Міністерство освіти і науки України

Національний авіаційний університет

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Імітаційне моделювання

Лабораторна робота №4

«Основи роботи з пакетом MATLAB»

Варіант№ 38

Роботу виконав:
студент групи СП-325
Козлов Олексій
Роботу прийняла:
Нечипорук О.П.

Київ – 2020

Моделювання випадкових чисел

Мета лабораторної роботи

1. Вивчення основ роботи з призначеним для користувача інтерфейсом системи Matlab;
2. Придбання практичних знань і навичок програмування в системі Matlab;
3. Отримання навичок розробки та реалізації програм на основі математичної моделі об'єкта;
4. Розробка комп'ютерної моделі руху тіла.

Хід роботи

Завдання

3. Побудувати графіки декількох траєкторій руху тіла, що відповідають декільком значенням часу досягнення мети і цільову точку. Як параметри задати координати цілі (X1, У1) і час досягнення мети, початком руху вважати точку (0,0).

8. Побудувати графік економічною траєкторії досягнення тілом цільової точки і цільову точку. Як параметри задати координати цілі (x1, y1), початком руху вважати точку (0,0). Траєкторія вважається економною, якщо мета досягається за мінімальної початковою швидкістю.

Формули:

Рівномірний рух тіла по осі x:

$$x(t) = x_0 + \cos(a)\, v_0 t$$

Рівноприскорений рух тіла по осі y:

$$y(t) = y_0 + \sin(a)\, v_0 t - \frac{1}{2} g t^2$$

Загальний час руху тіла:

$$t1 = \frac{v_0 \sin(a) + \sqrt{v_0^2 \sin(a) + 2 g y_0}}{g}$$

Кут початкової швидкості при направлені тіла в ціль:

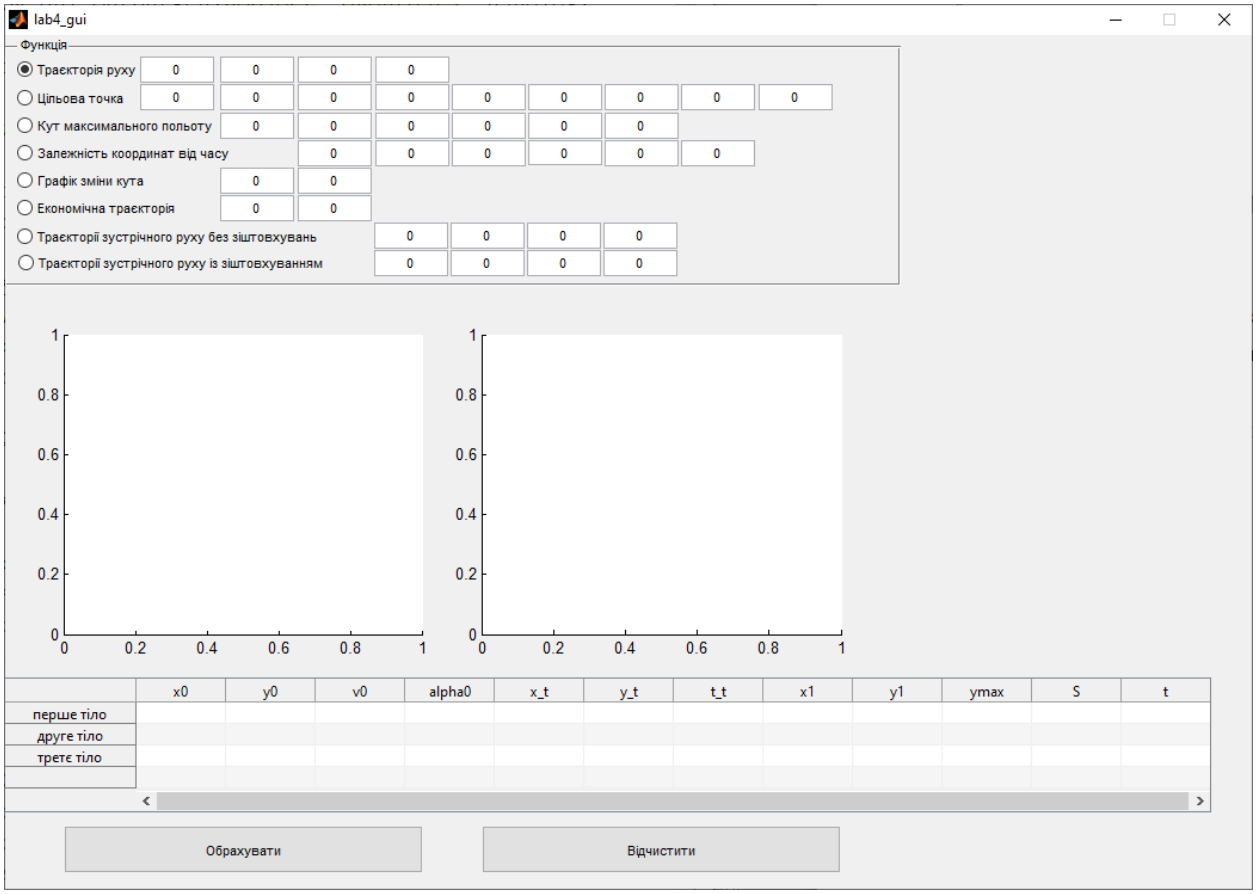
$$a_t = arctg(\frac{{}^1/_2 g t_t^2 + y_t}{x_t})$$

Початкова швидкість при направлені тіла в ціль:

$$v_{0t} = \frac{x_t}{\cos(a_t) t_t}$$

Реалізація на мові MATLAB

1. Інтерфейс користувача



2. Код програми

```
function table_fill(bodys,handles)
    Data = [zeros(1,12);zeros(1,12);zeros(1,12)];
    for i = (1: length(bodys))
        Data(i,:) =
[bodys(i).x0,bodys(i).y0,bodys(i).v0,bodys(i).alpha
,bodys(i).x_t,bodys(i).y_t,bodys(i).t_t,bodys(i).x1
,bodys(i).y1,bodys(i).ymax,bodys(i).S,bodys(i).t];
    end
    set(handles.uitable2,'Data',Data)

```

```
    function new_bodys =
move_in_target_econom(bodys)
        G = 9.8;
        for i = (1: length(bodys))
            bodys(i).a =
atan((bodys(i).y_t+0.5*G*1^2)/bodys(i).x_t);
            bodys(i).alpha = bodys(i).a*180/pi;
            bodys(i).v0 =
bodys(i).x_t/cos(bodys(i).a)*1;
            bodys(i).t =
(bodys(i).v0*sin(bodys(i).a)+(bodys(i).v0^2*sin(bod
ys(i).a)^2+2*G*bodys(i).y0)^0.5)/G;
            bodys(i).x1 =
bodys(i).x0+bodys(i).v0*cos(bodys(i).a)*bodys(i).t;
            bodys(i).ymax =
bodys(i).y0+bodys(i).v0*sin(bodys(i).a)*(bodys(i).t
/2) - 0.5 * G * (bodys(i).t/2)^2;
            bodys(i).ts =
[0:0.1:bodys(i).t,bodys(i).t];
            bodys(i).xs =
bodys(i).x0+bodys(i).v0*cos(bodys(i).a)*bodys(i).ts
;
            bodys(i).ys =
bodys(i).y0+bodys(i).v0*sin(bodys(i).a)*bodys(i).ts
- 0.5 * G * bodys(i).ts.^2;
            bodys(i).S = abs(bodys(i).x1-
bodys(i).x0);
        end
        new_bodys = bodys;

    function new_bodys = move_in_target(bodys)
        G = 9.8;
        for i = (1: length(bodys))
            bodys(i).a =
atan((bodys(i).y_t+0.5*G*bodys(i).t_t^2)/bodys(i).x
_t);
            bodys(i).alpha = bodys(i).a*180/pi;
            bodys(i).v0 =
bodys(i).x_t/cos(bodys(i).a)*bodys(i).t_t;
            bodys(i).t =
(bodys(i).v0*sin(bodys(i).a)+(bodys(i).v0^2*sin(bod
ys(i).a)^2+2*G*bodys(i).y0)^0.5)/G;
            bodys(i).x1 =
bodys(i).x0+bodys(i).v0*cos(bodys(i).a)*bodys(i).t;
```

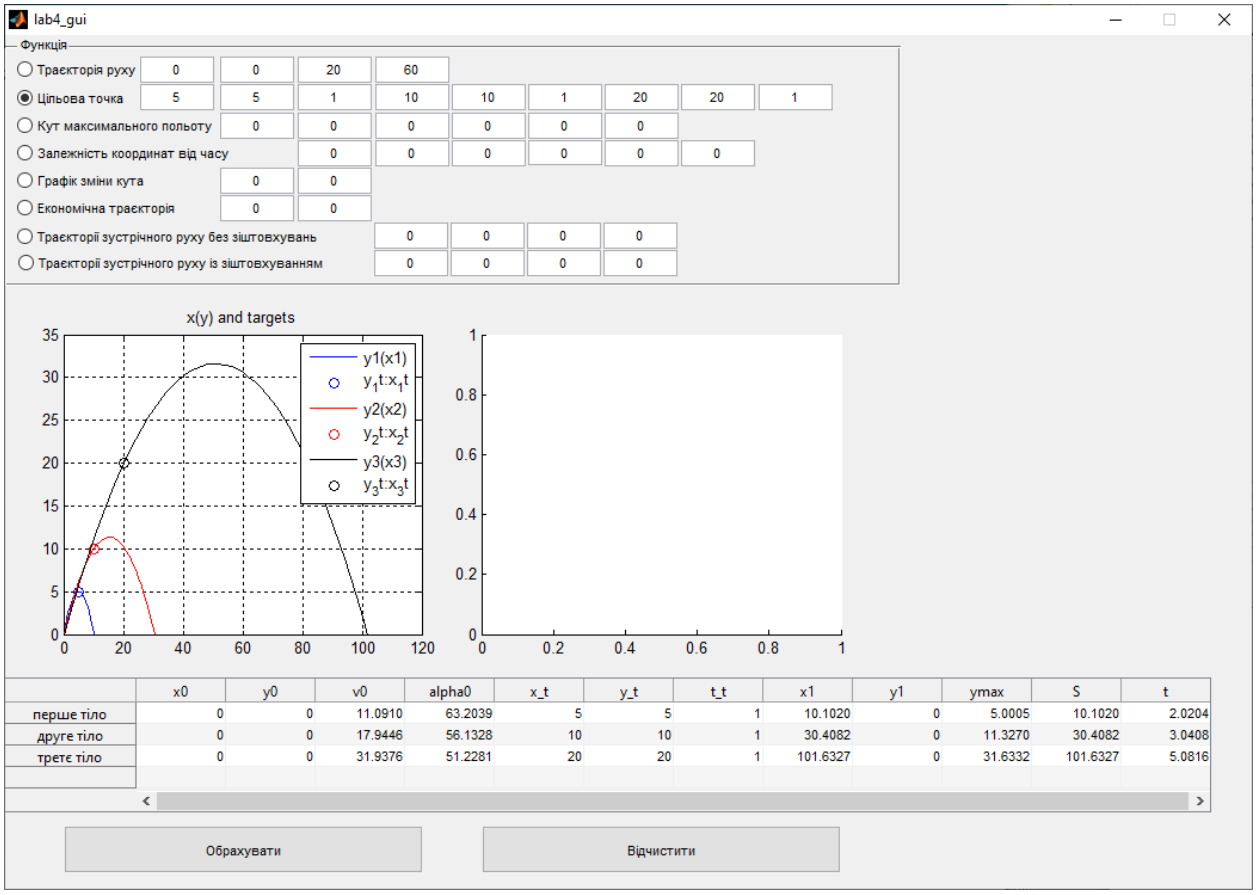
```
34.             bodys(i).ymax =
  bodys(i).y0+bodys(i).v0*sin(bodys(i).a)*(bodys(i).t
  /2) - 0.5 * G * (bodys(i).t/2)^2;
35.             bodys(i).ts =
  [0:0.1:bodys(i).t,bodys(i).t];
36.             bodys(i).xs =
  bodys(i).x0+bodys(i).v0*cos(bodys(i).a)*bodys(i).ts
  ;
37.             bodys(i).ys =
  bodys(i).y0+bodys(i).v0*sin(bodys(i).a)*bodys(i).ts
  - 0.5 * G * bodys(i).ts.^2;
38.             bodys(i).S = abs(bodys(i).x1-
  bodys(i).x0);
39.         end
40.         new_bodys = bodys;
41.
42.     % --- Executes on button press in pushbutton1.
43.     function pushbutton1_Callback(hObject,
  eventdata, handles)
44.         body =
  struct('x0',0,'y0',0,'alpha',0,'a',0,'v0',0,'x_t',0
  ,'y_t',0,'t_t',0,'x1',0,'y1',0,'ymax',0,'S',0,'t',0
  ,'ts',[],'xs',[],'ys',[]);
45.         bodys = [body; body; body];
46.
47.         if get(handles.radiobutton2,'Value')
48.             bodys(1).x_t =
  str2double(get(handles.edit5,'String'));
49.             bodys(1).y_t =
  str2double(get(handles.edit6,'String'));
50.             bodys(1).t_t =
  str2double(get(handles.edit7,'String'));
51.
52.             bodys(2).x_t =
  str2double(get(handles.edit8,'String'));
53.             bodys(2).y_t =
  str2double(get(handles.edit9,'String'));
54.             bodys(2).t_t =
  str2double(get(handles.edit10,'String'));
55.
56.             bodys(3).x_t =
  str2double(get(handles.edit11,'String'));
57.             bodys(3).y_t =
  str2double(get(handles.edit12,'String'));
58.             bodys(3).t_t =
  str2double(get(handles.edit13,'String'));
```

```
            bodys = move_in_target(bodys);
            table_fill(bodys,handles)

            hold(handles.axes1,'off')
            cla(handles.axes2,'reset')
            plot(handles.axes1,
  bodys(1).xs,bodys(1).ys)

            grid(handles.axes1, 'on')
            hold(handles.axes1,'on')
            plot(handles.axes1,
  bodys(1).x_t,bodys(1).y_t,'o')

            plot(handles.axes1,
  bodys(2).xs,bodys(2).ys,'red')
            plot(handles.axes1,
  bodys(2).x_t,bodys(2).y_t,'or')

            plot(handles.axes1,
  bodys(3).xs,bodys(3).ys,'black')
            plot(handles.axes1,
  bodys(3).x_t,bodys(3).y_t,'oblack')

            title(handles.axes1,'x(y) and targets')

  legend(handles.axes1,{'y1(x1)','y_1t:x_1t','y2(x2)'
  ,'y_2t:x_2t','y3(x3)','y_3t:x_3t'});
            grid(handles.axes1, 'on')
        end
        if get(handles.radiobutton6,'Value')
            bodys(1).x_t =
  str2double(get(handles.edit28,'String'));
            bodys(1).y_t =
  str2double(get(handles.edit29,'String'));

            bodys = move_in_target_econom(bodys);
            table_fill(bodys,handles)

            cla(handles.axes2,'reset')
            hold(handles.axes1,'off')
            plot(handles.axes1,
  bodys(1).xs,bodys(1).ys)
            grid(handles.axes1, 'on')
            hold(handles.axes1,'on')
            plot(handles.axes1,
  bodys(1).x_t,bodys(1).y_t,'o')
```

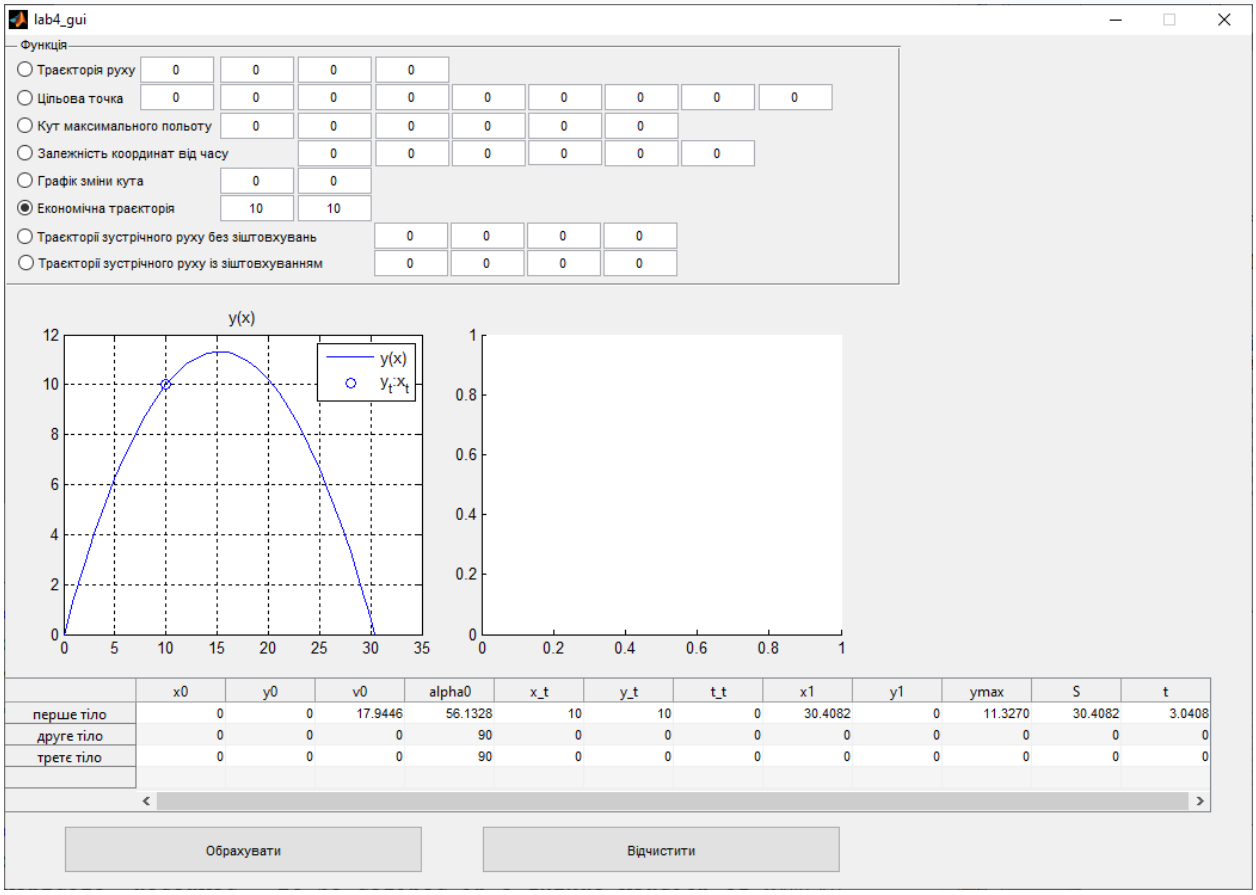
```
93.             title(handles.axes1,'y(x)')
94.
  legend(handles.axes1,{'y(x)','y_t:x_t'});
95.         end
```

3. Завдання

3.1. Побудувати графіки декількох траєкторій руху тіла, що відповідають декільком значенням часу досягнення мети і цільову точку. Як параметри задати координати цілі (X1, У1) і час досягнення мети, початком руху вважати точку (0,0).

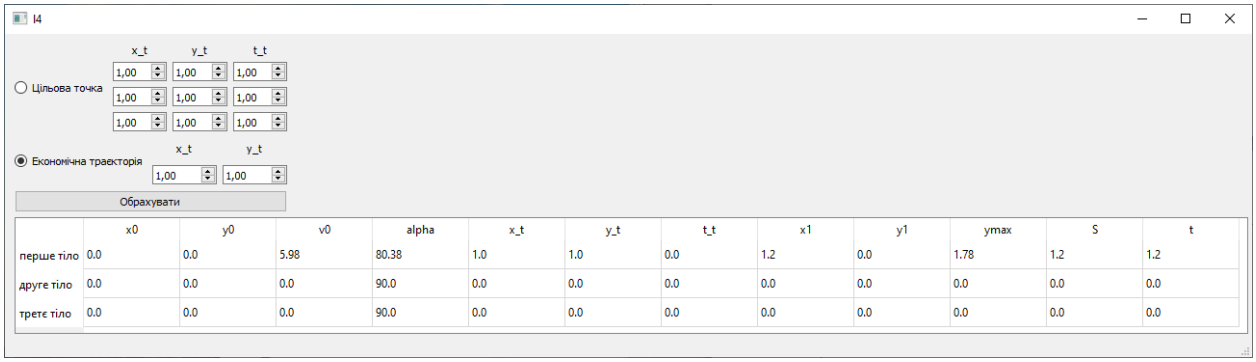


3.2. Побудувати графік економічною траєкторії досягнення тілом цільової точки і цільову точку. Як параметри задати координати цілі (x1, y1), початком руху вважати точку (0,0). Траєкторія вважається економною, якщо мета досягається за мінімальної початковою швидкістю.



Реалізація на мові Python

1. Інтерфейс користувача



2. Код програми

```
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import *
from math import *
```

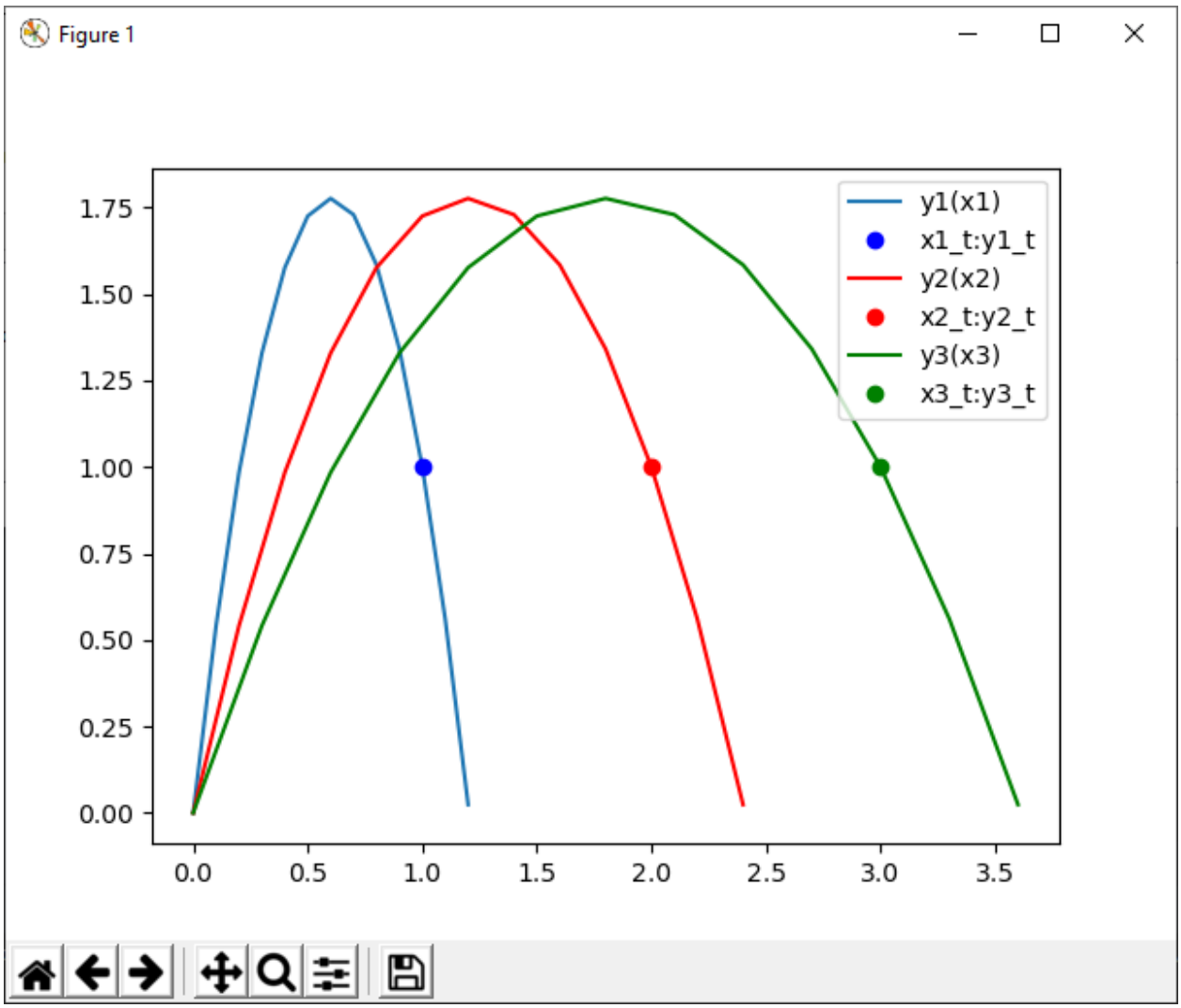
```
from matplotlib import pyplot as plt
from numpy import *
from IM_L4_ui import Ui_MainWindow
import sys

class body:
    x0 = 0
    y0 = 0
    alpha = 0
    a = 0
    v0 = 0
    x_t = 0
    y_t = 0
    t_t = 0
    x1 = 0
    y1 = 0
    ymax = 0
    S = 0
    t = 0
    ts = []
    xs = []
    ys = []

    def move_to_target(self):
        G = 9.8
        self.a = atan2(self.y_t + 0.5 * G * self.t_t ** 2, self.x_t)
        self.alpha = self.a * 180 / pi
        self.v0 = self.x_t / cos(self.a) * self.t_t
        self.t = (self.v0 * sin(self.a) + sqrt(self.v0 ** 2 * (sin(self.a) **
2) + 2 * G * self.y0)) / G
        self.x1 = self.x0 + self.v0 * cos(self.a) * self.t
        self.ymax = self.y0 + self.v0 * sin(self.a) * (self.t / 2) - 0.5 * G *
(self.t / 2) ** 2
        self.ts = arange(0, self.t, 0.1)
        self.xs = self.x0 + self.v0 * cos(self.a) * self.ts
        self.ys = self.y0 + self.v0 * sin(self.a) * self.ts - 0.5 * G *
self.ts ** 2
        self.S = abs(self.x1 - self.x0)

    def move_in_target_econom(self):
        G = 9.8
        self.a = atan2(self.y_t + 0.5 * G, self.x_t)
        self.alpha = self.a * 180 / pi
        self.v0 = self.x_t / cos(self.a)
        self.t = (self.v0 * sin(self.a) + sqrt(self.v0 ** 2 * (sin(self.a) **
2) + 2 * G * self.y0)) / G
        self.x1 = self.x0 + self.v0 * cos(self.a) * self.t
        self.ymax = self.y0 + self.v0 * sin(self.a) * (self.t / 2) - 0.5 * G *
(self.t / 2) ** 2
        self.ts = arange(0, self.t, 0.1)
        self.xs = self.x0 + self.v0 * cos(self.a) * self.ts
        self.ys = self.y0 + self.v0 * sin(self.a) * self.ts - 0.5 * G *
self.ts ** 2
        self.S = abs(self.x1 - self.x0)

class MyWindow(QMainWindow, Ui_MainWindow):

    def __init__(self):
        QMainWindow.__init__(self)
        self.setupUi(self)
```

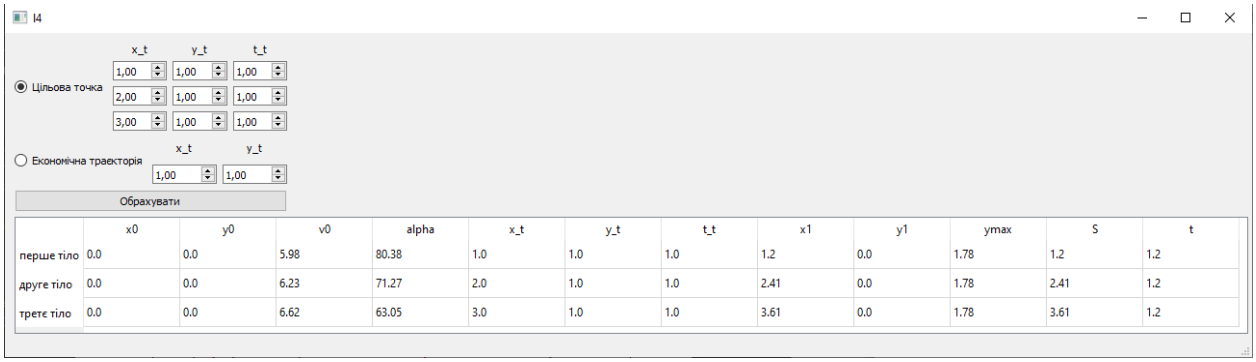
```
        self.setWindowTitle("l4")
        self.radioButton.setChecked(True)
        self.pushButton.clicked.connect(self.Compute_all)

    def fill_table(self, bodys):
        for idx in range(3):
            self.tableWidget.setItem(idx, 0,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].x0)))))
            self.tableWidget.setItem(idx, 1,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].y0)))))
            self.tableWidget.setItem(idx, 2,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].v0)))))
            self.tableWidget.setItem(idx, 3,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].alpha)))))
            self.tableWidget.setItem(idx, 4,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].x_t)))))
            self.tableWidget.setItem(idx, 5,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].y_t)))))
            self.tableWidget.setItem(idx, 6,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].t_t)))))
            self.tableWidget.setItem(idx, 7,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].x1)))))
            self.tableWidget.setItem(idx, 8,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].y1)))))
            self.tableWidget.setItem(idx, 9,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].ymax)))))
            self.tableWidget.setItem(idx, 10,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].S)))))
            self.tableWidget.setItem(idx, 11,
QTableWidgetItem(str(float("{0:.2f}".format(bodys[idx].t)))))

    def Compute_all(self):
        bodys = [body(), body(), body()]

        if self.radioButton.isChecked():
            bodys[0].x_t = float(self.doubleSpinBox.text().replace(',', '.'))
            bodys[1].x_t = float(self.doubleSpinBox_4.text().replace(',',
'.'))
            bodys[2].x_t = float(self.doubleSpinBox_7.text().replace(',',
'.'))

            bodys[0].y_t = float(self.doubleSpinBox_2.text().replace(',',
'.'))
            bodys[1].y_t = float(self.doubleSpinBox_5.text().replace(',',
'.'))
            bodys[2].y_t = float(self.doubleSpinBox_8.text().replace(',',
'.'))

            bodys[0].t_t = float(self.doubleSpinBox_3.text().replace(',',
'.'))
            bodys[1].t_t = float(self.doubleSpinBox_6.text().replace(',',
'.'))
            bodys[2].t_t = float(self.doubleSpinBox_9.text().replace(',',
'.'))
            for current_body in bodys:
                current_body.move_to_target()
            self.fill_table(bodys)

            fig, ax = plt.subplots()
            ax.plot(bodys[0].xs, bodys[0].ys, label='y1(x1)')
            ax.plot(bodys[0].x_t, bodys[0].y_t, 'bo', label='x1_t:y1_t')
            ax.plot(bodys[1].xs, bodys[1].ys, 'r', label='y2(x2)')
```

```
            ax.plot(bodys[1].x_t, bodys[1].y_t, 'ro', label='x2_t:y2_t')
            ax.plot(bodys[2].xs, bodys[2].ys, 'g', label='y3(x3)')
            ax.plot(bodys[2].x_t, bodys[2].y_t, 'go', label='x3_t:y3_t')
            leg = ax.legend()
            plt.show()

        if self.radioButton_2.isChecked():
            bodys[0].x_t = float(self.doubleSpinBox_10.text().replace(',',
'.'))
            bodys[0].y_t = float(self.doubleSpinBox_11.text().replace(',',
'.'))
            for current_body in bodys:
                current_body.move_in_target_econom()

            self.fill_table(bodys)

            fig, ax = plt.subplots()
            ax.plot(bodys[0].xs, bodys[0].ys, label='y(x)')
            ax.plot(bodys[0].x_t, bodys[0].y_t, 'bo', label='x_t:y_t')
            leg = ax.legend()
            plt.show()


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    my_app = MyWindow()
    my_app.show()
    sys.exit(app.exec_())
```

3. Завдання

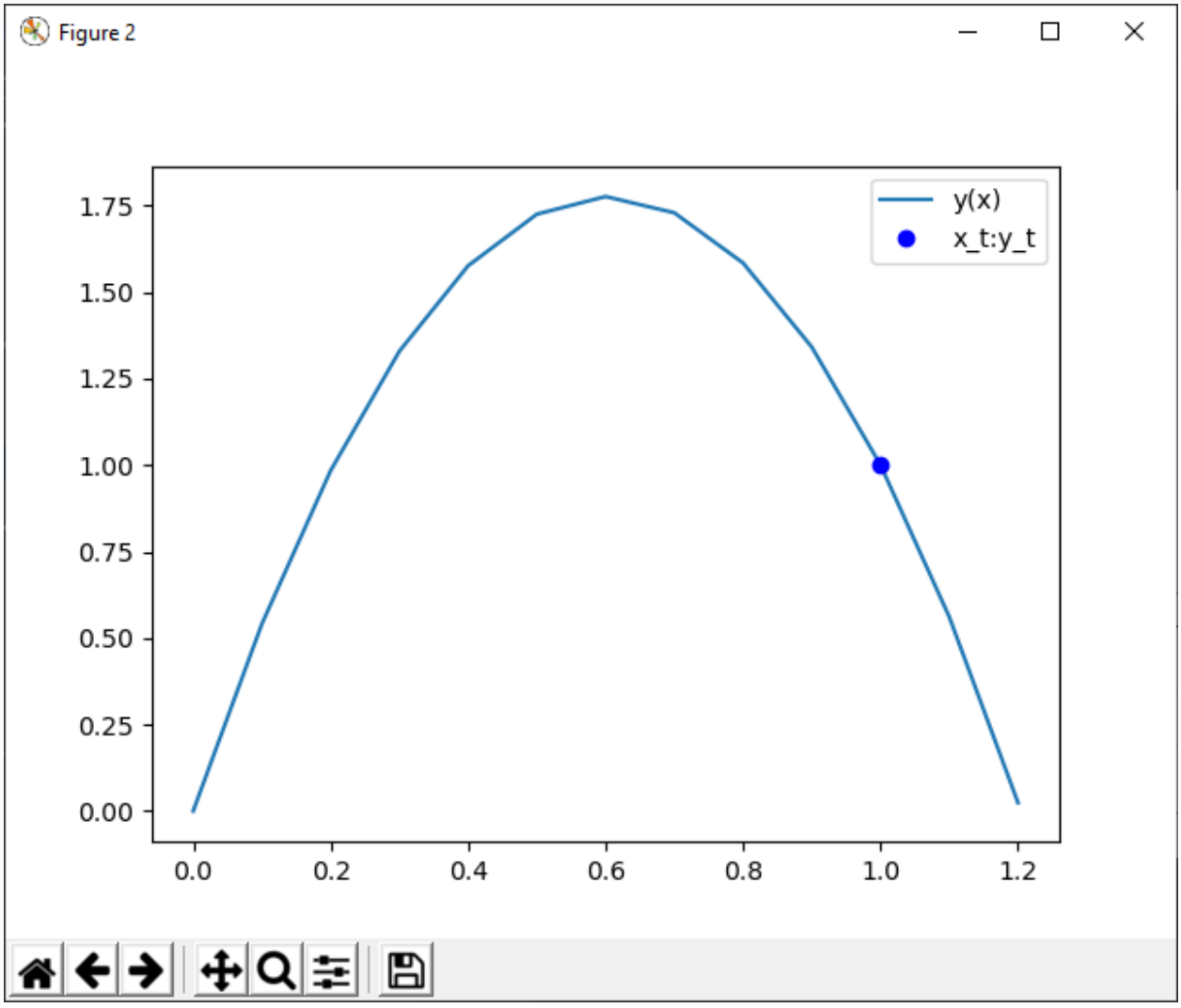
3.1. Побудувати графіки декількох траєкторій руху тіла, що відповідають декільком значенням часу досягнення мети і цільову точку. Як параметри задати координати цілі (X1, У1) і час досягнення мети, початком руху вважати точку (0,0).

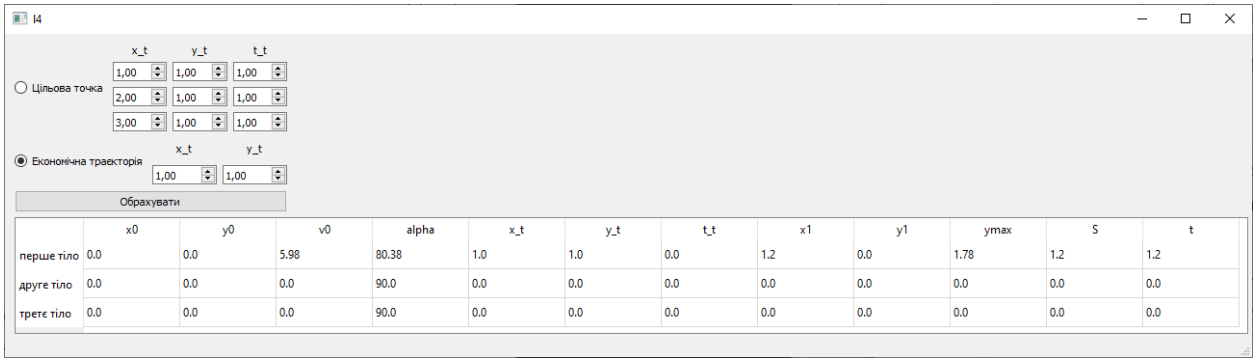


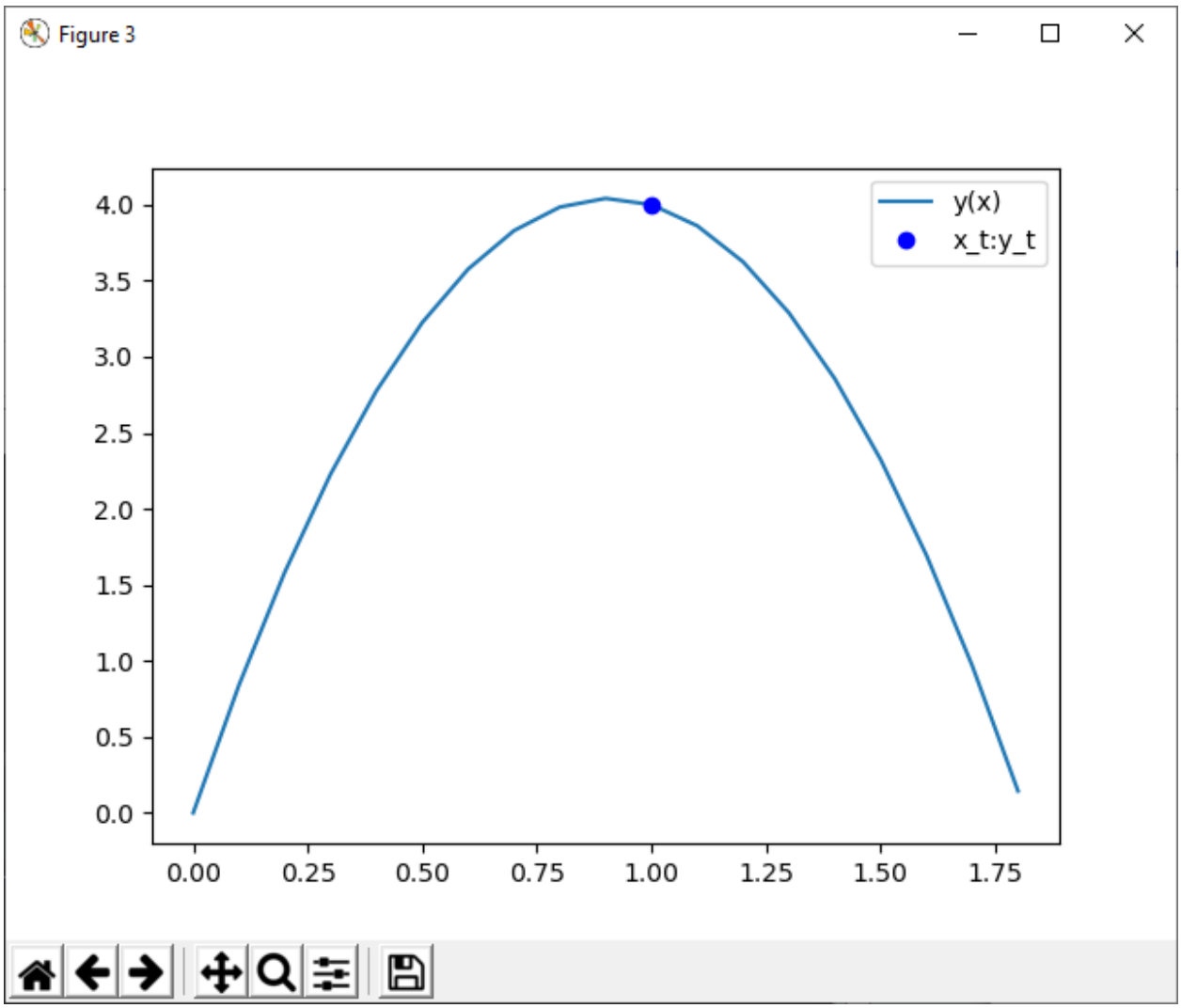


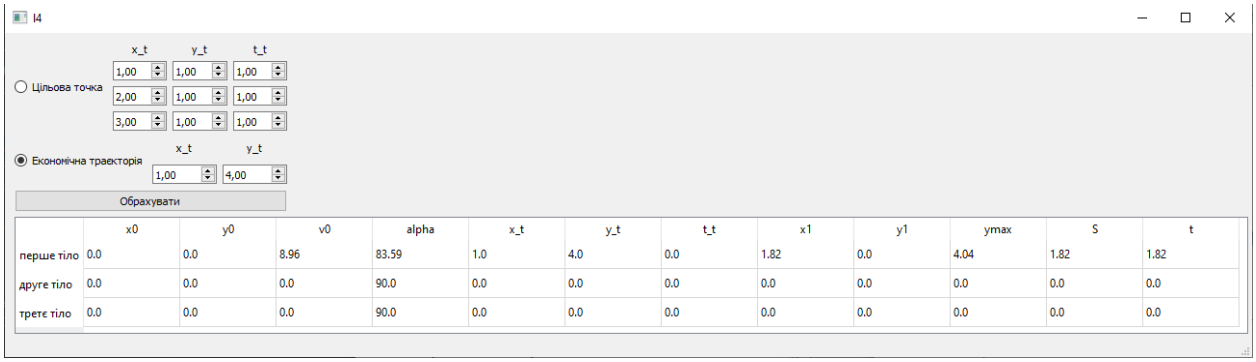
	x0	y0	v0	alpha	x_t	y_t	t_t	x1	y1	ymax	S	t
перше тіло	0.0	0.0	5.98	80.38	1.0	1.0	1.0	1.2	0.0	1.78	1.2	1.2
друге тіло	0.0	0.0	6.23	71.27	2.0	1.0	1.0	2.41	0.0	1.78	2.41	1.2
третє тіло	0.0	0.0	6.62	63.05	3.0	1.0	1.0	3.61	0.0	1.78	3.61	1.2

3.2. Побудувати графік економічною траєкторії досягнення тілом цільової точки і цільову точку. Як параметри задати координати цілі (x1, y1), початком руху вважати точку (0,0). Траєкторія вважається економною, якщо мета досягається за мінімальної початковою швидкістю.









Висновок: В ході виконання роботи ми вивчили основи роботи з інтерфейсом користувача системи Matlab, придбали практичні навички програмування на мові системи Matlab, отримали навички розробки та реалізації програм на основі математичної моделі об'єкта на прикладі розробки комп'ютерної моделі руху тіла.