Практичне заняття № 18

Тема. Програмування алгоритмів чисельних процесів.

Мета роботи: освоїти простіши методи програмування, навчитись користуватися операторами розгалуження та операторами циклів.

Вправа 1. Знайомство з процедурою використовування циклів для побудови графіків функцій та обчислення сум ряду.

Мова програмування в системі **MATLAB** достатньо проста і складається з основного набору конструкцій: операторів розгалуження та циклів [44]. Простота мови програмування забезпечена великою кількістю вбудованих функцій, які дозволяють вирішувати задачі з різних предметних галузей.

Цикл **for** використовується для повторення дії оператора у разі, коли число повторень наперед відоме. У циклі **for** використовується лічильник циклу, його початкове значення, крок та кінцеве значення вказують через двокрапку (**:**). Блок операторів, що розміщується у середині циклу, повинен закінчуватися словом-оператором **end,** яке вказує на закінчення циклу.

Приклад 1. Побудувати графік функції $f(x,\alpha)=e^{\alpha x^2}\cos x^2$ на відрізку [–1,1] для значень параметра $\alpha\in[-0.5,0.5]$.

Побудова графіку буде виконуватись за допомогою наступних команд та операцій:

```
>> x=[–1:0.01:1];for alfa=–0.5:0.1:0.5
y=exp(alfa*x.^2).*cos(x.^2);
plot(x,y);hold on
end
hold off
```

Зауважимо, що у випадку, коли крок (лічильник циклу) дорівнює одиниці, то його вказувати не обов'язково.

Приклад 2. Обчислити суму $\sum_{n=1}^{10}\frac{x^n}{n!}$.

При різних значеннях x буде потрібен **m**–файл, текст якої приведений далі. Зверніть увагу, що **m**–файл **sum1** може бути викликаний як від числа, так і від масиву значень, завдяки застосуванню поелементних операцій.

```
function s=sum1(x)
x=[–2:0.01:2];
s=0; for k=1:10
s=s+x.^k/factorial(k);
end
```

Для виконання операції знаходження суми достатньо ввести у командний рядок им'я **m**–файлу.

Вправа 2. Навчитись виконувати операції з невідомим числом певних дій.

Цикл **for** підходить для повторення заданого числа певних дій [33]. У тому випадку, коли число повторів наперед невідоме і визначається в ході виконання блоку операторів слід організувати цикл **while**. Цикл **while** працює, поки виконана умова циклу. Приклад 1. Знайти суму усіх перших негативних елементів вектора x =[–1 –2 –3 6 7].

Поперше формуємо **m**–файл з циклом. **M**–файл **negsum** формує суму всіх перших негативних компонентів вектора:

```
function s=negsum(x)
s=0;
n=1;
while x(n) <0
s=s+x(n);
n=n+1;
end
```

Після формування **m**–файлу треба ввести у командний рядок вектор, суму негативних елементів якого необхідно знайти:

```
>> x=[–1 –2 –3 6 7 ]
x =
   –1   –2   –3    6    7
>> negsum(x)
ans =
   –6
```

У якості операторів відношення використовуються наступні символи: >, <, >=, <= == (рівно), ~= (не рівно).

M–файл **negsum** має один недолік: якщо всі елементи масиву – негативні числа, то **n** стає більше довжини масиву **x**, що приводить до помилки, наприклад:

```
>> b=[–2 –7 –1 –9 –2 –5 –4];
s=negsum(b)
??? Index exceeds matrix dimensions.
```

Окрім перевірки значення **x(n)** варто подбати про те, щоб значення **n** не перевершувало довжини вектора **x**. Вхід в цикл повинен здійснюватися тільки при одночасному виконанні умов **n<=length(x)** і **x(n) <0**, тобто необхідно застосувати логічного оператора "**i**", що позначається в системі **MATLAB** символом **&.**

Треба змінити умову циклу на складну: **k<=length(x)& x(k) <0**.

Якщо перше з умов не виконується, то друга умова перевірятися не буде, саме тому вибраний такий порядок операндів.

Як результат маємо, що **m**–файл **negsum** працює вірно для будь–яких вихідних векторів.

Логічний оператор "або" позначається символом вертикальної межі “|”, а заперечення – за допомогою символа “~” (тильди). Нижче приведені логічні операції спадання їх пріоритета:

- оператор заперечення “~” ;
- оператори “>”, “<”, “>=”, “<=”, “==”, “~=” (відношення);
- оператор “&” (логічне "і");
- оператор “|” (логічне "або").

Для зміни порядку виконання логічних операторів використовуються круглі дужки.

Вправа 3. Знайомство з вкладеними циклами.

Цикли можуть бути вкладені один в одного.

Приклад 1. Знайти суму елементів матриці $A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$, які розташовані вище головної діагоналі.

Для розв’язування цієї задачі слід використати два цикли **for**, причому початкове значення лічильника внутрішнього циклу залежить від поточного значення лічильника зовнішнього циклу [33]. Перелік необхідних команд наведений нижче:

```
>> A=[1 2 3;4 5 6;7 8 9];
>> [n m]=size(A);
s=0;
for i=1:n
for j=i+1:m s=s+A(i,j);
end
end
>> s
s =
   11
```

Вправа 4. Застосування розгалужених циклів та використання перемикачів у ході роботи програми.

Розгалуження в ході роботи програми здійснюється за допомогою конструкції **if–elseif–else**.

Приклад 1. Знайти суму всіх позитивних елементів вектора **x**. Цю задачу можна виконати за допомогою команд:

```
>> x=[–1 2 3 –5 –6 –7 9]; s=0;
>> for k=1:length(x)
if x(k) >0
s=s+x(k);
end
end
s
s =
   14
```

Якщо хід програми повинен змінюватися залежно від декількох умов, то слід використовувати повну конструкцію **if–elseif–else**. Кожна з гілок **elseif** в цьому випадку повинна містити умову виконання блоку операторів, розміщених після неї. Важливо розуміти, що умови перевіряються підряд, перша виконана умова приводить до роботи відповідного блоку, виходу з конструкції **if–elseif–else** і переходу до оператора, наступного за **end**. У останньої гілки **else** не повинно бути ніякої умови. Оператори, що знаходяться між **else** і **end**, працюють в тому випадку, якщо всі умови виявилися невиконаними.

Приклад 2. Написати **m**–файл для обчислення кусково–заданої функції:

$$f(x)=\begin{cases}1-e^{-1-x}, & x<-1\\ x^2-x-2, & -1\le x\le 2\\ 2-x, & x>2\end{cases}$$

Перша умова $x<-1$ перевіряється в гілці **if**. Умову $-1\le x$ не потрібно включати в наступну гілку **elseif**, оскільки в цю гілку програма заходить, якщо попередня умова **(**$x<-1$**)** опинилася не виконаною. Умову $x>2$ перевіряти не треба – якщо не виконано дві попередні умови, то змінна x набуває величини більше ніж два. Перелік команд для виконання задачі наведений нижче:

```
function f=pf(x)
if x<–1
f=1–exp(–1–x);
elseif x<=2
f=x^2–x–2;
else f=2–x;
end
```

Хід роботи програми може визначатися завданням значення деякої змінної (перемикача). Такий спосіб розгалуження програми заснований на використанні команди перемикання **switch**. Змінна–перемикач поміщається після команди **switch** через пропуск. Оператор **switch** містить блоки, що починаються із слова **case**, після кожного **case** записується через пропуск те значення перемикача, при якому виконується даний блок. Останній блок

починається із слова **otherwise**, його оператори працюють у тому випадку, коли жоден з блоків **case** не був виконаний. Якщо хоч би один з блоків **case** виконаний, то відбувається вихід з тіла команди **switch** і перехід до команди, наступної за **end**.

Приклад 3. Знайти кількість одиниць і мінус одиниць в масиві **x** і знайти суму всіх елементів, відмінних від одиниці і мінус одиниці.

Слід перебрати всі елементи масиву в циклі, причому в ролі змінної–перемикача буде поточний елемент масиву. Розглянемо приклад, у якому по заданому масиву повертаєтья число мінус одиниць в першому вихідному аргументі, число одиниць – в другому, а суму всіх елементів – в третьому. Для утворення етапів розв′язання поставленої задачі створюємо **m**–файл:

```
function [m,p,s]=mpsum(x)
m=0;
p=0;
s=0;
for i=1:length(x)
switch x(i);
case –1, m=m+1;
case 1, p=p+1;
otherwise s=s+x(i);
end
end
```

Визначаємо вектор **x** та виконуємо поставлену задачу:

```
>> x=[–1 –1 2 0 1 2 ]; [m,p,s]=mpsum(x)
m =
    2
p =
    1
s =
    4
```

Зауважимо, що блок **case** може бути виконаний не тільки при одному певному значенні перемикача, але і тоді, коли перемикач приймає одне з декількох допустимих значень. В цьому випадку значення вказуються після слова **case** у фігурних дужках через кому, наприклад: **case {1,2,3}**.

Дострокове завершення циклу команд **while** або **for** здійснюється за допомогою оператора **break**.

Приклад 4. По заданому масиву **x=**[–1 –1 2 0 1 2**]** утворити новий масив **y(n)=x(n+1)/x(n)** до першого нульового елементу **x(n)**, тобто до тих пір, поки має сенс виконання операції ділення. Номер першого нульового елементу в масиві **x** наперед невідомий (в масиві **x** може і не бути нулів).

Розв′язок задачі полягає в послідовному обчисленні елементів масиву **y** і припиненні обчислень при виявленні нульового елементу в **x**. Послідовні команди, які виконують задачу, записані у наступному **m**–файлі:

```
function y=dv(x)
for n=1:length(x)–1
if x(n)==0
break
end
y(n)=x(n+1)/x(n);
end
```

Задамо масив **x** та обчислимо елементи нового масиву **y**:

```
>> x=[–1 –1 2 0 1 2 ];
>> y=dv(x)
y =
    1    –2     0
```

Практичні завдання лабораторної роботи № 18.

Виконати наступні завдання :

Завдання 1. Написати **m**–файл для обчислення кусково–заданої функції (див. табл. 18.1).

Таблиця 18.1 – Варіанти до завдання №1

№ п/п	Функції
1	$f(x)=\begin{cases} -1, & -3\le x\le -1 \\ x, & -1< x\le 1 \\ e^{1-x}, & 1< x\le 3 \end{cases}$
2	$f(x)=\begin{cases} \sqrt{x}, & 0\le x\le 1 \\ 1, & 1< x\le 3 \\ (x-4)^2, & 3< x\le 5 \end{cases}$
3	$f(x)=\begin{cases} \ln x, & 1\le x\le e \\ x/e, & e< x\le 9 \\ 9e^{8-x}, & 9< x\le 12 \end{cases}$
4	$f(x)=\begin{cases} \sin x, & -2\pi\le x\le 0 \\ -x^3, & 0< x\le 1 \\ \cos \pi x, & 1< x\le 3\pi \end{cases}$
5	$f(x)=\begin{cases} \arcsin x-1, & 0\le x\le 1 \\ \frac{\pi}{2}-x, & 1< x\le \frac{\pi}{2} \\ \cos x, & \frac{\pi}{2}< x\le \pi \end{cases}$

Закінчення таблиці 18.1

6	$f(x)=\begin{cases} \lvert x\rvert, & -2\le x\le 1 \\ \sin\frac{\pi}{2}x, & 1<x\le 2 \\ (2-x)^3, & 2<x\le 3 \end{cases}$
7	$f(x)=\begin{cases} (x-1)^2, & -2\le x\le 1 \\ \cos\frac{\pi}{2}x, & 1<x\le 3 \\ 1-e^{3-x}, & 3<x\le 8 \end{cases}$
8	$f(x)=\begin{cases} e^x, & -2\le x\le -1 \\ \frac{\lvert x\rvert}{e}, & -1<x\le 1 \\ e^{-x}, & 1<x\le 2 \end{cases}$
9	$f(x)=\begin{cases} e^{x+1}, & -2\le x\le -1 \\ x^2, & -1<x\le 1 \\ (2-x)^3, & 1<x\le 2 \end{cases}$
10	$f(x)=\begin{cases} x^2\log_2 x, & 1\le x\le 2 \\ x^3/2, & 2<x\le 3 \\ x^x/2, & 3<x\le 3.5 \end{cases}$
11	$y(x)=\begin{cases} \sin x & -4\pi\le x\le -\pi \\ 3(x/\pi+1)^2 & -\pi<x\le 0 \\ 3e^{-x} & 0<x\le 5 \end{cases}$
12	$y(x)=\begin{cases} x^2, & -2\le x\le 1 \\ 2-x, & 1\le x\le 3 \\ x-4, & 3\le x\le 5 \end{cases}$
13	$f(x)=\begin{cases} \cos x, & -3\pi\le x\le 0 \\ x^2, & 0\le x\le 2 \\ \sin\pi x, & 2\le x\le 3\pi \end{cases}$
14	$f(x)=\begin{cases} \lvert x\rvert, & -3\le x\le 0 \\ \cos\frac{\pi}{2}x, & 0\le x\le 2 \\ (1-x)^2, & 2\le x\le 4 \end{cases}$

Завдання 2. Забезпечити виконання умов задачі (див. табл. 18.2):

Таблиця 18.2 – Варіанти до завдання № 2

№ п/п	Задачі
1	Визначити кількість від'ємних компонентів вектора $x=[1\ \ 7\ -1\ \ 3\ \ -2\ \ 5\ -1\ \ 0\ \ 3\ ]$, розташованих між його максимальним і мінімальним елементами.
2	Підрахувати число нулів і одиниць в заданій матриці $A=\begin{bmatrix}1 & 0 & 0 & 0 & 1\\ 1 & 1 & 1 & 0 & 1\\ 0 & 0 & 1 & 0 & 1\\ 1 & 1 & 1 & 0 & 0\end{bmatrix}$.
3	Визначити кількість додатних компонентів вектора $x=[1\ \ 7\ -1\ \ 3\ \ 2\ \ 5\ -1\ \ 0\ \ 3\ ]$, розташованих між його максимальним і мінімальним елементами.
4	Обчислити подвійну суму: $\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{a^{i+j}}{(i+j)^2}$, де: $A=\|a_{ij}\|=\begin{bmatrix}0 & -1 & 3 & -1 & 1\\ 0 & -1 & 1 & 2 & 5\\ 1 & -3 & 6 & -1 & 2\\ 2 & -1 & 1 & -4 & -1\end{bmatrix}$, де $n\times m$ – розмір матриці A.
5	Підсумувати від'ємні елементи матриці $A=\begin{bmatrix}0 & -1 & 1 & -1 & 2\\ 3 & -1 & 1 & 5 & 1\\ 1 & -1 & 2 & -1 & 3\\ 2 & -1 & 1 & -4 & -1\end{bmatrix}$, які лежать нижче за головну діагональ.
6	Замінити додатні компоненти вектора $x=[1\ \ 7\ -1\ \ 3\ \ 5\ -1\ \ 3\ ]$ сумою всіх його від'ємних компонентів.
7	Підсумувати позитивні елементи матриці $A=\begin{bmatrix}0 & -1 & 3 & -1 & 2\\ 3 & -1 & 1 & 5 & 4\\ 1 & 1 & 2 & -1 & 3\\ 2 & -1 & 7 & -4 & -1\end{bmatrix}$, які лежать вище головної діагоналі.

Закінчення таблиці 18.2

8	Для матриці $A=\|\|a_{ij}\|\|=\begin{bmatrix}0 & -1 & 3 & -1 & 2\\ 3 & -1 & 1 & 2 & 5\\ 1 & -8 & 1 & -1 & 3\\ 2 & -1 & 1 & -4 & -1\end{bmatrix}$ знайти значення виразу: $w(x)=\sum_{i=1}^{n}\prod_{j=1}^{m}a_{ij}$.
9	Знайти суму $s(x)=\sum_{n=1}^{m}nx^n$, якщо компоненти вектора x набувають значення від -3 до 3 з кроком $h=0.1$ при $m=20$.
10	Обчислити суму $s(x)=\sum_{n=0}^{m}\frac{x^n}{n!}$, якщо компоненти вектора x набувають значення від -5 до 5 з кроком $h=0.1$ при $m=20$.
11	Обчислити суму: $s(x)=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{x^i}{(1+j)^3}$, де $x=[1\ \ 7\ -1\ \ 3\ \ 2\ \ 5\ -1\ \ 0\ \ 3\ ]$.
12	Підсумувати від'ємні елементи матриці A, які лежать вище головної діагоналі: $A=\begin{bmatrix}0 & -1 & 1 & -1 & -2\\ 3 & -1 & 1 & -5 & -1\\ 1 & 1 & 2 & -1 & 3\\ 2 & -1 & 1 & 4 & -1\\ 0 & 2 & 3 & 6 & 3\end{bmatrix}$.
13	Підрахувати число мінус одиниць і одиниць в заданій матриці: $A=\begin{bmatrix}0 & -1 & 1 & -1 & 2\\ 3 & -1 & 1 & 5 & 1\\ 1 & 1 & 2 & -1 & 3\\ 2 & -1 & 1 & 4 & -1\end{bmatrix}$.
14	Для матриці $A=\|\|a_{ij}\|\|=\begin{bmatrix}0 & -1 & 3 & -1 & 2\\ 3 & -1 & 1 & 2 & 5\\ 1 & -8 & 1 & -1 & 3\\ 2 & -1 & 1 & -4 & -1\end{bmatrix}$ знайти значення виразу $w(x)=\sum_{i=1}^{n}\sum_{j=1}^{m}a_{ij}$, де $n\times m$ – розмір матриці A.

Контрольні питання

1. У яких випадках використовується цикл **for-end**?
2. Наведіть приклад обчислення суми за допомогою циклу **for-end.**
3. Як працює цикл **while-end?**
4. Яка конструкція використовується в ході роботи програми для здійснення розгалуження?
5. Для чого використовується команда-перемикач **switch?**
6. Поясніть, як здійснюється дострокове завершення циклу **while** або **for?**