УДК 004.9:535.3
Гаєв Є.О.
Національний авіаційний університет, Київ, Україна

MATLAB-ПРОЄКТ «АНАЛОГОВИЙ ГОДИННИК»

DOI: 10.14308/ite000707

Викладання програмування у старшій школі або на перших курсах університету може допомагати в «оживленні» математики, найбільш складної дисципліни для учнів і студентів. З цією метою ми шукаємо такі задачі на програмування, які були б привабливими для них, достатньо прості та одночасно апелювали б до важливих розділів математики і фізики.

Пропонується проєкт (розробка кількох уроків) із створення MATLAB-програми аналогового годинника на екрані комп'ютера, який має відразу кілька освітньо-дослідницьких цілей. Спершу студенти мають поставлену складну задачу розділити на кілька простіших, програмування яких більш зрозуміле. Розв'язуючи ці останні, вони пов'язують програмування з побудовою графіків, тригонометрією й аналітичною геометрією, проводять фізичні міркування й «експеримент», встановлюють «емпіричні» рівняння і врешті решт все це «замикають» програмуванням і насолоджуються естетичним віджетом аналогового годинника на екрані комп'ютера.

Надаємо програми з поясненнями (коментарями). Одночасно це слугує вступом до MATLAB. Бачимо, що таке «легке програмування» не затуляє змістовної частини роботи. Тому пропонуємо це середовище для якнайширшого використання в освіті, зокрема у викладанні математики і фізики.

Ключові слова: *освітні методи в математиці і фізиці, програмування, MATLAB, метод власних відкриттів студентів.*

Вступ

Сьогоднішні учні бачили багато естетичних віджетів «Годинник» – і без стрілочок (цифрові), і зі стрілочками різних форм (аналогові). Проблема у тому, щоб вони самі створили щось подібне та оволоділи потрібними навичками. Таке завдання є цікавим і змістовним, бо надає можливість пов'язати комп'ютерну дисципліну з іншими, такими як математика і фізика.

В Інтернеті багато пропозицій, як побудувати «годинник із стрілками» на різних мовах програмування [1-3], у тому числі у середовищі MATLAB [4,5]. Їхній головний недолік з позиції викладача: вони пропонують скопіювати, замість пояснити і навчити. Іншим недоліком є те, що обрана мова програмування часто фокусує труднощі на власне програмуванні, а не на фундаментальних задачах, які стоять за проблемою і які програміст має розв'язати. MATLAB, який ми пропагуємо як середовище розробки, більшість суто програмістських аспектів робить дуже легкими (наприклад, заповнення вектору не потребує циклу) і дозволяє сфокусуватися саме на змістовних проблемах.

Така розробка продовжує наші пропозиції щодо «легкого програмування» як основи викладання всіх фізико-математичних і технічних дисциплін в комп'ютерну епоху [6-13]. В свою чергу, більше про MATLAB-програмування можна дізнатися в [15-16,19].

1. Ставимо задачу. Розкладаємо на простіші.

Ми хочемо створити комп'ютерну програму в математично-програмному середовищі MATLAB, яка буде демонструвати годинник з рухомими стрілочками (аналоговий). Дізнатися

локальний час у середовищі MATLAB, подібно до інших мов програмування високого рівня, проблеми не складає. Команда (знак «>>» позначає, що виконуємо її у командному рядочку)

```
>> Date=clock;                                                    (1)
```

робить змінну *Date* вектором з шести елементів, де *Hour=Date*(4), *Minute=Date*(5) і *Second=Date*(6) є, відповідно, локальні година, хвилина і секунда. Завдання полягає у тому, аби показати їх геометрично на зразок того, як це відбувається у годиннику [17]. Щоб це реалізувати, слід розв'язати низку попередніх задач і створити підпрограми, що їх розв'язують. Важливо систематично надавати учням приклади такого аналізу, «розчленування предмету пізнання» [18], типових у сфері інтелектуальної діяльності. У цьому випадку ми бачимо наступні задачі, кроки розробки:

1. Створити циферблат з типовими рисками і цифровими позначеннями.
2. Створити підпрограму *Arm*(k_1, k_2, ..) з кількома параметрами k_1, k_2, які дозволяли б малювати сімейство стрілочок різних форм у якомусь фіксованому положенні. За певних значень параметрів ми могли б отримати годинникову стрілку *HArm*, хвилинну *MArm* та секундну *SArm* у цьому фіксованому положенні.
3. Програма *Arm*, яку ми запропонуємо, матиме справу з кількома точками на площині, які слід з'єднати лінією і заповнити замкнений контур тим чи іншим кольором. Однак, як побудувати стрілку з довільним кутом щодо системи координат? Виникає пропозиція розглянути перетворення координат з рухомої системи, що обертається разом зі стрілками і де останні є нерухомими, до нерухомої системи, жорстко пов'язаної з циферблатом. Це відома проблема фізики.
4. Щоб пов'язати попередній пункт з положенням реальної стрілки (годинникової чи іншої), треба отримати зв'язок між значенням певного часу, хвилини або секунди з кутом її положення. Це пропонується зробити на зразок уявного фізичного експерименту та встановити «експериментальні рівняння» $\varphi_h = F_1(H)$, $\varphi_m = F_2(M)$, $\varphi_s = F_3(S)$ для, відповідно, годинної, хвилинної та секундної стрілочок.
5. Навчившись робити кроки 2 – 4, лишається виконувати їх щосекунди певну кількість разів. Більшість учнів зрозуміють, що це можливо за допомогою стандартного блоку програмування ***for***.

Виробивши такий план, йдемо за ним.

2. Створення циферблату

На цьому етапі доцільно показати учням два можливі шляхи, які відрізняються не лише використаними командами, а й, головне, *типом даних* у їх аргументах.

2.1. Із текстовими аргументами. У командному вікні MATLAB виконуємо:

```
>> %Етап 1, створення циферблату. Перший шлях.
>> H_dial=ezplot('cos(t)', 'sin(t)', [0, 2*pi]);
>> title('Мій годинник')                                          (2)
>> set(H_dial, 'Color', 'k',   'LineWidth', 2)  %Можливість змінювати властивості лінії
```

Отримуємо коло в окремому (графічному) вікні (позначка >> називається *prompt* (запрошення), і позначатиме, що справа відбувається у Command Line. Знак процента – коментар у MATLAB). На цьому шляху ми використовуємо команду **EAZY PLOT** («легкий графік»), яка значною мірою бере на себе те, що учні знають про побудову графіка «по точках». Ця інтелектуальна команда «розуміє» лише аргументи у текстовому форматі (в апострофах '). Остання команда дозволяє керувати властивостями кривої з іменем (хендлом) *H_dial*, що побудована (у цьому випадку – її кольором і товщиною). Повний перелік її властивостей надасть команда ***get***(*H_dial*). Подробиці в [14–16].

2.2. Методичні міркування. Хоча команда *ezplot* є зручною як у практиці програмування, так і у педагогічному аспекті, розробник MATLAB чомусь вирішив позбутися

її в майбутніх (після 2019 р.) релізах. Такий само *текстовий тип даних* для функцій (тут – тригонометричних) використовує й інший фрагмент коду, аналогічний (2):

```
>> H_dial=fplot(@(t)cos(t), @(t)sin(t), [0, 2*pi]); axis equal
>> title('Мій годинник')                                                    (3)
>> set(H_dial, 'Color', 'g', 'LineWidth',3) %Керування властивостями лінії
>> axis([-1.1 1.1, -1.1 1.1])  %Експериментуйте: без цих команд, і з ними
>> axis off
```

(ми використали деякі відмінності щодо (2), аби мотивувати читача (учня) експериментувати з командами і зрозуміти їх призначення самостійно).

На відміну від багатьох книжок з програмування, з MATLAB, зокрема, ми ставимо питання «Як не зробити учня мавпою?», «Як знайти оптимум між командами, які все за нього зроблять, і тим, що він має зробити самостійно і, таким чином, засвоювати матеріал?» [6-16]. Наша відповідь у тому, аби максимально зменшити рутинну роботу і сфокусуватися на тих ментальних діях, які підлягають засвоєнню і оволодінню. Якщо подібний MATLAB-проєкт виконується не вперше, то увага і зусилля учнів на технічні питання (більшість з таких термінів позначена тут фонтом), на усвідомлення поняття *тип даних* (текстовий, як 'cos(t)' та 'Color'; чисельний, як *x* і *y* унизу; хендл *H_dial* тощо) більше не витрачаються.

2.3. Побудова графіків «по точках», як вчили в школі. На цьому шляху учень використовує, власне, матричну «філософію[4] MATLAB» і, порівнюючи з (2) і (3), краще усвідомлює автоматизм використаних там команд і дій. Викладемо цей шлях:

```
>> %Етап 1, створення циферблату. Другий шлях.
>> t=0: pi/50: 2*pi;  %Створення упорядкованого чисельного вектору t, аргументу
>> %Створення чисельних векторів  x і  y:
>> x=cos(t); y=sin(t);                                                      (4)
>> H_dial=plot(x,y); axis equal
>> set(H_dial, 'Color', 'r', 'LineWidth',5)
>> axis off
```

Отримали поки що лише коло у графічному вікні, рис. 1. Учні можуть додати йому того чи іншого «красивого» вигляду за власним естетичним смаком. З цією метою ми вказали засоби керування зовнішнім виглядом циферблату. Математичний бік справи полягає у тому, що інкремент $\frac{\pi}{50}$ (крок) змінної *t* має бути підібраний «експериментально» таким чином, аби коло було доволі гладким (рис. 1), а остаточна програма достатньо швидкою. Якщо ж інкремент обрати $\frac{\pi}{6}$, то циферблат буде, як на рис. 3. Відмітимо також, що «просте» обчислення чисельних векторів *t, x* і *y* також є наслідком «матричної філософії»; в інших мовах програмування для цього потрібні цикли *for*.

2.4. «Прикрашення» циферблату. Додамо до кола деякі «прикраси», аби зображення більше наблизилося до уявлень про циферблат. Наприклад, позначимо часові ділення на колі, які розташовані з кутовою відстанню $\varphi = 2\pi/12$ один від одного:

```
>>%Побудова поділів циферблату («каменів»)
>> hold on
>> JewelAngle=2*pi/12;  %кут Fi між «каменями»
>> Angles=(1 : 12)*JewelAngle;  %Всі кути «каменів» щодо ОУ
>> JewelX=sin(Angles); JewelY=cos(Angles);  %Координати всі «каменів»      (5)
>> HJewel=plot(JewelX, JewelY, 'o',  'MarkerFaceColor', 'y');  %Будуємо «камені»
>> set(HJewel, 'MarkerSize', 8)  %збільшуємо їх розмір
```

[4] Так ми називаємо все те, що ховається за назвою **MAT**rix **LAB**oratory, MATLAB [14-16].

Зверніть увагу, що у (5) ми відраховуємо кути і координати від координатної осі *OY*, на відміну від (4), де відраховуємо від *OX*. Тут також працює «матрична філософія», яка дозволяє MATLAB-програмістові обходитися без ***for***.

Тепер нанесемо цифрові надписи біля відповідних «каменів». Тут обійтися без ***for*** не вдається. Пропонуємо такий код з поясненнями:

```
>> for i=1:12
   Text=num2str(i);    %перетворення чисел i=1,2,3,… у текстовий формат
   Htext(i)=text(1.1*JewelX(i), 1.1*JewelY(i), Text);
end
```

Він почергово розміщує текст ’1’, ’2’, . . . , ’12’ у змінній *Text* та «друкує» його близько до точки з координатами (*JewelX*(*i*), *JewelY*(*i*)). Додатковою командою

```
>> set(Htext, 'FontWeight', 'bold',  'FontSize',12)
```

можемо керувати параметрами шрифту. Тут *Htext*, *H_dial* і *HJewel* вище – ще один тип даних, т.з. *хендл*, вказівник на графічний об’єкт, призначений для управління його властивостями. Останнє відбувається за допомогою команд ***set*** і ***get*** , приклади яких наведено. Вектори координат *JewelX* і *JewelY* продовжено на 1.1 довжини, аби надписи опинилися поза кругом.

Циферблат побудовано, див. рис. 1. Якщо всі команди за однією з цих побудов зібрати в одному *m*-файлі, то отримуємо MATLAB-програму *DialPlate.m* з наступних текстом

Лістинг 1 файлу *DialPlate.m*

```
%Програма побудови циферблату годинника шляхом 3.2
H_dial=fplot(@(t) cos(t), @(t) sin(t), [0, 2*pi]); axis equal
title('Мій годинник')
set(H_dial, 'Color', 'g', 'LineWidth', 3) %Управління властивостями лінії
axis([-1.1 1.1, -1.1 1.1])  %Експериментуйте: без цих команд, і з ними
axis off
%Побудова ділень циферблату («каменів»)
hold on
JewelAngle=2*pi/12;  %кут Fi між «каменями»
Angles=(1 : 12)*JewelAngle;  %Всі кути «каменів» щодо ОУ
JewelX=sin(Angles); JewelY=cos(Angles);  %Координати всіх «каменів»
% Робимо надписи:
HJewel=plot(JewelX, JewelY, 'o',  'MarkerFaceColor', 'y');  %Будуємо «камені»
set(HJewel, 'MarkerSize', 8)  %збільшуємо їх розмір
```

Тепер *DialPlate* стає «командою» MATLAB (програмою). Для побудови циферблату виконуємо її у командному рядку:

```
>> DialPlate
```

3. Стрілка у початковому положенні.

Тепер побудуємо стрілку у вертикальному положенні, рис. 2. Її форма залежить від чотирьох параметрів h_1, h_2, l_1 і l_2, що дозволить будувати доволі різноманітні стрілки. Вершини стрілки нумеруємо цифрами 1, 2, 3, …, 7, і останню вершину, 8, спрямовуємо у положення 1, щоб замкнути лінію контуру. Очевидно, що координати вершин стрілки наступні:

1$(h_1, 0)$, **2**(h_1, l_1), **3**(h_2, l_1), **4**$(0, l_2)$

і далі симетрично:

5$(-h_2, l_1)$, **6**$(-h_1, l_1)$, **7**$(-h_1, 0)$ і **8**$(h_1, 0)$ (тобто та ж **1**).

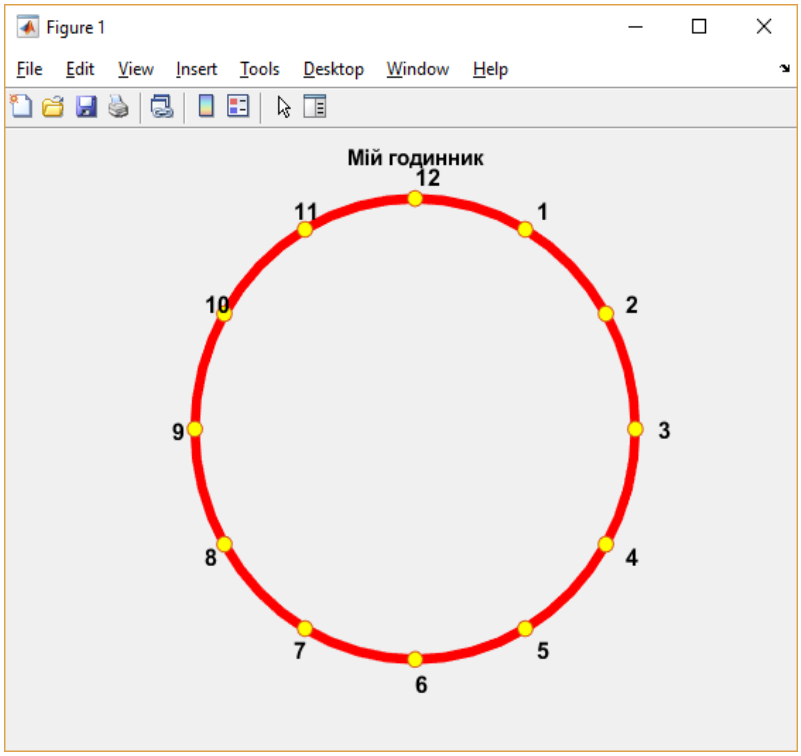

Рис. 1. Вигляд циферблату

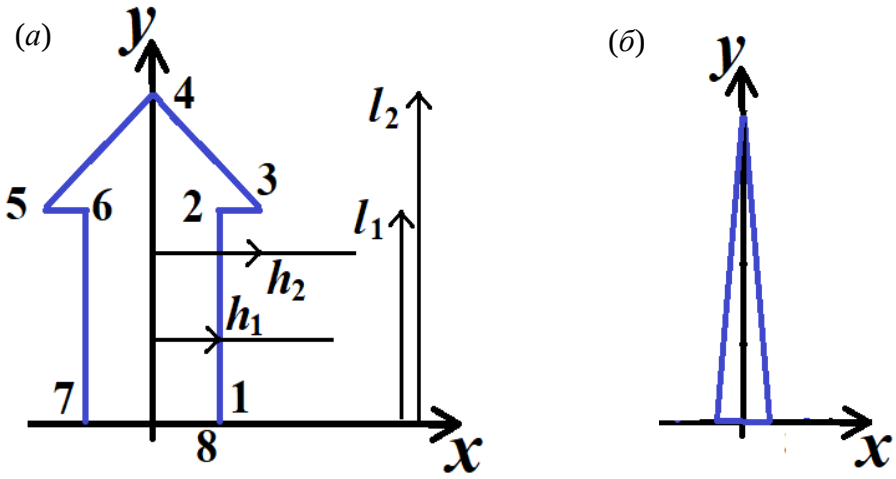

Рис. 2. (а) Схема стрілки 1-2-3-4-5-6-7-8 з параметрами h_1, h_2, l_1 і l_2 і (б) один з її частинних випадків ($h_2 = 0$, $l_2 = l_1$)

За форматом MATLAB-команди *plot*, маємо всі *x*- і *y*-координати зібрати в окремих векторах *ArmX*=[*h*1,*h*1,*h*2,0,-*h*2,-*h*1,-*h*1,*h*1] і *ArmY*=[0, *l*1,*l*1, *l*2,*l*1,*l*1,0,0]. Реалізуємо це, однак уже не в командному вікні як (1) – (5), а в редакторі *m*-файлів (тобто запрошення «>>» не пишемо). Текст програми (лістинг) виглядає таким чином:

Лістинг 2. Програма «*Arm.m*»

```
function [ArmX, ArmY]=Arm(h1,h2,  l1,l2,  Color)
%Побудова вертикальної стрілки кольору Color, яка залежить від h1, h2, l1 і l2.
%Приклади запуску в командному вікні:
%>> Arm(.1, .15, .9, 1,'y')
%>> figure, DialPlate; [X, Y]=Arm(.1, .1,  0,1, 'm');
% >> figure, DialPlate;  Arm(.05, .09, .7, .9,'m');
%Copyright: Name, Date
```

```
ArmX=[h1,h1,h2,0,-h2,-h1,-h1,h1];  %X-координати разом
ArmY=[0, l1,l1, l2,l1,l1,0,0];    %X-координати разом
plot(ArmX, ArmY, 'k')  %%%Контур стрілочки чорним кольором
hold on
fill(ArmX, ArmY, Color) %%%Залити середину замовленим кольором
axis([-1.1 1.1, -1.1 1.1])  %%%Спробуйте без цієї команди
hold off          %%%Спробуйте без цієї команди, аби її зрозуміти
```

Потрібні технічні пояснення. *m*-файл (програма) може бути і без ключового слова ***function***, як попередня програма. Та замість такого типу MATLAB-програми (скрипту) краще обрати інший, *m*-функцію [15,16]. У такому разі вхідна інформація подається через аргументи, а програма має пов'язати остаточний її результат з ними. Тут, як пояснюють коментарі, будуються стрілки на зразок (*а*) і (*б*), рис. 2. Ця *m*-функція має 5 вхідних аргументів; перші чотири управляють формою стрілки згідно рис. 2, останній управляє її кольором. Вона створює два вихідних аргументи *ArmX* і *ArmY*, кожний з яких буде матрицею вимірністю 1 на (вектором). При запуску програми з командного вікна останні можна не вказувати (див. приклади запуску), якщо їх подальше використання не передбачається.

Все закоментоване після ключового слова ***function*** грає роль допомоги (Help). Структура його зрозуміла [15,16]. Обов'язково вимагати її від учнів! Зокрема, у хелповій частині наводять приклади виконання програми. Бачимо, що така проста і коротка програма дійсно здатна породжувати доволі різноманітні стрілки.

Далі ця програма буде викликатися у ролі допоміжної (як підпрограма). У такому випадку будувати зображення не слід. Тобто рядочки, позначенні трьома знаками %, слід буде закоментувати.

Останній рядочок «хелпової» частини також вважаємо принципово важливим: систематичне вживання поняття *Copyright* виховує самоповагу у майбутніх програмістів.

Завдання для самостійного виконання учнями може бути таким: побудувати стрілку у вертикальному положенні; спрямовану униз; побудувати в одному з двох можливих горизонтальних положень.

4. Стрілка під довільним кутом до *Oy*

Візьмемо довільний кут φ до *Oy* і побудуємо ту саму стрілку. Згідно з планом розділу 2, вважатимемо, що разом із нею обертається система координат Ox_1y_1, у якій стрілочка нерухома. Доцільно звернути увагу, що така задача перетворення координат є типовою у механіці і фізиці. Запропонуємо учням не «пригадувати формули», як вони часто висловлюються, а самостійно (нехай вже не в перший раз) їх вивести.

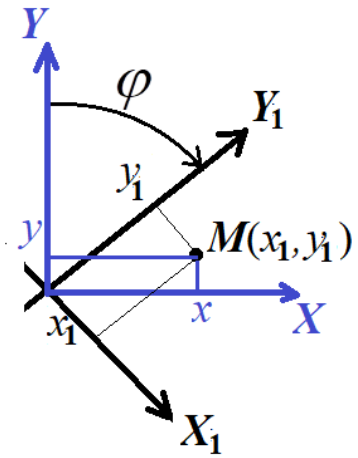

Рис. 3. Нагадування задачі про перетворення координат (x_1,y_1) в (x,y) точки M

Задача звучить наступним чином. Нехай одна з точок стрілки *M* має координати (x_1, y_1) в рухомій системі координат Ox_1y_1. Остання повернута відносно того ж центра *O* на кут φ. Знайти координати *M* у нерухомій системі Oxy. Для кожної точки за допомогою рис. 3 отримуємо:

Кут φ_1 радіус-вектору *OM* щодо осі $OУ_1$

$$tg(\varphi_1) = \frac{x_1}{y_1}, \quad \varphi_1 = arctg\left(\frac{x_1}{y_1}\right);$$

Кут φ_2 радіус-вектору *OM* щодо осі *OУ*

$$\varphi_2 = \varphi + \varphi_1 = \varphi + arctg\left(\frac{x_1}{y_1}\right).$$

Звідси маємо потрібні координати

$$x = |OM|\sin\varphi_2 = \sqrt{x_1^2 + y_1^2}\sin\varphi_2 \quad \text{і}$$

$$y = |OM|\cos\varphi_2 = \sqrt{x_1^2 + y_1^2}\cos\varphi_2 .$$

Наступна програма виконує такі перетворення.

Лістінг 3. Програма «*FiTransform.m*»

```
function [xM,yM]=FiTransform(xM1,yM1, Fi, Color)
%Для точок (xM1,yM1) у системі координат №1, яка рухається разом із стрілкою годинника,
%та повернута на кут Fi (у градусах) відносно Oy,
%шукаються їх координати (xM,yM) у нерухомій системі Oxy.
%Приклади запуску у командному вікні:
%>> DialPlate; [xM1,yM1]=Arm(.05, .09,  0.7,.9, 'b');
%>> FiTransform(xM1,yM1,120, 'g'); %під кутом 120 град. (4 години)
% Copyright: Name, Date.

TgFi1=xM1 ./ yM1; %тангенс кута точок стрілки у рухомій системі координат №1
Fi1=atan(TgFi1);      % кут точок стрілки у рухомій системі координат №1 (радіани)
Fi2=Fi1 + Fi*pi/180; % кут точок стрілки у нерухомій системі координат (радіани)
L=sqrt(xM1 .^ 2 +yM1 .^ 2); % відстані точок стрілки до центру
xM=L .* sin(Fi2);  % координати точок стрілки у нерухомій системі координат
yM=L .* cos(Fi2);
fill(xM,yM, Color)
```

Коментарі і схожі назви змінних роблять програму легко зрозумілою. Кожній точці *M*1(*xM*1,*yM*1) у рухомій системі координат, що повернута на кут *Fi* (у градусах) відносно нерухомої, зіставляються її координати *M*(*xM*,*yM*) в останній.

Та важливо зауважити, що ця програма побудована за «матричною філософією MATLAB», тобто якщо на вхід подається матриця *xM*1 та *yM*1 (координати всіх точок), то і на виході отримуємо матриці такої вимірності *xM* і *yM*, і кожна пара (*xM*(*i*), *yM*(*i*)) – всі «виправлені» координати стрілки під заданим кутом *Fi*. Реалізація «філософії» потребує операції «з крапкою» **.*** або **.^** [14-16]. Наведені у хелпі приклади запуску програми дозволяють її перевірити. Перед цим у підпрограмі *Arm.m* рядочки, вище відмічені %%% наприкінці, слід закоментувати.

5. Емпіричне визначення кута для стрілок трьох видів

Тепер ставимо задачу, щоб програма малювала стрілку в залежності від певної заданої години, хвилини і секунди. Відповідно, програми (у подальшому тексті – підпрограми) мають називатися *hArm*, *mArm* і *sArm*. Їх аргументами мають бути, перш за все, потрібні значення години *H*, хвилини *M* та секунди *S* (і власні геометричні та кольоровий параметри), визначені

за розділом 1. Для цього слід мати функціональну залежність $\varphi = f_h(H)$, так само, як і $\varphi = f_m(M)$, $\varphi = f_s(S)$. Встановимо її на зразок того, як це робиться на підставі фізичного експерименту.

Наступну таблицю можна вважати результатом такого «експерименту»:

Таблиця №1.

Кут годинникової стрілки у залежності від H=Date(4) за (1)

Година *H*	1	2	3	4	. . .	12
Кут φ	30^o	60^o	90^o	120^o	. . .	360^o

Слід знайти функцію $\varphi = f_h(H)$. Зрозуміло, що це має бути лінійна функція: значення φ подвоюється (потроюється), коли подвоюється (потроюється) *H*, тобто найбільш загальний її запис такий

$$\varphi = A_h H + B_h . \quad (6)$$

Достатньо двох рівнянь, щоб знайти коефіцієнти A_h і B_h, наприклад:

$$\begin{cases} 30 = A_h + B_h, \\ 60 = 2A_h + B_h. \end{cases}$$

Звідси маємо $A_h = 30$ і $B_h = 0$. Можна перевірити, що встановленій функції підкоряються й інші дані таблиці. Таким чином, якщо задати годину H і геометричні параметри стрілки *h*1, *h*2, *l*1 і *l*2, програмою *hArm.m* розміщуємо її на циферблат під потрібним кутом:

Лістінг 4. Програма *hArm.m*

```
%Скрипт-програма hArm.m побудови годинникової стрілки
%з геометричними параметрами h1, h2, l1 і l2 і кольору Color, які задані раніше.
%Приклади запуску у командному вікні:
%Встановлюємо H=2; Color='c'; у тілі програми, і
% >> hArm   %без аргументів, бо це програма-скрипт.
%Copyright: Name, Date
H=2; Color='c'; %%%встановлюємо перед кожним запуском
Ah=30; Bh=0;
Fi=Ah*H+Bh;
DialPlate;
[xM1, yM1]=Arm(.05, .09, .7, .9, Color);
[xM, yM]=FiTransform(xM1,yM1, Fi, Color);
fill(xM,yM, Color)
```

Аналогічно створюємо програми *mArm* і *sArm*. Якщо їх повторити одна за одною, утримуючи попередні побудови командою *hold on*, маємо на годиннику потрібні (замовлені) годину, хвилину та секунду.

Доцільно звернути увагу учнів, що створено періодичну функцію, яка дає ті самі побудови для *H*=0, *H*=12, *H*=24, . . ., *H*=1, *H*=13, *H*=25, . . . тощо, тобто клас періодичних функцій тригонометричними не обмежується.

Лишається проблема, що це одноразовий показ часу (хоча інколи пропонують саме такі програми [5]). Тепер слід повторювати все це у циклі з інтервалом 1 секунда, як у наступній остаточній програмі.

6. Постійно діючий годинник

Штучну затримку можна виконати за допомогою МАТЛАБ-команди *pause*(*DelayTime*). Саме так це роблять і в інших мовах програмування. Деякі додаткові новації пояснені нижче.

Лістінг 5. Програма *MyWatch.m*

```
%Script-програма MyWatch
%із стрілками, які рухаються з часом.
%Запускати з командного рядочка:
% >>  MyWatch
%Copyright Name, Date

%До майбутнього циферблату додаємо кнопку «Stop»
hFig=figure;
Hui=uicontrol('String', 'Stop',  'FontWeight','bold');
set(Hui,'ForegroundColor', [1,1,1])
set(Hui,'BackgroundColor', [.6,0 .3])
%Вчимо кнопку «Stop» реагувати на натискання:
set(Hui, 'Callback','StopSignal=1';);
StopSignal=0;
for i=1:10^10  %Велика кількість циклів для i=1, 2, . . .
    Date=clock; %Визначення машинного часу:
    H=Date(4);
    M=Date(5);
    S=Date(6);  %Таким чином, визначені година, хвилина і секунда
    hArm, mArm, sArm  % DialPlate, циферблат будується у першому скрипті
  %поставити кружечок, «гайку», у центрі обертання:
  plot(0,0,'o', 'MarkerSize',10, 'MarkerFaceColor', 'r');
     pause(1)
    hold off
    if StopSignal = =1
        close(hFig)
        break
    end
end
```

Тобто на кожному кроці циклу *i*=1,2, . . . будується певна картинка. Це відбувається у режимі *hold on* («утримуй попередні побудови»), а перед початком нового циклу (перед відповідним *end*) включається режим *hold off*, щоб усі побудови розпочати спочатку. Використовуються вже розроблені підпрограми *hArm* та інші (процедурна парадигма програмування). Кожну побудову прикрашає додавання кружечка у центрі обертання розміром *MarkerSize*=10, рис. 3.

Та головною новизною, ще не обговореною, є наступна. Без неї годинник можливо «виключити» лише примусово, <Ctrl–C>. Аби це виправити, до циферблату додано кнопку «Stop», рис. 3. Із лістінгу 5 легко зрозуміти, як це програмується: потрібний GUI-елемент створюється MATLAB-командою *uicontrol*; серед його властивостей є *Callback* (реакція на подію), якому ми «доручаємо» виконати команду «*StopSignal*=1;». (Важливо відмітити, що GUI-елементи у MATLAB були створені вже в об’єктно-орієнтованій парадигмі). Наприкінці кожного циклу значення змінної *StopSignal* перевіряється і коли вона дорівнює 1, фігура з хендлом *hFig* закривається, цикли перериваються.

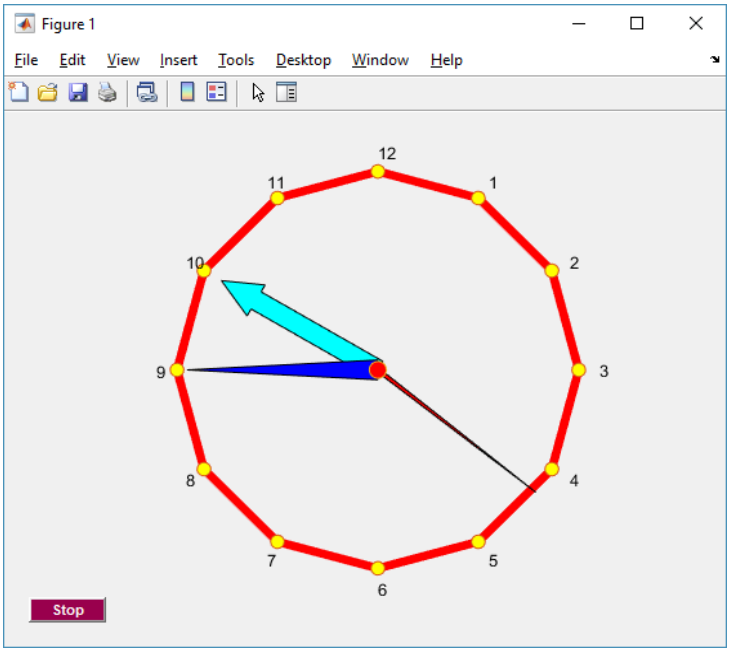

Рис. 3. Годинник в остаточному вигляді. Додана кнопка для його зупинки.

7. Недоліки програми. Подальші можливості

Можна порадити студентові зробити кілька варіантів цього віджету, рис. 3, які відрізнялися б формою циферблату, стрілочок і шрифтом надписів. Така «гра» закріплює розуміння MATLAB-команд *ezplot, plot, set* тощо [14-16], які були використані.

Деякі покращення програми запропоновані наприкінці попереднього розділу. Лишається кілька математичних і суто програмістських недоліків у викладеній програмі. Якщо секундна стрілка рухається увесь час (точніше, кожну секунду), то хвилинна пересувається ривками після того, як секундна переходить ділення «12». Ще гірше – годинникова: вона зрушується лише раз на годину. Між тим, у реальному аналоговому годиннику стрілки можуть займати й проміжне положення. Причина цього явища – у спрощеній апроксимації (6). Щоб його уникнути, слід замінити (6) на більш загальну залежність, наприклад, для кута годинникової стрілки: $\varphi = A_h H + A_m M + B_h$. Цю чергову задачу для учнів тут не розглядаємо.

Більш суттєвий недолік полягає у тому, що цей віджет повністю «забирає» командне вікно середовища MATLAB. Останнє можна звільнити лише натисканням на кнопку «Stop». Не дивно, бо процесор весь час контролює цикли, які виконуються програмою; не «залишає» програму навіть тоді, коли вона час *DelayTime*=1 простоює. Можливий штучний вихід із такого становища: саме середовище, програму MATLAB, запустити ще раз і продовжувати роботу у новому. У такому випадку операційна система комп'ютера, яка «навчена багатозадачності», керує двома середовищами в окремих *thread* (процесах).

Ситуація краща у, скажімо, середовищі Java-розробок (Eclipse або інші): тут є засоби паралелізації. Починаючи з версії 2007 р. MATLAB має також окремий засіб (toolbox) паралелізації. Цієї теми тут не торкаємось: це може слугувати завданням для самостійної роботи.

Починаючи з версії 2006 р. фірма MathWorks розробляла для MATLAB також засоби об'єктно-орієнтованої парадигми (ООП). На сьогодні всі засоби графіки є ООП-об'єкти. Однією з їхніх властивостей є можливість працювати в окремому *thread* (процесі); це також дозволяє «звільнити» командний рядок [4]. Таким чином відкриваються нові горизонти самостійної роботи для тих учнів, кого саме програмування приваблює більше.

Враховуючи сказане, пропонуємо такі подальші завдання для учнів:

7.1. Запропонуйте інше «сімейство стрілок» і реалізуйте з ним цю програму.

7.2. Зробіть більш плавний рух годинникової стрілки, використавши також значення хвилин.

7.3. Розберіться з об'єктно-орієнтованим програмуванням у MATLAB і зробіть аналогічну програму з використанням цієї парадигми (це доволі складне завдання призначається для досвідчених і завзятих учнів).

Висновки: чому вчить програма «Годинник»

1. Ця робота має на меті не лише запропонувати певні завдання учню з програмування, а й розглядає останнє як ефективний засіб залучити учнів до вивчення математики, фізики й інших дисциплін. Вона продовжує нашу агітацію за методи активного навчання, за «власні відкриття» у ході навчання [12,13]. Підтвердженням цього слугують розробки наших студентів [6-10].
2. Метод «до науки через програмування» спирається на захопленні майже всіх сучасних студентів передовими комп'ютерними технологіями. Отже, можна очікувати на його значну ефективність.
3. У роботі над програмою учень має звертатися до потрібних математичних і фізичних задач, що дозволяє йому як повторити цей матеріал, так і побачити міждисциплінарні зв'язки, науку в її єдності. У цьому випадку учню слід використовувати тригонометрію, зв'язок між градусною і радіанною мірою кутів, створювати «експериментальну» таблицю залежності кута від значень *H*, *M* та *S*, встановлювати «емпіричні залежності» $\varphi = f(H)$.
4. Реалізація задачі у привабливій і сучасній власній графічній комп'ютерній програмі дає насолоду «любителю комп'ютерів» і мотивує його до подальшого навчання. Таке програмування має бути достатньо простим для більшості учнів, а шлях від ідеї до реалізації достатньо коротким. Навряд чи більшість професійних мов програмування Delphi, C, C^{++}, Java, Python тощо задовольняють таким умовам. Проте середовище MATLAB – саме таке.
5. Реалізуючи названі положення, у статті викладено основні етапи створення такої MATLAB-програми. Вчителі України доволі вільні у виборі засобів, які використовують у курсі інформатики. Однак MATLAB на сьогоднішній день серед них відсутній. Між тим у США його вивчають наприкінці школи, і студенти-початківці MATLAB у певному об'ємі знають. Сподіваємося, що викладене тут буде корисним як вчителям математики і фізики, так і вчителям програмування. Про вихід MATLAB на освітнє поле України свідчить нещодавно започаткована конференція [19].
6. Автор також сподівається, що Міністерство освіти і науки України погодить названі педагогічні підходи як до навчання взагалі, так і до програмування, і сприятиме визнанню середовища MATLAB в освітній сфері країни.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Програмування на Python (2016). Скрипт аналогового годинника на Python. Відновлено з http://opython.blogspot.com/2016/07/python.html.
2. БлогNot. (2016). Програма годинника на мові Javascript. Відновлено з http://blog.kislenko.net/show.php?id=1520.
3. CyberForum. (2012). Аналоговий годинник на C++. Відновлено з http://www.cyberforum.ru/cpp-beginners/thread424764.html.
4. Scott Frasso. (2016). Scottsclock. MATLAB Central File Exchange. Retrieved from https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/9503/versions/3/previews/scottsclock.m/index.html.
5. Veksler, A. (2008). Analog clock. Retrieved from https://de.mathworks.com/matlabcentral/fileexchange/20528-analog-clock.

6. Гаєв, Є. О., Рожок, О. & Овчарчин, Н. (2014). Звук та музика в курсі програмування. *Інженерія програмного забезпечення*, 19 (3), 41-48. Відновлено з http://er.nau.edu.ua/bitstream/NAU/39439/1/Гаєв-Рожок-Овчарчін_9714-24829-1-SM.pdf.
7. Гаев, Е., Мартич, М., & Тарак, Г. (2015). Программы моделирования случайных явлений для изучения программирования и математики. *Информационные технологии в образовании*, 23 (2), 30-42. Извлечено из http://ite.kspu.edu/issue-23/p-30-42.
8. Гаев, Е. А. & Малинина, Д. (2017). Параметрическая роза – предмет математики, программирования, эстетики. *Информационные технологии в образовании,* 1 (30), 9-24. Извлечено из http://ite.kspu.edu/issue_30/p-9-24.
9. Gayev, Ye., Khavray, K., Skoroded, A. … Pruss, T. (2017). Digital Laboratory of Information Processes Theory: an innovative educational approach. *Proceedings of the XIII International Scientific and Technical Conference "Avia-2017"* (pp. 638–641). Kyiv: National Aviaton University. Retrieved from http://avia.nau.edu.ua/doc/avia-2017/AVIA_2017.pdf.
10. Gayev, Ye. A., & Kalmikov, V. V. (2017).The Travelling Salesman Problem in the engineering education programming curriculum. *Proceedings of National aviation university*, 3(72), 90-98. Retrieved from http://jrnl.nau.edu.ua/index.php/visnik/article/view/11989/16164.
11. Гаєв, Є. О. (2018). MATLAB-програма дисперсії світла на призмі та метод навчання на «Власних відкриттях». *Інформаційні технології в освіті*, 3 (36), 30-35. Відновлено з http://ite.kspu.edu/Issue_36/p-30-45.
12. Гаев, Е. А. (2016). Программирование: путь к сердцу студента. *Тезисы докладов международной научной онлайн-конференции «Тараповские чтения – 2016»*, Харьков, 1-4 марта 2016 г. (с. 23). Харьков: Цифрова друкарня. Извлечено из http://theormech.univer.kharkov.ua/tar-conf/Book_of_abstracts_2016.pdf.
13. Gayev, Ye. A., & Azarskov, V. M. (2018). Educational "Own Discoveries" Method by an easy MATLAB-Programming for Engineers. *Materials of 34th International CAE Conference and Exhibi-tion "Evolving Engineering Simulation: the Age of the Digital Twin", Vicenza, Italy, 8-9 October 2018* (p. 95). Retrieved from https://www.academia.edu/37811759/Educational_Own_Discoveries_Method_by_an_easy_MATLAB-Programming_for_Engineers_How_to_teach_future_engineers_that_are_students_yet.
14. Gayev, Ye. A. & Nesterenko, B. N.(2006). *MATLAB for Math and Programming* (Textbook). Zaporizhzhia: Polygraph. Retrieved from http://www.exponenta.ru/educat/systemat/gayev/index.asp
15. Азарсков, В. М. & Гаєв, Є. О. (2019). Програмування та математика з MATLAB. *Сучасне програмування для інженерів*, ч. 1. Київ: Інтерсервіс.
16. Гаєв, Є. О. & Азарсков, В. М. (2016). Складні типи даних та алгоритми, інтелектуальні програми. *Сучасне програмування*. Ч. 2 (модулі 3 - 5). Київ: НАУ.
17. Wikipedia. (2019). *History of the modern clock*. Retrieved from https://en.wikipedia.org/wiki/Clock.
18. Wikipedia. (2020). *Analysis*. Retrieved from https://en.wikipedia.org/wiki/Analysis.
19. Romanenko, V. H., Solomakha, T.S. & Gayev, Ye. (Eds.) (2019). MATLAB and computer calculations in education, science and engineering (Conference Thesis). Retrieved from https://www.researchgate.net/publication/332655611_MATLAB_and_computer_calculations_in_education_science_and_engineering.

REFERENCES (TRASLATED AND TRANSLITERATED)

1. Programming in Python (2016). Analog clock script in Python. Retrieved from http://opython.blogspot.com/2016/07/python.html.
2. BlogNot. (2016). Clock script in Javascript. Retrieved from http://blog.kislenko.net/show.php?id=1520.
3. CyberForum. (2012). Analog clock in C++. Retrieved from http://www.cyberforum.ru/cpp-beginners/thread424764.html.

4. Scott Frasso. (2016). Scottsclock. MATLAB Central File Exchange. Retrieved from https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/9503/versions/3/previews/scottsclock.m/index.html.
5. Veksler, A. (2008). Analog clock. Retrieved from https://de.mathworks.com/matlabcentral/fileexchange/20528-analog-clock.
6. Gayev, Ye., Rozhok, O. & Ovcharchyn., N. (2014). Sound and music in course of programming. *Software Engineering*, 19 (3), 41-48. Retrieved from http://er.nau.edu.ua/bitstream/NAU/39439/1/Гаєв-Рожок-Овчарчін_9714-24829-1-SM.pdf.
7. Gayev, Ye., Martych, M. & Tarak, H. (2015). Programs for modelling random events for the sake of learning both programming and mathematics. *Information technologies in education*, 23 (2), 30-42. Retrieved from http://ite.kspu.edu/issue-23/p-30-42.
8. Gayev, Ye.A. & Malynyna,. D. (2017). Parametric rose as a subject of mathematics, programming, aesthetics. *Information technologies in education*, 1 (30), 9-24. Retrieved from http://ite.kspu.edu/issue_30/p-9-24.
9. Gayev, Ye., Khavray, K., Skoroded, A. … Pruss, T. (2017). Digital Laboratory of Information Processes Theory: an innovative educational approach. *Proceedings of the XIII International Scientific and Technical Conference "Avia-2017"* (pp. 638–641). Kyiv: National Aviaton University. Retrieved from http://avia.nau.edu.ua/doc/avia-2017/AVIA_2017.pdf.
10. Gayev, Ye. A., & Kalmikov, V. V. (2017).The Travelling Salesman Problem in the engineering education programming curriculum. *Proceedings of National aviation university*, 3(72), 90-98. Retrieved from http://jrnl.nau.edu.ua/index.php/visnik/article/view/11989/16164.
11. Gayev, Ye. O. (2018). Matlab-program for light dispersion on prizm and "own discoveries" educational method. *Information technologies in education*, 3 (36), 30-35. Retrieved from http://ite.kspu.edu/Issue_36/p-30-45.
12. Gayev, Ye. O. (2016). Programming: the path to the student's heart. *Abstracts of the International Conference «Tarapov Readings»* (p. 23). Kharkiv, Ukraine. March 1–15, 2016. Kharkiv: Tsyfrova drukarnia. Retrieved from http://theormech.univer.kharkov.ua/tar-conf/Book_of_abstracts_2016.pdf.
13. Gayev, Ye. A., & Azarskov, V. M. (2018). Educational "Own Discoveries" Method by an easy MATLAB-Programming for Engineers. *Materials of 34th International CAE Conference and Exhibi-tion "Evolving Engineering Simulation: the Age of the Digital Twin", Vicenza, Italy, 8-9 October 2018* (p. 95). Retrieved from https://www.academia.edu/37811759/Educational_Own_Discoveries_Method_by_an_easy_MATLAB-Programming_for_Engineers_How_to_teach_future_engineers_that_are_students_yet.
14. Gayev, Ye. A. & Nesterenko, B. N.(2006). *MATLAB for Math and Programming* (Textbook). Zaporizhzhia: Polygraph. Retrieved from http://www.exponenta.ru/educat/systemat/gayev/index.asp
15. Azarskov, V. & Gayev, Ye. (2019). MATLAB Programming and Mathematics. *Modern programming for engineers*, part 1. Kyiv: Interservis.
16. Gayev, Ye. A. & Azarskov, V. (2016). Complex Data Types and Algorithms, Intelligent Programs. *Modern programming*, part 2 (sec. 3 - 5). Kyiv: Nation Aviation University.
17. Wikipedia. (2019). *History of the modern clock*. Retrieved from https://en.wikipedia.org/wiki/Clock.
18. Wikipedia. (2020). *Analysis*. Retrieved from https://en.wikipedia.org/wiki/Analysis.
19. Romanenko, V. H., Solomakha, T.S. & Gayev, Ye. (Eds.) (2019). MATLAB and computer calculations in education, science and engineering (Conference Thesis). Retrieved from https://www.researchgate.net/publication/332655611_MATLAB_and_computer_calculations_in_education_science_and_engineering.

Стаття надійшла до редакції 05.09.2019.
The article was received 05 September 2019.

Yevhen Gayev
National aviation university, Kyiv, Ukraine

MATLAB-PROJECT "ANALOGUE WATCH"

Learning programming in secondary school or on on first years in higher educational institution may facilitate making mathematical or physical problem "alive". With this in view, we are looking for such tasks for programming that are attractive to students, sufficiently easy but employed important and feasible mathematics and physics.

A project has been suggested (it consists of several lessons and home works) how to create a MATLAB-program of analogue watch on the computer screen that pursues several educational aims. The main steps of creating such a MATLAB program are explained in the article. Initially, students are to split the whole task to several simpler ones that they are able to program. To solve them, students employ plotting graphs, trigonometry and analytical geometry, physical consideration and mental experiment, look for empirical equation to fit data, complete everything with rather easy programming and enjoy finally with pleasant widget of analogue watch on computer screen. Thus, the implementation of the task in a beautiful and modern own graphic computer program enjoy the student and motivate him to further study. This way of programming in secondary school or on first years in higher educational institution should be simple enough for most students, and the path from idea to implementation should be short enough.

Programs are given with comments. They serve, at the same time, as an introduction to MATLAB. It may be seen that its environment do not hidden the sense of the work but rather facilitates it. That is why this environment is suggested for wide use in mathematics and physics learning in Ukrainian education.

Key words: methodology of mathematics and physics, programming, MATLAB, method of own students' discoveries.