Зміст

Вступ

1. Програмне середовище

1.1 Загальні характеристики

Для швидкої та нескладної організації моделювання складних високорівневих задач дискретної математики використовують різноманітні обчислювальні середовища, в яких застосовується програмна технологія скриптів. Серед таких середовищ найбільш відомим і популярним (і доволі дорогим) є Matlab. Такоє є альтернативне вільне програмне забезпечення, таке як Octave, Numerical Python і Scilab.

У всіх цих середовищ для чисельного моделювання є загальні властивості: мова, яка забезпечує зручну маніпуляцію з векторами і матрицями, інструменти для візуалізації чисельних результатів, велика кількість простих у використанні бібліотек, які спеціалізовані для вирішення математичних задач у окремих галузях математики і техніки. Середовище може використовуватись у інтерактивному режимі — як інтерпретатор процедур або виконувати програми, описані у файлах (сценарії чи скрипти або макроси). Така розробка та виконання сценаріїв дає набагато менше навантаження на програміста, ніж при програмування на будь-якій імперативній мові високого рівня.

Scilab — це вільне програмне забезпечення, створене в INRIA (Франція) для дослідження алгоритмів вирішення різноманітних задач чисельної обробки. Scilab широко використовується як з ОС Windows, так і Linux. Це середовище прийнято в багатьох університетах і компаніях по всьому світу. За своїми властивостями, інфраструктурою, підтримкою спільноти користувачів та простотою функціональності для наукових обчислень та інженерії Scilab переважає такі середовища, як Octave та Python. Тому лабораторні роботи по курсу пропонується виконувати у програмному середовищі Scilab.

1.2 Інсталяція Scilab

Програма для інсталяції середовища Scilab зберігається за адресою: https://www.scilab.org/download і є вільною для використання. Можна вибрати інсталяційну програму для ОС Windows, GNU/Linux, Mac OS X з розрядністю 32 чи 64. Бажано використати версію 5, яка супроводжується пакетом (Atom) для креслення графів.

Після запуску програми інсталяції вона автоматично розпаковується у вибраний каталог і середовище Scilab відкривається для використання. При цьому бажано вибрати українську мову для інсталяції. Вікно середовища показано на рис.1.

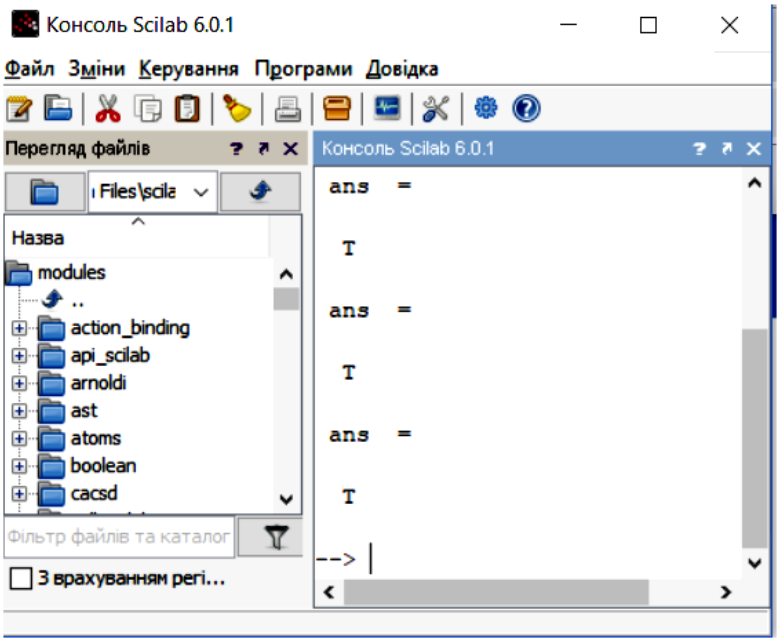

Рис. 1 — Загальний вид середовища Scilab

Другим кроком інсталяції є підключення пакету для креслення графів. Для цього у меню налаштування «Програми» — «Керування модулями — Atoms» у каталозі «Графи» слід вибрати пакет «Metanet» і натиснути кнопку «Встановити» (Рис.2).

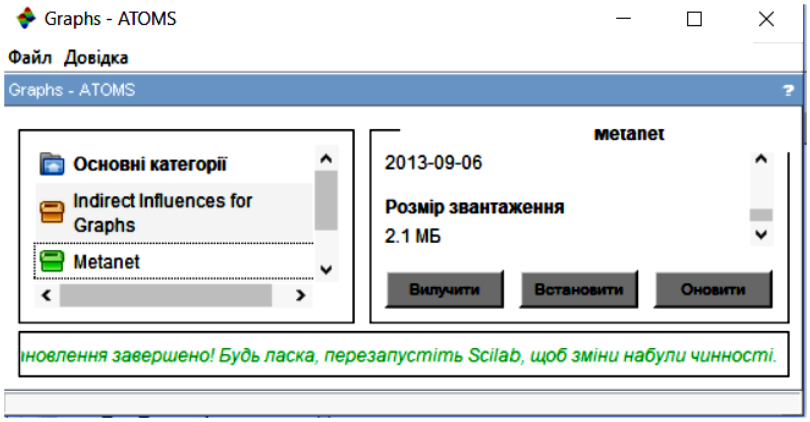

Рис. 2 — Вікно для встановлення додаткових пакетів середовища Scilab

Пакет «Metanet» відповідної версій буде знайдений у сховищі на сайті Scilab, переписаний і встановлений у відповідне місце середовища на комп'ютері користувача з відповідним настроюванням зв'язків.

1.3 Архітектура Scilab

1.3.1 Характеристика Scilab

Система Scilab складається з інтерпретатора, редактора та програмного середовища, яке допомагає формувати та виконувати команди та скрипти, які написані мовою скриптів Scilab.

Основним блоком системи є інтерпретатор. Він виконує у інтерактивному режимі команди Scilab та скрипти, які вводяться у вікні консолі системи. При цьому скриптом є програма, яка написана мовою Scilab і збережена як файл з розширенням .SCI з іменем, за яким цей скрипт запускається з консолі чи викликається іншими скриптами, якщо він організований як бібліотечний скрипт.

Об'єкти, які обробляються скриптами, є змінні, структури та файли. Змінні, як правило, представлені числами з плаваючою комою у форматі Double і є динамічними змінними. Структурами є динамічні масиви та записи. Змінні, які об'явлені у функціях чи скриптах, що викликаються, є локальними.

Передача змінних та структур у скрипти та функції, які викликаються, виконується, як правило, за посиланням.

Результати виконання команд відображаються негайно у вікні консолі або у графічному вікні. У останньому випадку автоматично кресляться графіки залежностей чи геометричні фігури (наприклад, граф) і текст.

Головне меню системи має команди для роботи з файлами, настроювання середовища, редактування команд та одержання довідкової інформації.

1.3.2 Середовище і меню Scilab

Пункт меню «Файл» призначений для роботи з файлами. Його команди:

• New Scilab открывает новое окно Scilab, фактически пакет запускается повторно;

• «Виконати» — запускає на виконання створений раніше Scilab-скрипт (файли з розширенням .SCE чи .SCI);

• «Відкрити файл» — відкриває вікно для завантаження файла, який створено раніше, рисунка чи моделі;

• «Завантажити середовище» — відкриває вікно для завантаження файлу, який був збережений раніше, з параметрами середовища, включаючи команди, введені з консолі, визначені та розраховані змінні та структури;

• «Зберегти середовище» — зберігання у файлі з розширенням .SOD усіх змінних і функцій, які визначені в данній сесії;

• «Змінити поточний каталог» — зміна поточного робочого каталогу, де зберігаються скрипти користувача;

• «Показати поточний каталог» — виводить в командний рядок консолі ім’я поточного каталога;

• «Вийти» — вихід з системи Scilab.

Пункт меню «Зміни» вміщує типові команди по редагуванню рядків поточної сесії у вікні консолі, а також у підпункті «Налаштування» дає змогу налаштувати шрифт, кольори ключових слів, літералів та ін.

Пункт меню «Керування» має команди, які можуть перервати чи призупинити виконання складних скриптів.

Пункт меню «Програми» призначений для виклику та настроювання допоміжних програм системи. Його команди:

• «SciNotes» — викликає редактор скриптів;

• «Xcos» — викликає програму – симулятор потокових скриптів;

• «Керування модулями — Atoms» — дає змогу підключити до системи різноманітні пакети, які розроблені спільнотою користувачів Scilab;

• «Перегляд змінних» — відкриває вікно, у якому зручно дослідити стан змінних, які використовуються у даній сесії;

• «Журнал команд» — відкриває вікно, у якому відображається послідовність у часі команд, які були виконані інтерпретатором.

Пункт меню «Довідка» — дає змогу знайти довідку на усі команди, функції, пакети Scilab, а також запустити демонстраційні скрипти з ними.

Після запуску системи Scilab у вікні консолі з’являється командний рядок, який помічено символами: « -->». У цьому рядку можна вводити команди Scilab. Якщо введена команда кінчається вводом Enter, то вона виконується і її результат відображається у консолі. Якщо вона кінчається символом «;» і Enter, то команда виконується, але її результат не виводиться у консолі.

Виконати оператори файла-скрипта можна кількома способами:

• з меню редактора SciPad викликати команду «Виконати» — «Зберегти і виконати» або виконати частину скрипту за відповідними командами;

• з меню Scilab викликати команду «Виконати» і вказати ім’я файла-скрипта;

• у командному рядку набрати команду **exec** “ім’я файла скрипта разом з розширенням”.

1.4 Синтаксис, оператори і функції Scilab

1.4.1 Коментарі

Коментарі у скрипті записуються після двох похилих, наприклад:

// це коментар

1.4.2 Змінні

Змінна декларується під час свойого першого присвоювання за оператором «=». При цьому тип змінної (число з плаваючою комою, ціле число, символ, рядок, булева змінна) визначається типом результату виразу у правій частині оператору присвоювання. Наприклад, змінній d з плаваючою комою присвоюється значення 5.0 у командному рядку:

--> d = 2. + 3. ;

Тут символи «-->» — запрошення до вводу команди.

Якщо у командному рядку чи скрипті виконується вираз без оператора присвоювання, то виконується неявне присвоювання системній змінній ans, наприклад:

--> 1 + 2

ans =

3.

Ідентифікатори змінних повинні мати латинські літери чи десяткові цифри, чи знак підкреслення, але починатись з літери. Системні змінні можуть починатись сецсимволами «%», «#», «!», «$», «?». Так, вбудовані константи:

%pi — число π=3.1415,

%inf — машинний символ нескінченності (∞),

%T — логічна 1,

%F — логічний 0.

Ідентифікатори з однаковим написанням, але з малими і великими літерами вважаються різними, тобто, вони є чутливими до висоти букв. Наприклад:

data = 1;

Data = 2;

є присвоюваннями різним змінним.

Щоб дізнатись наявне значення змінної, слід набрати її ім’я у командному рядку і натиснути Enter.

Щоб відмінити (стерти) змінну у системі, використовують функцію **clear**:

clear data Data;

1.4.3 Змінні типу вектора і матриці

Змінні типу вектора і матриці зберігають одновимірний та двовимірний масиви, відповідно. Вектор розглядається як матриця, яка по одній з розмірностей має одиничний розмір. Тобто, вони розрізняються як рядки (вектори) та стовпці (ковектори).

Нумерація елементів масивів починається з 1. Щоб створити вектор v довжиною k, який складається з елементів арифметичної прогресії, використовують операцію:

v = x1:dx:xk;

де x1 — значення першого елемента вектора, dx — крок зміни значення, xk — значення останнього елемента масиву, тобто, другий елемент дорівнюватиме x1+ dx, третій — x1+ 2*dx і т.д. до xk. За замовчуванням, dx =1, тобто, за операцією

Y = 0:5;

сформується масив Y довжиною 6, який складається з елементів 0,1,…,5.

Іншим чином такий масив — вектор-рядок — задається простим переліченням його елементів через пробіл:

Y = [0 1 2 3 4 5];

При завданні вектора-стовпця його елементи перелічують через крапку з комою:

Yc = [0; 1; 2; 3; 4; 5];

До елемента вектора звертаються за його індексом у круглих дужках:

y2 = Y(2);

Послідовність сусідніх елементів виділяють за допомогою двокрапки:

--> Y(2:4)

ans =

1. 2. 3.

Ввід елементів матриці також виконують у квадратних дужках, перелічуючи елементи по рядках. Причому елементи розділяються пробілами чи комами, а рядки — крапкою з комою. Наприклад:

--> X = [0 1 2; 1 2 3; 2 3 4]

X =

0. 1. 2.

1. 2. 3.

2. 3. 4.

Крім того, таку матрицю можна формувати з векторів-рядків:

v1= [0 1 2]; v2 = [1 2 3]; v3 = [2 3 4];

X = [v1; v2; v3];

Щоб звернутись до елемента матриці, застосовують круглі дужки:

x12 = X(1,2);

Щоб прочитати зразу цілий рядок матриці, наприклад, другий, використовують двокрапку:

v2 = X(2,:);

а якщо стовпець — то також:

c2 = X(:, 2);

Пустий масив позначається як «[]». За допомогою його можна видалити з матриці заданий стовпець:

X(:,3) = [];

або рядок:

X(2,:) = [];

Так само з вектора можна видалити заданий елемент:

v1(2) = [];

Одинична матриця *I* одержується з такими самими розмірами, як матриця A за функцією:

eye(A);

або з розмірами 4 на 5:

eye(4,5);

Матриця таких самих розмірів, яка заповнена одиницями, формується функціями:

ones(A);

ones(4,5);

1.4.4 Операції і вирази

Операції використовуються у формуванні виразів, які стоять у правій частині оператора присвоювання. Над скалярними змінними виконуються такі самі операції, як і у мові Сі, тобто, додавання, віднімання, множення, логічні операції і т.і. Операції порівняння наступні:

- a==b;
- a~=b або a<>b;
- a<b;
- a<=b;
- a>b;
- a>=b.

Над векторами і матрицями виконуються наступні операції:

- + , – — додавання і віднімання;
- ‘ транспонування;
- * матричне множення та множення на число;
- ^ піднесення до ступеня;
- \ , / ліве та праве ділення;

• **.*** поелементне множення матриць;

• **.^** поелементне піднесення до ступеня;

• **./** поелементне ділення.

Як і операції, у виразах можуть брати участь математичні функції, які повертають один результат. Якщо це скалярна функція, яка застосовується до вектора чи матриці, то насправді вона виконується з кожним елементом окремо.

1.4.5 Керуючі оператори

Основним оператором розгалуження керування є оператор if. Його проста форма виглядає як

if *умова*

блок операторів1

else

блок операторів2

end

Тут *умова* — логічний вираз, *блок операторів* — один чи кілька операторів або викликів функцій, які обмежені ключовими словами if, else, end. Складна форма цього оператора перевіряє кілька умов:

if *умова1*

блок операторів1

elseif *умова2*

блок операторів2

else

блок операторів

end

Селективний оператор виконує кілька розгалужень в залежності від значення параметра *параметр*:

select *параметр*

case *значення1* **then** *блок операторів1*

case *значення2* **then** *блок операторів2*

…

else *оператори*

end

В мові Scilab є два види операторів цикла: оператор **while** з передумовою і оператор **for**.

Перший з них має вид:

while *умова*

блок операторів

end

Тут *умова* — логічний вираз; оператори в блоці операторів будуть виконуватись циклічно поки логічна умова істинна.

Оператор **for** виконує цикл задану кількість разів:

for x=xn:hx:xk

блок операторів

end

Тут x — змінна параметра циклу, xn — початкове значення x, xk — кінцеве значення x, hx — крок циклу. Якщо крок циклу дорівнює 1, то hx можна не застосовувать.

Оператор **break** перериває виконання циклу.

Оператор **continue** виконує переривання ітерації циклу і перехід до наступної ітерації.

1.4.5 Функції

Функції, які використовуються в Scilab, можна розділити на два класи:

• вбудовані;

• визначені користувачем.

Багато вбудованих функцій співпадають з функціями мови Сі. Це, наприклад тригонометричні функції, логарифм, експонента, квадратний корінь. Але є функції, які специфічні для мови Scilab, такі як нижче.

• **halt**('Press a key') — зупинка виконання скрипта поки не буде натиснута кнопка.

• **diag**(vm, [k]) — повертає матрицю, яка вміщує діагональ матриці vm, причому при k = 0 — це центральна діагональ, а при k>0, k<0 — діагональ, яка на |k| позицій вища чи нижча за центральну.

• **eye**(m,n) — повертає матрицю з одиничною діагоналлю розміром m*n.

• **ones**(m,n) — повертає матрицю з усіма одиницями розміром m*n.

• **zeros**(m,n) — повертає матрицю з усіма нулями розміром m*n.

• **rand**() — повертає одне випадкове значення у діапазоні (0 – 1.0).

• **rand**("normal") — встановлює генератор випадкових чисел у режим генерації з нормальним законом розподілення.

• **rand**("seed",k) — встановлює початковий параметр генератора випадкових чисел у k, за замовчуванням k=0.

• **rand**(m,n) — повертає матрицю з випадковими числами розміром m·n у діапазоні (0 – 1.0).

• [n,m]=**size**(A) — повертає розміри n,m матриці A.

• n = **size**(V) — повертає розміри n вектора V.

• **and**(A) — повертає Логічне І усіх елементів булевої матриці.

• **or**(A) — повертає Логічне Або усіх елементів булевої матриці.

• **cat**(dims,A,B)— конкатенація матриць A і B по рядках при dims=1 чи по стовпцях при dims=2.

• **modulo**(n,m) — остача від ділення n на m.

Основним типом Scilab є число з плаваючою комою. Щоб визначити напевно цілі числа, об’єкт (число, матриця) треба перетворити функціями

• **round**(X) — округлення до об’єкта з цілих,

• **floor**(X) — округлення до меншого об’єкта з цілих,

• **int32**(X) — у об’єкт з 32-розрядними цілими,

• **int8**(X) — з байтовими цілими,

• **iconvert**(X, it) — перетворення у байт без знаку при it = 11 чи слово без знаку при it = 14.

Про синтаксис і семантику інших функцій можна дізнатись у довідці Scilab.

Scilab має декілька вбудованих пакетів для вирішення математичних задач у окремих областях математики і комп’ютерної науки. У кожному з пакетів зібрані колекції специфічних функцій. У пакеті Graphics зібрані функції, які призначені, в основному, для виводу графіків різних функціональних залежностей у графічному вікні. Також в ньому є функції, які корисні для виводу графічних примітивів, які є корисними для відображення графів у цьому вікні, такі як наступні.

• **plot2d**([0,200],[0,100],[-1,-1],"022") — формує прямокутне поле у графічному вікні розміром 100x200 одиниць.

• **xstring**(x,y,”string”) — вивід рядка string починаючи з позиції з координатами (x,y).

• **xnumb**(x,y,data) — вивід числа data починаючи з позиції з координатами (x,y).

• **xarrows**([x1;x2], [y1;y2], w, c) — стрілка з координатами початку (x1;y1), кінця (x2;y2), розмірами вістря w і кольором c (c=1 –чорний, 2– синій і т.д.).

• **xsegs**([x1;y1], [x2;y2], c) — пряма з координатами початку [x1;y1], кінця [x2;y2] і кольором c.

• **xarc**(x,y,w,h,a1,a2) — малювання дуги еліпса, який вписаний у прямокутник з координатами (x,y) верхнього лівого кута і шириною w та висотою h, яка обмежена променями з кутами a1 і a2, які задаються у 64-х частках градуса. Наприклад, коло діаметром 4 одиниць з центром у точці (10,20) задається функцією xarc(8,22,4,4,0,360*64);

• **clf** — очистити графічне вікно.

Наприклад, малювання простого графа кодується наступним скриптом.

```
clf;
plot2d([0;100],[0;100],0);
xarc(8,22,4,4,0,360*64);
xnumb(8,18,1);
xarc(48,22,4,4,0,360*64);
xnumb(48,18,2);
xarrows([12;48],[20;20],  50, 2);
```

В результаті виконання цього скрипта одержується зображення, як на рис.3.

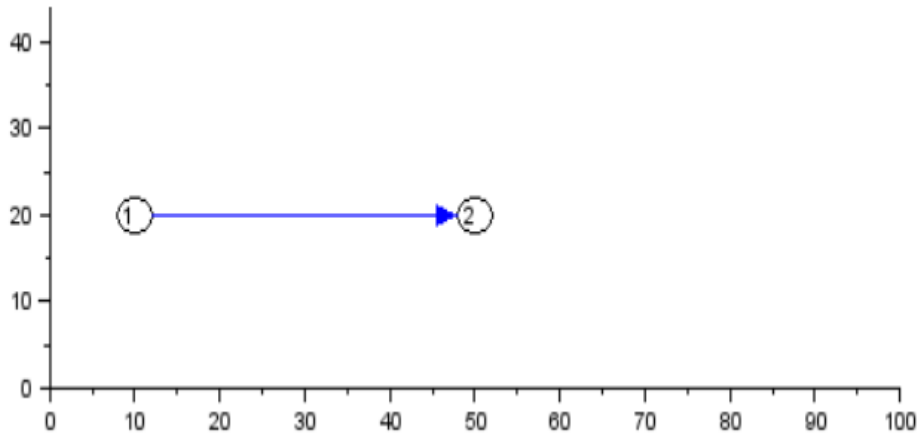

Рис.3 — Зображення графа, побудоване функціями з пакету Graphics

Функції, які визначені користувачем перед своїм використанням об’являються так, як у наступному прикладі.

```
function  [x,y] = adj(a,b)
    x = a - 2;
    y = b + 2;
endfunction;
function  node(x,y,n)
    [c,d] = adj(x,y);
    xarc(c,d,4,4,0,360*64);
    xnumb(c,d-4,n);
endfunction;
```

Тут функція **adj** повертає два результати, а функція **node** — жодного. Використання останньої функції зручне для виводу у графічному вікні зображень вершин. Виклик цієї функції

```
node(30,40,13)
```

ставить зображення вершини графа у точку з координатами (30,40) і номером 13.

Декларація функції ставиться на початку пакету перед її викликами. Якщо функцій багато, доцільно їх зібрати в один скрипт, який перед використанням функцій викликається командою **exec**.

1.5 Модуль Metanet

Примітка. *Інформація про модуль Metanet надається для порівняння, які бувають інструменти. Застосовувати його не обов'язково: пакет Graphics має достатньо можливостей для виконання лабораторних робіт, і навпаки — в Metanet є не все необхідне.*

Metanet — це набір інструментів Scilab для обчислень з графами і мережами. Він поставляється як пакет користувача (Atom) Scilab разом з графічним вікном для відображення та модифікації графів.

При представленні графа вважається, що дуга (arc) (*i*, *j*) іде від початкової вершини (tail) *i* до кінцевої вершини (head) *j*. Ненаправлена дуга, як правило, називається ребром (edge). Граф можна задати кількістю вершин, списком

початкових вершин і списком кінцевих вершин, які є понумерованими. У Scilab ці списки представлені векторами рядків, які мають імена tail і head, відповідно. Отже, якщо розглядається *i*-та дуга, то вона інцидентна вершинам з номерами tail(*i*) та head(*i*). Це стандартне представлення графів в Metanet.

1.5.1 Спискова структура даних графу

Metanet використовує спискову структуру даних графу для представлення графів. Список графу містить 33 елементи (не рахуючи першого елементу — визначення типу — 'graph'). Тільки перші п'ять елементів повинні обов'язково мати значення в списку, всі інші можуть бути порожніми векторами [] і для них використовується значення за замовчуванням. Ці п'ять необхідних елементів є:

• **name** — ім'я назви графу (рядок),

• **directed** — дорівнює 1, якщо граф направлений або дорівнює 0, якщо граф — ненаправлений,

• **node number** — число вершин,

• **tail** — вектор номерів вершин – початків дуг,

• **head** — вектор номерів вершин – кінців дуг.

Вектори **tail, head** не повинні бути пустими.

Доступ до кожного елементу списку можна отримати, використовуючи його назву. Наприклад, якщо g - список графу, і треба отримати елемент числа вузлів, то потрібно лише ввести:

g ('node_number')

і якщо треба змінити це значення на 10, то потрібно лише ввести:

g ('node_number') = 10

Також до елементів структури є доступ з синтаксисом мови Сі — через крапку. Наприклад, оператор

g.nodes.graphics.defaults.diam=22;

присвоює нове значення діаметра вершини за замовчуванням.

Функція **check graph** перевірки графа перевіряє список на відсутність невідповідностей в його елементах.

Наступні імена також використовуються для доступу до елементів списку:

node name – вектор імен вершин, які мають бути різними, за замовчуванням вони дорівнюють номерам вершин.

node_type — вектор типів вершин — цілі числа: 0 – звичайна вершина, зображається як коло, 1 — вершина-стік, коло з напрямленою в нього стрілкою 2 — вершина-джерело, коло з витікаючою стрілкою.

node_x — вектор просторових координат X усіх вершин.

node_y — вектор просторових координат X усіх вершин.

node_color — вектор кольорів усіх вершин від 0 (чорний) до 16 (білий), за замовчуванням — чорний.

node_diam — вектор діаметрів усіх вершин у відносних одиницях, за замовчуванням дорівнює елементу структури **default_node_diam** = 20.

node_border — вектор товщини кола вершин у пікселах, за замовчуванням дорівнює елементу структури default_node_border.

node_font_size — вектор висоти літер, яка буває 8, 10, 12, 14, 18 чи 24, за замовчуванням дорівнює елементу структури **default_font_size** = 12.

node_demand — вектор вагів вершин, за замовчування вага дорівнює 0.

edge_name — вектор імен дуг, які мають бути різними, за замовчуванням дорівнюють номеру дуги.

edge_color — вектор кольорів дуг, за замовчуванням – чорні, тобто, 0.

edge_width — вектор ширини дуг у пік селах, за замовчуванням дорівнює елементу структури **default_edge_width** = 1.

edge_hi_width — вектори ширини вибраних (higlighted) дуг в пікселах, за замовчуванням дорівнює елементу структури **default_edge_hi_width** = 2.

edge_font_size — таке саме, **node_font_size**, але для дуг.

edge_length — вектор довжини дуг для задач типу „мінімальний маршрут”, за замовчування вага дорівнює 0.

edge_cost — вектор вартості дуг для задач типу „максимальний потік”, за замовчування вартість дорівнює 0.

edge_min_cap — вектор мінімальної ємності дуг для задач типу „максимальний потік”, за замовчування вартість дорівнює 0.

edge_max_cap — аналогічно **edge_min_cap.**

edge_q_weight — вектор квадратичних ваг дуг для задач типу „максимальний потік”, за замовчування вартість дорівнює 0.

edge_q_orig — аналогічно **edge_q_weight.**

edge_weight — вектор ваги дуг для задач типу „мінімальне дерево”, за замовчування вартість дорівнює 0.

node_label — вектор довільних міток на вершинах.

edge_label — вектор довільних міток на дугах.

У пакеті Metanet зібрані функції, які допомагають задавати і графічно зображати графи, а також вирішувати поширені задачі з графами. Кілька перших функцій є наступними.

g = **make_graph**(‘name’,directed,n,tail,head) **—** створює граф у виді спискової структури даних g, яка має ім’я ‘name’, ознаку направленості directed (0– ненаправлений, 1– направлений) , кількість вершин n, вектори вершин tail і head. Наприклад,

g=**make_graph**('graph1',1,0,[1 1 4 3],[4 3 1 3]);

Для зображення цього графа у графічному вікні треба вершинам дати конкретні координати:

g.nodes.graphics.x=[40,33,75,42];

g.nodes.graphics.y=[7,61,43,120];

show_graph(g) — показує граф g у графічному вікні Scilab так, як на рис. 4.

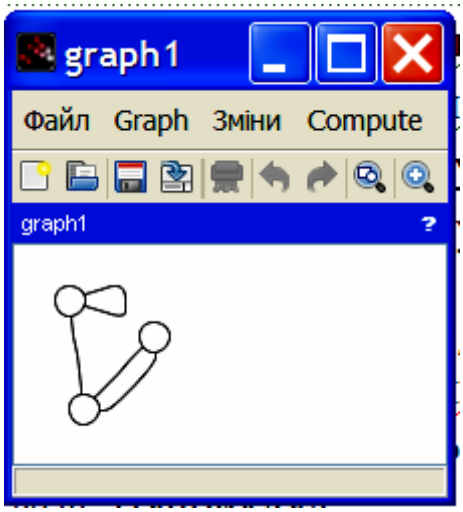

Рис. 4 — Граф, введений функцією make_graph

plot_graph(g) — показує граф g у графічному вікні, яке створюється пакетом Metanet. У цьому вікні можна редагувати граф.

as=**graph_2_mat**(g); — перетворення опису графу у розріджену матрицю. Граф на рис. 4 роздрукується як матриця:

```
 as  =
(    4,    4) sparse matrix
(    1,    1)       1.
(    1,    2)       1.
(    1,    3)     - 1.
(    3,    2)     - 1.
(    4,    1)     - 1.
(    4,    3)       1.
```

g1=**mat_2_graph**(as, directed); — перетворення опису графу з розрідженої матриці as у опис списком. При цьому матриця повинна складатись з цілих чисел. Щоб утворити розріджену матрицю xs з повної матриці x, використовується функція:

xs = **sparse**(x);

Тут матриця x може бути як матриця суміжності (node-node) або матриця інцидентності (node-arc). Тоді останнім параметром функції **mat_2_graph** ставлять ключове слово 'node-node' або 'node-arc', відповідно.

Протилежною до функції **sparse** є функція **full:**

```
-->full(as)
 ans  =
    1.    1.  - 1.    0.
    0.  - 1.    0.    0.
  - 1.    0.    1.    0.
    0.    0.    0.    0.
```

g = **gen_net**('name',directed,v) — генерування випадкового графу з вектором параметрів v, який має рівно 12 елементів – цілих чисел:

— початковий параметр генератора випадкових чисел,

— число вузлів,

— число джерел,

— число витоків,

— мінімальна ціна,

— максимальна ціна,

— потужність входу (Input supply),

— потужність виходу (Output supply),

— мінімальна ємність,

— максимальна ємність,

— процент дуг зважених вартістю (0 — 100),

— процент дуг зважених ємністю (0 — 100).

2 Лабораторні роботи

2.1 Лабораторна робота 1. Графічне представлення графів

Мета лабораторної роботи

Метою лабораторної роботи №1. «Графічне представлення графів» є набуття практичних навичок представлення графів у комп'ютері та користування системою Scilab.

Постановка задачі

1. Представити у програмі напрямлений і ненапрямлений графи із заданими параметрами:

— число вершин n;

— розміщення вершин;

— матриця суміжності A.

Параметри задаються на основі номера залікової книжки студента — десяткового числа $n_1\ n_2\ n_3\ n_4$

Число вершин n дорівнює $10 + n_3$.

Розміщення вершин:

— колом при $n_4 = 0,1$;

— прямокутником (квадратом) при $n_4 = 2,3$;

— трикутником при $n_4 = 4,5$;

— колом з вершиною в центрі при $n_4 = 6,7$;

— прямокутником (квадратом) з вершиною в центрі при $n_4 = 8,9$.

Наприклад, при $n = 10$ розміщення вершин прямокутником з вершиною в центрі повинно виглядати так, як на прикладі графа рис.5.

Матриця A напрямленого графа за варіантом формується за командами Scilab:

rand("seed", $n_1n_2n_3n_4$);

T = **rand**(n,n) + **rand**(n,n);

A = **floor**((1.0 - n_3*0.02 - n_4*0.005 - 0.25)*T)

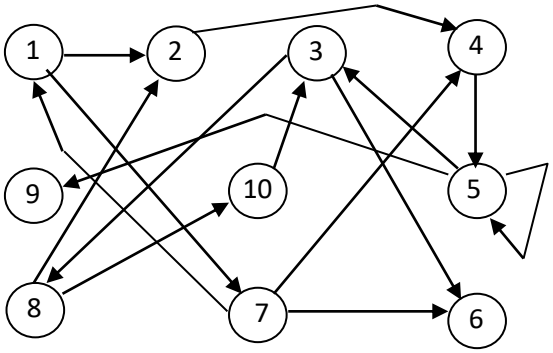

Рис.5 — Приклад графа

2. Створити скрипт для формування зображення напрямленого і ненапрямленого графів у графічному вікні Scilab.

***Зверніть увагу**, що пакет Metanet застосовувати не обов'язково: пакет Graphics має достатньо можливостей для виконання лабораторних робіт, і навпаки — в Metanet є не все необхідне (немає номерів вершин).*

Зміст звіту

1. Загальна постановка задачі та завдання для конкретного варіанту.

2. Текст програми-скрипту для Scilab.

3. Згенеровані матриці суміжності напямленого і ненапрямленого графів.

4. Скриншоти напрямленого і ненапрямленого графів, які побудовані за варіантом.

2.2 Лабораторна робота 2. Характеристики графів

Мета лабораторної роботи

Метою лабораторної роботи №2. «Характеристики графів» є дослідити характеристики графів та навчитись визначати їх на конкретних прикладах.

Постановка задачі

1. Представити напрямлений граф із заданими параметрами так само, як у лабораторній роботі №1.

Відміна: матриця А напрямленого графа за варіантом формується за командами Scilab:

rand("seed", $n_1n_2n_3n_4$);

T = **rand**(n,n) + **rand**(n,n);

A = **floor**((1.0 – n_3*0.01 – n_4*0.01 - 0.3)*T)

Перетворити граф у ненапрямлений.

2. Визначити степені вершин напрямленого і ненапрямленого графів. Програма на екран виводить степені усіх вершин ненапрямленого графу і напівстепені виходу та заходу напрямленого графу. Визначити, чи граф є однорідним та якщо так, то вказати степінь однорідності графу.

3. Визначити всі висячі та ізольовані вершини. Програма на екран виводить перелік усіх висячих та ізольованих вершин графу.

Зміст звіту

1. Загальна постановка задачі та завдання для конкретного варіанту.
2. Текст програми-скрипту для Scilab.
3. Згенеровані матриці суміжності
4. Таблиці степенів, напівстепенів вузлів.
5. Перелік висячих та ізольованих вершин.
6. Скриншоти заданих орієнтованого і неорієнтованого графів.
7. Висновки.

2.3 Лабораторна робота 3. Зв’язність графа

Мета лабораторної роботи

Метою лабораторної роботи №3. «Зв’язність графа» є вивчення методу дослідження графа за допомогою операції транзитивного замикання.

Постановка задачі

1. Представити напрямлений граф із заданими параметрами так само, як у лабораторній роботі №1.

Відміна: матриця А напрямленого графа за варіантом формується за командами Scilab:

rand("seed", $n_1n_2n_3n_4$);

T = **rand**(n,n) + **rand**(n,n);

A = **floor**((1.0 - n_3*0.005 - n_4*0.005 - 0.27)*T)

2. Створити скрипт для Scilab для обчислення наступних результатів:

1) матриця суміжності;
2) півстепені вузлів;
3) всі шляхи довжини 2 і 3;
4) матриця досяжності;
5) компоненти сильної зв’язності;
6) матриця зв’язності;
7) граф конденсації.

Шляхи довжиною 2 і 3 слід шукати за матрицями A^2 і A^3, відповідно. Для спрощення пошуку маршрутів, перед обчисленням цих матриць у матриці А слід заповнити нулями головну діагональ. Матрицю досяжності та компоненти сильної зв’язності слід шукати за допомогою операції транзитивного замикання.

Зміст звіту

1. Загальна постановка задачі та завдання для конкретного варіанту.

2. Текст програми-скрипту для Scilab.

3. Згенеровані матриці суміжності, півстепенів вузлів, шляхів довжини 2 і 3 (навести матриці A^2 і A^3, а також перелік знайдених маршрутів), досяжності, зв’язності, суміжності графа конденсації.

4. Скриншоти заданого графа і графа конденсації.

2.4 Лабораторна робота 4. Обхід графа

Мета лабораторної роботи

Метою лабораторної роботи №4. «Обхід графа» є вивчення методу дослідження графа за допомогою обходу його вершин в глибину або в ширину.

Постановка задачі

1. Представити напрямлений граф із заданими параметрами так само, як у лабораторній роботі №1. Відміна: матриця A за варіантом формується за командами Scilab:

rand("seed", $n_1n_2n_3n_4$);

T = **rand**(n,n) + **rand**(n,n);

A = **floor**((1.0 - n_3*0.01 - n_4*0.005 - 0.15)*T)

2. Створити скрипт для Scilab для обходу в глибину при n_4 — парному і для обходу в ширину — при непарному. Обхід починати з вершини, яка має вихідні дуги. При цьому у скрипті:

— встановити функцію **halt** у точці призначення номеру черговій вершині,

— виводити зображення графа у графічному вікні перед кожною зупинкою по функції **halt**.

3. Під час обходу графа побудувати дерево обходу. Вивести побудоване дерево у графічному вікні.

Зміст звіту

1. Загальна постановка задачі та завдання для конкретного варіанту.

2. Текст програми-скрипту для Scilab.

3. Згенерована матриця суміжності.

4. Матриця дерева обходу і матриця відповідності вершин і одержаної нумерації.

5. Скриншоти зображення графа з одержаною нумерацією та дерева обходу.

2.5 Лабораторна робота 5. Мінімальний кістяк графа

Мета лабораторної роботи

Метою лабораторної роботи №5. «Мінімальний кістяк графа» є вивчення методів розв'язання задачі знаходження мінімального кістяка графа.

Постановка задачі

1. Представити зважений ненапрямлений граф із заданими параметрами так само, як у лабораторній роботі №1. Відміна: матриця A за варіантом формується за командами Scilab:

rand("seed", $n_1n_2n_3n_4$);

T = **rand**(n,n) + **rand**(n,n);

A = **floor**((1.0 - n_3*0.01 - n_4*0.005 - 0.05)*T)

Матриця ваг W формується за наступними командами:

Wt = **round**(**rand**(n,n)*100 .* A);

B = Wt & **ones**(n,n);

Wt = (**bool2s**(B & ~B') + **bool2s**(B & B') .* **tril**(**ones**(n,n),-1)) .* Wt;

W = Wt + Wt';

2. Створити скрипт для Scilab для знаходження мінімального кістяка за алгоритмом Краскала при n_4 — парному і за алгоритмом Пріма — при непарному. При цьому у скрипті:

— встановити функцію **halt** у точці додавання чергового ребра до кістяка,

— виводити зображення графа у графічному вікні перед кожною зупинкою по функції **halt**.

3. Під час обходу графа побудувати дерево його кістяка. Вивести побудоване дерево у графічному вікні. При зображенні як графа, так і його кістяка, вказати ваги ребер.

Зміст звіту

1. Загальна постановка задачі та завдання для конкретного варіанту.

2. Текст програми-скрипту для Scilab.

3. Згенерована матриця суміжності та матриця суміжності ненапрямленого графу.

4. Матриця знайденого кістяка графу.

5. Скриншоти зображення графа зі вказаними вагами ребер та знайденого кістяка.

2.6 Лабораторна робота 6. Пошук найкоротшого шляху

Мета лабораторної роботи

Метою лабораторної роботи №6. «Пошук найкоротшого шляху» є вивчення методу дослідження графа за допомогою алгоритму Дейкстри.

Постановка задачі

1. Представити зважений ненапрямлений граф із заданими параметрами так само, як у лабораторній роботі №5.

2. Створити скрипт для Scilab для пошуку найкоротшого шляху до кожної вершини при n_4 — парному і найдовшого — при непарному. Пошук виконувати з першої вершини. Якщо вона ізольована, то з другої і так далі. При цьому в скрипті:

— встановити функцію **halt** у точці призначення чергової активної вершини,

— виводити зображення графа у графічному вікні перед кожною зупинкою по функції **halt**, виділяючи окремими кольорами тимчасові й постійні вершини.

3. Під час обходу графа побудувати до кожної вершини графа найкоротший (найдовший) шлях від стартової вершини.

Зміст звіту

1. Загальна постановка задачі та завдання для конкретного варіанту.

2. Текст програми-скрипту для Scilab.

3. Згенерована матриця суміжності та матриця суміжності ненапрямленого графу.

4. Перелік знайдених шляхів та їх довжин.

5. Скриншоти зображення графа зі вказаними вагами ребер.