ОСВІТА ТА ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

УДК 004.9+501:372.8

Гаєв Є.О., Рожок О., Овчарчин Н.
Національний авіаційний університет

ЗВУК ТА МУЗИКА В КУРСІ ПРОГРАМУВАННЯ

Пропонується включати математичні аспекти звуку та його програмування до університетського курсу комп'ютерних наук та програмування. Це зробити найлегше, коли ці дисципліни вивчаються у середовищі MATLAB. Викладено наш досвід, пропозиції та прості розробки, що створені разом із студентами. Розглянуто якнайпростіші так і складніші приклади синтезування звуку з точки зору цього предмету вивчення, так і способу покращення знань студентів з програмування.

Предлагается включать математические аспекты звука и его программирование до университетского курса компьютерных наук и программирования. Это сделать легче, когда эти дисциплины изучаются в среде MATLAB. Изложены наш опыт, предложения и простые разработки, созданные вместе со студентами. Рассмотрены попроще так и более сложные примеры синтезирования звука с точки зрения этого предмета изучения, так и способа улучшение знаний по программированию.

It is proposed to include audio and mathematical aspects of its programming to the university course in computer science and programming. This is easiest to do when these subjects are studied in an environment MATLAB. Expounded our experience, and offers a simple design, created with the students. Yaknayprostishi considered and sophisticated examples of synthesizing sound in terms of the subject of study, and the way to improve students' knowledge of programming.

Ключові слова: *програмування, MATLAB, цифровий звук, викладання та вивчення програмування.*

Вступ

Комп'ютерні науки стали ледь не найголовнішими дисциплінами сьогоднішньої системи вищої освіти. На сьогоднішній день вони включають в себе не тільки власне програмування і різні мови програмування, а також багато самостійних, окремих дисциплін – алгоритми, структури даних, операційні системи та комп'ютерні мережі, комп'ютерна графіка, штучний інтелект та інші [1-5]. Природно, що наявність такої кількості комп'ютерних дисциплін ставить викладача перед питанням вибору методики викладання.

Загальна теза про "всемогутність" комп'ютерної науки, що закріплює інтерес студентів до навчання та до галузі в цілому, отримає ще одне фактичне підтвердження, якщо серед інших питань розглянути тему про "цифровий звук". Не відволікаючи студента від основних питань, ця тема здивує студента комп'ютерними можливостями та стимулюватиме його до дослідницької діяльності.

Використання звуку в кожній з відомих нині мов програмування – Pascal, C, C++, Java і тд. – завдання не з простих і потребує знання мови в значному обсязі. Проте, на початку викладання програмування все більшу популярність здобуває "матрична лабораторія" MATLAB, яка є чудовим інструментом не лише оволодіння математикою [6], а й програмуванням в цілому [7-12]. У даній статті викладено звукові можливості MATLAB як в плані викладання, так і в плані вивчення програмування в процесі розробки акустичних і музичних MATLAB-програм.

Постановка задачі

Запис, зберігання і відтворення звуку в комп'ютерних науках стало можливим лише після створення спеціальних пристроїв для комп'ютера (звукової карти, динаміка, т.п.), кодування сигналів в спеціальних форматах та збереження у файлах, розробки програмних засобів і протоколів їх взаємодії зі звуковими файлами. Відповідні проблеми, що почали розроблятися з 1970-х гг., багато в чому вже вирішені [13–19], хоча поліпшення і подальша розробка тривають. Темою даної статті є робота з готовими пристроями та інструментами, використання звуків в курсі програмування та алгоритмізації, їх відтворення і вивчення, а також

програмування звуків і музики. Принципові питання даної проблематики обговорюються в роботах [20-22] та ін. Даний неповний список джерел та пошук в Інтернеті показують, що певна література з цього питання доступна, але вона дуже обмежена. Крім того, зазначені джерела присвячені складним та серйозним аспектам обробки звуку та орієнтовані, як правило, на професіоналів. З них одна лише книга [14], також розрахована на акустиків-професіоналів, має справу з середовищем MATLAB. Визначення ж "мінімального набору" відомостей для демонстрації цифрового звуку, так само як і його використання у викладанні основ програмування, ставиться тут вперше.

Вершиною використання звуку у практиці людства є музика. Та за останні роки у цій галузі культури виникла нова сфера, так звана комп'ютерна музика, музикально-комп'ютерні технології [15]. Багато студентів, що вивчають комп'ютерні науки та програмування, цікавляться нею, володіють тією чи іншою з існуючих складних програм для створення цифрової музики [16]. Та проблема полягає не у тому, аби створити конкурентно-спроможну аналогічну програму [16], а у тому, щоб виділити основні і прості елементи з цієї галузі комп'ютерних наук та надати студентам можливість їх вивчати шляхом створення доволі простих власних програм, експериментувати з такими можливостями.

Найпростіший звук в MATLAB

Демонструючи щонайширші можливості зазначеного математичного пакета, доцільно здивувати студентів виконанням наступної MATLAB-команди:

>> load handel, sound(*y*)

Вони почують фрагмент ораторії Генделя. (Пояснення: "запрошення" >> тут і далі означає запуск на виконання в командному вікні MATLAB; після налагодження, ці команди слід поставити у текст програми безпосередньо; ім'ям файлу, що завантажується можуть бути і "*train*", "*laughter*", деякі інші. Передбачається, що з численних додаткових інструментів MATLAB повинен бути встановлений AudioVideo Toolbox із стандартним розміщенням в ProgramFiles\MATLAB\...\AudioVideo\). Тепер учні охоче збільшують привабливість всіх своїх програм вставкою в кінець коду або його етапів тих чи інших звуків. В якості наступного кроку, можна потішити студентів "стисненням" і "розтягуванням" цих або записаних ними (див. далі) звуків,

>> FS = 4000; sound (y, FS)

>> FS=12000; sound (y, FS)

(нормальне звучання при частоті відтворення *FS*=8192 *Гц*), або їх "складанням"

*>> Y*1 = [*y*; *y*; *y*]; *sound*(*y*1) (1)

(спираємося на свободу матричних операцій MATLAB; в даному випадку – подовження стовпчику даних). Одночасно слід продемонструвати "портрет" того чи іншого звуку

>> plot(*y*)

(подібний до зображеного на рис. 1, ліворуч та рис. 2), обговорити фізичне походження і математичний зміст "сигналу *y*".

Виникає питання і про запис звуку. Опишемо команди, які реалізують це в MATLAB.

У версіях MATLAB до v.2012a, запис через мікрофон з частотою *Fs* протягом часу *T, сек*, може бути здійснена наступним фрагментом коду:

>> Fs = 8192; *T =3*;

>>Info=[*'Кажіть щось ', num2str(T), ' секунд*:*'*];

>> disp(*Info*)

>> y =wavrecord(*T***Fs*, *Fs*); (2a)

>> disp('*Запис закінчено*!')

Отриманий запис *y* (фактично – числовий вектор) можна відтворити командою *sound*(*y, Fs*), або побудувати її числовий графік:

>> plot(*y*)

>>title('*bf*'*Портрет" запису через мікрофон*')

>> xlabel('*Час*, *it*{*сек*}')

>> ylabel('*Сила звуку*')

Параметр *Fs* у ході запису аналогового сигналу (голосу) спочатку грає роль частоти дискретизації, а у команді *sound* – частоти відтворення. Її значення *Fs*=8192 *Гц* є такою, при якій звучання та тембр голосу достатньо якісні.

Зазначена команда запису, однак, в старших версіях MATLAB замінена на іншу, що використовує об'єктно-орієнтовану парадигму програмування. А саме:

>>% створення аудіо-об'єкта ***r****:*

>> ***r*** *= audiorecorder*(*4000, 8, 1*);

>>% включаємо мікрофон:

>> record(***r***); (3)

>>% зупинка запису:

>> stop(***r***);

В цьому фрагменті коду відбувається наступне. Створюється аудіо-об'єкт ***r*** з частотою дискретизації *Fs*=4000 *Гц* (варіанти

для більш якісного запису 8000, 11025, 22050 і більше *Гц*), 1 канал з "глибиною" запам'ятовування 8 байт. Аудіо-об'єкт ***r*** поки "порожній", з деякими полями, про які можна дізнатися командою *get*(***r***). Запит допомоги

```
>> help audiorecorder
```

дозволяє дізнатися і про методи об'єкта ***r***. До них крім вже використаних методів *record* і *stop* входять і методи *play*, *getaudiodata* та інші. Ось їх використання у можливому фрагменті коду:

```
>>% Відтворення запису r:
>> P = play(r);
>>% Перетворення запису r
>> % у числовий вектор:
>> mySpeech=getaudiodata(r);
>> %Побудова його графіку:
>> Plot(mySpeech)
```

Хоча це і не відноситься до програмування, але тут якраз вдалий привід надати студентам фундаментально-важливе поняття про дискретне перетворення Фур'є як спосіб показати "частотний портрет" записаного звуку без складних математичних подробиць. Для цього до запису *mySpeech* слід застосувати MATLAB-команду *fft* (Fast Fourier Transform).

Та спочатку продемонструємо в інтерпретації [10], як саме MATLAB визначає „спектральний склад” сигналів. Створимо тестовий сигнал у часі t, $y = a_1 \sin \omega_1 t + a_2 \sin \omega_2 t$ (беремо задля простоти лише дві частоти ω_1 і ω_2, що його складають); „виміряємо” його у дискретні проміжки часу $t = 0 : dt : N \cdot dt$, де N – кількість вимірів. Наприклад:

```
>>%Спектр з частот 20 і 40 Гц:
>>Omeg1=20; a1=2; Omeg2=40; a2=-3;
>>%Обравши крок, моменти вимірів:
>> dt=0.0001; N=20000; t=0:dt:N*dt;
>>%Вектор „вимірів”:
>>y=a1*sin(Omeg1*t)+a2*sin(Omeg2*t);
>>P=2*pi*max([1/Omeg1,1/Omeg2]);
```

Маємо „портрет сигналу”:

```
>>plot(t,y)
>> xlabel('Час, \it{сек}')
>> ylabel('Звуковий сигнал')
>>title(['Сигнал; Період=',num2str(P)])
```

Його періодом є $T = max(\frac{2\pi}{\omega_1}, \frac{2\pi}{\omega_2})$.

Задача полягає у тому, аби знайти складові частоти тестового сигналу та порівняти з тими, що його утворили. Обернена задача: маємо якийсь сигнал y та рівномірний масив значень часу t, коли його отримано. Тепер покажемо, як за цією вхідною інформацією отримати спектральний склад сигналу:

```
>>% інкремент часу,кількість вимірів:
>> DT=t(2)-t(1); n=length(t);
>>% отримуємо дійсну частину FFT:
>> Y=fft(y); ReY=real(Y);
>> % отримуємо простір частот:
>> Omega=2*pi*(1:n)/(n*DT);
>> % спектральний „портрет”:
>>figure, plot(Omega(1:100), ReY(1:100))
>>  grid on
>> xlabel('Частота, \it{Гц}')
>> ylabel('Дійсна частина FFT')
>>title('Частотний портрет сигналу')
```

По „плесках” можна зробити наближений висновок про частоти, з яких сигнал утворено. У даному числовому випадку знайдемо $\omega \approx 20$ і $\omega \approx 40$ *Гц*; див. рис. 1, праворуч. На причинах можливих похибок не зупиняємось.

Практика показує, що положення 2.1 – 2.3 студенти освоюють легко і швидко. Але даний матеріал вже дає можливості для власних творчих фантазій студентів в програмуванні. Як приклад – рис. 2, графічний інтерфейс програми студентки 1-го курсу НАУ А.Грігорук (створення інтерфейсу командою *guide* див. нижче).

Натискання кнопки "Record" дозволяє записати через мікрофон будь-яку фразу в заданий у програмі час, фрагменти коду (2) або (3). Кнопка "Listen" дає можливість прослухати цей запис через вбудовані динаміки комп'ютера (фрагмент коду (1)). Кнопка "Plot initial Soundwave" дозволяє побачити "портрет" записаної фрази у вигляді графіка: по горизонтальній осі час в секундах, по вертикалі величина звукового сигналу. Варіанти програмної реалізації таких дій також подані вище.

Оскільки записаний звук – числовий вектор (умовно названий ***r***), то його елементи нескладно переписати в зворотному порядку, від останнього до першого. В MATLAB це виконується за допомогою оператора

```
>> r1 = r(end : -1: 1);
```

де ключове слово *end* закріплено за довжиною масиву і обчислюється автоматично. Тепер кнопка "Reverse" дозволяє прослухати запис "з кінця наперед", а кнопка "Plot Soundwave backwards" – побачити "портрет" перевернутого запису.

Аналогічно можна досліджувати "огрубіння" записаного звуку командою

```
>> r2 = r(1: 2: end);
```

(елементи запису через один пропускаються).

Таким чином, наведені прості команди MATLAB дають учням можливість самостійно використовувати звук у власних програмах, поміркувати про фізичну і математичну природу звуку та проекспериментувати з цим. Складніші програми обговорюються далі.

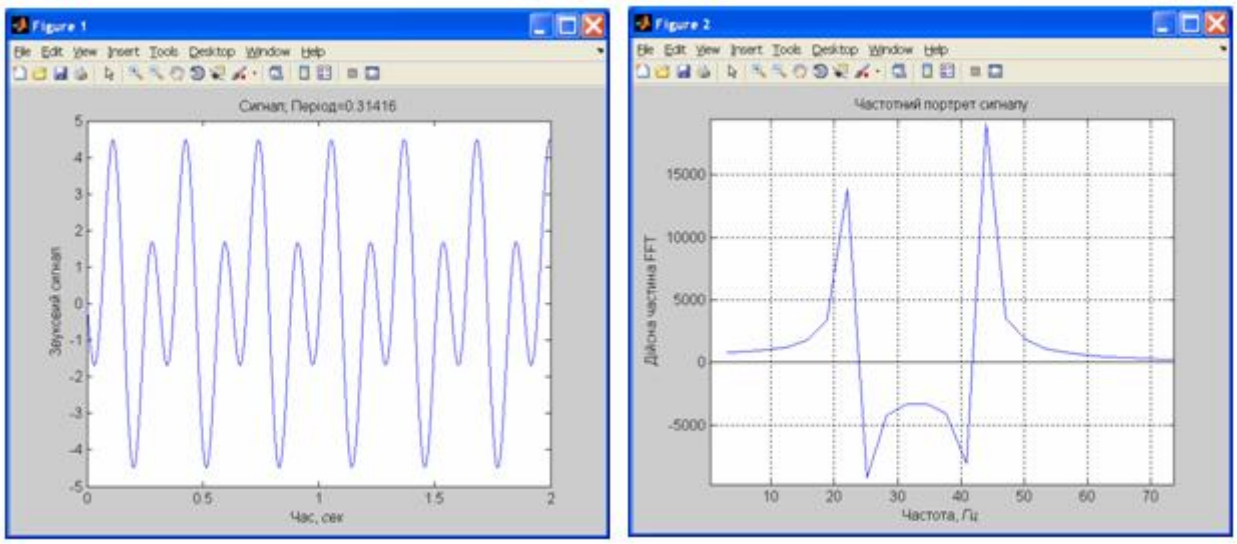

Рис. 1. „Звичайний” портрет сигналу (ліворуч) та його частотний склад за *fft* (прворуч)

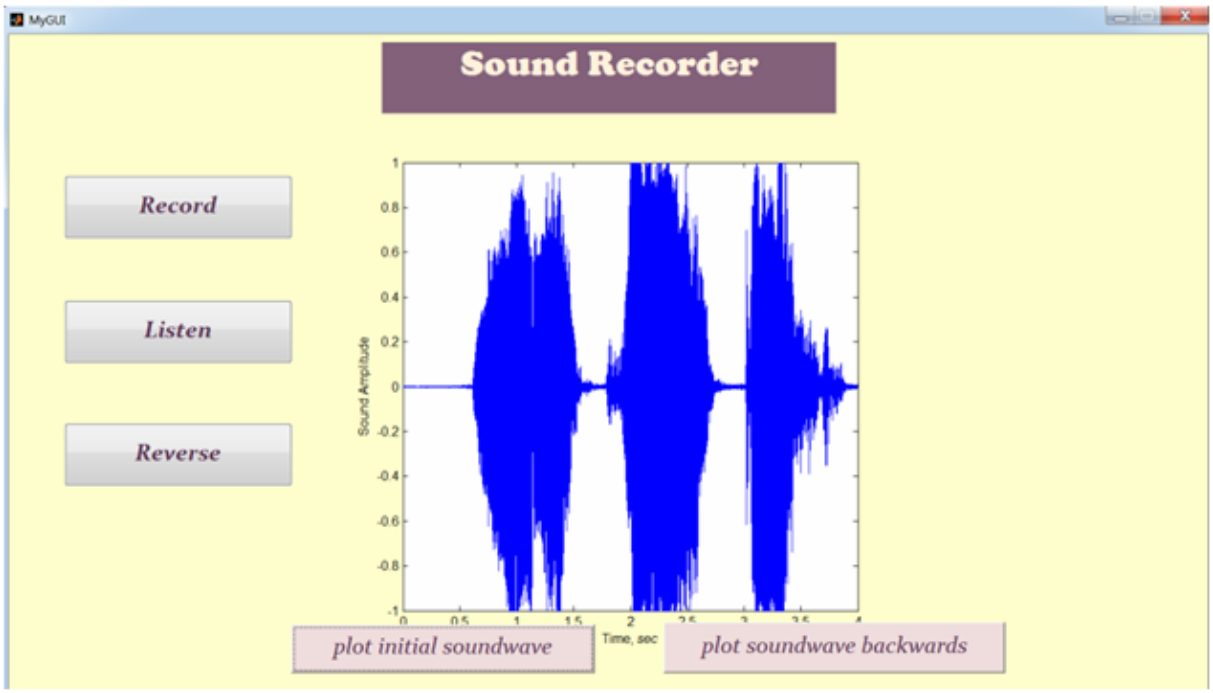

Рис. 2. Графічний інтерфейс для програми запису, відтворення та “перевертання” звук

Програмування звуку в MATLAB

Легко створити гармонічний звук (числовий вектор) *y* за допомогою функції *sin*():

```
>> F=2093; T=3;
>>yC=sin(0:T*550*pi);                (4)
>> sound(yC, F)
```

При даних параметрах – частоті відтворення *F*=2093 *Гц*, отримаємо звук "До" першої октави ("C" в німецькій нотації) [23]; його тривалість дорівнює близько *T*=3 секунди. Звук "Ре" ("D" в німецькій нотації) отримаємо, якщо, відповідно до теорії музики

[23], помножимо частоту відтворення на $2^{\frac{1}{6}}$; інтервал дискретизації тут дорівнює 1. В командному вікні це виглядає так:

```
>>F=F*2^(1/6),yD=sin(0:300*pi*F/F0);
>>sound(yD, F)                    (5)
```

(частота F=2349 *Гц*). Повторивши останню команду, аналогічним чином відтворимо звук "Мі" ("E"), частота F=2637 *Гц*. Продовжуючи множити частоту відтворення на $2^{\frac{1}{6}}$, можна "виконати" найпростішу гаму. Тривалість кожної ноти буде близько 1 *с*, що забезпечується подовженням числового вектора щоразу на *F/F0*.

Графічний інтерфейс для виконання гами

Викликаємо у командному вікні середовище створення графічного інтерфейсу

```
>> guide,
```

у вікні GUI (Graphical User Interface) розташовуємо сім елементів PushButton (за кількістю нот від До3 до До4), тобто створюємо дизайн програми у вигляді клавіш рояля, рис. 3, і в відповідному *m*-файлі за кожної "клавішею" прописуємо коди за зразком (4), (5). Послідовне натискання клавіш зліва направо і потім назад "виконує" гаму До-мінор. Звичайно, можна виконати і інший простий музичний фрагмент в межах однієї октави при однаковій тривалості звучання нот.

Тембр звуку

Вище ми утворювали „чистий", гармонійний звук однієї частоти, що називається фундаментальною для даної ноти. Та музикальні твори цінують за „багатство" звуку, що виражається перш за все у понятті „тембр" звуку (timbre). Під цим розуміють додавання до фундаментальної частоти звуків кратних частот із зменшеною амплітудою, [24,25]. Тобто, замість синусоїди з фундаментальною частотою ω_0 виконують звук

$$y = a_0 \sin\omega_0 + a_2 \sin 2\omega_0 + ... + a_k \sin k\omega_0,$$

де $a_k < a_{k-1} < ... < a_2 << a_0$. Кількість частот k та амплітуд $a_2,..., a_k$ – справа суб'єктивного вкусу. Нами підобрано тембр, що визначається доданком

$$dy = \frac{a_0}{2}\sin 3\omega_0 + \frac{a_0}{2^2}\sin 5\omega_0 + \frac{a_0}{2^3}\sin 7\omega_0 + \frac{a_0}{2^4}\sin 9\omega_0,$$

див. далі.

Простий синтезатор

Програма синтезу числових векторів *t*,

```
t=mySynth(freq, duration, timbred),   (6)
```

що відповідають тій чи іншій ноті згідно до п. 3.1, зберігається у файлі *mySynth.m* і складається з кількох підпрограм. До підпрограми *freqParse*(*note*), що визначає частоту *freq*, відповідну тій чи іншій ноті *note*, звертатимемося з текстовим аргументом *note*, який буде складатися з двох або трьох символів, що позначають ноту, її модифікацію (дієз або бемоль) і номер октави. наприклад:

'C # 4' - До дієз четвертої октави,

'E1' - Мі першої октави,

'Ab3' - Ля бемоль малої октави.

Названа підпрограма в основних рисах виглядає таким чином:

```
function freq = freqParse(note)
% Обраховує значення частоти, Гц.
  if (ischar(note) && length(note) > 1)
    octave = note (end);
    note_ = note(1:end-1);
% Частота еталонної Ля 1-ї октави:
    base = 440;
      switch(note_)
        case 'Cb' % нота До
          freq = base/(2^(10/12));
        case 'C'
          freq = base/(2^(9/12));
        case 'C#' % До дієз
          freq = base/(2^(8/12));
        case 'Db' % нота Ре бемоль
          freq = base/(2^(8/12));
        case 'D' % нота Ре
          freq =base/(2^ (7/12));
  . . . . . .
        case 'B' % Сі бемоль
          freq =base*2^ (1/12);
        case 'H' % нота Ре
          freq = base*2^(2/12);
        case 'H #'
          freq =base*2;
        otherwise
          error ('Немає такої ноти!')
      end
% зміна частоти враховуючи октаву:
      switch(octave)
        case '0'
          freq = freq/16;
        case '1'
          freq = freq/8;
        . . . . . .
        case '7'
          freq = freq*8;
      end
  end
```

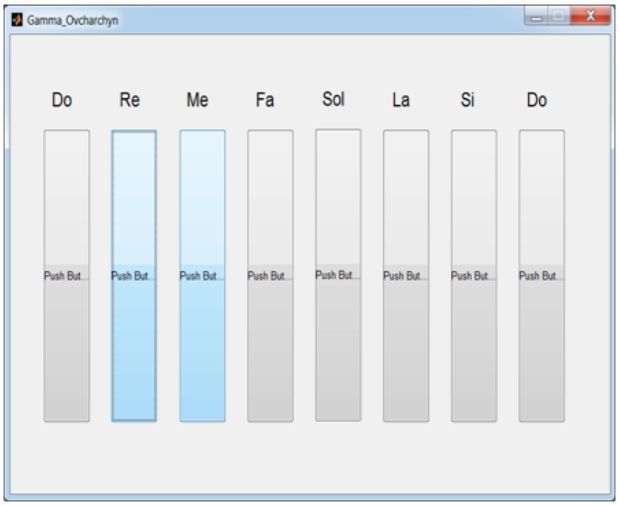

Рис. 3. Графічна програма, що відтворює третю октаву; кожна клавіша відповідає певній стандартні ноті (надписана). Може виконувати гамм та кілька більш складних музичних фрагментів

Пояснимо її. 1). Перший оператор функції *freqParse* – умовний ***if…end***; він аналізує текстовий аргумент *note* і відокремлює від нього текстову складову аргументу *note_* (один або 2 перші символи) і цифрову складову *octave* (останній символ).

2) Далі перший блок ***switch ... end*** по базовій частоті *base* = 440 *Гц* обчислює частоту звучання тієї чи іншої ноти *freq*. Наприклад, нота 'До дієз' ('C#' в міжнародній нотації) звучить з частотою *freq=base*/(2^(8/12))=77.18 *Гц*.

Рис. 4. Фрагмент музичного твору, що відтворено у програмі *havaNagila.m*

Якщо текст, поданий на вхід програми *mySynth* не відповідає ні одній ноті, отримаємо діагностику "*Немає такої ноти*!", оператор *error* у разі *otherwise*.

3) Другий блок ***switch...end*** призначений для уточнення значення цієї частоти відповідно до *octave*, тобто до цифри в кінці текстового аргументу *note*. Як бачимо, для цього, знайдена в першому блоці величина частоти *freq* множиться або ділиться на деяку ступінь двійки.

Іншими аргументами програми *mySynth.m* є *duratio*n і *timbred*, що визначають, відповідно, тривалість синтезованих сигналів у секундах та їх тембр, тобто доданок до основної синусоїди. Останній аргумент приймає значення 0 (гармонічний сигнал не змінюється) або 1 (сигнал модифікується). Він використовується у підпрограмі *oscillator.m*. Ця підпрограма утворює числовий вектор, що відповідає „замовленій" ноті, її тривалості та тембру.

В результаті програма (6) видає числовий вектор *t*, що відповідає частоті *freq* замовленої ноти *note*, її тривалості *duration* та присутності або відсутності модифікації її тембру *timbred*. Далі цей вектор може бути „виконаний" командою (5).

3.4. Виконання музичного твору

На основі описаного синтезатора стандартних музичних нот можна програмувати цифрову музику за її нотним записом. Наводимо скрипт *havaNagila.m*, що програмує в MATLAB фрагмент популярної мелодії, ноти якого показані на рис. 3.

```
timbre = 1; T=44100;
mus = [ ]; %пустий спочатку вектор
%поступове накопичення вектору mus
mus = [mus  mySynth('C4', 0.5, timbre)];
mus=[mus  mySynth('C4', 0.75, timbre)];
mus=[mus  mySynth('E4', 0.25, timbre)];
mus=[mus  mySynth('C#4',0.25, timbre)];
mus=[mus  mySynth('C4', 0.25, timbre)];
mus =[mus  mySynth('E4', 0.5, timbre)];
mus =[mus  mySynth('E4', 0.75, timbre)];
mus =[mus mySynth('G4', 0.25, timbre)];
mus =[mus mySynth('F4', 0.25, timbre)];
mus =[mus mySynth('E4', 0.25, timbre)];
mus =[mus mySynth('F4', 0.5, timbre)];
```

```
mus =[mus mySynth('F4', 0.75, timbre)];
mus =[mus mySynth('Ab4', 0.25, timbre)];
mus =[mus mySynth('G4', 0.25, timbre)];
mus =[mus mySynth('F4', 0.25, timbre)];
mus =[mus mySynth('G4', 0.5, timbre)];
mus =[mus mySynth('E4', 0.25, timbre)];
mus =[mus mySynth('C#4', 0.25, timbre)];
mus =[mus mySynth('C4', 0.75, timbre)];
% „Виконання” вектора mus
sound(mus, T);
```

Як пояснено у коментарях, програма поступово утворює вектор *mus*, що є спочатку пустим, та поступово накопичує числові елементі. Наприкінці, „виконання” музичного вектора *mus* здійснюється командою *sound* з частотою відтворення *T* (такою ж вона має бути у програмі синтезу *mySynth.m*). Кількість елементів в *mus* є 155 000 при *T*=44100, і 38 750 при *T*=5000 *Гц*.

Висновки

Таким чином, середовище рішень математичних задач і програмування MATLAB дозволяє просто, без відволікання на спеціальні питання, використовувати звук і музику як для їх самостійного відтворення та вивчення їх фізико-математичних аспектів, так і як доповнення до задач програмування. Саме останньому дана робота призначена. Використання звуку та музики може бути хорошим, захоплюючим для студентів розвитком стандартного курсу програмування.

У даній роботі ми надали як відомі MATLAB-команди, що використовують звук і можуть покращити якість і привабливість вже існуючого в лектора матеріалу з програмування, так і нові прості розробки, що можуть суттєво розширити арсенал лекцій з даної дисципліни.

Подальшим розвитком теми можуть бути й інші проблеми, робота над якими поведе учнів до нового кола актуальних практично значущих завдань. У їх числі можна назвати, наприклад, розпізнавання звуку, набір тексту за диктуванням, створення голосового пароля для комп'ютера тощо. Цьому будуть присвячені наступні публікації з проблеми.

Список використаних джерел

1. Computer Science Curricula 2013. http://www.acm.org/education/CS2013-final-report.pdf

2. Рекомендации по преподаванию информатики в университетах Computing Curricula 2001: Computer Science. – С.-Петербург, 2002 (http://window.edu.ru/window/library?p_rid=23885).

3. Стаття англ. Вікіпедії "Computer science", http://en.wikipedia.org/wiki/Computer_science

4. Статті російської Вікіпедії “Информатика”, “Компьютерные науки”, https://ru.wikipedia.org/wiki.

5. Бекман И.Н. Компьютерные науки. Курс лекций. М.: МГУ им. М.В.Ломоносова. http://profbeckman. narod.ru/

6. Чен К., Джиблин П., Ирвинг А. MatLab в математическом исследовании. – М.: Мир, 2001. – 346 с.

7. Austin M., Chancogne D. Introduction to Engineering Programming: in C, MATLAB, and Java, 1999.

8. Лазарєв Ю.Ф. Початки програмування в середовищі MatLAB. Навч. посібник. – К.: “Політехніка”, 2000. – 396 с

9. Кондратов В.Е., Королев С.Б. MATLAB как система программирования научно-технических расчетов. – М.: Мир, 2002. – 350 с.

10. Гаєв Є.О., Нестеренко Б.М. Універсальний математичний пакет MatLab і типові задачі обчислювальної математики. Навчальний посібник.– К.: НАУ, 2004. – 176 с.

11. Gayev Ye.A., Nesterenko B.N. MATLAB for Math and Programming: Textbook. – Zaporozhye: Polygraph, 2006 – 102 p.

12. Гаєв Є.О., Нестеренко Б.Н. Опыт и предложения по использованию MATLAB в курсах математики и информатики\\ Тези XI Міжнародної наукової конференції ім. акад.. М.Кравчука. К.: КПІ, 2006.

13. Гордеев О. Программирование звука в Windows. 1999.

14. Ананьев А.Б., Ананьева Е.А., Путилова А.Ю. MATLAB для акустиков, а также всех, кто собирается создавать и обрабатывать различного рода сигналы: учебное пособие. - К.: [ПП ВФ], 2007. - 192 с.

15. Статті російської Вікіпедії “Компьютерная музыка”, “Музыкально-компьютерные технологии”, “Звуковой и музыкальный компьютинг”, https://ru.wikipedia.org/ wiki.

16. Sound and Music Computing, List of Software Tools, http://smcnetwork.org/view/software.

17. Чесебиев И.А. Компьютерное распознавание и порождение речи. М.: Спорт и культура-2008. 128 с.

18. Digital audio http://en.wikipedia.org /wiki/Digital_sound.
19. Цифровий звук. Стаття в українській Вікіпедії http://uk.wikipedia.org/wiki
20. Музыченко Е. Низкоуровневое программирование звука в Windows Компьютер Пресс №6, 2000. http://www.rsdn.ru/article/multimedia/winsnd.xml
21. Программирование звука в Windows (C++) http://soundcoding.ru/
22. Секунов Н. Обработка звука на PC в подлиннике.
23. https://ru.wikipedia.org/wiki/ Статьи “ДО (нота)”, “РЕ (нота)”, “Октавная система”.
24. https://ru.wikipedia.org/wiki/, Стаття "Тембр".
25. https://en.wikipedia.org/wiki/Timbre, стаття "Timbre".

Інформація про авторів:

Гаєв Євген Олександрович – д. т. н, професор, професор кафедри систем управління літальних апаратів Національного авіаційного університету. Наукові інтереси: математика, механіка рідини та газу, алгоритми та програмування.
E-mail: Ye_Gayev@voliacable.com

Рожок Олександр – студент другого курсу кафедри систем управління літальних апаратів Інституту аеронавігації Національного авіаційного університету. Наукові інтереси: програмування, перетворення Фур’є, складні алгоритми, цифрова музика.
E-mail: mizzdev@gmail.com

Овчарчин Назарій – студент другого курсу кафедри систем управління літальних апаратів Інституту аеронавігації Національного авіаційного університету. Наукові інтереси: програмування, складні алгоритми, цифрова та звичайна музика.
E-mail: nazacheres@gmail.com