системі MATLAB, зокрема програмування з використанням m-файлів; навчитися керувати планувальником задач і робочими процесами, здійснювати обмін даними з робочими процесами, виконувати згортку цифрових зображень в системі MATLAB за допомогою паралельних робочих процесів.

Завдання: вивчити можливості пакетів розширень MATLAB Distributed Computing Toolbox та MATLAB Distributed Computing Engine при роботі з m-файлами, встановити параметри планувальника задач JobManager і робочих процесів Worker, виконати згортку цифрових зображень із використанням m-файлів за допомогою чотирьох робочих процесів.

Обладнання: персональний комп'ютер (ПК).

Програмне забезпечення: система MATLAB (R2007b-R2011b).

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Обробка цифрових зображень в системі MATLAB

Цифрові зображення в системі MATLAB [1, 2] зчитуються з графічних файлів (формату *.bmp;*.gif;*.jpg;*.png;*.tif) за допомогою функції «imread» або створюються програмно. Яскравість (інтенсивність) зображень зручно нормувати до діапазону від 0 до 1. Зображення у відтінках сірого записується у вигляді прямокутної матриці $m_f = (m_f(i, k))$ (рис. 4.1):

$$m_f = \begin{bmatrix} m_f(1,1) & \dots & m_f(1,k) & \dots & m_f(1,N) \\ \dots & \dots & \dots & \dots & \dots \\ m_f(i,1) & \dots & m_f(i,k) & \dots & m_f(i,N) \\ \dots & \dots & \dots & \dots & \dots \\ m_f(M,1) & \dots & m_f(M,k) & \dots & m_f(M,N) \end{bmatrix}, \qquad (4.1)$$

де $m_f(i, k)$ – елемент матриці, в якому записується значення яскравості пікселя з індексами i, k;

i – номер рядка пікселів; k – номер стовпця пікселів;

$i = 1, ..., M$; $k = 1, ..., N$; M – висота зображення в пікселах;

N – ширина зображення в пікселах.

Математична модель для кольорових цифрових зображень складніша, оскільки для кожного каналу кольору (червоного, зеленого, синього) в моделі RGB (Red, Green, Blue) створюється окрема двовимірна матриця. У цій роботі всі зображення обробляються у відтінках сірого.

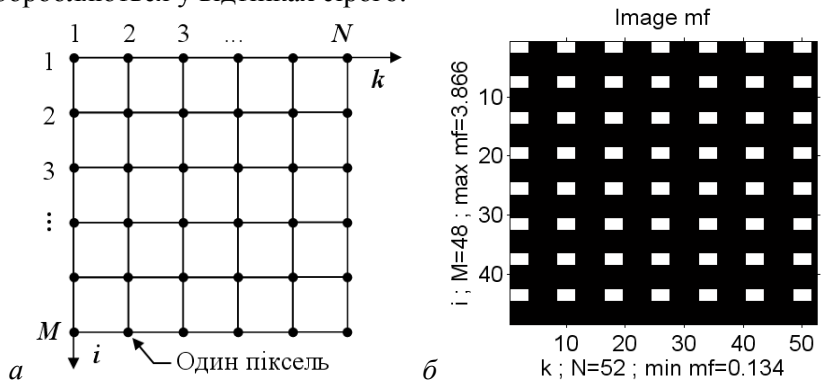


Рис. 4.1. Система координат *ik* цифрового зображення m_f в системі MATLAB: *а* – система координат *ik*;
б – система координат для модельованого зображення

Цифрові зображення в системі MATLAB записуються в графічні файли (формату *.bmp;*.gif;*.jpg;*.png;*.tif) за допомогою функції «imwrite».

1.2. Згортка цифрових зображень в системі MATLAB

Просторова фільтрація цифрових зображень m_f виконується в системі координат *ik* (рис. 4.1) за допомогою згортки зображення $m_f = (m_f(i, k))$ з ядром фільтра $w = (w(m, n))$ за формулою [1]

$$m_g(i,k) = \sum_{m=1}^{Mw} \sum_{n=1}^{Nw} m_f(i-m+m_c, k-n+n_c) \cdot w(m,n), \qquad (4.2)$$

де $m_g=(m_g(i, k))$ – фільтроване зображення (такого ж розміру, що й m_f);
$i = 1, ..., M$, $k = 1, ..., N$; $m = 1, ..., M_w$, $n = 1, ..., N_w$;
M_w – висота ядра фільтра в пікселах;
N_w – ширина ядра фільтра в пікселах;
$m_c = (M_{w2} + 1)$ – центр ядра фільтра за висотою;
$n_c = (N_{w2} + 1)$ – центр ядра фільтра за шириною;
M_{w2}, N_{w2} – цілі частини від половини розміру ядра фільтра *w*, які обчислюються за формулами:

$$M_{w2} = [M_w / 2], \quad N_{w2} = [N_w / 2]. \qquad (4.3)$$

Операція згортки зображення m_f *з* ядром *w* спрощено записується у вигляді

$$m_g = m_f * w. \quad (4.4)$$

Згортка в системі MATLAB [1] виконується командою:

$$m_g = \text{filter2}\ (m_f, \text{w, shape}), \quad (4.5)$$

де при значенні параметра shape = 'same' розмір фільтрованого зображення m_g співпадає з розміром початкового m_f.

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання для всіх варіантів *V* 1...*V* 100

1. На комп'ютері з встановленою системою MATLAB запустити службу **MDCE** [3]; якщо служба відсутня, то її потрібно проінсталювати (див. лабораторна робота № 2). Інформацію про запущений планувальник задач та робочі процеси (в кластері) для вузла (node) отримати командою **nodestatus**

```
>>!path\toolbox\distcomp\bin\nodestatus
    -remotehost hostN -infolevel 3,
```
(4.6)

де

- path – шлях до папки (каталогу), в яку інстальовано MATLAB (наприклад, path = 'C:\MATLABR2007B'). Кореневий каталог установки MATLAB можна визначити, наприклад, командою: >> path = matlabroot.
- ключ «-remotehost» встановлює назву хоста hostN, на якому буде виконано запуск процесу планувальника; у нашому випадку як hostN потрібно вказати ім'я комп'ютера (Панель управления/Система/Имя компьютера).
- ключ «-infolevel» вказує рівень інформативності системних повідомлень (1 – мінімальна, 3 – детальна).

Закрити всі запущені робочі процеси командою «stopworker» і всі планувальники задач командою «stopjobmanager».

2. Виконати запуск планувальника задач **JobManagerNV** у командному вікні MATLAB командою

```
>>!path\toolbox\distcomp\bin\startjobmanager
    -name JobManagerNV -remotehost hostN -v,
```
(4.7)

де

- ключ «-name» встановлює назву планувальника задач JobManagerNV для варіанта *V* (наприклад, для варіанта *V* = 0 назва планувальника «JobManagerN0»);
- hostN – ім'я комп'ютера;

- ключ «-v» дозволяє показувати системні повідомлення при виконанні команди.

Виконати запуск робочого процесу **workerNV1** командою

```
>>!path\toolbox\distcomp\bin\startworker
 -name workerNV1 -remotehost hostN -jobmanager JobManagerNV,
```
(4.8)

де

- ключ «-name» визначає ім'я першого процесу workerNV1 для варіанта *V* (наприклад, для варіанта $V = 0$ назва робочого процесу «workerN01»);
- ключ «-remotehost» встановлює назву хоста hostN;
- ключ «-jobmanager» встановлює назву планувальника задач JobManagerNV для варіанту *V* (аналогічно як в команді 4.7), з цим планувальником JobManagerNV буде асоційований робочий процес workerNV1 (буде отримувати від нього підзадачі).

Аналогічно запустити другий, третій і четвертий робочі процеси: workerNV2, workerNV3, workerNV4.

3. Створити головний **m-файл «p_CS_Lab_4_V.m»** (*V* – номер варіанта), в якому записувати команди пунктів 3, 6. Всі m-файли, які належать цій лабораторній роботі, повинні знаходитися **в одній папці**.

Посилання pJmV (*V* – номер варіанта) **на процес планувальника «JobManagerNV»** отримати за допомогою функції «findResource»:

```
pJmV = findResource('scheduler', 'type', 'jobmanager',...
    'Name','JobManagerNV','LookupURL','hostN'),
```
(4.9)

якій як параметри передати тип процесу ('jobmanager'), назву процесу ('JobManagerNV'), ім'я вузла ('hostN'); інші параметри встановлюються за замовчуванням.

Для створення **паралельної задачі JobV** (*V* – номер варіанта) використати команду

```
pJobV = pJmV.createParallelJob.
```
(4.10)

Результатом виконання даної команди є посилання **pJobV** на паралельну задачу:

```
pJobV = distcomp.paralleljob.
```

Мінімальну Qw_min = 4 і максимальну Qw_max = 4 кількість робочих процесів для паралельної задачі встановити командами:

```
set(pJobV,'MinimumNumberOfWorkers',Qw_min);
set(pJobV,'MaximumNumberOfWorkers',Qw_max).
```
(4.11)

4. Початкове зображення m_f створити в модулі «**p_Create_Image**», розміри m_f обчислити за формулою:

$$M = 48 + V_0,\ N = 52 + V_0, \qquad (4.12)$$

де V_0 – молодша цифра номеру варіанта V.

Яскравість зображення m_f обчислити за формулою:

$$m_f(i,k) = \left(\sin\left(\frac{2\pi \cdot i}{6+V_0}\right)+1\right) + \left(\sin\left(\frac{2\pi \cdot k}{6+V_1+V_0}\right)+1\right), \qquad (4.13)$$

де i = 1, ..., M, k = 1, ..., N; V_0 – молодша цифра номеру варіанта V;

V_1 – старша цифра номеру варіанта V.

Зображення m_f перетворити до бінарного за правилами:

якщо m_f (i,k) < a_mf, то m_f (i,k) = min_mf; (4.14)

якщо m_f (i,k) >= a_mf, то m_f (i,k) = max_mf;

де min_mf – мінімальне значення m_f, max_mf – максимальне значення m_f, a_mf – поріг, який обчислюється виразом:

a_mf = (min_mf + max_mf) · 0.85. (4.15)

5. Створити **паралельну підзадачу TaskV** у вигляді **m-файлу «fint*V*1.m»** (V – номер варіанта), який буде виконуватися всіма робочими процесами одночасно, а запускатися підзадача повинна з файла **«p_CS_Lab_4_V.m»**. Зв'язок паралельної задачі (посилання pJobV) з m-файлом «fint*V*1.m» встановити командою:

set(pJobV,'FileDependencies',{'fint*V*1.m'});

За допомогою паралельної задачі виконати згортку початкового зображення m_f *з* ядром w фільтра. Згортку кожної області (1-4) зображення m_f (рис. 4.2) виконувати робочим процесом з відповідним номером за допомогою ядра w фільтра. Поділ зображення m_f на області виконати згідно з варіантом (рис. 4.2), кожну область пронумерувати (як для V = 0).

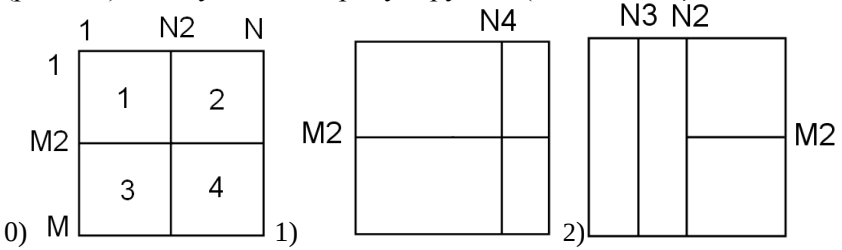
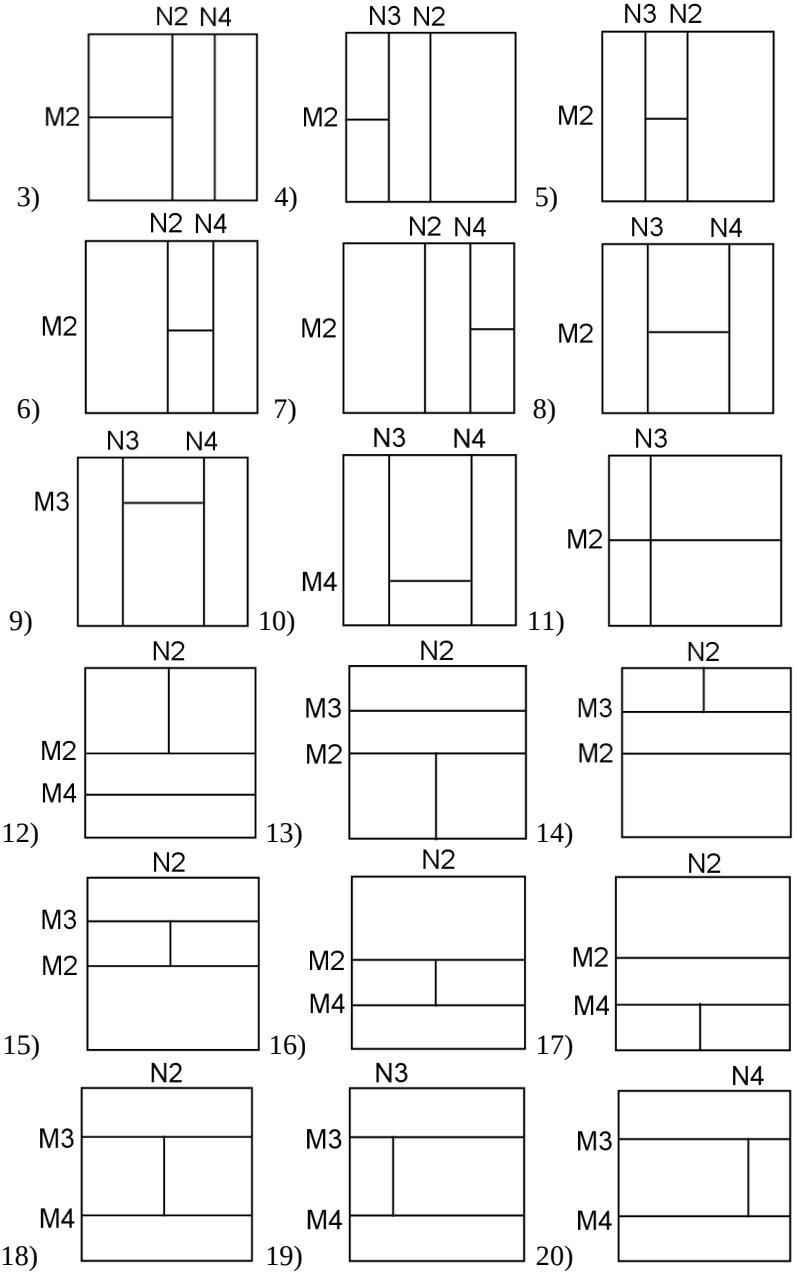


Рис. 4.2. Поділ зображення m_f на 4 області; лівіше від кожного зображення вказано його номер V_N; для варіантів V < 21 номер зображення V_N = V; для інших варіантів V_N =*mod*(V/20).



Продовження рис. 4.2.

Для поділу зображення m_f на області використовуються такі межі за висотою: M2=[M/2], M3=[M/4], M4=[M·3/4], та шириною: N2=[N/2], N3=[N/4], N4=[N·3/4].

У кожному робочому процесі із зображення m_f потрібно отримати відповідну йому область m_{f1} і створити ядро фільтра w (табл. 4.1). Згортку області m_{f1} з ядром фільтра w виконати функцією: mg1 = filter2(w, mf1, shape).

Команди пункту 5 записати в m-файлі **«fint*V*1.m»** паралельної підзадачі, де кожен процес повертає область mg1 після згортки.

Таблиця 4.1

Ядра фільтра *w* для 4 областей зображення m_f;
V_0 – молодша цифра варіанта V; для варіантів $V > 9$ центральний елемент ядра *w* замінити на V_1 – старшу цифру варіанта V

V_0	w1	w2	w3	w4
0	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$	$\begin{pmatrix}0&0&0\\1&1&1\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$	$\begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix}$
1	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$	$\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}$	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$	$\begin{pmatrix}0&0&0\\1&1&1\\0&0&0\end{pmatrix}$
2	$\begin{pmatrix}1&0&1\\0&1&0\\1&0&1\end{pmatrix}$	$\begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix}$	$\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}$	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$
3	$\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}$	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$	$\begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix}$
4	$\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\\1&1&1\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$
5	$\begin{pmatrix}1&0&1\\0&1&0\\1&0&1\end{pmatrix}$	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$	$\begin{pmatrix}0&0&0\\1&1&1\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$

Продовження табл. 4.1

6	$\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}$	$\begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix}$	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$	$\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}$
7	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$	$\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}$	$\begin{pmatrix}0&0&0\\1&1&1\\0&0&0\end{pmatrix}$	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$
8	$\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}$	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$	$\begin{pmatrix}1&0&1\\0&1&0\\1&0&1\end{pmatrix}$	$\begin{pmatrix}0&0&0\\1&1&1\\0&0&0\end{pmatrix}$
9	$\begin{pmatrix}1&0&1\\0&1&0\\1&0&1\end{pmatrix}$	$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$	$\begin{pmatrix}0&1&0\\1&1&1\\0&1&0\end{pmatrix}$	$\begin{pmatrix}0&0&1\\0&1&0\\1&0&0\end{pmatrix}$

6. Для передачі m-файлу **«fint*V*1.m»** до робочого процесу використати об'єкт **pTaskV**. Об'єкт **pTaskV**, який є дочірнім об'єктом для об'єкта pJobV, створити методом createTask:

pTaskV=createTask(pJobV,'fintV1',1,{M,N,mf}). (4.16)

В об'єкт pTaskV через параметри передається кількість вихідних аргументів (в нашому випадку – 1) і вхідні аргументи (M,N,mf). Результатом виконання даної команди є об'єкт:

pTaskV=distcomp.task.

Після цього задачу з посиланням pJobV потрібно направити планувальнику за допомогою команди: submit(pJobV).

Задача починає виконуватися тоді, коли вільними є MinimumNumberOfWorkers процесів. Момент завершення виконання задачі визначається командою

waitForState(pJobV);

після чого результати виконання задачі потрібно повернути командою results=getAllOutputArguments(pJobV).

Із загального результату results отримати результати кожного процесу, наприклад, для процесу 1: results1=results(1,:).

Отримані результати для процесів записати в матриці, наприклад, для процесу 1: mg1=cell2mat(results1).

На основі обчислених областей (mg1, mg2, mg3, mg4) отримати як з фрагментів зображення-результат mg і показати його.

7. Додаткове завдання. Згортку областей зображення виконувати 2 рази (з тим самим ядром фільтра).

У звіті описати хід виконання роботи, лістинги створених m-файлів «p_CS_Lab_4_V.m», «fintV1.m», «p_Create_Image», а також початкове зображення m_f і зображення після згортки m_g.

3. ПРИКЛАД ВИКОНАННЯ РОБОТИ (ВАРІАНТ $V = 0$)

1. На комп'ютері з встановленою системою MATLAB запустив службу **MDCE**. Інформацію про запущений планувальник задач та процеси (в кластері) отримав командою **nodestatus**. Закрив усі запущені процеси і всі планувальники задач.

2. Виконав запуск планувальника задач **JobManagerN0**, виконав запуск робочих процесів **workerN01, workerN02, workerN03, workerN04.**

3. Створив головний m-файл «**p_CS_Lab_4_0.m**» (див. лістинг 1).

Посилання **pJm0** на процес планувальника **«JobManagerN0»** отримав за допомогою функції «findResource»:

```
pJm0=findResource('scheduler', 'type', 'jobmanager',...
    'Name','JobManagerN0','LookupURL','hostN');
```

Вивів список властивостей і методів об'єкта **pJm0** (вставити список).

Створив **паралельну задачу** з посиланням **pJob0** командою

```
pJob0 = pJm0.createParallelJob.
```

Встановив мінімальну (4) і максимальну (4) кількість робочих процесів для паралельної задачі.

Лістинг 1. Головний m-файл «p_CS_Lab_4_0.m»

```
clear all; close all; clc; % clc (Clear Command Window);
disp('  ---------- p_CS_Lab_4_V01 ---------- ');
                    %% Parallel program % JobManager
pJm0=findResource('scheduler', 'type', 'jobmanager',...
'Name','JobManagerN0','LookupURL','microsof-140e50.');
  disp('pJm0='); disp(pJm0);
% ------------------------------------------------------------------
  p_Create_Image; % create initial image mf
%% createParallelJob
pJob0=pJm0.createParallelJob;      disp('pJob0='); disp(pJob0);
Qw=4;   % Quantity of Workers
 set(pJob0,'MinimumNumberOfWorkers',Qw);
```

```
 set(pJob0,'MaximumNumberOfWorkers',Qw);
 set(pJob0,'FileDependencies',{'fint01.m'});
%%% createTask
pTask0=createTask(pJob0,'fint01',1,{M,N,mf});
  disp('pTask0='); disp(pTask0);
%%
 submit(pJob0);  % Send pJob0 to JobManager
 waitForState(pJob0); % wait State(pJob0) == finished
%%
  results=getAllOutputArguments(pJob0);
  results1=results(1,:);  results2=results(2,:);
  results3=results(3,:);   results4=results(4,:);
 %  disp('results1='); disp(results1);
  mg1=cell2mat(results1); mg2=cell2mat(results2);
  mg3=cell2mat(results3);   mg4=cell2mat(results4);
%  disp('mg1=');  disp(mg1);
% Result image mg
    mg=zeros(0);
% fragment - worker 1, 11111111111111111
for i=1:1:M2
  for k=1:1:N2
    mg(i,k)=mg1(i,k);
  end; % k
end; % i
 % fragment - worker 2,  22222222222222222222
for i=1:1:M2
  for k=(N2+1):1:N
    mg(i,k)=mg2(i,k-N2);
  end; % k
end; % i
 % fragment 333333333333333333333
for i=(M2+1):1:M
  for k=1:1:N2
    mg(i,k)=mg3(i-M2,k);
  end; % k
end; % i
% fragment 4444444444444444444444
for i=(M2+1):1:M
  for k=(N2+1):1:N
    mg(i,k)=mg4(i-M2,k-N2);
  end; % k
end; % i
```

Лабораторна робота № 4

4. Початкове зображення m_f створив у модулі «p_Create_Image» (рис. 4.3а, лістинг 2).

Лістинг 2. Модуль «p_Create_Image»

```
M=48; % width of image (pixels)
N=52; % height of image (pixels)
 M2=floor(M/2);   N2=floor(N/2);
 M3=floor(M/4);    N3=floor(N/4);
 M4=floor(M*3/4);   N4=floor(N*3/4);
mf=zeros(0);
for i=1:1:M
  for k=1:1:N
    mf(i,k)=((sin(2*pi*i/6)+1)+(sin(2*pi*k/6)+1));
  end; % k
end; % i
 min_mf=min(min(mf)); max_mf=max(max(mf));
 a_mf=(min_mf+max_mf)*0.85;
 % limit of brightness mf
for i=1:1:M
  for k=1:1:N
    if mf(i,k)<a_mf
       mf(i,k)=min_mf;
    end;
    if mf(i,k)>=a_mf
       mf(i,k)=max_mf;
    end;
  end; % k
 end; % i

% ------------------------ Image mf Show -----------------------------
 Mode_Image_mf_Show=1;  % 1 - show
if (Mode_Image_mf_Show==1)    % scale_XY (i, k) - pixels
set(0,'DefaultAxesFontSize',19,'DefaultAxesFontName','Arial Cyr');
 set(0,'DefaultTextFontSize',19,'DefaultTextFontName','Arial Cyr');
    figure('Color','w');
    % imshow(mf,[]);
    imshow(mf,[],'InitialMagnification','fit');
        Qmap=256;
       mapS = gray(Qmap); cmap='grey'; % grey / black - grey - white   %
mapS=jet(Qmap); cmap='jet'; % jet /dark blue - spectrum - dark red
  %  mapS = bone(Qmap); cmap='bone'; % bone/white-grey+blue-black
  % mapS=copper(Qmap); cmap='copper'; %copper/black-brown-yellow
  % mapS = pink(Qmap); cmap = 'pink'; % pink / white - brown - black
```

```
 % mapS = hsv(Qmap); cmap = 'hsv'; % hsv / read - green - blue - red
 % mapS = cool(Qmap); cmap = 'cool'; % cool / blue - purple
 % map = hot(Qmap); cmap = 'hot'; % hot / black - red - white
 % mapS = prism(Qmap); cmap='prism'; % prism / multiple All spectrum
 %  mapS = spring(Qmap); cmap = 'spring';  % spring / rose – yellow
%mapS = summer(Qmap); cmap = 'summer'; % summer / green-yellow
 %  mapS = autumn(Qmap); cmap = 'autumn'; % autumn / red - yellow
  %  mapS = winter(Qmap); cmap = 'winter'; % winter / blue - green
  % mapS=1-mapS;  % inverse color map
     colormap(mapS);
    % not colorbar;
     colorbar;
  axis on;
    axis('image'); % pixel  - square
   st='i ; M= ';     st=strcat(st,num2str(M));
    st=strcat(st,' ; max mf= ');
    st=strcat(st,num2str(max_mf,'%7.3f')); % '%5.1e'
     ylabel(st);
   st='k ; N= ';     st=strcat(st,num2str(N));
    st=strcat(st,' ; min mf= ');
    st=strcat(st,num2str(min_mf,'%7.3f'));
      xlabel(st);
   st='Image mf';     title(st);
end; %   show mf
```


5. Створив **паралельне завдання** у вигляді m-файлу **«fint01.m»** (лістинг 4.3)**.**

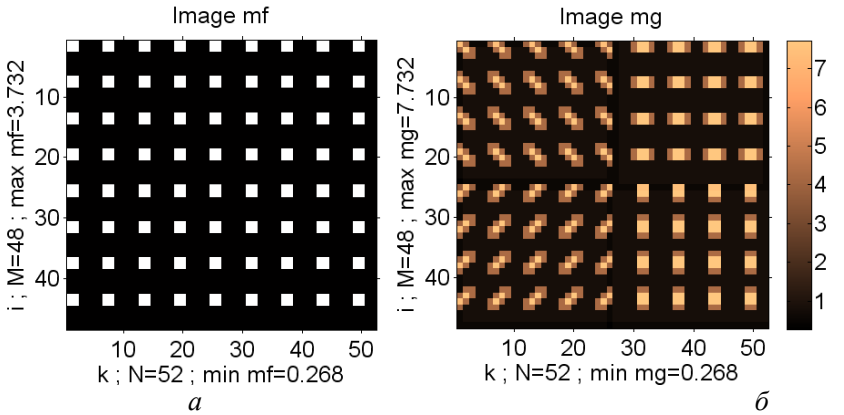
Лістинг 4.3. Модуль «fint01.m»

```
function pint01=fint01(M,N,mf);
  % parallel convolution of image mf, mg = mf * w
  % mg - result image;  w - kernel of filter
  M2=floor(M/2);   N2=floor(N/2);  M3=floor(M/4);    N3=floor(N/4);
  M4=floor(M*3/4);   N4=floor(N*3/4);
    mg1=zeros(0);  % fragment of image mg
     w=zeros(0);
    Mw=3; % width of w
    Nw=3; % height of w
   for m=1:1:Mw
     for n=1:1:Nw
       w(m,n)=0;
     end; % n
   end; % m
```

```
 % 1111111111111111111111111111111111111111111111111111
if labindex==1     % fragment 1
   % w - kernel of filter
   w(1,1)=1;    w(2,2)=1;   w(3,3)=1;
  % convolution
    shape='same'; % size mf
   mf1=mf(1:1:M2,1:1:N2);
   mg1=filter2(w,mf1,shape);
end; % labindex=1
% 2222222222222222222222222222222222222222222222222222
if labindex==2
   w(2,1)=1;   w(2,2)=1;   w(2,3)=1;
    shape='same'; % size mf
   mf1=mf(1:1:M2,(N2+1):1:N);
   mg1=filter2(w,mf1,shape);
end; % labindex=2
 3333333333333333333333333333333333333333333333333333333
if labindex==3
   w(3,1)=1;   w(2,2)=1;   w(1,3)=1;
    shape='same'; % size mf
   mf1=mf((M2+1):1:M,1:1:N2);
   mg1=filter2(w,mf1,shape);
end; % labindex=3
% 4444444444444444444444444444444444444444444444444444444
if labindex==4
   w(1,2)=1;   w(2,2)=1;   w(3,2)=1;
    shape='same'; % size mf
   mf1=mf((M2+1):1:M,(N2+1):1:N);
   mg1=filter2(w,mf1,shape);
 end; % labindex=4
pint01=mg1;  % result fragment of image
```

За допомогою паралельної задачі виконав згортку початкового зображення m_f *з* ядром *w* фільтра. Згортку кожної області (1-4) зображення m_{f1} виконував процесом із відповідним номером за допомогою ядра *w* фільтра (табл. 4.1). Поділ зображення m_f на області виконав згідно з варіантом (рис. 4.2).

6. У результаті виконання файлу **«p_CS_Lab_4_0.m»**, який викликає **паралельну підзадачу** у вигляді m-файлу **«fint01.m»,** виконав згортку зображення m_f з ядрами фільтра *w* (окремо для кожної області). У результаті отримано зображення m_g (рис. 4.3б), для якого згортка кожної області виконана з різними ядрами фільтра *w*.



а *б*

Рис. 4.3. Початкове зображення m_f (*а*) та зображення після згортки m_g (*б*); (кольорова палітра зображення m_g – «copper»)

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Яка модель цифрового зображення використовується в системі MATLAB?
2. Як виконується згортка цифрового зображення з ядром фільтра?
3. Які дії потрібно виконати перед запуском планувальника задач?
4. Як створюється початкове зображення m_f? Пояснити код модуля «p_Create_Image» та отримане зображення m_f.
5. Як виконується згортка областей початкового зображення m_f з ядрами w фільтра? Пояснити код модуля «fintV1.m».
6. Якими параметрами обмінюються головний модуль і модуль підзадачі «fintV1.m»?
7. Як на основі обчислених матриць областей (mg1, mg2, mg3, mg4) отримується все зображення-результат m_g?
8. Проаналізуйте, як взаємодіють планувальник і робочі процеси при паралельних обчисленнях.
9. Поясність, як розпаралелюється процес фільтрації зображення з використанням m-файлів.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB / Москва: Техносфера, 2006. 616 с.
2. Лазарєв Ю. Ф., Лазарєв Ю. Ф. Довідник з MATLAB. Електронний навчальний посібник з курсового і дипломного проектування / Київ: НТУУ «КПІ», 2013. 132 с.
3. Оленёв Н. Н., Печёнкин Р. В., Чернецов А. М. Параллельное программирование в MATLAB и его приложения / Москва: Вычислительный центр им. А. А. Дородницына РАН, 2007. 120 с.

ЛАБОРАТОРНА РОБОТА № 5
МОДЕЛЮВАННЯ РЕШІТЧАСТИХ ТОПОЛОГІЙ ОБЧИСЛЮВАЛЬНИХ СИСТЕМ

Мета: вивчити основні характеристики решітчастих топологій обчислювальних систем, зокрема плоскої решітчастої топології, способи використання дексрипторної графіки в системі MATLAB; навчитися моделювати решітчасті топології в системі MATLAB, виконувати їх тривимірну візуалізацію й обчислювати основні параметри решітчастих топологій.

Завдання: вивчити характеристики решітчастих топологій обчислювальних систем і способи тривимірної візуалізації двовимірних масивів у системі MATLAB, виконати розрахунок структури та параметрів заданої решітчастої топології, провести її тривимірну візуалізацію, побудувати маршрут між вузлами решітчастої топології з використанням дескрипторної графіки.

Обладнання: персональний комп'ютер (ПК).

Програмне забезпечення: система MATLAB (R2007b-R2011b).

1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1. Решітчасті топології обчислювальних систем

Обчислювальна система є мережею, вузли якої зв'язані трактами передачі даних – каналами. У ролі вузлів можуть виступати процесори, модулі пам'яті, пристрої вводу-виводу та групи перерахованих пристроїв. Організація внутрішніх комунікацій обчислювальної системи називається **топологією**. Топологію мережі міжз'єднань (ММЗ) обчислювальної системи визначає множина вузлів ***N***, об'єднана множиною каналів ***C***.

Залежно від вибраної стратегії комутації розрізняють мережі з комутацією **з'єднань** і комутацією **пакетів**. В обох випадках інформація пересилається у вигляді **пакета**. Пакет є групою бітів, для позначення якої використовується термін «**повідомлення**».

У мережах із комутацією **з'єднань** шляхом відповідної установки комутуючих елементів створюється тракт від вузла-джерела до вузла-приймача; такий тракт зберігається, поки весь пакет не досягне вузла-приймача.

Мережі з комутацією **пакетів** припускають, що повідомлення самостійно знаходять шлях до місця призначення. На відміну від мереж із комутацією з'єднань, маршрут може змінюватися. Пакет

послідовно проходить через вузли. Черговий вузол запам'ятовує прийнятий пакет у своєму буфері й аналізує його. Залежно від завантаженості мережі пакет або негайно пересилається до наступного вузла, або пакет додається до черги, яка зберігається у буфері. Якщо **розмір черги** стає більший за допустимий, то за однією зі стратегій маршрутизації відбувається «скидання хвоста» (tail drop) – відмова від прийому нових пакетів.

До **статичних топологій** ММЗ відносять такі, де між вузлами можливий один фіксований шлях. У процесі обчислень статична топологія залишається незмінною. Комутація шляхів або не виконується зовсім, або виконується до початку обчислень.

Одним із видів статичних двовимірних топологій є решітчасті (mesh) топології. Конфігурація решітчастої топології визначається структурою двовимірних масивів даних. Розрізняють такі решітчасті топології (рис. 5.1):

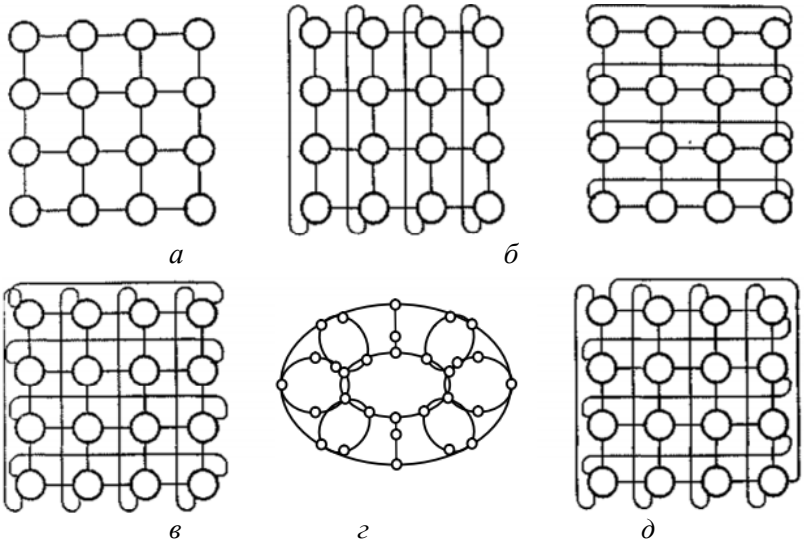


Рис. 5.1. Решітчасті топології: *а* – плоска; *б* – циліндрична; *в, г* – тороїдальна; *д* – вита тороїдальна

Плоска решітчаста топологія передбачає, що кожен її вузол сполучений із найближчим сусідом (рис. 5.1а). Така мережа розмірності **m × m** має такі характеристики: діаметр **D = 2(m-1)**; порядок вузла **d =4**; кількість зв'язків **I=2N-2m**; ширина бісекції **B = m**.

Якщо провести операцію топологічного загортання (wraparound) плоскої матриці, з'єднавши інформаційними трактами однойменні вузли лівого та правого стовпців або однойменні вузли верхнього і нижнього рядків плоскої матриці, то з плоскої конструкції отримуємо топологію **циліндра** (рис. 5.1б). У топології циліндра кожен рядок (або стовпець) матриці є кільцем.

Якщо провести топологічне загортання плоскої матриці в обох напрямах, то отримаємо **тороїдальну** топологію мережі (рис. 5.1в). Двовимірний тор на базі решітки **m × m** володіє такими параметрами: діаметр **D = 2 min(m/2)**; порядок вузла **d =4;** кількість зв'язків **I=2N**; ширина бісекції **B = 2m**. Об'ємний вигляд тороїдальної топології для масиву 4 × 8 показано на рис. 5.1г.

Крім загортання до плоскої решітки, може застосовуватися операція **скручування** (twisting). Суть операції в тому, що замість кілець всі вузли об'єднуються в розімкнену або замкнуту **спіраль,** тобто вузли, розташовані з протилежних країв плоскої решітки, з'єднуються з певним зсувом. Якщо горизонтальні петлі об'єднані у вигляді спіралі, то утворюється так звана мережа типу **ILLIAC.** На рис. 5.1д зображена подібна конфігурація ММЗ, що характеризується такими параметрами: діаметр **D = m-1**; порядок вузла **d =4**; кількість зв'язків **I=2N**; ширина бісекції **B = 2m.**

1.2. Побудова тривимірних поверхонь у системі MATLAB

У системі MATLAB двовимірні масиви $f_s = (f_s(i, k))$, де $i = 1,..., m$, $k = 1,..., n$, можуть бути візуалізовані у вигляді поверхонь, при цьому висота певної точки поверхні (вісь z) визначається значенням відповідного елемента масиву f_s. При моделювання решітчастих топологій кожному елементу масиву f_s відповідає вузол мережі. Використовується три способи візуалізації поверхні:

1. Функція «surf (mk,mi,fs)» – візуалізації решітки вузлів (grid) із зафарбовуванням областей між вузлами; колір поверхні залежить від її висоти.

2. Функція «surfl (mk,mi,fs,S_Light, K_Light)» – візуалізації решітки вузлів (grid) із зафарбовуванням областей між вузлами; колір поверхні залежить від її освітлення; параметри S_Light задають положення джерела світла, параметри K_Light описують відбивну здатність матеріалу поверхні.

3. Функція «mesh (mk,mi,fs)» візуалізації решітки вузлів (grid) без зафарбовування областей між вузлами.

В масивах mk та mi записуються номери елементів (вузлів) вздовж осей x (індекси k) та y (індекси i) відповідно.

В режимі тривимірної (3D) візуалізації вигляд поверхні визначається такими параметрами (рис. 2.7):

1. Опція «shading interp», яка забороняє візуалізацію сітки (grid) поверхні.

2. Параметри, які задають положення поверхні у просторі: AZS (Azimuth) – азимут, ELS (Elevation) – кут піднесення.

3. Параметри джерела світла S_Light=[AZL,ELL]: AZL (Azimuth) – азимут, ELL (Elevation) – кут піднесення.

4. Параметри відбивної здатності матеріалу поверхні K_Light=[K_background, K_diffuse, K_specular, K_glossy], які враховують вклад фонового, дифузного, дзеркального освітлення, а також ефект глянцю (наприклад, K_Light=[0.02,0.025,0.0,0.0]).

5. Прозорість поверхні alpha, яка змінюється від 0 до 1, при «0» поверхня повністю прозора, при «1» – повністю непрозора.

1.3. Дескрипторна графіка в системі MATLAB

Дескриптор в системі MATLAB – це змінна, в яку записується вказівник на об’єкт. Дескриптор є унікальним числовим значенням, що дозволяє відрізнити один об’єкт від іншого. Наприклад, дескриптор hDp графіку: hDp = plot3(...).

Дескрипторна графіка дає можливість повного контролю над елементами програми, зокрема, над графічними об’єктами. Наприклад, отримавши дескриптор hDp графіку, можна зчитати та/або змінити всі його властивості:

```
wp=get(hDp,'LineWidth');    % зчитати у wp товщину лінії графіка
set(hDp,'LineWidth',wp+1); % збільшити товщину лінії графіка на 1
set(hDp,'Visible', 'off');        % зробити графік невидимим
set(hDp,'Visible', 'on');         % зробити графік видимим
```

2. ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Завдання для всіх варіантів *V* 1...*V* 100

1. Створити **m-файл «p_CS_Lab_5_NV.m»** (*V* – номер варіанту), в якому записувати команди для моделювання плоскої

решітчастої топології обчислювальної системи. Розміри решітки $m \times n$ обчислити за формулою:

$$m = 25 + V_0,\ n = 25 + V_1, \qquad (5.1)$$

де V_0 – молодша цифра номеру варіанта V; V_1 – старша цифра номеру варіанта V.

Для кожного вузла решітки обчислити розмір черги пакетів, який зберігається в буфері, і записати його в масив $f_s = (f_s(i, k))$, де $i = 1,..., m$, $k = 1,..., n$. Значення елементів масиву f_s обчислити як суму двох двовимірних розподілів Гауса за формулою:

$$f_s(i,k) = \sum_{p=1}^{2} a_{cp} \cdot \exp\left(-\frac{(i - m_{cp})^2 + (k - n_{cp})^2}{2\sigma_{cp}^2} \right), \qquad (5.2)$$

де p – номер двовимірного розподілу Гауса;

a_{cp} – амплітуда розподілу;

m_{cp} – центр розподілу вздовж осі i;

n_{cp} – центр розподілу вздовж осі k;

σ_{cp} – середнє квадратичне відхилення (СКВ) розподілу.

Значення параметрів (a_{cp}, m_{cp}, n_{cp}, σ_{cp}) обчислити за формулами:

$$a_{c1} = 1 + V_0,\ a_{c2} = 0.5 \cdot a_{c1}, \qquad (5.3)$$

$$m_{c1} = [m/2] + V_0,\ m_{c2} = [0.15 \cdot m] + V_0, \qquad (5.4)$$

$$n_{c1} = [n/2] + V_1,\ n_{c2} = [0.15 \cdot n] + V_1, \qquad (5.5)$$

$$\sigma_{c1} = 0.15 \cdot m,\ \sigma_{c2} = 0.1 \cdot m. \qquad (5.6)$$

2. Виконати тривимірну візуалізацію масиву f_s за допомогою функцій «surf», «surfl» та «mesh» (по черзі). Уточнити просторове положення поверхні за допомогою опції «Tools/ Rotate 3D». Отримані екранні форми додати до звіту.

3. Створити маршрут № 1 між діагональними елементами масиву f_s і показати його як набір точок на поверхні за допомогою функції plot3. Маршрут може проходити або вздовж осі i, або вздовж осі k. Координати вузлів маршруту зберігати в масивах mrk, mri, mrz. Отримати дескриптор hDp графіку plot3. За допомогою дескриптора збільшити товщину лінії графіка на 1, а також 3 рази по черзі зробити його невидимим і видимим (із затримкою pause(0.3)).

4. Створити та показати на поверхні масиву f_s маршрут № 2 між центрами розподілів (m_{c1}, n_{c1}) та (m_{c2}, n_{c2}). Візуально оформити маршрут № 2 по іншому, ніж маршрут № 1.

5. Обчислити параметри (діаметр, порядок вузла, кількість зв'язків, ширина бісекції) решітчастої топології розміру $m \times m$ для плоскої та тороїдальної топологій. Отримані параметри додати до звіту.

6. Додаткове завдання. Обчислити параметри (діаметр, порядок вузла, кількість зв'язків, ширина бісекції) решітчастої топології розміру $m \times n$ для плоскої та тороїдальної топологій. Отримані параметри додати до звіту.

У звіті описати хід виконання роботи, лістинг створеного m-файлу «p_CS_Lab_5_NV.m» з поясненнями, отримані екранні форми масиву f_s з двома маршрутами, а також параметри (діаметр, порядок вузла, кількість зв'язків, ширина бісекції) решітчастої топології.

3. ПРИКЛАД ВИКОНАННЯ РОБОТИ (ВАРІАНТ $V = 0$)

1. Створив **m-файл «p_CS_Lab_5_N0.m»** (V – номер варіанта), в якому записав команди для моделювання плоскої решітчастої топології обчислювальної системи. Розміри решітки $m \times n$ обчислив за формулою (5.1):

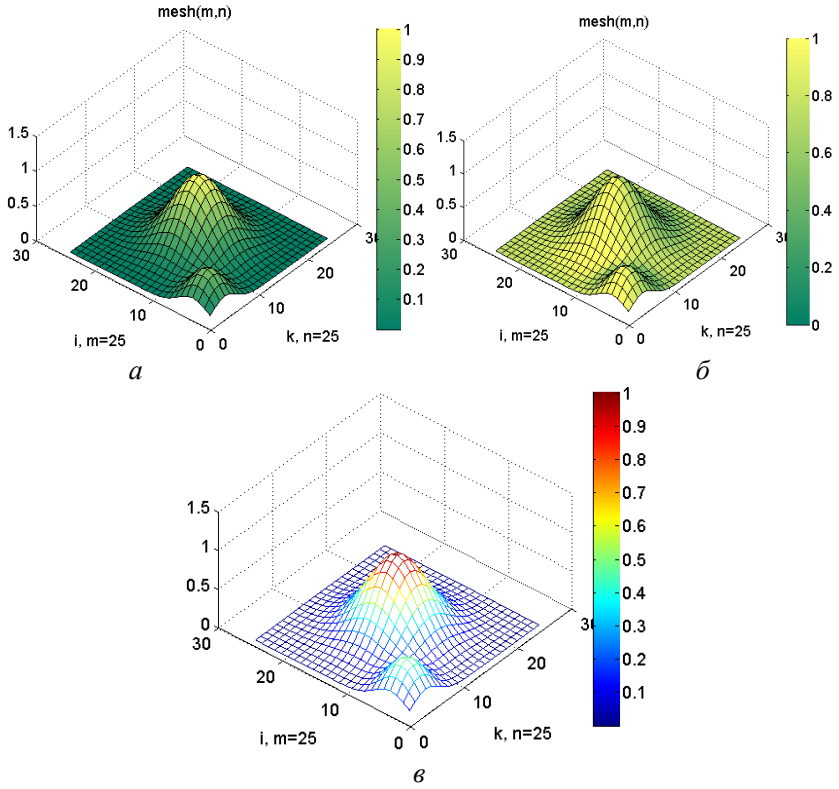
$$\mathrm{m} = 25,\ \mathrm{n} = 25.$$

Для кожного вузла решітки обчислив розмір черги пакетів, який зберігається в буфері, і записав його в масив $f_s = (f_s(i, k))$, де $i = 1,..., m$, $k = 1,..., n$. Значення елементів масиву f_s обчислив як суму двох двовимірних розподілів Гауса за формулою (5.2):

$$f_s(i,k) = \sum_{p=1}^{2} a_{cp} \cdot \exp\left(-\frac{(i - m_{cp})^2 + (k - n_{cp})^2}{2\sigma_{cp}^2} \right),$$

де $a_{c1} = 1$, $a_{c2} = 0.5$,
$m_{c1} = [\mathrm{m}/2]$, $m_{c2} = [0.15 \cdot \mathrm{m}]$,
$n_{c1} = [\mathrm{n}/2]$, $n_{c2} = [0.15 \cdot \mathrm{n}]$,
$\sigma_{c1} = 0.15 \cdot \mathrm{m}$, $\sigma_{c2} = 0.1 \cdot \mathrm{m}$.

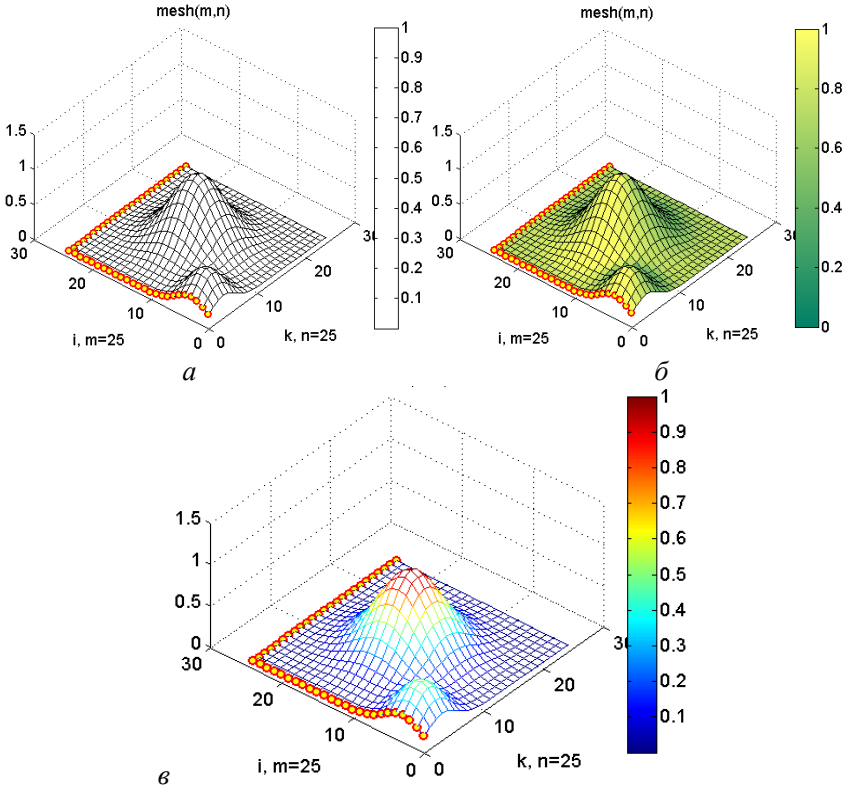
2. Виконав тривимірну візуалізацію масиву f_s за допомогою функції «surf», «surfl» та «mesh» (рис. 5.2).



а *б* *в*

Рис. 5.2. Тривимірна візуалізація масиву f_s за допомогою функцій «surf» (*а*), «surfl» (*б*) та «mesh» (*в*)

3. Створив маршрут № 1 між діагональними елементами масиву f_s і показав його як набір точок на поверхні за допомогою функції plot3 (рис. 5.3). Отримав дескриптор hDp графіку plot3. За допомогою дескриптора збільшив товщину лінії графіка на 1, а також 3 рази зробив його невидимим і видимим (із затримкою pause(0.3)).



а *б* *в*

Рис. 5.3. Тривимірна візуалізація масиву f_s та маршруту № 1 за допомогою функцій «surf» (map = white) (*а*), «surfl» (*б*) та «mesh» (*в*)

4. Створив і показав на поверхні масиву f_s маршрут № 2 (аналогічно до маршруту № 1).

5. Обчислив параметри (діаметр, порядок вузла, кількість зв'язків, ширина бісекції) решітчастої топології розміру $m \times m$ для плоскої та тороїдальної топологій:

Flat mesh:

Diameter D= 48, d= 4

Connections I= 1200, Bisect B= 25

Tor:

Diameter Dt= 24, dt= 4

Connections It= 1250

Bisect Bt= 50

Лістинг 1. Головний m-файл «p_CS_Lab_5_N0.m»

```
clear all; close all; clc; % clc (Clear Command Window);
disp('   ---------- p_CS_Lab_5_N0 ---------- ');
   m=25;  % mesh height (Node)
   n=25;   % mesh width (Node)
mc1=round(m*0.5);  % peak 1
nc1=round(n*0.5);
   sigmac1=m*0.15;
   ac1=1;                  % amplitude peak 1
mc2=round(m*0.15);  % peak 2
nc2=round(n*0.15);
   sigmac2=m*0.1;
   ac2=0.5;                % amplitude peak 2
 % set Node parameter ----------------------------------------------
   fs=zeros(0);
  for i=1:1:m
    for k=1:1:n
      dmn2=((i-mc1)^2+(k-nc1)^2); % (distance to (mc1,nc1)) ^2
       fs(i,k)=ac1.*exp(-dmn2./(2*(sigmac1)^2));
      dmn2=((i-mc2)^2+(k-nc2)^2); % (distance to (mc2,nc2)) ^2
       fs(i,k)=fs(i,k)+ac2.*exp(-dmn2./(2*(sigmac2)^2));
    end; % k
  end; % i
   mi=1:1:m;   mk=1:1:n;
% route (маршрут )-----------------------------------------------
 mri=zeros(0);   % route i
 mrk=zeros(0);  % route k
 mrz=zeros(0);   Qr=m+n-1;
 for i=1:1:m
     p=i;     mri(p)=i;     mrk(p)=1;      mrz(p)=fs(i,1)+ac1*0.01;
 end;
 for k=2:1:n
     p=p+1;     mri(p)=m;     mrk(p)=k;     mrz(p)=fs(m,k);
 end;
% Show fs 3D / Surface  ------------------------------------------
   Mode_Show_fs_3D=1;  % 1 - 3D
 if Mode_Show_fs_3D==1
   figure('Color','w');
      % Light parameters ---------------------------------------
         AZL=-75;  % AZL - Azimuth Light
         ELL=35;   % ELL - Elevation Light, 35
        S_Light=[AZL,ELL];  % S_Light=[AZL.ELL];
```

```
      %    K_Light=[K_background, K_diffuse, K_specular, K_glossy];
        K_Light=[0.02,0.025,0.0,0.0];
    %   surf(mk,mi,fs); % grid + Color, Color = f(fs); real Scale z
     surfl(mk,mi,fs,S_Light,K_Light); % grid+Lighting, Scale z [0..1]
    %   mesh(mk,mi,fs);   % mesh, grid
     alpha(1.0);  % 0 - surface transparency, 1 - not transparency
      hold on;  % hold surface
      AZS=-50;   % Azimuth Surface, -40
      ELS=53;    % Elevation Surface, 53
     view(AZS,ELS);  % Rotate Surface 3D
     % shading interp;  % not grid
     % shafing faceted; % RGB Color
      % box on; % 3D cube
      % box off;
      Qmap=256;
      map = gray(Qmap); cmap='grey'; % grey / black - grey - white
     %  map = white; cmap='white';
% map=jet(Qmap); cmap='jet'; % jet /dark blue - spectrum - dark red
% map = bone(Qmap); cmap='bone'; % bone / white-grey + blue-black
% map = copper(Qmap); cmap='copper';
% map = pink(Qmap); cmap = 'pink'; % pink / white - brown - black
% map = hsv(Qmap); cmap = 'hsv'; % hsv / read - green - blue - red
       % map = cool(Qmap); cmap = 'cool'; % cool / blue - purple
       % map = hot(Qmap); cmap = 'hot'; % hot / black - red - white
       % map = prism(Qmap); cmap='prism'; % prism
      %  map = spring(Qmap); cmap = 'spring';  % spring / rose - yellow
map = summer(Qmap); cmap = 'summer'; % summer / green - yellow
%  map = autumn(Qmap); cmap = 'autumn'; % autumn / red - yellow
      %  map = winter(Qmap); cmap = 'winter'; % winter / blue - green
        colormap(map);
        colorbar;
     grid on;
      % grid off;
Mode_Point_Show=1;
  if Mode_Point_Show==1
    hDp=plot3(mrk, mri, mrz, 'or',...
      'MarkerEdgeColor','r',...
      'MarkerFaceColor','y',...
      'MarkerSize',6, 'LineWidth',1);
     wp=get(hDp,'LineWidth');
     set(hDp,'LineWidth',wp+1);
  end; % Mode_Point_Show==1
```

```
    axis on;    % axis off;
    st='i, m=';    st=strcat(st,num2str(m));       ylabel(st);
    st='k, n= '; st=strcat(st,num2str(n));    xlabel(st);   st='mesh(m,n)';
 title(st);
   for dp=1:1:0
 pause(0.3); set(hDp,'Visible', 'off');  pause(0.3);  set(hDp,'Visible', 'on');
   end; % dp
end; % if Mode_Show_fs_3D==1
```

КОНТРОЛЬНІ ЗАПИТАННЯ ТА ЗАВДАННЯ

1. Що таке топологія обчислювальної системи?
2. У чому полягає різниця між мережами з комутацією з'єднань і мережами з комутацією пакетів?
3. Які топології обчислювальних систем відносять до статичних?
4. Опишіть основні характеристики плоскої решітчастої топології.
5. Поясніть значення параметрів мереж міжз'єднань: діаметр, порядок вузла, кількість зв'язків, ширина бісекції. Наведіть приклади.
6. Опишіть основні види решітчастих топологій. Проаналізуйте їх параметри.
7. Опишіть три способи візуалізації поверхні на основі двовимірного масиву, які використовуються в системі MATLAB.
8. Як створюються та використовуються дескриптори графічних об'єктів у системі MATLAB?
9. Поясніть програмний код створеного файлу «p_CS_Lab_5_NV.m».
10. Поясність отримані екранні форми масиву f_s з двома маршрутами. Проаналізуйте, які параметри мережі можна отримати на основі маршруту № 1.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB / Москва: Техносфера, 2006. 616 с.
2. Лазарєв Ю. Ф., Лазарєв Ю. Ф. Довідник з MATLAB. Електронний навчальний посібник з курсового і дипломного проектування / Київ: НТУУ «КПІ», 2013. 132 с.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DCT – Distributed Computing Toolbox (набір інструментів для розподілених обчислень), пакет розширень для паралельного програмування в MATLAB R2007b.

MDCE – MATLAB Distributed Computing Engine (розподілений обчислювальний рушій MATLAB), пакет розширень для паралельного програмування в MATLAB R2007b.

MIMD – multiple instruction, multiple data (множинний потік команд, множинний потік даних).

MPI (Message Passing Interface) – інтерфейс передачі повідомлень.

ММЗ – мережа міжз'єднань.

ЗМІСТ

Навчальне видання

КОМП'ЮТЕРНІ СИСТЕМИ

Методичні вказівки до лабораторних робіт

Укладачі:
Баловсяк Сергій Васильович,
Одайська Христина Савеліївна

Відповідальний за випуск Г. І. Воробець

Літературний редактор О. В. Колодій

Підписано до друку 23.12.2021. Формат 60 х 84/16
Папір офсетний. Друк різографічний. Ум. -друк. арк. 4,0.
Обл.-вид. арк. 4,2. Тираж 50. Зам. Н-148
Видавництво та друкарня
Чернівецького національного університету
58002, Чернівці, вул. Коцюбинського,2
Свідоцтво суб'єкта видавничої справи
ДК № 891 від 08.04.2002 р.