УДК 004.9:535.3

Гаєв Є.О.

Національний авіаційний університет, Київ, Україна

MATLAB-ПРОГРАМА ДИСПЕРСІЇ СВІТЛА НА ПРИЗМІ ТА МЕТОД НАВЧАННЯ НА «ВЛАСНИХ ВІДКРИТТЯХ»

DOI: 10.14308/ite000672

MATLAB-програмування фізичних задач доволі просте. Воно захоплює студентів, надає їм сили подолати певні труднощі та дійти до красивої програми з графічним інтерфейсом. Студенти зосереджуються на фізичному формулюванні задачі, обирають потрібний математичний апарат. Тим самим студенти реалізують метод навчання «Шляхом власних відкриттів», який автор пропагує в останніх роботах. У даному випадку це ілюстровано створенням MATLAB-програми, яка демонструє (імітує) розділення білого світла на «веселку» після його проходження через скляну призму. MATLAB надає інструмент «нескладного програмування», що дозволяє студентові більш ефективно вивчати фізику. Окрім законів шкільної геометричної оптики, потрібно знати лише геометрію, тригонометрію та початки аналітичної геометрії, тему «Пряма на площині».

Матеріал статті надасть лекторові з інформатики кілька вправ з алгоритмізації і програмування. Вчителю математики – переконливу ілюстрацію практичного використання її певних розділів (тригонометрія та рівняння прямої). Вчителю фізики – модель та програму для комп'ютерних експериментів з оптики. А учням та студентам, зрештою – доступні їм науково-навчальні вправи, курсову роботу тощо, які демонструють як красоту наук, так і їх взаємний зв'язок.

***Ключові слова**: дисперсія світла, програмування, MATLAB, геометрична оптика, аналітична геометрія.*

Вступ

Справжній учитель відрізняється тим, що захоплюється своєю дисципліною сам та вміє запалити любов і цікавість до неї в своїх учнях. Є різні «вчені методики», як це робити. Одна з них – пережити самим учням ті чужі відкриття, які вони мають вивчати. На цьому базується методика «Навчання шляхом відкриттів» [1,2].

Розвиток кінематографу та телебачення надали, починаючи з 60-х років минулого сторіччя, найширші можливості учням багато чого побачити на власні очі. Це, однак, є пасивним навчанням. Лабораторні експерименти у фізиці та хімії незамінні у навчанні у школі та в університеті тим, що вимагають активного відношення з боку студентів і використовуються в освіті вже більше як 150 років. Розвиток комп'ютерних технологій призвів до виникнення розрахункових практикумів та лабораторних робіт майже з усіх навчальних дисциплін; вони, однак, зрозумілі лише студентам університетів старших років навчання. Виконання таких робіт, безумовно, надзвичайно корисно. Та вони, однак, також можуть бути віднесені до доволі пасивного навчання, бо завдання студентам готують «старші».

Стрімкий розвиток комп'ютерних технологій останніх років, зокрема методи візуалізації розрахунків, готують чергові революційні зміни в освіті та її методах. Наприклад, англійський геній Стівен Вольфрам, творець математичного пакету *Mathematica* та освітнього сайту *Wolfram|Alpha*, пропагує вивчати всю математику через програмування, а саме – програмування клітинних автоматів, яке надасть все розмаїття об'єктів дослідження [3]. Автор статті утримується більш консервативних поглядів, та все ж у роботах [4-11], як і у даній, намагається пропагувати програмування як метод активного

навчання, який одного студента (або кілька, команду студентів-розробників) залучає до створення програмного комплексу (фізичного змісту, як тут), і на цьому шляху зосереджує їх на осмисленні потрібних законів даної науки. Результат роботи надає широкому загалу інших студентів можливість проводити віртуальні експерименти з відповідної дисципліни. Такий підхід відповідає сучасним закордонним тенденціям [12-14]. Більш того, чудовий досвід такого плану є й в Україні. Так, автор [15] створив цілу бібліотеку Паскаль-програм, що дозволяє проводити візуалізовані комп'ютерні експерименти з багатьох фізичних проблем. Та більшу частину свого коду автор не розголошує. На відміну від цього, ми детально викладаємо основні проблеми програмування, бо студент (або викладач фізики) має його, або його модифікацію, відтворити задля «власного відкриття» з цієї науки. MATLAB, на відміну від інших середовищ програмування, дозволяє якнайменше відволікатися на техніку програмування та якнайглибше сфокусуватись на проблемі дослідження.

Про актуальність проблеми свідчить також те, що автори [16] анонсують задачу створення навіть «віртуальної фізичної лабораторії». Одна з можливих її задач, моделювання руху м'яча до баскетбольної корзини, реалізована в MathCAD, [17]. Така комерційна розробка створена Інститутом педагогіки АПН України разом з корпорацією «Квазар – Мікро» [18]. Аналогічною, але більш розвиненою, є розробка Virtual Physical Laboratory (VPLab) британської National Physical Laboratory, з якої деякі програми можна завантажити безкоштовно [19].

1. Постановка задачі, стан питання, його теоретичні аспекти

Ми хочемо створити комп'ютерну програму у математично-програмному середовищі MATLAB, яка буде демонструвати відомий оптичний ефект "утворення веселки" внаслідок проходу променю білого світла через трикутну призму, відомий як дисперсія світла, або заломлення променю [20,21]. Це чудовий, хоча й широко відомий фізичний ефект, який вабить молодь вивчати фізику. Це також доволі складне завдання для програміста; ми не знайшли жодної готової програми на якійсь алгоритмічній мові. Підручники з програмування обмежуються лише допоміжною задачею заломлення світла на одній поверхні розділу середовищ різної оптичної щільності [21], яких для нашої задачі потрібно щонайменше дві. Наша задача складніше останньої ще у тому, що треба виконати цикл за кількома довжинами хвиль.

Цією роботою ми продовжуємо наш цикл "MATLAB-програмування для навчання" [4-11]. Хочемо показати, як назване програмно-математичне середовище можна з ефектом використати для розв'язання складної фізичної проблеми. Її ефективність полягає не лише у тому, що такою програмою вчитель фізики може пояснити учням певне явище, надати їм можливість зробити кілька комп'ютерних експериментів. Важливим є також те, що студент, отримавши завдання на створення такої програми, краще зосереджується на притаманних їй фізичних законах, практично побачить зв'язок з математикою та відчує насолоду від остаточного результату. Такий навчальний метод "власних відкриттів" [1; 2; 7-11], на нашу думку, найкращим чином готує майбутніх науковців та дослідників. Використання програмування задля вивчення фізики пропагують й інші викладачі вищої школи [15]. Ми вважаємо, одначе, що працювати з MATLAB студентові набагато легше; кінцевий результат досягається меншими зусиллями, не відволікаючись на "технічні" питання програмування, та значно швидше. Тому ми називаємо таку роботу "легке" програмування і саме його пропонуємо впроваджувати у систему як старшої шкільної, так і університетської освіти.

Уважають, що «виникнення веселки» у деяких оптичних експериментах було відомо ще з часів Аристотеля. Та лише юний ще професор Кембриджського університету Ісаак Ньютон надав цьому остаточне пояснення (1672 р., [21; 24]). Він підставив під вузький пучок сонячного світла, який ішов з маленького отвору у віконці, трикутну скляну призму, і на протилежній стіні отримав красиву кольорову смугу з усіма кольорами веселки: червоним, жовтогарячим, жовтим, зеленим, блакитним, синім, фіолетовим. З описаного досліду молодий геній зробив важливий висновок: розкладання білого світла в кольоровий спектр

означає, що воно є складеним, сумішшю всіх кольорів веселки. І цей факт означає також, що промені різних кольорів заломлюються у призмі по-різному: найбільше фіолетові промені, найменше – червоні. Це явище і його пояснення стали називати дисперсією світла [20-22]. Спробуємо його симулювати (тобто відобразити у його основних рисах) у комп'ютерній програмі.

1.1. Постановка задачі визначає те, яку програму будемо створювати. Уявимо темну кімнату з джерелом білого світла на стіні ("Ліхтар"); після його включення промінь іде горизонтально і зустрічає трикутну скляну призму. Остання підвішена за її вершину з можливістю бути повернутою вправо-вліво (потрібен "інструмент" для такого повороту). Промінь двічі перетинає грані призми, вхідну й вихідну, і двічі заломлюється, розпадаючись кожного разу на кілька кольорових променів. На екрані спостерігаємо остаточну "картинку". Повертаючи призму, маємо зміни у картинці.

Маємо, таким чином, уявлення про остаточний вигляд майбутньої програми на екрані комп'ютера, показаний на останньому рис. 4. Слайдер обрано як засіб повертати призму навколо точки *A*. Після фіксації призми натискуємо кнопку «Світло!» і спостерігаємо фізичний ефект на екрані зліва.

1.2. Фізичний аспект проблеми. Він полягає у тому, аби розуміти закони заломлення світла, а потім їх зімітувати у програмі. Припустимо, промінь з кутовим коефіцієнтом k_L відносно осі *OX* системи координат перетинає грань призми, яка має кутовий коефіцієнт k. (Індексом *L, Light*, позначаємо відношення до світлового променю). Це означає, що промінь та грань складають наступні кути до осі *Ox*: $\varphi_L = \arctan k_L$ та $\varphi = \arctan k$ відповідно[4]. Кутовий коефіцієнт нормалі до грані є

$$k_n = -1/k \tag{1}$$

та її кут до *Ox* складає

$$\varphi_n = \arctan k_n . \tag{2}$$

Схема рис 1 допомагає зрозуміти, що кут між променем та нормаллю складатиме

$$\alpha_{L1} = \varphi_n - \varphi_L \tag{3}$$

(рівняння прямих надписано над ними).

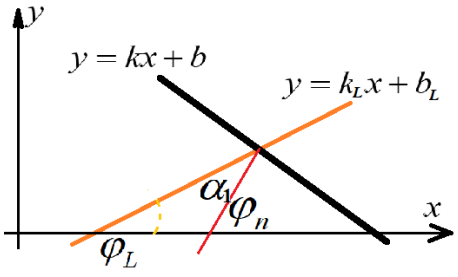

Рис. 1. До знаходження кута заламаного променю.

Ефект заломлення полягає у тому, що заломлений промінь піде під кутом α_2 до нормалі таким, що

$$\frac{\sin\alpha_2}{\sin\alpha_1} = \frac{n_1}{n_2} = n_{12},$$

де n_1 і n_2 оптичні щільності першого середовища та другого. Коефіцієнт заломлення n_{12} менше одиниці для переходу «повітря – скляна призма», $n_{12} < 1$, та більше одиниці для

[4] Ця функція реалізована у MATLAB під назвою *atan*, а *tg* як *tan*.

зворотного переходу «скляна призма – повітря», $n_{21} > 1$ закон Снеліуса [17]. Окрім цього, n_{12} залежить від частоти світла променю, про що мова нижче у розділі 4. В результаті маємо розрахункові формули:

$$\alpha_2 = \arcsin\left(n_{12} \sin \alpha_1\right) \text{ – кут заломленого променю до нормалі,} \quad (4)$$

$$\varphi_{L2} = \varphi_n - \alpha_2 \text{ – кут заломленого променю відносно осі } OX, \quad (5)$$

та

$$k_{L2} = \tan \varphi_{L2} \text{ кутовий коефіцієнт заломленого променю до } Ox. \quad (6)$$

З урахуванням останнього, можна записати рівняння заломленого променю через точку торкання K_1 чи K_2.

1.3. Математичний апарат проблеми. Всі потрібні математичні знання, як бачимо, обмежуються елементарною геометрією, тригонометрією та однією з задач аналітичної геометрії «рівняння прямої через дану точку $K(x_K, y_K)$ заданого кутового коефіцієнту k». Її рівняння, як відомо, таке:

$$y = y_K + k(x - x_K). \quad (7)$$

Підсумуємо. Якщо промінь з кутовим коефіцієнтом k_L перетинає грань призми з кутовим коефіцієнтом k у точці $K(x_K, y_K)$, то треба послідовно виконувати обчислення за (1) – (7). Маючи вже рівняння для променю та грані призми виду (7), потрібно буде знайти їх точку перетину K_1 чи K_2. Можна починати конструювання програми. Обираємо для цього середовище MATLAB [4-6].

Аби виклад був прозорішим і зрозумілішим, важливо ввести зручну і послідовну систему позначень як у даному тексті, так і у програмі. Кут усіх прямих, що далі розглядатимуться, до осі *Ox* будемо позначати як φ, а кут між променем та нормаллю до прямої як α. Індекси позначатимуть належність до променю світла (*L*), до нормалі (*n*), до першої (K_1) або другої точки перетину (K_2) призми.

2. Конструювання MATLAB-програми

2.1. Спочатку створюємо графічний інтерфейс майбутньої програми (GUI, Graphical User Interface). Для цього MATLAB пропонує спеціальне середовище (рис. 2), яке запускається наступною командою у командному вікні:

```
>> guide
```

(позначка >> називається *prompt*, і позначатиме, що справа відбувається у Command Line). Тут обираємо такі UI-елементи (User Interface), які забезпечують потрібну функціональність. А саме, для даної задачі – статичний текст *StaticText* для майбутніх незмінних надписів; графічне вікно *axis1*, де буде «відбуватись дія» – показ призми та променю; повзун (слайдер) *slider* та дві *PushButton*, кнопки для натискання – одна для "включення" світла, друга для отримання інформації чи допомого. Подвійним кликом на них викликається Property Inspector, що дозволяє встановлювати значення тим чи іншим властивостям даного UI-об'єкту. Більш детально це викладено в [5,23]. Та тут важливо підкреслити кілька головних моментів: 1) серед властивостей *slider* параметрам *Min* та *Max* слід надати значень відповідно найменшого -10^o та найбільшого $+10^o$ значень повороту призми до її крайнього лівого та крайнього правого положень; 2) властивості *Tag* для слайдеру *slider* та для кнопок *PushButton* надати певні змістовні імена, наприклад *Rotate, Light* та *Help* відповідно. Заготовка такого GUI показана на рис. 2. Радимо зробити графічне вікно близьким до квадрату, інакше будуть помітні викривлення фігур та їх кутів внаслідок геометричних перетворень.

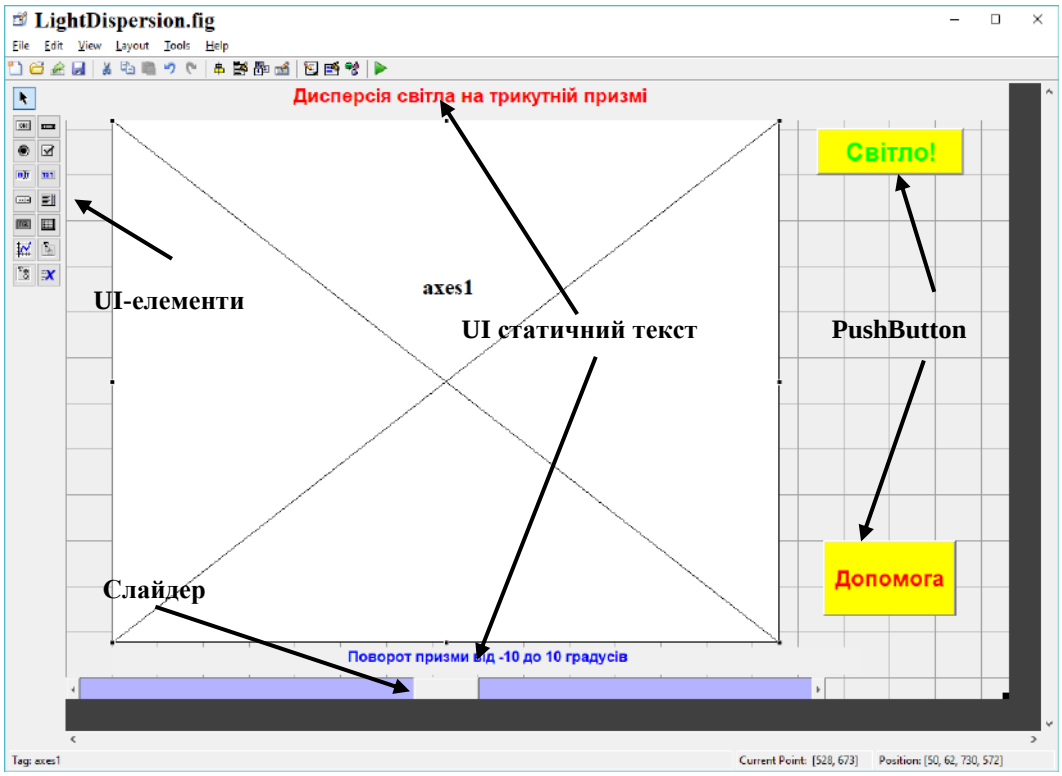

Рис. 2. Дизайн графічного вікна програми у GUI-редакторі guide.

Зберігаємо таку розробку з іменем, наприклад, *LightDispersion*. Усі графічні побудови збережуться у файлі *LightDispersion**.fig***. Одночасно у редакторі *m*-файлів буде автоматично створено файл *LightDispersion**.m*** для подальшого програмування обраних UI-елементів та пов'язаних з ними дій.

Далі працюємо з цим супроводжувальним *m*-файлом, який вже містить автоматично згенерований код для «ручної» конкретизації, де з кожним UI-елементом пов'язані певні функції *function*, не більше двох на UI-елемент. Ті з них, які мають постфікс «*_CreateFcn*» програмувати, як правило, не треба; вони призначені для завдання зовнішнього вигляду даного UI-елементу, що був встановлений через Properties Inspector. Завдання коду потрібно лише для функцій із постфіксом «*_Callback*», «зворотній зв'язок», [5,23]. Тому, значний об'єм *m*-файлу помічений коментарем "% DO NOT EDIT".

Та у незначне порушення цього правила, прив'язуємо до першої функції *LightDispersion_OpeningFcn* *m*-файлу *LightDispersion**.m*** код (команди) початкової побудови (хоча це рекомендовано лише для «просунутих» користувачів MATLAB):

```
function LightDispersion_OpeningFcn
% Нижче залишені деякі автоматичні коментарі та додані власні.
%Choose default command line output for LightDispersion
global hPrizma1 hPrizma2  Beta  Xa Ya Xb Yb Xc Yc L LightY
handles.output = hObject;
%контрольні видачі для налагоджування:
%get(handles)  %аби отримати та оглянути всі «handles»
%Дані прямокутника:
Xa=0.7;Ya=1; %(точка підвісу призми A)
Xb=0.8;Yb=0.1; Xc=0.6; Yc=0.1;  %(Вершини призми B і C)
Beta=atan((Xb-Xa)/(Ya-Yb));        %Кут розкриття такої призми
hRect=plot([0,1,1,0,0],[0,0,1,1,0]); hold on, %Будуємо
прямокутник
```

```
    set(hRect.Parent,'Color','k')    %Встановлюємо чорний колір
'k' у ньому
    LightX=1;  LightY=.5;                %Положення «Ліхтаря»
    hLight=plot(LightX,LightY,'oy'); %Малюємо жовтий «Ліхтар»
    set(hLight,'MarkerFaceColor','y')
    %Малюємо контур призми, заповнюємо її світлим синім кольором
[.7 .7 1]:
    hPrizma1=plot([Xa, Xb, Xc, Xa], [Ya, Yb, Yc, Ya],
'Color','k');
    hPrizma2=fill([Xa, Xb, Xc, Xa], [Ya, Yb, Yc, Ya], [.7 .7 1]);
    % Побудова значка – підвісу призми у точці A
    plot(Xa,Ya-.01,'wp','MarkerSize',30,'MarkerFaceColor','w')
    %Довжина сторон призми для контролю:
    L=sqrt((Xb-Xa)^2+(Yb-Ya)^2);
    % Update handles structure
    guidata(hObject, handles);%<-- Залишок автоматичного коду, не
змінювати!
```

У наведеному коді кожна дія пояснена коментарем (пояснювальним текстом за знаком %). Після запуску програми з командного рядка MATLAB

>> LightDispersion

матимемо на екрані зображення «чорної кімнати» з підвішеною призмою та «виключеним» Ліхтарем. Кнопка «Світло!» натискається, повзун слайдеру пересувається, та жодних подій це поки що не викликає. Важливо, що деякі введені тут величини (*Beta*, *Xa*, *Ya* тощо), які мають використовуватися також й в інших функціях, позначені тут як ***global***, «спільна область пам'яті». Цей рядочок коду обов'язково має бути першим. Так само у глобальну область пам'яті передаються вказівники (хендли, *handle*) на призми *hPrizma1* та *hPrizma2*. Хендли зберігають інформацію про всі властивості складних об'єктів, що створюються. Вони не лише дозволяють змінювати та встановлювати їм певні значення, як, наприклад, колір графічного вікна

set(hRect.Parent, 'Color', 'k') %Встановлюємо чорний колір 'k' у ньому

а й також знищувати ці об'єкти, коли потрібно. У даному фрагменті коду довелося заповнювати внутрішність трикутника призми (команда *fill*) певним кольором. Той, що було отримано за умовчанням, був занадто темний. Колір можна встановити або завданням у текстовому форматі, як вище, або як трикомпонентний числовий вектор *Color*=[*Red, Blue, Green*], у якому кожна з компонент від нуля до 1. Підібрати потрібний колір дозволила зручна MATLAB-команда

>> FindColor=uisetcolor. *(8)*

2.2. Створимо код, що відповідає слайдеру. Останній має повертати призму навколо точки півісу *A*. Код слід додати до функції *Rotate*_Callback. У ньому команда «*AngleGrade=get*(*hObject,'Value'*)» використана відповідно до автоматично згенерованих «вказівок» (Hints). Вона отримує значення між -10^o та 10^o, яке видає слайдер, та призначає його змінній *AngleGrade*. Далі йде обробка цієї величини відповідно до наступних геометричних перетворень.

Перш за все, слід зобразити призму у новій позиції, що відповідає її повороту на цей кут. Зміщення слайдеру вправо чи вліво видає цей кут у градусах; *Alfa=AngleGrade**π / 180, тобто α, його відповідник у радіанах, рис. 3. Кут β, половина кута розкриття призми, береться з попередньої функції за посередництво спільної області пам'яті *global*. Точка B, що

віддалена на L від точки підвісу A, тепер повернута на кут $\alpha+\beta$. Тобто маємо її нові координати

$$x_b = x_a + L\sin(\alpha+\beta),\ y_b = y_a - L\cos(\alpha+\beta).$$

Вершина С спочатку утворювала з вертикаллю кут β. Її також повернуто на кут α у тому чи іншому напрямку; приймемо задля конкретності, що направо. Тепер пряма АС утворює з вертикаллю кут $\beta-\alpha$. Аналогічно попередньому, координати вершини С є тепер

$$x_c = x_a - L\sin(\beta-\alpha),\ y_c = y_a - L\cos(\beta-\alpha).$$

Такі ж самі міркування мають місце при повороті призми вліво, тобто на кут $-\alpha$, тому наведені формули зберігаються. Тепер стає зрозумілим той код, що ми додаємо до функції слайдеру *Rotate*_Callback: в *m*-файлі *LightDispersion.m*:

```
    function Rotate_Callback(hObject, eventdata, handles)
    % Деякі автоматичні коментарі збережені:
    % Hints: get(hObject,'Value') returns position of slider
    global hPrizma1 hPrizma2 Xa Ya Xb1 Yb1 Xc1 Yc1 Alfa Beta k1 k2
L% AngleGrade
    global hLight1 hLight2 hNorm
    AngleGrade=get(hObject,'Value');%Кут, на який повернуто призму
слайдером
    Alfa=AngleGrade*pi/180; %у радіани
    %Точка B після повороту призми:
    Xb1=Xa+L*sin(Alfa+Beta); Yb1=1-L*cos(Alfa+Beta);
    %тут Alfa+Beta -- кут 1ї грані відносно вертикалі, Радіани
    L1=sqrt((Xb1-Xa)^2+(Yb1-Ya)^2);  %задля перевірки L1=L
    %Точка C після повороту призми:
    Xc1=Xa-L*sin(-Alfa+Beta); Yc1=1-L*cos(-Alfa+Beta);
    L2=sqrt((Xc1-Xa)^2+(Yc1-Ya)^2);  %задля перевірки L2=L
    %тут -Angle+Beta -- кут 2-ї грані відносно вертикалі, Радіани
    %Знищуємо попередньо-побудовані об"єкти:
    delete(hPrizma1),
    delete(hPrizma2)
    delete(hLight1),
    delete(hLight2),
    delete(hNorm)
    %Будуємо нові геометричні об"єкти:
    hPrizma1=plot([Xa,   Xb1,   Xc1,   Xa],   [Ya,   Yb1,   Yc1,
Ya],'Color','k');
    hPrizma2=fill([Xa, Xb1, Xc1, Xa],[Ya, Yb1, Yc1, Ya], [.7 .7
1]);
```

Ураховано, що деякі геометричні об’єкти *hLight1*, *hLight2* та *hNorm* існують, бо створені в іншій функції нижче. Доступ до них тут і там надає спільна пам’ять *global*.

3. Програмуємо «включення світла»

Натискання кнопки «*Світло*!», рис. 4, є найбільш складним для програмування. Її результатом має бути горизонтальна лінія до перетину з призмою (симуляція горизонтального променю світла) у точці K_1, перше його заломлення й пряма до перетину із

другою гранню призми у точці K_2, друге заломлення та подальший хід до екрану. Також можуть бути потрібними деякі допоміжні побудови (лінії).

3.1. Горизонтальний промінь від «Ліхтаря» до K_1. Перша поверхня призми відхилена від вертикалі (осі Oy) на кут $\alpha+\beta$. З властивостей трикутника зрозуміло, рис. 3, що вона складає кут $\alpha+\beta+\pi/2$ з віссю OX, тобто кутовий коефіцієнт прямої AB є $k_1=\tan\left(\alpha+\beta+\pi/2\right)$.

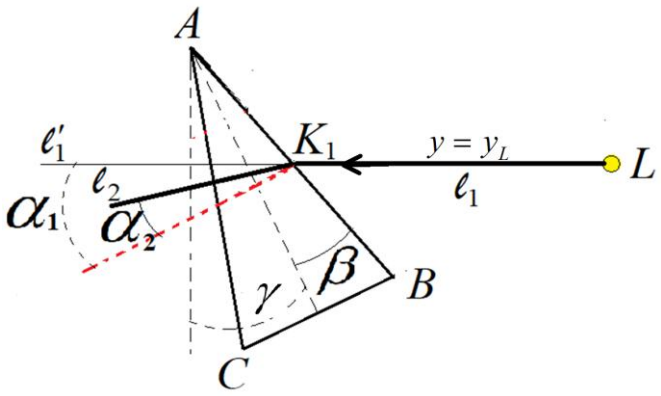

Рис. 3. Геометричні параметри призми після повороту; схема променю після першого заломлення

Рівнянням цієї прямої AB, згідно (7), буде $y=y_A+k_1(x-x_A)$. Оскільки промінь «з ліхтаря», з точки $L(x_L,y_L)$ є горизонтальним, $y=y_L$, то координату x_{K1}, де він торкається грані у точці K_1, знайдемо як

$$x_{k1}=x_a+\frac{(y_L-y_a)}{k_1}.$$

MATLAB-команди

```
Yk1=LightY;
Xk1=(Yk1-Ya)/k1+Xa;% (Xk1,LightY) - точка дотику K1
%Проводимо промінь світла до призми:
hLight1=plot([LightX Xk1],[LightY Yk1],'w', 'LineWidth',3);
hLight2=plot(Xk1,Yk1,'w<', 'MarkerSize',10);
```

будують такий «промінь» білим кольором (`'w'`) завтовшки 3 пікселі, та наприкінці ставить знак стрілки «справа наліво» `'<'` білим кольором `'w'` розміром (`'MarkerSize'`) 10. Ці геометричні об'єкти отримали хендли `hLight1` та `hLight2` відповідно.

Тепер побудуємо відрізок нормалі до прямої AB. За (1) маємо *кутовий коефіцієнт нормалі* $k_{n1}=-1/k_1$, а рівняння її за (7) має вигляд

$$y=y_L+k_{n1}(x-x_{K1}), \tag{9}$$

бо вона проходить через точку $K_1(x_{K1},y_L)$. Аби побудувати її короткий відрізок, беремо трохи менший $x=x_{test1}=x_{K1}-0.1$ та трохи більший $x=x_{test2}=x_{K1}+0.1$ за (9) визначаємо

відповідний йому $y = y_{test} = y_L + k_{n1}(x_{test} - x_{K1})$, і тепер проводимо білу пунктирну лінію між ними. Ось відповідний фрагмент коду:

```
%Побудова нормалі 1
k1N=-1/k1;
%її рівняння: Y=Yk1+k1N*(X-Xk1)
Xtest1=Xk1-.1; Ytest1=Yk1+k1N*(Xtest1-Xk1);%близька точка на
ній
Xtest2=Xk1+.1; Ytest2=Yk1+k1N*(Xtest2-Xk1);
hNorm=plot([Xtest1, Xtest2],[Ytest1, Ytest2],'w:');
```

Тепер час проводити заломлений промінь. Той профіль, що прийшов, має кут до нормалі першої грані призми, за (3), $\alpha_{K1} = \varphi_{n1}$, бо його кутовий коефіцієнт $k_L = 0$ і тому $\varphi_L = \arctan k_L = 0$. Знаходимо за (4) – (6) кутовий коефіцієнт заломленого променю та інші параметри цієї прямої:

$$\alpha_{2K1} = \arcsin\left(n_{12} \sin \alpha_{K1}\right), \quad \varphi_{LK1} = \varphi_{n1} - \alpha_{2K1}, \quad k_{LK1} = \tan \varphi_{LK1}. \tag{10}$$

де, як приклад поки що, приймемо $n_{12} = 0.4$. За цим кутовим коефіцієнтом слід будувати заломлений промінь як пряму, що проходить через K_1. Її рівняння подібно до (9):

$$y = y_{K1} + k_{LK1}(x - x_{K1}). \tag{11}$$

3.2. Промінь від K_1 до K_2. У нашій імітаційній побудові тепер промінь-пряму слід провести від K_1 до K_2**.** Друга поверхня призми відхилена від вертикалі (осі Oy) на кут $\alpha - \beta$, рис. 3. Оскільки вона проходить через точку $A(x_a, y_a)$, то її рівняння за (7) виглядатиме як

$$y = y_a + k_2(x - x_a), \tag{12}$$

де $k_2 = \tan(\frac{\pi}{2} + \alpha - \beta)$. Прирівнюємо її до (9), аби знайти точку $K_2(x_{K2}, y_{K2})$ перетину другої поверхні з променем,

$$y_{K1} + k_{LK2}(x_{K2} - x_{K1}) = y_a + k_2(x_{K2} - x_a).$$

Звідси

$$x_{K2} = \frac{y_a - y_{K1}}{k_{LK1} - k_2} + \frac{k_{LK1}x_{K1} - k_2 x_a}{k_{LK1} - k_2}.$$

Підставляючи це значення в (10) або в (11) знаходимо й іншу координату точки перетину, y_{K2}. Виписувати аналітичну формули для неї не доцільно.

3.3. Друге заломлення променю у точці K_2. Аналогічно до п. 3.1, у точці K_2 будуємо нормаль до другої грані призми у точці K_2. Її кутовий коефіцієнт є $k_{n2} = -1/k_2$, а рівняння, аналогічно до (9), є

$$y = y_{K2} + k_{n2}(x - x_{K2}). \tag{13}$$

Знов беремо $x = x_{test1} = x_{K2} - 0.1$ та $x = x_{test2} = x_{K2} + 0.1$ за (13) визначаємо відповідні ним $y = y_{testI} = y_{K2} + k_{n2}(x_{testI} - x_{K2})$, і тепер проводимо білу пунктирну лінію між (x_{test1}, y_{test1}) та (x_{test2}, y_{test2}). Вона показує нормаль, відповідно до якої заломлення "відбувається"; така побудова допомагає налагоджувати програму, та в її остаточному вигляді її знято.

Будуємо тепер другий заломлений промінь. Промінь, що прийшов, має кут φ_{LK1} до осі OX за (10), та до нормалі другої грані призми, за (3),

$$\alpha_{K2} = \varphi_{n2} - \varphi_{L2},$$

бо його кутовий коефіцієнт $k_L = k_{LK1}$ вже відмінний від нуля, де $\varphi_{L2} = \arctan k_{LK1}$. Знаходимо за формулами (4) – (6) кутовий коефіцієнт заломленого променю:

$$\alpha_{2K2} = \arcsin\left(n_{21}\sin\alpha_{K2}\right),\ \varphi_{LK2} = \varphi_{n2} - \alpha_{2K2},\ k_{LK2} = \tan\varphi_{LK2}.$$

Де слід вже брати $n_{21} = 1/n_{12}$, бо промінь виходить з призми до повітря. За цим кутовим коефіцієнтом слід будувати заломлений промінь як пряму, що проходить через K_2. Її рівняння подібно до (9):

$$y = y_{K1} + k_{LK1}(x - x_{K1}). \tag{14}$$

3.4. Промінь до екрану. Останній промінь (14) ведемо до "екрану", тобто до $x = 0.1$. Перетин відбувається при $x = 0.1$, і з останньої точки до $x = 0$ ведемо його "проекцію" горизонтально. Тобто маємо в решті решт білий промінь від "ліхтаря" до призми у $К_1$, заломлений промінь між $К_1$ до $К_2$ у середині призми (тут вже має відбуватися дисперсія променів, їх розшарування за кольором у залежності від n_{12} та n_{21}, наступний розділ), другий заломлений промінь до екрану та "полоси спектру" на екрані.

Даний етап розробки доцільно завершити тестуванням програми. А саме: приймаємо $n_{12} = 1$; хід променю через призму до екрану має бути прямолінійним, без заломлень.

4. Власне дисперсія світла.

Всі дії від $К_1$ до проекції на екран доцільно розмістити наприкінці *m*-файлу *LightDispersion* в окремій функції, скажімо `ColoredLightPath(i)`, що залежитиме від номера кольору *i*. Тепер код, пов'язаний із кнопкою «Світло!», має лише такий короткий вигляд:

```
for i=1:7      %якщо обрано лише 7 кольорів, як тут
    ColoredLightPath(i)                                  (15)
end
```

однак на початку функції `ColoredLightPath` маємо помістити таблицю обраних для симуляції кольорів та коефіцієнти заломлення на границі "повітря–скло" n_{12} для них. В п. 1 ми назвали характерні кольори для симуляції: червоний, жовтогарячий, жовтий, зелений, блакитний, синій та фіолетовий. Обмежимося даними 7 кольорами. Функція (8) MATLAB дозволяє підібрати "векторний еквівалент" *red-blue-green* для кожного з них, а саме:

```
vColor= [ 0.5   0.2  0.6;   %фіолетовий
          0     0.5  0.7;   %синій
          0.3   0.7  0.9;   %блакитний
          0     1    0;     %зелений                    (16)
          1     0.9  0.2;   %жовтий
          0.85  0.3  0.1;   %жовтогарячий
          1     0    0];    %червоний
```

(16) є насправді матрицею вимірністю 7*3, і звертатися до певного кольору слід на зразок, скажімо, синього *vColor*(2, :).

Кожний з названих кольорів має свій коефіцієнт заломлення n_{12} на границі "повітря–скло". Фізичний механізм відмінності n_{12} для кожного кольору, пов'язаний із різною швидкістю світла певної частоти у різному середовищі [19], для даної симуляції значення немає, так само як і частота відповідних електромагнітних коливань. Важливо, що $n_{12} < 1$. Так, для червоного кольору $n_{12} = 0,6086$, а для фіолетового $n_{12} = 0,5934$. Ще раз підкреслимо, що ми маємо справу не з точним фізичним розрахунком, а з симуляцією даного фізичного явища. Якщо прийняти названі значення n_{12}, то навряд ми зможемо показати явище на такому малому масштабі. Отже, приймаємо довільний вектор коефіцієнтів заломлення:

DispCoeff=[0.56 0.59 0.63 0.67 0.71 0.77 0.8];

За ним і відбувається циклічна побудова (15). В результаті бачимо на "екрані" ліворуч результат "розпаду" білого світла на спектр кольорів у звичному для веселки порядку, рис. 4.

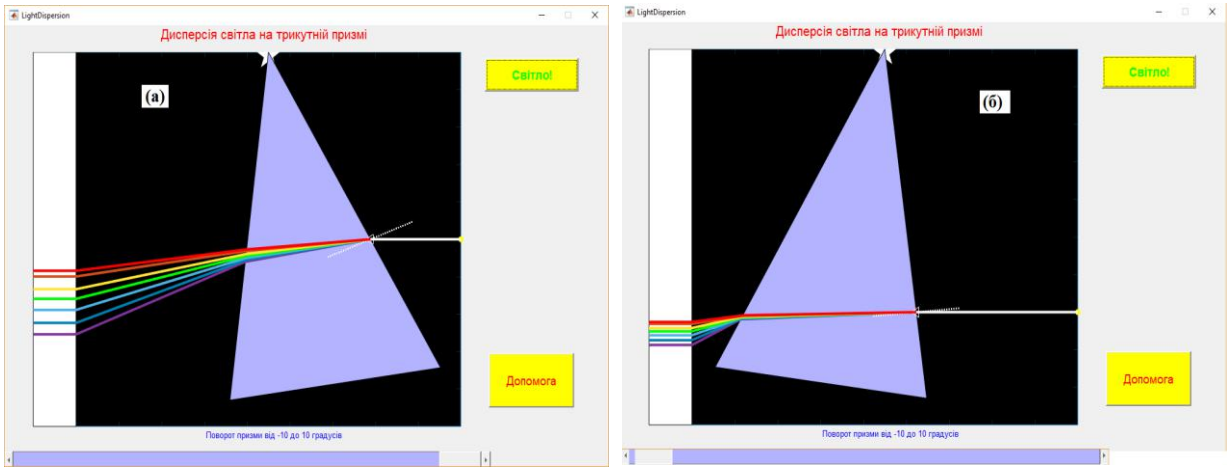

Рис 4. Дисперсія світла призмою, повернутою вправо (а) та уліво (б) з "ліхтарем" у зниженому положенні

Наприкінці слід задіяти кнопку `PushButton` «Допомога», як це вимагають сучасні стандарти програмування. Для цього до функції `Help_pushbutton_Callback`, що їй відповідає, слід додати рядок коду з готовою вже формою довідки:

```
helpdlg({' Текст довідки '}, ' Назва довідки ')
```

Про цю команду можна дізнатися з командного вікна MATLAB за запитом:

```
>> help helpdlg
```

Зразок такої довідки дає рис. 5.

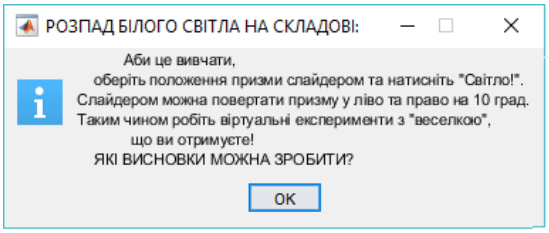

Рис. 5. Вікно "допомоги"

5. Навчання на «власних відкриттях»

Робота студента над такою програмою змушує його концентруватися над законами, які "управляють" даними фізичними об'єктами; студент нібито відтворює шлях, який дана наука вже колись пройшла. Безумовно, студент стикається з певними труднощами з математики та

з програмування. Та перемога над ними, красивий остаточний вигляд програми надає йому сил та піднесення дійти до кінця та, зрештою, відчути задоволення собою та подоланою проблемою. Математичне та програмне середовище MATLAB дозволяє мінімізувати "технічні" труднощі на цьому шляху. Сказане – лише перший аргумент за педагогічний метод "шляхом власних відкриттів".

Безумовно, студента мотивує також і те, як віднесуться до його програми оточуючі – друзі, вчителі, батьки. І тут суттєво, що вони мають тепер можливість робити певні віртуальні експерименти під керівництвом вчителя. Надамо їх короткий опис.

Перш за все, це зміни у розташуванні кольорового спектру для різних положень скляної призми, які задаються зсувом повзуна слайдеру, рис. 4. Далі можливо досліджувати цей спектр шляхом підняття або опускання "ліхтаря" *LightY*, для чого потрібні невеличкі зміни у програмі, п. 2.1.

Також можна дослідити зміну середовищ «повітря» та «скло» на якесь інше, де має місце аномальне заломлення (наприклад, «вода – повітряна порожнина»). Для цього у коефіцієнти *DispCoeff* слід поміняти на такі, що більше одиниці та зменшуються, наприклад *DispCoeff=1./DispCoeff*.

Можна надати наступним студентам розвиток цієї програми, наприклад, проходження світла через сферичну лінзу. Можна зробити більш детальну дискретизацію частотного діапазону світла (більше ніж 7, як тут); тоді зміна кольорів на екрані буде поступова, без розривів, як тут.

Та більш принциповим здається ще одне "власне відкриття" студентами. Вони можуть "емпірично" натикнутися на те, що у разі застосування першої з формул (10) можуть утворитися комплексні числа! Причина зрозуміла: аргумент функції $\arcsin\left(n_{12}\sin\alpha_{K1}\right)$ вийшов за межі $\left|n_{12}\sin\alpha_{K1}\right|\le 1$, тобто стає більше одиниці. Студент у розпачі: чому це так? Що робити?

І тут у вчителя виникає можливість ще одного "відкриття". Нагадаємо студентам, що в історії науки неодноразово були випадки відкриттів "на кінчику пера", а точніше – з формально-математичних ускладнень (планета Уран В.Гєршелєм, нейтріно В.Паулі). У даному випадку також слід припустити за Юджином Вігнером про "дивну ефективність математики у природничих науках" [25]. А саме, "Щось" має ховатися за відмовою математики працювати, коли α наближається до 90^{o}! І дійсно, цей факт слід пов'язати з так званим повним внутрішнім відбиттям, яке у фізиці відоме і має технічні застосування [26].

Висновки

1. "Метод власних відкриттів" має заохочувати студента до вивчення наук у школі чи в університеті. Це особливо стосується такої дисципліни, як фізика.
2. Програмування фізичних задач спирається на захопленні майже всіх сучасних студентів передовими комп'ютерними технологіями, надає їм можливість вивчати "складні" дисципліни активним застосуванням програмування.
3. Програмування не має бути складним і відволікати студента від фізичної суті проблеми. Таким є програмування у середовищі MATLAB, яке спрощує "технічні" складнощі надаючи студентові багату бібліотеку готових функцій для математики та візуалізації результату.
4. У такій роботі студент одночасно звертається до допоміжних математичних задач, що дозволяє йому побачити міждисциплінарні зв'язки, науку в її єдності.
5. Реалізуючи ці положення, у статті викладено основні етапи створення MATLAB-програми дисперсії світла на призмі. Сподіваємося, що це буде корисним як учителям фізики, так і вчителям програмування. Другі отримали навчальний матеріал із програмування, а перші – програму, з якою можна робити певні віртуальні експерименти.

6. Природно, що дана програма не є вільною від недоліків. Виправлення деяких та її подальші можливі модифікації можуть слугувати завданнями для наступних студентів-розробників. Кілька прикладів. Доцільно зробити дискретизацію хвильового діапазону видимого світла (15) не на 7 кольорів, як тут, а на більше, що дасть не дискретний спектр, а ближчий до неперервного. Спрощене завдання можна дати з одним лише заломленням (скажімо, промінь із кімнати в акваріум). Складнішим завданням буде побудувати промені крізь оптичну лінзу.

Подяка

Уперше таку програму ініціював та зробив студент автора Антон Федорук у своїй курсовій роботі наприкінці першого курсу Національного авіаційного університету (2015 р.). Він поставив та розв'язав її самостійно, та от пояснити її у супроводжувальному тексті не зміг. (Взагалі, "пояснити" – це слабке місце вітчизняних студентів, "закладене" недоліками нашої освітньої системи). Крім того, деякі місця у програмі не були розроблені на достатньому рівні внаслідок браку часу. Та все ж автор висловлює йому свою вдячність. Нещодавно зробити таку програму зажадав дипломник автора з коледжу комп'ютерних технологій, та справитися з нею не зміг. Автор вдячний і йому за мобілізацію.

Ця стаття в остаточному вигляді обговорювалася із учителем фізики фізико-математичного ліцею №145 м. Києва заслуженим учителем України О.Г. Розенвайном, якому автор висловлює свою щиру подяку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Discovery Learning. (2018). Retrieved from https://en.wikipedia.org/wiki/Discovery_learning.
2. Обучение путём открытия. (2018). Відновлено з https://ru.wikipedia.org/wiki/Обучение_путём_открытия.
3. Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.
4. Gayev, Y. O. & Nesterenko B.N. (2006). *MATLAB for Math and Programming: Textbook. Zaporozhye: Polygraph.* Retrieved from http://www.exponenta.ru/educat/competit/nagrada1_2015.asp.
5. Азарсков, В.М. & Гаєв, Є.О. (2014). *Сучасне програмування. Модулі 1,2: "Програмування та математика із другом MATLABом"*. Київ.: НАУ.
6. Гаєв, Є.О. & Азарсков, В.М. (2014). *Сучасне програмування. Частина 2 (модулі 3 – 5) "Складні типи даних та алгоритми, інтелектуальні програми"*. Київ: НАУ.
7. Гаєв, Є.О., Рожок, О. & Овчарчин, Н. (2014). Звук та музика в курсі програмування. *Інженерія програмного забезпечення*, 3(19), 41 – 48.
8. Гаев, Е.А., Мартич, М. & Тарак, Г. (2015). Программы моделирования случайных явлений для изучения программирования и математики. *Информационные технологии в образовании*, 23, 30 – 42. Відновлено з http://ite.kspu.edu/webfm_send/829.
9. Гаев, Е.А. & Малинина, Д. (2017). Параметрическая роза – предмет математики, программирования, эстетики. *Информационные технологии в образовании*, 1, 9-24. Відновлено з http://ite.kspu.edu/ru/webfm_send/929.
10. Gayev, Y.A., Khavray, K. & Skoroded, A. (2017). Digital Laboratory of Information Processes Theory: an innovative educational approach. *Матеріали XIII Міжнародної науково-технічної конференції "Авіа-2017". 19-21 квітня, Київ: НАУ, 2017. Секц. 9.42*, с. 638–641. Відновлено з http://avia.nau.edu.ua/doc/avia-2017/AVIA_2017.pdf.
11. Gayev, Ye.A. & Kalmikov, V.V. (2017). The Travelling Salesman Problem in the engineering education programming curriculum. *Proceedings of National aviation university*. Retrieved from http://jrnl.nau.edu.ua/index.php/visnik/article/view/11989/16164.
12. Stephen, J.Ch. (2007). *MATLAB Programming for Engineers*. Thomson Learning.

13. Wagon, S. (2010). *Mathematica in Action: Problem Solving Through Visualization and Computation*. Springer Pbl.
14. Cundy, M.H. & Rollett, A.P. (1974). *Mathematical Models*. Oxford University Press.
15. Овруцький, А.М. (2006). *Фізика на Паскалі: Практикум*. Дніпропетровськ.
16. Кравцов, Г.М., Баев, А.С., Лемещук, А.И. & Орлов, В.В. (2017). Мультимедийный редактор виртуальной физической лаборатории в системе дистанционного обучения «Херсонский виртуальный университет». *Information Technologies in Education*, 4 (33), 63- 79.
17. Флегантов, Л.О. & Антонець, А.В. (2017). Комп'ютерне моделювання механічного руху тіла засобами MathCAD. *Інформаційні технології в освіті*, 1 (30), с. 97 - 109.
18. Будкевич, Т.В. (2011). Програмні засоби навчання фізики, хімії і біології. *Комп'ютер у школі та сім'ї*, 4, 35 - 41.
19. The Virtual Physical Laboratory (VPLab). Software Screenshots. (2018). Retrieved from http://vplab.ndo.co.uk/screenshots_gallery.
20. Заломлення. (2017). Відновлено з https://uk.wikipedia.org/wiki/Заломлення.
21. Дисперсія світла: історія відкриття та опис явища. (2017). Відновлено з http://stylezhinki.ru/osobistist/7649-dispersija-svitla-istorija-vidkrittja-ta-opis.html.
22. Закон Снеліуса. (2018). Відновлено з https://uk.wikipedia.org/wiki/Закон_Снеліуса.
23. Бодриев, И.Б., Бандеров, В.В. & Задворнов, О.А. (2010). *Разработка графического пользовательского интерфейса в среде MATLAB. Учебн. пособие*. Казань: Казанский гос.ун-т.
24. Элементарный учебник физики (Под ред. академика Г.С.Ландсберга) т. 3 (Колебания, волны, оптика. Строение атома). (1964). М.: «Наука».
25. Вигнер, Е. (1968). Непостижимая эффективность математики в естественных науках. *Успехи физических наук*, 94 (3), 535 - 546. Відновлено з http://ogs-seminar.narod.ru/materials/effectiveness_of_mathematics.pdf.
26. Повне внутрішнє відбиття. (2016). Відновлено з https://uk.wikipedia.org/wiki/Повне_внутрішнє_відбиття.

REFERENCES

1. Discovery Learning. (2018). Retrieved from https://en.wikipedia.org/wiki/Discovery_learning.
2. Learning by Discovery Method. (2018). Retrieved from https://ru.wikipedia.org/wiki/Обучение_путём_открытия.
3. Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.
4. Gayev, Y. O. & Nesterenko B.N. (2006). *MATLAB for Math and Programming: Textbook.* Zaporozhye: Polygraph. Retrieved from http://www.exponenta.ru/educat/competit/nagrada1_2015.asp.
5. Azarskov, V. M. & Gayev, Y. O. (2014). *Modern programming. Modules 1,2: "Programming and mathematics with the mate MATLAB"*. Kyiv: NAU.
6. Gayev, Y.O. & Azarskov, V.M. (2016). *Modern programming. Part 2 (modules 3 – 5) "Complex data types and algorithms, intellectual programs"*. Kyiv: NAU.
7. Gayev, Y.O., Rozhok, O. & Ovcharnyn, N. (2014). Sound and music in programming. *Software Engineering*, 3(19), 41 – 48.
8. Gayev, Y.A., Martich, M. & Tarak, G. (2015). Programs for modelling random events for the sake of learning both programming and mathematics. *Information Technology In Education*, 23, 30 – 42. Retrieved from http://ite.kspu.edu/webfm_send/829.
9. Gayev, Y.A. & Malinina, D. (2017). Parametric rose as a subject of mathematics, programming, aesthetics. *Information Technology In Education*, 1, 9-24, Retrieved from http://ite.kspu.edu/ru/webfm_send/929.
10. Gayev, Y.A., Khavray, K. & Skoroded, A. (2017). Digital Laboratory of Information Processes Theory: an innovative educational approach. *Materiali XIII Mizhnarodnoi naukovo-technichnoi*

conferencii "Avia-2017". 19-21 April, Kyiv: NAU, 2017. Sect. 9.42, p. 638–641. Retrieved from http://avia.nau.edu.ua/doc/avia-2017/AVIA_2017.pdf.

11. Gayev, Ye.A. & Kalmikov, V.V. (2017). The Travelling Salesman Problem in the engineering education programming curriculum. *Proceedings of National aviation university*. Retrieved from http://jrnl.nau.edu.ua/index.php/visnik/article/view/11989/16164.
12. Stephen, J.Ch. (2007). *MATLAB Programming for Engineers*. Thomson Learning.
13. Wagon, S. (2010). *Mathematica in Action: Problem Solving Through Visualization and Computation*. Springer Pbl.
14. Cundy, M.H. & Rollett, A.P. (1974). *Mathematical Models*. Oxford University Press.
15. Ovrutski, A.M. (2006). *Physics in Pascal: Practicum*. Dnipropetrovsk.
16. Kravtsov, H., Baiev, A., Lemeshchuk, O., Orlov, V. (2017). Multimedia editor of virtual physical laboratory in distance learning system «Kherson virtual university». *Information Technology In Education*, 4 (33), 63 - 79.
17. Flehantov, L. & Antonets, A. (2017). Computer simulation the mechanical movement body by means of MathCAD. *Information Technology In Education*, 1 (30), 97 – 109.
18. Budkevich, T.V. (2011). Programming means to learn physics, chemistry and biology. *Computer in the school and family*, 4, 35 - 41.
19. The Virtual Physical Laboratory (VPLab). Software Screenshots. (2018). Retrieved from http://vplab.ndo.co.uk/screenshots_gallery.
20. Refraction. (2017). Retrieved from https://uk.wikipedia.org/wiki/Заломлення.
21. Dispersion of light: the history of discovery and description of the phenomenon (2017). Retrieved from http://stylezhinki.ru/osobistist/7649-dispersija-svitla-istorija-vidkrittja-ta-opis.html.
22. Snell's law. (2018). Retireved from https://uk.wikipedia.org/wiki/Закон_Снеліуса.
23. Bodriev, I.B., Banderov, V.V. & Zadvornov, O.A. (2010). *Development of graphical user interface in MATLAB. Textbook.* Kazan: Kazan State university.
24. Elementary physics text book (Edited by Academician G.S.Landsberg), v. 3 (Oscillations, waves, optics. Atom structure).(1964). Moscow: «Nauka».
25. Wigner, E. (1968). The Unreasonable Effectiveness of Mathematics. *Natural Sciences, Comm. Pure and Appl. Math.*, 131, 1 (1960). In: Uspekhi fisicheskikh nauk, 1968, 94 (3), 535 - 546. Retrieved from http://ogs-seminar.narod.ru/materials/effectiveness_of_mathematics.pdf.
26. The total internal reflection. (2016). Retireved from https://uk.wikipedia.org/wiki/Повне_внутрішнє_відбиття.

Стаття надійшла до редакції 29.07.2018
The article was received 29 July 2018.

Yevgeny Gayev
National aviation university, Kyiv, Ukraine

MATLAB-PROGRAM FOR LIGHT DISPERSION ON PRIZM AND "OWN DISCOVERIES" EDUCATIONAL METHOD

MATLAB-programming of physical problems is quite easy. It captures students, encourages them to overcome certain difficulties and obstructions on the way to a pleasant program with graphical interface. Students focus to physical problem formulation, choose mathematical tools required. This way they realize educational method «Path to own Discoveries» being propagated in last publication by the author. In this article, this is illustrated by creation of MATLAB-program that displays (imitates) white light dispersion to a «rainbow» when it passes through a glass prism. MATLAB provides capabilities of an «easy programming» that facilitates students to learn physics more effectively. Besides school laws of geometrical optics, they need to know only school geometry, trigonometry and basics of analytic geometry (namely, the topic «Line on a plane»).

This article delivers a few exercises with algorithms and programming to the informatics lecturer. To the teacher of mathematics it provides convincing illustration for practical use of some its chapters (trigonometry, line equations etc.). Teacher of physics gets a model and the program for virtual computer experiments in optics. But the students, finally, get simple scientific and educational tools, term works etc. that display pleasure of sciences and their mutual intrinsic relationships.

Keywords: light dispersion, programming, MATLAB, geometrical optics, analytical geometry.

Гаев Е.О.
Национальный авиационный университет, Киев, Украина

MATLAB-ПРОГРАММА ДИСПЕРСИИ СВЕТА НА ПРИЗМЕ И МЕТОД ОБУЧЕНИЯ НА "СОБСТВЕННЫХ ОТКРЫТИЯХ"

MATLAB-программирование физических задач довольно просто. Оно увлекает студентов, придает им силы преодолеть определенные сложности и прийти к красивой программе с графическим интерфейсом. Студенты сосредотачиваются на физической формулировке задачи, выбирают необходимый математический аппарат. Тем самым студенты реализуют метод обучения «Путем собственных открытий», который автор пропагандирует в последних работах. В данном случае это иллюстрировано созданием MATLAB-программы, которая демонстрирует (имитирует) разделение белого света на «радугу» после его прохождения через стеклянную призму. MATLAB надает инструменты «легкого программирования», что и позволяет студенту овладевать физикой более эффективно. Кроме законов школьной геометрической оптики необходимо знать лишь геометрию, тригонометрию и начала аналитической геометрии, тему «Прямая на плоскости».

Материал данной статьи предоставляет лектору по информатике несколько упражнений по алгоритмизации и программированию. Учителю математики – убедительную иллюстрацию практического использования ее определенных разделов (тригонометрии и уравнений прямой). Учителю физики – модель и программу для компьютерных экспериментов по оптике. А учителям и студентам, самое главное, доступные им научно-обучающие упражнения, курсовую работу и т.п., которые демонстрируют как красоту наук, так и их взаимную связь.

Ключевые слова: дисперсия света, программирование, MATLAB, геометрическая оптика, аналитическая геометрия.