МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ А.С. ПУШКИНА»

Физико-математический факультет

Кафедра общей и теоретической физики

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ В МЕХАНИКЕ

Курсовая работа по специализации «Компьютерное моделирование физических процессов» специальности 1-31 04 08 Компьютерная физика

Выполнил студент 3 курса гр. КФ-3

Дорофейчик А.Р. _______________

(подпись)

Научный руководитель

кандидат физ.-мат. наук, доцент

Плетюхов В.А _______________

(подпись)

Брест 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разработанные в курсовой работе проекты решения физических задач написаны на языке MATLAB и отлажены в среде MATLAB.

MATLAB – это высокоуровневый язык технических расчетов, интерактивная среда разработки алгоритмов и современный инструмент анализа данных. MATLAB по сравнению с традиционными языками программирования (C/C++, Java, Pascal, FORTRAN) позволяет на порядок сократить время решения типовых задач и значительно упрощает разработку новых алгоритмов. MATLAB представляет собой основу всего семейства продуктов MathWorks и является главным инструментом для решения широкого спектра научных и прикладных задач, в таких областях как: моделирование объектов и разработка систем управления, проектирование коммуникационных систем, обработка сигналов и изображений, измерение сигналов и тестирование, финансовое моделирование, вычислительная биология и др.

Ядро MATLAB позволяет максимально просто работать с матрицами реальных, комплексных и аналитических типов данных. Содержит встроенные функции линейной алгебры (LAPACK, BLAS), быстрого Фурье преобразования (FFTW), функции для работы с полиномами, функции базовой статистики и численного решения дифференциальных уравнений. Все встроенные функции ядра MATLAB разработаны и оптимизированы специалистами и работают быстрее или так же, как их эквивалент на C/C++.

Ключевые возможности:

- Платформонезависимый, высокоуровневый язык программирования ориентированный на матричные вычисления и разработку алгоритмов
- Интерактивная среда для разработки кода, управления файлами и данными
- Функции линейной алгебры, статистики, анализ Фурье, решение дифференциальных уравнений и др.
- Богатые средства визуализации, 2-D и 3-D графика
- Встроенные средства разработки пользовательского интерфейса для создания законченных приложений на MATLAB
- Средства интеграции с C/C++, наследование кода, ActiveX технологии
- Удобное моделирование в среде Simulink
- И многое другое

В данной работе рассматривается использование возможностей средств визуального программирования GUIDE для создания графического интерфейса пользователя GUI(Graphic User Interface) с помощью инструментальных средств для визуально-ориентированного программирования и проектирования приложений с GUI.

В работе в качестве предметной области для компьютерного моделирования выбраны физические задачи. Рассматриваются особенности технологии математического моделирования на компьютере. Рассмотрены вопросы постановки математических задач на базе фундаментальных физических законов.

В связи с вышесказанным целью курсовой работы является разработка решения физических задач на языке MATLAB с применением возможности создания графического интерфейса пользователя GUI.

Для достижения цели необходимо решить следующие задачи:

-Изучить учебно-методическую и научную литературу по теме работы.

-Изучить основы работы с средой MATLAB, основы создания графического интерфейса пользователя GUI.

-Изучить математические модели физических задач.

-Разработать алгоритмы решения и их реализацию на языке MATLAB в среде MATLAB.

-Изучить особенности визуализации результатов моделирования физических задач с использованием инструмента PLOT Tool

В работе рассмотрены задачи: моделирование движения тела, брошенного под углом к горизонту; моделирование движения математического маятника, численного моделирования орбиты (Задача Кеплера).

Курсовая работа состоит из введения, двух глав, заключения и списка литературы. Первая глава включает в себя основы работы с MATLAB основы работы с приложением GUIDE входящим в сотав среды компьютерного моделирования MATLAB. Вторая глава включает в себя описание математических моделей физических задач, методов их решения, а также кодов программ, реализующих рения на языке MATLAB. Также, во второй главе показаны результаты решения в виде рисунков с графиками, и скриншотов программ.

ГЛАВА 1. Основы работы в среде MATLAB

1.1 Основы MATLAB

Интерпретирующий язык программирования системы MATLAB создан таким образом, что любые (подчас весьма сложные) вычисления можно выполнять в режиме прямых вычислений, то есть без подготовки программы пользователем. При этом MATLAB выполняет функции суперкалькулятора и работает в режиме командной строки. Работа с системой носит диалоговый характер и происходит по правилу «задал вопрос – получил ответ». Пользователь набирает на клавиатуре вычисляемое выражение, редактирует его (если нужно) в командной строке и завершает ввод нажатием клавиши ENTER. Опишем основные моменты работы с программой:

– для указания ввода исходных данных используется символ >>;
– данные вводятся с помощью простейшего строчного редактора;
– для блокировки вывода результата вычислений некоторого выражения после него надо установить знак ; (точка с запятой);
– если не указана переменная для значения результата вычислений, то MATLAB назначает такую переменную с именем ans;
– знаком присваивания является привычный математикам знак равенства =, а не комбинированный знак :=, как во многих других языках программирования и математических системах;
– встроенные функции (например, sin) записываются строчными буквами, и их аргументы указываются в круглых скобках;
– результат вычислений выводится в строках вывода (без знака >>);
– текстовые комментарии в программах вводятся с помощью символа %, например так: % It is factorial function.
– производится синтаксический контроль программного кода по мере ввода текста. При этом используются несколько цветовых выделений (ключевые слова языка программирования – синий цвет; операторы, константы и переменные – черный цвет; комментарии после знака % – зеленый цвет; символьные переменные (в апострофах) – зеленый цвет; синтаксические ошибки – красный цвет)

Для наглядности введём в окно Comman Window тригонометрическую функцию для вычисления арккосинуса угла заданного в радианах. Для справки:

acos(X) – возвращает арккосинус для каждого элемента X. Для действительных значений X в области [–1,1] acos(X) возвращает действительное значение из диапазона [0,pi], для действительных значений X вне области [–1, 1] acos(X) возвращает комплексное число.

Пример:

\>>Y = acos (0.5)

Y = 1.0472

Отображение результата выполненной команды в программе на рисунке 1.1.1.

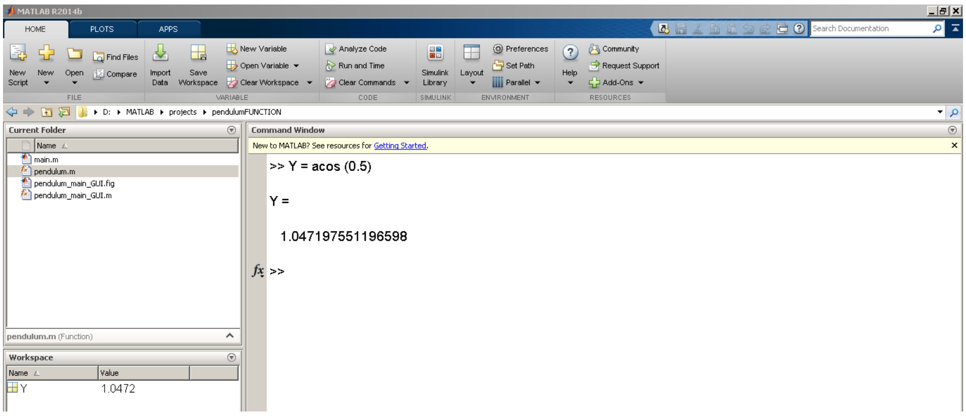

Рисунок 1.1.1 – Выполнение обратной тригонометрической функции

Программы в системе MATLAB представлены m-файлами. Для подготовки, редактирования и отладки m-файлов служит специальный многооконный редактор (рис. 1.1.2). Он выполнен как типичное приложение Windows. Редактор можно вызвать командой edit из командной строки или командой New -> M-file из меню File. После этого в окне редактора можно создавать свой файл, пользоваться средствами его отладки и запуска. Перед запуском файла его необходимо записать на диск, используя команду File -> Save as в меню редактора.

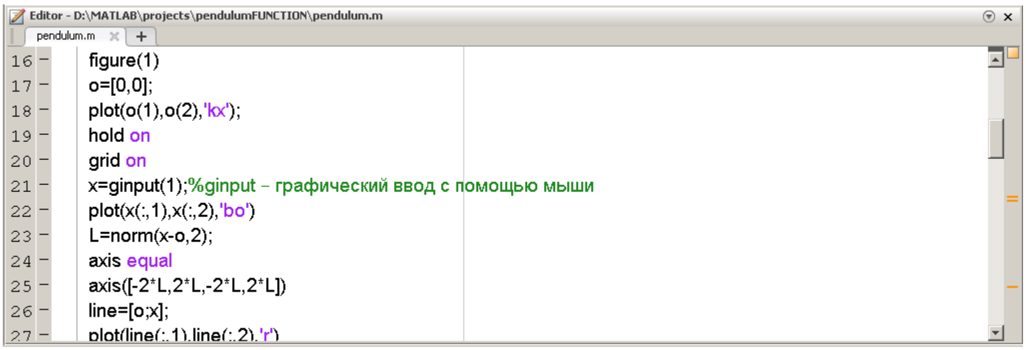

Рисунок 1.1.2 – Многооконный редактор **Editor**

После записи файла на диск можно заметить, что команда **Run** в меню **Tools**(Инструменты) редактора становится активной (до записи файла на диск она пассивна) и позволяет произвести запуск файла. Запустив команду **Run**, можно наблюдать исполнение m-файла – в нашем случае это моделирование математического маятника и построение рисунка с графиком затухающих колебаний маятника в графическом окне. Для удобства работы с редактором/отладчиком строки программы в нем нумеруются в последовательном порядке. Редактор является многооконным. Окно каждой программы оформляется как вкладка. Редактор-отладчик позволяет легко просматривать значения переменных. Для этого достаточно подвести к имени переменной курсор мыши и задержать его – появится всплывающая подсказка с именем переменной и ее значением.

Также при написании программного кода были задействованы М-файл-функции. Он является типичным полноценным объектом языка программирования системы MATLAB. Одновременно он является полноценным модулем с точки зрения структурного программирования, поскольку содержит входные и выходные параметры и использует аппарат локальных переменных. Структура такого модуля с одним выходным параметром выглядит следующим образом:

function var=f_name(Список_параметров)
%Основной каментарий
%Дополнительный каментарий
Тело файла с любыми выражениями
var=выражение

М-файл-функция имеет следующие свойства:

– он начинается с объявления function, после которого указываются имя переменной var – выходного параметра, имя самой функции и список ее

– входных параметров;

– функция возвращает свое значение и может использоваться в виде name(Список_параметров) в математических выражениях;

– все переменные, имеющиеся в теле файла-функции, являются локальными, то есть действуют только в пределах тела функции;

– файл-функция является самостоятельным программным модулем, который общается с другими модулями через свои входные и выходные параметры;

– правила вывода комментариев те же, что у файлов-сценариев;

– файл-функция служит средством расширения системы MATLAB;

– при обнаружении файла-функции он компилируется и затем исполняется, а созданные машинные коды хранятся в рабочей области системы MATLAB.

Последняя конструкция var=выражение вводится, если требуется, чтобы функция возвращала результат вычислений. Приведенная форма файла-функции характерна для функции с одним выходным параметром. Если выходных параметров больше, то они указываются в квадратных скобках после слова function. При этом структура модуля имеет следующий вид:

```
function [var1,var2,…]=f_name(Список_параметров)
%Основной камментарий
%Дополнительный каментарий
Тело файла с любыми выражениями
var1=выражение
var2=выражение
...
```

Такая функция во многом напоминает процедуру. Ее нельзя слепо использовать непосредственно в математических выражениях, поскольку она возвращает не единственный результат, а множество результатов – по числу выходных параметров. Если функция используется как имеющая единственный выходной параметр, но имеет ряд выходных параметров, то для возврата значения будет использоваться первый из них. Это зачастую ведет к ошибкам в математических вычислениях.

Также был использован условный оператор **if** , в общем виде записывается следующим образом:

```
if Условие
    Инструкция_1
        elseif Условие
    Инструкция_2
    else
        Инструкция_3
end
```

Пока условие возвращает логическое значение 1(то есть "истина"), выполняються инструкции, составляющие тело структуры if ..end. При этом оператор end указывает на конец перечня инструкций. Инструкции в списке разделяются оператором , (запятая) или ; (точка с запятой). Если Условие не выполняется (даёт логическое значение 0, то есть "ложь"), то Инструкции также не выполняются.

Причём в качестве операторов_отношения используются следующие операторы: ==, <, >, <=, >= или ~= .Все эти операторы представляют собой пары символов без пробелов между ними.

Циклы типа for…end обычно используются для организации вычислений с за данным числом повторяющихся циклов. Конструкция такого цикла имеет следующий вид:

for var=Выражение, Инструкция, ..., Инструкция end

"Выражение" чаще всего записывается в виде s:d:e, где s – начальное значение переменной цикла var, d – приращение этой переменной и e – конечное значение управляющей переменной, при достижении которого цикл завершается. Возможна и запись в виде s:e (в этом случае d=1). Список выполняемых в цикле инструкций завершается оператором end. Следующие примеры поясняют применение цикла для получения квадратов значений переменной цикла:

```
>> for i=1:5 i^2, end;
ans = 1
ans = 4
ans = 9555
ans = 16
ans = 25
```

Также в ходе курсовой работы был использован цикл типа while выполняется до тех пор, пока выполняется Условие:

```
while Условие
        Инструкции
end
```

Досрочное завершение циклов реализуется с помощью операторов break или continue. Цикл while часто используется для подготовки итерационных программ, завершающих свою работу по какому-либо условию.

1.2 Средства визуального программирования GUIDE

Главным средством в визуально-ориентированном проектировании GUI является приложение GUIDE (GUI Designer). При работе с инструментом GUIDE можно создавать окна GUI путем выбора мышью нужных элементов управления и перемещения их в окно GUI. Таким образом, могут создаваться различные элементы интерфейса, например кнопки, раскрывающиеся списки, линейки прокрутки и т. д. При этом возможно программирование событий, которые возникают при обращении пользователя к заданным элементам управления. Визуальное программирование совмещается с объектно-ориентированным, в частности в первом широко используется свойство наследования признаков родительских объектов их потомками.

Приложение с GUI может состоять из одного окна (основного) или нескольких окон и осуществлять вывод графической и текстовой информации как в основное окно приложения, так и в отдельные окна. MATLAB имеет ряд функций создания стандартных диалоговых окон для открытия и сохранения файлов, печати, выбора шрифта для текстовых объектов, создания окон для ввода данных и др. Эти средства (объекты) можно использовать в приложениях пользователя. Для открытия окна инструмента GUIDE надо использовать команду:

```
>> guide
```

При этом инструмент запускается и появляется диалоговое окно GUIDE **Quick Start** (рисунок 1.2.1). Это окно имеет две вкладки:

- **Create New GUI** – создание нового приложения с GUI;
- **Open Existing GUI** – открытие уже существующего приложения с GUI.

На вкладке создания нового приложения **Create New GUI** выбираем **Blank GUI** – пустое окно приложения.

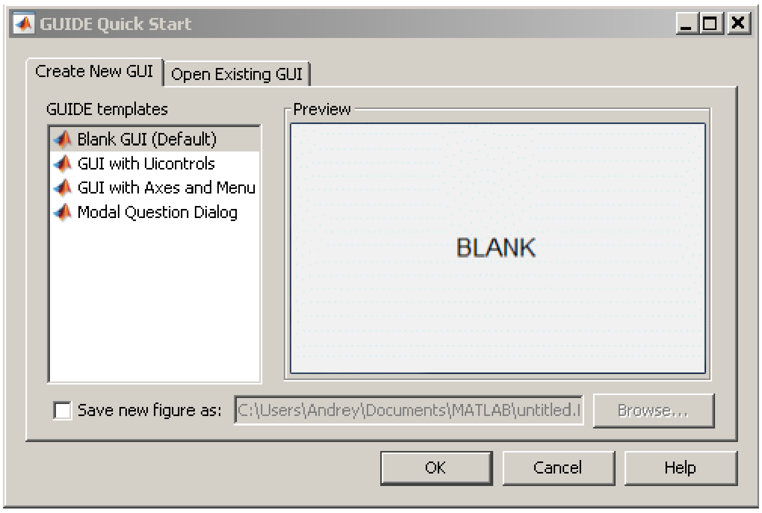

Рисунок 1.2.1. -Диалоговое окно GUIDE Quick Start

Выбрав мышью на вкладке Create New GUI заготовку Blank GUI и нажав клавишу OK, можно наблюдать инициализацию инструмента GUIDE. По окончании инициализации появляется основное окно среды GUIDE, содержащее поле окна приложения, вертикальную панель инструментов для добавления элементов интерфейса, горизонтальную панель инструментов и обычное меню (рис. 1.2.3). С назначением кнопок панелей можно ознакомиться по подсказкам, которые появляются, если установить курсор мыши на нужную кнопку панели и задержать его на пару секунд. Одна из таких подсказок видна на рис. 1.2.3 для кнопки Button Group.

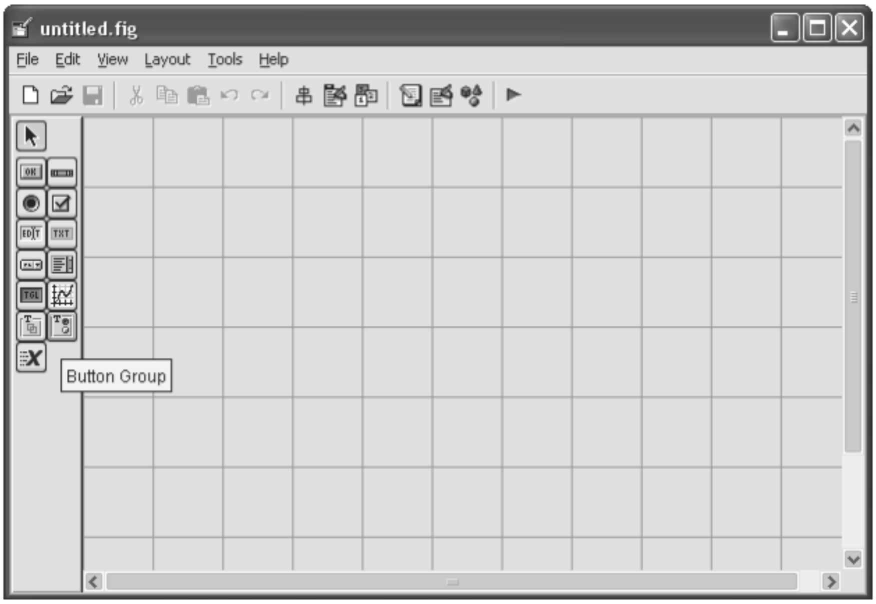

Рисунок 1.2.3 - Пустое окно приложения с GUI

Окно рисунка 1.2.3. имеет титульную строку, меню и две панели инструментов –обычную горизонтальную и вертикальную с набором объектов GUI. Горизонтальная панель инструментов с номерами кнопок показана на рис. 1.2.4. Эта панель позволяет работать с окном приложения с GUI, не обращаясь к его меню.

Начнём с добавления кнопки в заготовку окна приложения. Для этого при помощи мыши выберите кнопку **Push Button** (ее пиктограмма содержит кнопку OK, а имя появляется на всплывающей подсказке) и щелчком мыши поместите кнопку на заготовку окна приложения.

Созданная описанным образом кнопка пока не действует. Чтобы кнопка выполняла какие-либо действия, она должна быть связана с программой обработки событий. Для каждого объекта или совокупности объектов в окне приложения с GUI такая программа, именуемая Callback, создается приложением GUIDE автоматически.

Добавим код программы расчёта траектории движения тела, брошенного под углом к горизонту, основанный на формулах 2.1.1. – 2.1.8. в результате получим:

```
global V alpha
V = str2double(get(handles.edit1,'String'));
alpha = str2double(get(handles.edit2,'String'));
function pushbutton1_Callback(hObject, eventdata, handles)
global V alpha
read_data(handles);
%V=20.0;
%alpha=35;
g=9.8; %гравитационная постоянная
alpha = alpha * pi /180.0; %градусы в радианы
t=0;
```

Рисунок 1.2.4 – Добавление расчётных формул в функцию объекта кнопка

ГЛАВА II. Решение физических проблем методами компьютерного математического моделирования и их визуализация

Математические модели можно разделить на следующие группы:

- детерминированные
- стохастические

Математическая модель называется детерминистической (детерминированной), если все ее параметры и переменные являются однозначно определяемыми величинами, а также выполняется условие полной определенности информации. В противном случае, в условиях неопределенности информации, когда параметры и переменные модели - случайные величины, модель называется стохастической (вероятностной). Отметим, что, говоря о математических моделях, мы имеем в виду сугубо прикладной аспект.

В физике математическое моделирование является чрезвычайно важным методом исследования. Наряду с традиционным делением физики на экспериментальную и теоретическую сегодня уверенно выделяется третий фундаментальный раздел - вычислительная физика. Причину этого в целом можно сформулировать так: при максимальном проникновении в физику математических методов, порой доходящем до фактического сращивания этих наук, реальные возможности решения возникающих математических задач традиционными методами очень ограниченны. Из многих конкретных причин выделим две наиболее часто встречающихся:

- нелинейность многих физических процессов и отсюда нелинейность описывающих их математических моделей
- необходимость исследования совместного движения многих тел, для которого приходится решать системы большого числа уравнений.

Численное моделирование в физике называют вычислительным экспериментом, поскольку оно имеет много общего с лабораторным экспериментом (Таблица 1).

Таблица 1 - Аналогии между лабораторным и вычислительным экспериментами

Лабораторный эксперимент	Вычислительный эксперимент
Образец Физический прибор Калибровка прибора Измерение Анализ данных	Модель Программа для компьютера Тестирование программы Расчёт Анализ данных

Численное моделирование (как и лабораторные эксперименты) чаще всего является инструментом познания качественных закономерностей природы. Важнейшим его этапом является анализ результатов, представление их в максимально наглядной и удобной для восприятия форме. Получение распечатки чисел еще не означает окончания моделирования (даже если эти числа верны). Важно представить результаты в виде графиков, диаграмм, траекторий движения динамических объектов для получения качественной информации. Здесь необходима помощь компьютера – возможность визуализации абстракций.

2.1 Моделирование движения тела брошенного под углом к горизонту

Рассмотрим движение тела брошенного под некоторым углом к горизонту с начальной скоростью v_0 без учёта сопротивления воздуха.

Разложим скорость v_0 на горизонтальную и вертикальную составляющие:

$$v_{0x} = v_0 \cos\alpha; \qquad (2.1.1)$$

$$v_{0y} = v_0 \sin\alpha; \qquad (2.1.2)$$

Движение по вертикали не равномерно. Оно является равнозамедленным до достижения верхней точки на траектории и равноускоренным после неё. Движение по горизонтали является равномерным. Для вертикальной составляющей получаем:

$$v_y = v_{0y} - gt; \qquad (2.1.3)$$

Найдём время достижения верхней точки траектории. Имеем в верхней точке $v_y = 0$. Тогда в верхней точке $v_{0y} = gt$ Отсюда время достижения верхней точки:

$$t = \frac{v_{0y}}{g} = \frac{v_0 \sin\alpha}{g}; \qquad (2.1.4)$$

Для нахождения траектории достаточно из текущих значений *x* и *y* исключить *t:*

$$x = v_{0x} t; \qquad (2.1.5)$$

$$t = \frac{x}{v_{0x}}; \qquad (2.1.6)$$

$$y = y_{oy} t - \frac{gt^2}{2}; \qquad (2.1.7)$$

Подставив (2.1.6) в (2.1.7) получим:

$$y = y_{oy} \frac{x}{v_{0x}} - \frac{g}{2} \frac{x^2}{v_{0x}^2} = tg\alpha \cdot x - \frac{g}{2v_{0x}\cos^2\alpha} x^2 \qquad (2.1.8)$$

Формула (2.1.8.) является уравнением параболы.

Для создания GUI приложения с графическим отображением брошенного тела под углом к горизонту необходимо выполнить следующую последовательность команд, сохранённую нами в файле throwGUI.m

```
function varargout = throwGUI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
               'gui_Singleton',  gui_Singleton, ...
               'gui_OpeningFcn', @throwGUI_OpeningFcn, ...
```

```
                'gui_OutputFcn',  @throwGUI_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
function throwGUI_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

guidata(hObject, handles);

function varargout = throwGUI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function edit1_Callback(hObject, eventdata, handles)
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit2_Callback(hObject, eventdata, handles)
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function  read_data(handles)
global V alpha
V = str2double(get(handles.edit1,'String'));
alpha = str2double(get(handles.edit2,'String'));
function pushbutton1_Callback(hObject, eventdata, handles)
global V alpha
read_data(handles);
%V=20.0;
%alpha=35;
g=9.8; %гравитационная постоянная
alpha = alpha * pi /180.0; %градусы в радианы
t=0;
i=1;
```

```
%расчтаем положение координат тела до достижения максимальной высоты
while (t <= (V*sin(alpha) / g))
        x1(i) = V*cos(alpha)*t;
        y1(i) = V*sin(alpha)*t - g*t*t / 2;
        t =t+ 0.01;
        i=i+1;
end
h=y1(end);
hmax=h;
l=x1(end);
t = 0;
i=1;
%расчтаем положение координат тела после достижения максимальной высоты
while (t <= (V*sin(alpha) / g))
    x2(i) = l + V*cos(alpha)*t;
    y2(i) = h - g*t*t / 2;
    i=i+1;
    t =t+ 0.01;
end
% построим график функций (x1, y1,x2,y2) задав цвет линиям,
% указав параметр 'r' после объявления абсцис и ординат.
plot(x1, y1,'r',x2,y2,'r');
%установим надпись возле оси x
xlabel('[m]')
%установим надпись возле оси y
ylabel('[m]')
```

Результат выполнения приведённой выше последовательности команд представлен на рисунке 2.1.1 - 2.1.3.

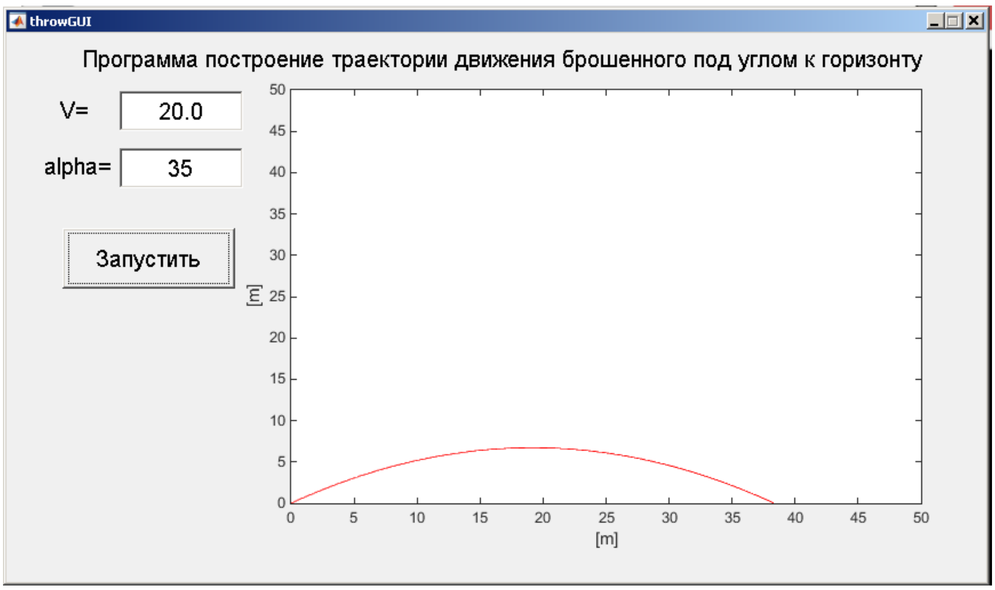

Рисунок 2.1.1. – Траектория движения тела при $v_0 = 20 м/с$ α=35^0

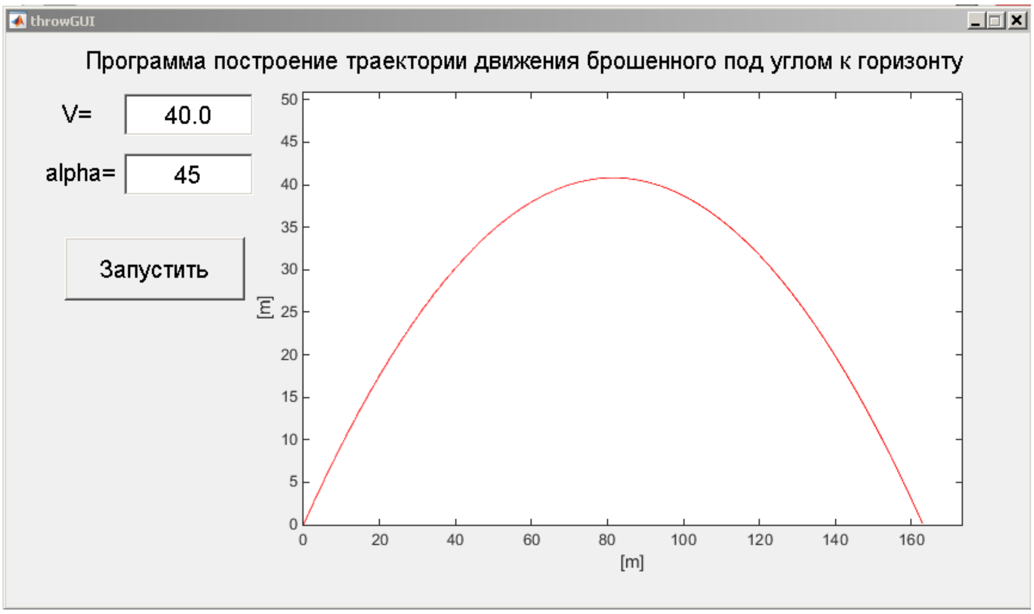

Рисунок 2.1.2. – Траектория движения тела при $v_0 = 40$м/с α=45^0

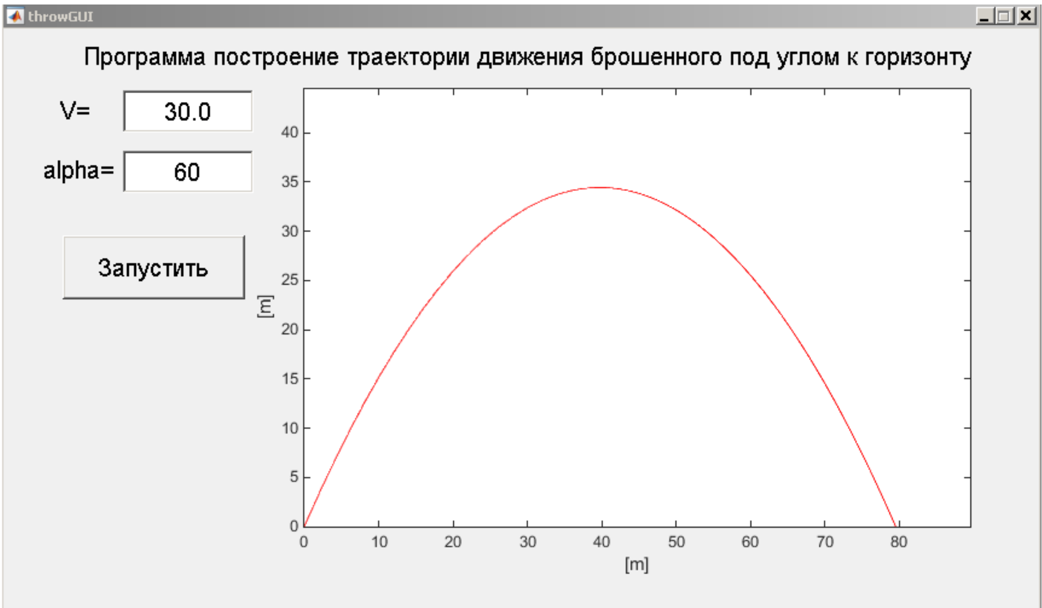

Рисунок 2.1.3. – Траектория движения тела при $v_0 = 30$м/с α=60^0

2.2 Численное моделирование орбиты и уравнения движения планет (Задача Кеплера)

Задача о движении планет в поле тяжести небесных светил, являющаяся частным случаем задачи о движении в поле центральных сил, известна на протяжении нескольких тысячелетий истории человечества и в настоящее время рассматривается как в школьных курсах физики, астрономии, так и в вузовских курсах классической механики и астрономии.

Большую часть наших знаний о движении планет объединили в себе законы Кеплера, полученные на основе анализа данных астрономических наблюдений, которые формулируются следующим образом:

1. Всякая планета движется по эллиптической орбите, в одном из фокусов которой находится Солнце.
2. Скорость планеты возрастает по мере удаления от Солнца таким образом, что прямая, соединяющая Солнце и планету, в равные промежутки времени заметает одинаковую площадь.
3. Для всех планет, вращающихся вокруг Солнца, отношение T^2 / R^3 одинаково (T- период обращения планет вокруг Солнца, R - большая полуось эллипса).

Отметим, что получить аналитическое решение задачи Кеплера удаётся только в случае рассмотрения движения двух тел, взаимодействующих по закону обратных квадратов. Это решение рассматривается во всех учебниках по классической механике. Задача Кеплера для трёх и более тел аналитического решения не имеет, может быть решена только численно, поэтому мы будем решать уравнение движения тела в центральном поле численным методом.

Для начала рассмотрим движение двух тел, взаимодействующих друг с другом, считая их при этом материальными точками (рисунок 2.2.1). Функция Лагранжа такой системы имеет вид:

$$L = \frac{m_1 \dot{\vec{r}}_1^2}{2} + \frac{m_2 \dot{\vec{r}}_2^2}{2} - U(|\vec{r}_1 - \vec{r}_2|) = \frac{m_1 \dot{\vec{r}}_1^2}{2} + \frac{m_2 \dot{\vec{r}}_2^2}{2} + \frac{\gamma m_1 m_2}{|\vec{r}_1 - \vec{r}_2|}, \qquad (2.2.1)$$

где $\vec{r}_1, \vec{r}_2$ - радиусы-векторы первого и второго тела соответственно, $U(|\vec{r}_1 - \vec{r}_2|)$ - потенциальная энергия взаимодействия тел, γ - гравитационная постоянная. Введём вектор взаимного расстояния обоих тел.

$$\overrightarrow{r_{12}} = \overrightarrow{r_1} - \overrightarrow{r_2}. \qquad (2.2.2)$$

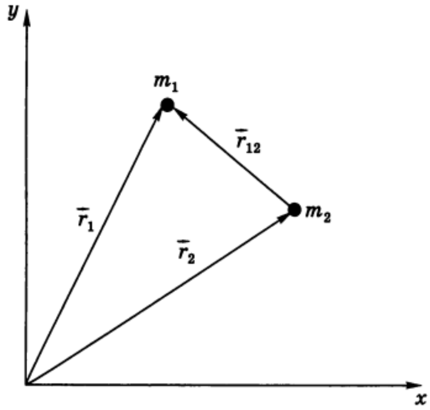

Рис.2.2.1. К решению задачи Кеплера

Тогда в системе отсчёта с началом координат в центре масс, рассматриваемой системы тел

$$m_1\overrightarrow{r_1} + m_2\overrightarrow{r_2} = 0. \qquad (2.2.3)$$

Из (2.2.2), (2.2.3) находим

$$\overrightarrow{r_1} = \frac{m_2}{m_1+m_2}\overrightarrow{r_{12}}, \qquad (2.2.4)$$

$$\overrightarrow{r_2} = -\frac{m_1}{m_1+m_2}\overrightarrow{r_{12}}, \qquad (2.2.5)$$

подставляя (2.2.4), (2.2.5) в (2.2.1) получаем

$$L = \frac{m_1\dot{\overrightarrow{r_{12}}}^2}{2} + U(|\overrightarrow{r_{12}}|) = \frac{m_1\dot{\overrightarrow{r_{12}}}^2}{2} + \frac{\gamma m(m_1+m_2)}{|\overrightarrow{r_{12}}|} \quad , \qquad (2.2.6)$$

где введено обозначение

$$m = \frac{m_1 m_2}{m_1+m_2}. \qquad (2.2.7)$$

Величину, определяемую в соответствии с (2.2.7), принято называть приведённой массой. Функция (2.2.6) формально совпадает с функцией Лагранжа одной материальной точки с массой m, движущейся в потенциале $U(|\overrightarrow{r_{12}}|)$, симметричным относительно начала выбранной системы отсчёта. Таким образом, задача о движении двух взаимодействующих тел сводиться к решению задачи о движении одного тела с массой m в заданном внешнем поле $U(|\overrightarrow{r_{12}}|)$, создаваемом неподвижным центром с массой $m_1 + m_2$. Отметим, что если масса одного из взаимодействующих тел значительно меньше массы другого тела, последнее можно рассматривать как неподвижный притягивающий центр и найденная зависимость $\overrightarrow{r_{12}}(t)$ будет описывать траекторию движения более лёгкого тела. В противном случае, решив задачу о движении тела с массой m в потенциале $U(|\overrightarrow{r_{12}}|)$, по зависимости $\vec{r}(t)$ в соответствии с (2.2.4),(2.2.5) находят траектории каждой частицы $\overrightarrow{r_1}(t), \overrightarrow{r_2}(t)$.

Воспользовавшись уравнениями Лагранжа (здесь обобщёнными координатами являются координаты радиуса-вектора $\overrightarrow{r_{12}}(t)$, обобщёнными скоростями - координаты вектора $\dot{\overrightarrow{r_{12}}}(t)$)

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\overrightarrow{r_{12}}}(t)} = \frac{\partial L}{\partial \overrightarrow{r_{12}}(t)}, \qquad (2.2.8)$$

Получим уравнения движения тела

$$m\frac{d^2\overrightarrow{r_{12}}(t)}{dt^2} = -\frac{\gamma m(m_1+m_2)}{|\overrightarrow{r_{12}}|^3}\overrightarrow{r_{12}}, \qquad (2.2.9)$$

которое при $m_1 >> m_2$ в полном соответствии с законом всемирного тяготения Ньютона принимает вид:

$$m_2\frac{d^2\overrightarrow{r_{12}}(t)}{dt^2} = -\frac{\gamma m_1 m_2}{|\overrightarrow{r_{12}}|^3}\overrightarrow{r_{12}}, \qquad (2.2.10)$$

Отметим два важных свойства силы тяготения, вытекающих из (2.2.10):

1. сила зависит только от расстояния между телами;

2. сила направлена по прямой, проходящей через центры взаимодействующих тел (такие силы называются центральными).

Следствием указанных свойств является сохранение момента импульса тела. Это означает что траектория движения тела в центральном поле лежит в плоскости, которой перпендикулярен вектор $\vec{L}$. Кроме того, движение тела ограничивается условиями сохранения полной энергии.

$$E = \frac{1}{2}m\dot{\overrightarrow{r_{12}}}^2 - \frac{\gamma m(m_1+m_2)}{|\overrightarrow{r_{12}}|}. \qquad (2.2.11)$$

И величины

$$\left[\dot{\vec{r}}_{12}\times\vec{L}\right]-\frac{\gamma m(m_1+m_2)}{|\vec{r}_{12}|}=\mathrm{co}nst. \qquad (2.2.12)$$

Для решения уравнений движения выберем прямоугольную систему координат, начало которой находится в центре масс (рис. 2.2.2.).

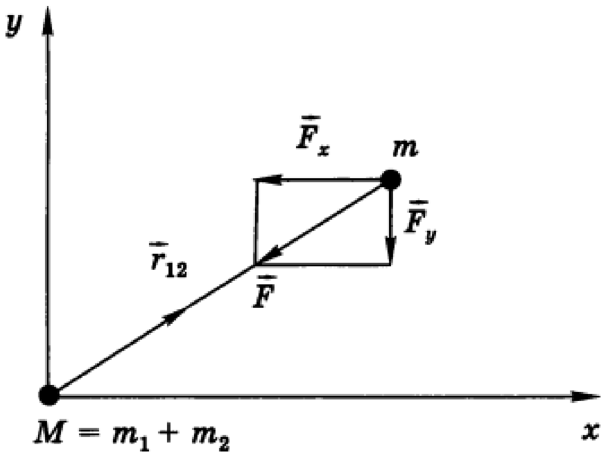

Рисунок 2.2.2 - Система координат, используемая для описания движения тел, под действием гравитационного взаимодействия

Отметим что в отличие от аналитического решения, наиболее просто получаемого в полярной системе координат, численное решение задачи Кеплера более удобно находить, используя декартову систему координат.

Уравнение движения (2.2.9) в выбранной системе координат имеют следующий вид:

$$m\frac{d^2x_{12}}{dt^2}=-\frac{\gamma m(m_1+m_2)}{|\vec{r}_{12}|^3}x_{12} \qquad (2.2.13)$$

$$m\frac{d^2y_{12}}{dt^2}=-\frac{\gamma m(m_1+m_2)}{|\vec{r}_{12}|^3}y_{12} \qquad (2.2.14)$$

Введя обозначение $M=m_1+m_2$ и сократив общие множители, запишем выражения (2.2.13), (2.2.14), составляющие систему ДУ второго порядка, в виде:

$$\frac{d^2x_{12}}{dt^2}=-\frac{\gamma M}{(x_{12}^2+y_{12}^2)^{3/2}}x_{12}, \qquad (2.2.15)$$

$$\frac{d^2y_{12}}{dt^2}=-\frac{\gamma M}{(x_{12}^2+y_{12}^2)^{3/2}}y_{12}, \qquad (2.2.16)$$

Исходя из численных решений системы уравнений (2.2.15), (2.2.16), проведём обезразмеривание этих уравнений, выбрав в качестве единиц измерения расстояния радиус орбиты R и время - период обращения T, соответствующие движению тела по окружности. Тогда можно ввести безразмерные переменные $X = x_{12} / R, Y = y_{12} / R, \tau = t / T$

Выполнив в (2.2.15), (2.2.16) замену переменных $x \to X$, $y \to Y$, $t \to \tau$, получаем

$$\frac{d^2 X}{d\tau^2} = -\frac{\gamma M T^2}{R^3 (X^2 + Y^2)^{3/2}} X, \tag{2.2.17}$$

$$\frac{d^2 Y}{d\tau^2} = -\frac{\gamma M T^2}{R^3 (X^2 + Y^2)^{3/2}} Y, \tag{2.2.18}$$

Как известно, при движении тела по окружности величина центростремительного ускорения a связана с радиусом круговой орбиты $|\vec{R}|$ и скоростью тела $|\vec{v}|$ соотношением

$$a = \frac{|\vec{v}|^2}{|\vec{R}|}. \tag{2.2.19}$$

При движении в гравитационном поле по окружности центростремительное ускорение обусловлено гравитационной силой. Следовательно,

$$\frac{m|\vec{v}|^2}{|\vec{R}|} = \frac{\gamma m M}{|\vec{R}|^2}, \tag{2.2.20}$$

откуда находим

$$|\vec{v}| = \left(\frac{\gamma M}{|\vec{R}|} \right)^{1/2}. \tag{2.2.21}$$

Выражение (2.2.20), являясь общим условием для любой круговой орбиты, позволяет найти зависимость периода движения от радиуса орбиты:

$$T = \frac{2\pi\left|\vec{R}\right|}{\left|\vec{v}\right|}, \qquad (2.2.22)$$

поэтому, подставив в (2.2.22) выражение (2.2.21), получим

$$T = \sqrt{\frac{4\pi^2 \, |\vec{R}|^3}{\gamma M}}. \qquad (2.2.23)$$

Подставляя выражение (2.2.22) в (2.2.17),(2.2.18), окончательно получаем обезразмеренную систему уравнений

$$\frac{d^2X}{d\tau^2} = -\frac{4\pi^2}{(X^2+Y^2)^{3/2}}X, \qquad (2.2.24)$$

$$\frac{d^2Y}{d\tau^2} = -\frac{4\pi^2}{(X^2+Y^2)^{3/2}}Y. \qquad (2.2.25)$$

Из уравнений (2.2.24), (2.2.25) видно, что они оказываются универсальными, поскольку в (2.2.24), (2.2.25) не входят ни период обращения тела вокруг центра поля, ни радиус орбиты. Следовательно, величина T^2 / R^3, входящая в (2.2.17), (2.2.18), одинакова для всех тел, совершающих движение в гравитационном поле по замкнутым траекториям. Данный результат является доказательством справедливости третьего закона Кеплера.

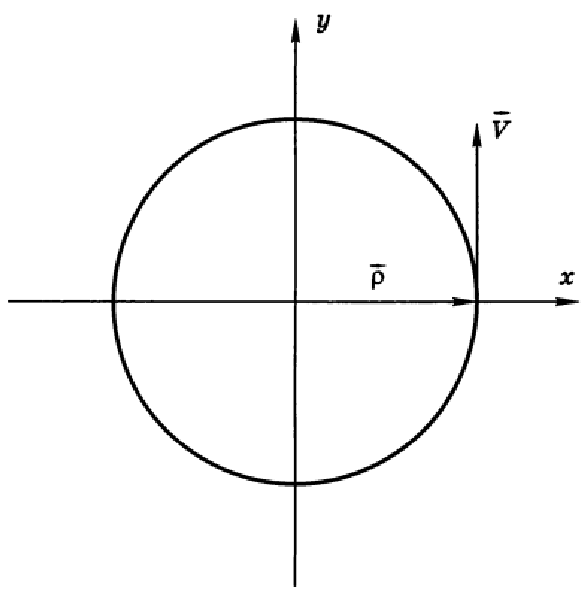

Рис 2.2.3. К выбору начальных условий численного интегрирования СДУ (2.2.24), (2.2.25)

При решении системы дифференциальных уравнений будем считать, что в начальный момент времени тело находилось в точке с радиус-вектором $\vec{r}=(R,0)$ скорость тела была направлена вертикально вверх $\vec{v}=(0,v)$ рис.2.2.3.

Так как система уравнений (2.2.24), (2.2.25) является безразмерной, необходимо также привести к безразмерному виду начальные условия. Выполнив, как и выше, замену переменных $\vec{r}=\vec{\rho}R, t=\tau T$, приводим начальные условия к следующему виду:

$$\vec{\rho}=(1,0), \tag{2.2.26}$$

$$\vec{V}=\left(0, v\frac{T}{R}\right), \tag{2.2.27}$$

где T определяется выражением (2.2.23).

Однако использовать конкретные числовые значения R, T, M для проверки законов Кеплера не требуется, так как безразмерные начальные условия также обладают известным универсализмом. Для того чтобы это показать, найдём безразмерную скорость тела, движущегося в гравитационном поле по окружности. Подставив (2.2.22) в (2.2.27), получим

$$\vec{V}=\left(0, 2\pi\right). \tag{2.2.28}$$

Из (2.2.28) видно, что для получения орбит, отличных от круговых достаточно задавать значения начальной скорости, отличные от 2π .

Для нахождения численного решения системы уравнений (2.2.24), (2.2.25), приведём её к эквивалентной системе уравнений первого порядка, выполнив замену переменных $X \to z_1, X' \to z_2, Y \to z_3, Z' \to z_4$:

$$\frac{dz_1}{d\tau}=z_2, \tag{2.2.29}$$

$$\frac{dz_2}{d\tau}=-\frac{4\pi^2 z_1}{\left[z_1^2+z_3^2\right]^{3/2}}, \tag{2.2.30}$$

$$\frac{dz_3}{d\tau}=z_4, \tag{2.2.31}$$

$$\frac{dz_4}{d\tau}=-\frac{4\pi^2 z_3}{\left[z_1^2+z_3^2\right]^{3/2}}. \tag{2.2.32}$$

Следуя общим правилам MATLAB решения систем ОДУ первого порядка, создадим m-функцию Orbit.m, возвращающую значения координат вектора-функции, стоящей в левой части системы (2.2.29) - (2.2.32):

```
function dy=Orbit(t,z)
% функция, возвращающая значения
% координат вектора-функции, стоящей в левой части системы (2.2.30)
dy=zeros(4,1); % задание вектора-столбца размерности 4x1
% задание координат вектора-функции, стоящей в правой части (2.2.30)
dy(1)=z(2);
dy(2)=-4*pi^2*z(1)/(z(1)^2+z(3)^2)^(3/2);
dy(3)=z(4);
dy(4)=-4*pi^2*z(3)/(z(1)^2+z(3)^2)^(3/2);
```

Далее необходимо выполнить в командном окне следующую последовательность команд, сохранённую нами в файле ZadachaKeplera.m:

```
% задание начальных условий системы ОДУ (2.2.30)
x0=1;
y0=0;
vx0=0;
vy0=2*pi*1.2;
[t,Y]=ode45('Orbit',[0:10^-3:2.5],[x0 vx0 y0 vy0]);% вычисление
                                        % численного
                                        % решение системы
                                        % ОДУ (2.2.30)
figure(1); plot(t,Y(:,1));title('построение зависимости x=x(t)'); % построение
зависимости x=x(t)
figure(2); plot(t,Y(:,2));title('построение зависимости x''=x''(t)'); % построение
зависимости x'=x'(t)
figure(3); plot(t,Y(:,3));title('построение зависимости y=y(t)'); %построение
зависимости y=y(t)
figure(4); plot(t,Y(:,4));title('построение зависимости y''=y(t)'); %  построение
зависимости y'=y(t)
X0=0; Y0=0; %задание координат притягивающего центра
figure(5); plot(Y(:,1),Y(:,3),X0,Y0,'o','MarkerSize',5);
title('построение траектория движения y=y(x) и притягивающего … центра'); %
построение траектория движения y=y(x)
```

Зависимость кинематических характеристик от времени, а также и траектория движения представлены на рисунке (2.2.4 - 2.2.8)

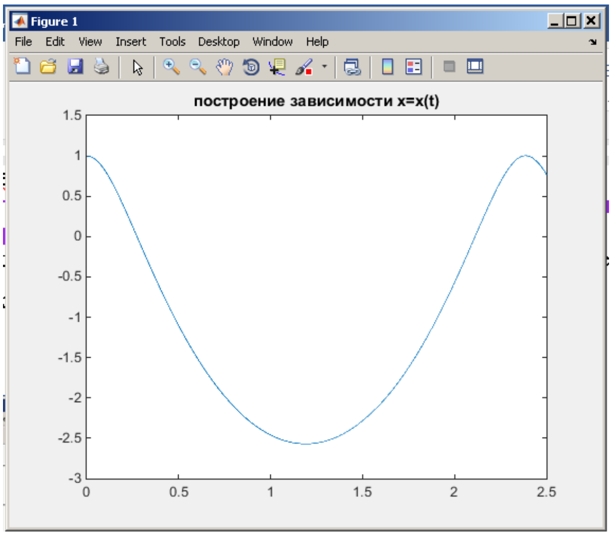

Рис.2.2.4. Построение зависимости x=x(t)

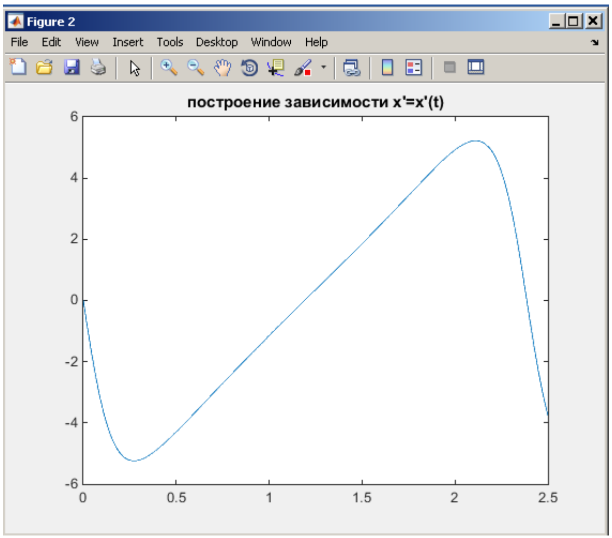

Рис.2.2.5 Построение зависимости x'=x'(t)

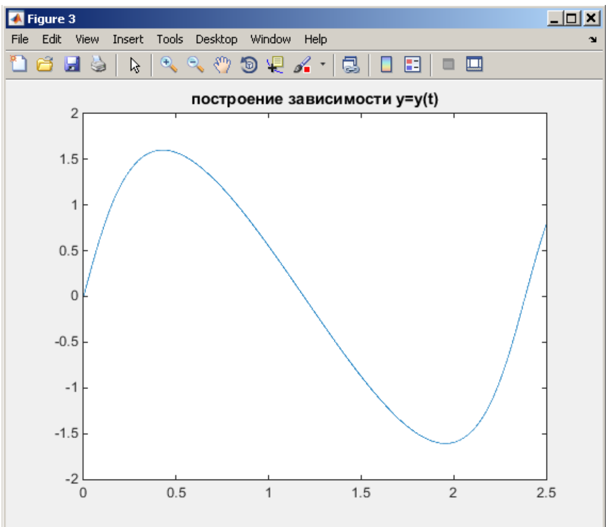

Рис.2.2.6. Построение зависимости y=y(t)

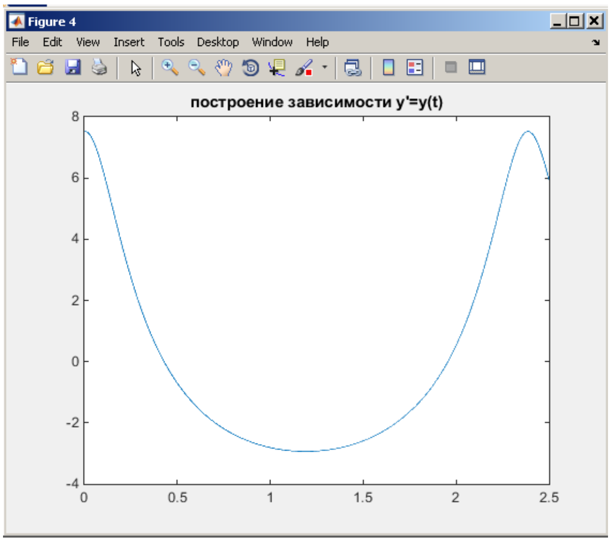

Рис.2.2.7. Построение зависимости y'=y(t)

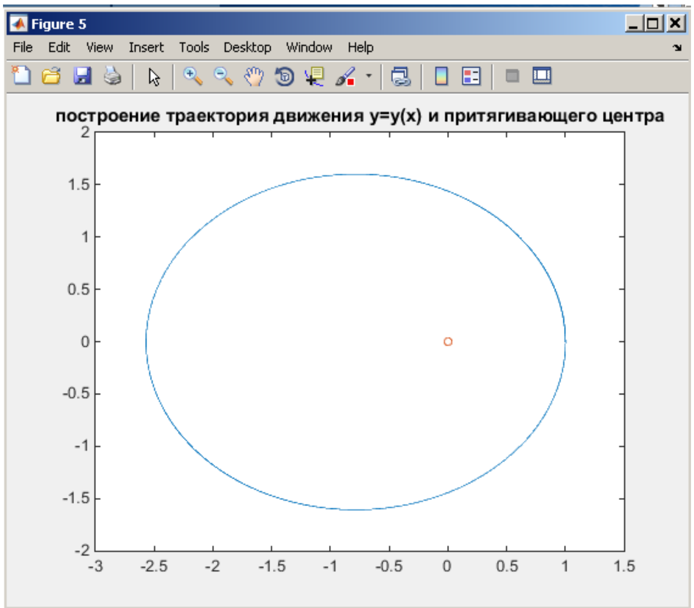

Рис.2.2.8. Построение траектории движения y=y(x) и притягивающего центра

Для вычисления эксцентриситета орбиты необходимо выполнить следующую последовательность команд, сохранённую нами в файле ZadachaKeplera.m

```
MinX=min(Y(:,1)); % вычисление минимального элемента в первом
              % столбце матрицы Y
MaxX=max(Y(:,1)); % вычисление максимального элемента в первом
              % столбце матрицы Y
a=(MaxX-MinX)/2; % вычисление длины первой полуоси
MinY=min(Y(:,3)); % вычисление минимального элемента во втором
```

```
                 % столбце матрицы Y
MaxY=max(Y(:,1)); % вычисление максимального элемента во втором
                 % столбце матрицы Y
b=(MaxY-MinY)/2; % вычисление длины второй полуоси
% вычисление эксцентриситета
if b<=a
    e=(1-(b/a)^2)^0.5
  else
    e=(1-(a/b)^2)^0.5
end
```

При запуске программы получим следующий результат:

e = 0.682490057625959

2.3 Модель математического маятника

Рассмотрим движение груза массой m, прикреплённого к одному из концов жёсткого стержня длиной L, другой конец которого закреплён в точке подвеса (рисунок 2.3.1.).

Такая система, как известно из опыта, будучи выведенной из положения равновесия будет совершать колебания. Будем пренебрегать трением в точке подвеса, массой стержня по сравнению с массой груза и считают, что вся масса груза приложена в одной точке.

Так как движение груза происходит по дуге окружности радиуса L с центром в точке O , то положение груза характеризуется углом отклонения стержня о вертикали θ. При движении по окружности линейная скорость и ускорение груза равны:

$$v = L\frac{d\theta}{dt}, \qquad (2.3.1)$$

$$a = L\frac{d^2\theta}{dt^2}. \qquad (2.3.2)$$

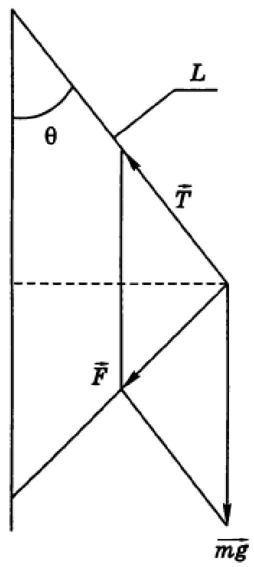

Рисунок 2.3.1. – Математический маятник

В используемой модели на математический маятник действуют две силы: сила тяжести $\overrightarrow{mg}$, направленная вертикально вниз, и сила реакции стержня (рисунок 2.3.1). Равнодействующая этих сил, равная, как видно из рисунка 2.3.1, $-\overrightarrow{mg}\sin\theta$, направлена в сторону уменьшения угла θ.

Следовательно, уравнение движения математического маятника записывается в виде:

$$mL\frac{d^2\theta}{dt^2} = -\vec{m}g\sin\theta \qquad (2.3.3)$$

или

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin\theta \qquad (2.3.4)$$

В общем случае уравнение (2.3.4) оказывается нелинейным. Его решение, как и решения большинства нелинейных уравнений, не выражается через элементарные функции. Отмеченная особенность (2.3.4) определяет необходимость использования для его решения численных методов.

Однако при достаточно малых углах, при которых $\sin\theta \approx \theta$, уравнение (2.3.4) становится линейным

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\theta \qquad (2.3.5)$$

вводя обозначение

$$\omega_0 = \sqrt{\frac{g}{L}} \qquad (2.3.6)$$

Период малых колебаний математического маятника Т

$$T = \frac{2\pi}{\omega_2} = 2\pi\sqrt{\frac{L}{g}} \qquad (2.3.7)$$

не зависит от амплитуды колебаний.

Получим выражение для полной энергии математического маятника, являющейся интегралом движения. Как видно из рисунка 2.3.1, потенциальная энергия математического маятника U, отсчитываемая от точки равновесия маятника, равна:

$$U = mgh = mgL(1-\cos\theta) \qquad (2.3.8)$$

Кинетическая энергия маятника равна

$$\frac{1}{2}mv^2 = \frac{1}{2}mL^2\left(\frac{d\theta}{dt}\right)^2,$$

поэтому полная энергия маятника задаётся следующим выражением:

$$E = \frac{1}{2} mL^2 \left(\frac{d\theta}{dt} \right)^2 + mgL(1 - \cos\theta). \qquad (2.3.9)$$

Уравнение (2.3.9) позволяет получить формулу, связывающую период колебания математического маятника и угол начального отклонения. Для этого решим уравнение (2.3.9) относительно $\frac{d\theta}{dt}$:

$$\frac{d\theta}{dt} = \pm\sqrt{\frac{2}{mL^2}(E - mgL(1 - \cos\theta))}. \qquad (2.3.10)$$

Из (2.3.10) видно, что переменные разделяются:

$$dt = \pm \frac{d\theta}{\sqrt{\frac{2}{mL^2}(E - mgL(1 - \cos\theta))}}. \qquad (2.3.11)$$

Интегрируя (2.3.11), находим выражение для периода колебаний математического маятника:

$$T = 2\int_{-\theta_0}^{\theta_0} \frac{d\theta}{\sqrt{\frac{2}{mL^2}(E - mgL(1 - \cos\theta))}}, \qquad (2.3.12)$$

где θ_0 - начальный угол отклонения маятника.

При движении маятника из начального положения с нулевой начальной скоростью $E = mgL(1 - \cos\theta)$, поэтому

$$T = \sqrt{\frac{2L}{g}} \int_{-\theta_0}^{\theta_0} \frac{d\theta}{\sqrt{\cos\theta - \cos\theta_0}}. \qquad (2.3.13)$$

Возможности MATLAB позволяют не только получить количественные характеристики движения математического маятника, являющиеся в известной мере статическими, но и получить динамическую картинку, демонстрирующую его движение.

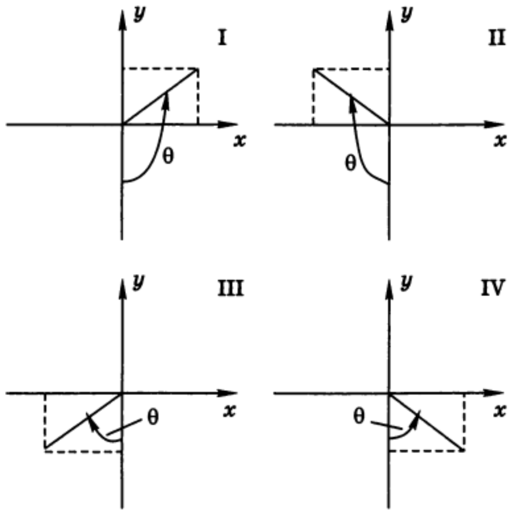

Рисунок 2.3.2. – К вычислению декартовых координат математического маятника

Для создания анимации движения математического маятника необходимо в качестве первого шага вычислить соответствующие декартовы координаты по известной зависимости угла отклонения маятника от времени, хранящейся в матрице Z, возвращённой функцией **ode45**. Как видно из рисунка 2.3.2, при нахождении маятника в первом, втором, третьем и четвёртом квадрантах координаты x, y вычисляются по следующим формулам, соответственно:

1. $x(t) = L\sin(\pi - \theta(t)), y(t) = L\cos(\pi - \theta(t)),$
2. $x(t) = L\sin(\pi - \theta(t)), y(t) = L\cos(\pi - \theta(t)),$
3. $x(t) = -L\sin\theta(t), y(t) = -L\cos\theta(t),$
3. $x(t) = L\sin\theta(t), y(t) = -L\cos\theta(t),$

Для создания анимационного клипа необходимо выполнить следующую последовательность команд, сохранённую нами в файле Pendulum.m

```
clear all  % очистка рабочей области
global omega
g=9.8; % ускорение свободного падения
L=1; % длина маятника
T=2*pi*(g/L)^0.5; % период колебаний
omega=2*pi/T; % циклическая частота
phi0=pi*0.995; % начальный угол
R0=[phi0 0]; % начальные условия
N=20000; % числов узлов временной сетки
[t Z]=ode45('Oscillator',[0:3*T/N:3*T],R0); % решение уравнения
```

```
                                    % движения
% вычисление декартовых координат маятника
for i=1:N+1
        if Z(i,1)>pi/2
           S1(i,1)=L*cos(Z(i,1)-pi/2);
           S1(i,2)=L*sin(Z(i,1)-pi/2);
       end;
       if Z(i,1)<-pi/2
          S1(i,1)=-L*cos(abs(Z(i,1))-pi/2);
          S1(i,2)=L*sin(abs(Z(i,1))-pi/2);
       end;
       if (-pi/2<=Z(i,1))&(Z(i,1)<=pi/2)
          S1(i,1)=L*sin(Z(i,1));
          S1(i,2)=-L*cos(Z(i,1));
       end;
 end;
% визуализация мгновенных значений координат маятника (рис. 8.7)
figure(1); plot(t,S1(:,1)); %,t,S1(:,2)
% координаты подвеса
Sa(1,1)=0;
Sa(1,2)=0;
% координаты маятника
Sa(2,1)=S1(1,1);
Sa(2,2)=S1(1,2);
% отображение маятника в момент времени t=0 (рис. 8.8)
figure(2); plot(Sa(:,1),Sa(:,2),Sa(2,1),Sa(2,2),'o');
axis([-1.2 1.2 -1.2 1.2]);
set(gca,'nextplot','replacechildren'); % включение режима перерисовки
                                    % графиков в одно и том же окне
k=1;
% создание анимационного клипа
for i=1:100:length(S1)
       Sa(2,1)=S1(i,1);
       Sa(2,2)=S1(i,2);
       C=plot(Sa(:,1),Sa(:,2),Sa(2,1),Sa(2,2),'o');
       F(k)=getframe;
       k=k+1;
end;
movie(F,1) % однократное воспроизведение клипа
```

Результат выполнения приведённой выше последовательности команд представлен на рисунке 2.3.3,2.3.4.

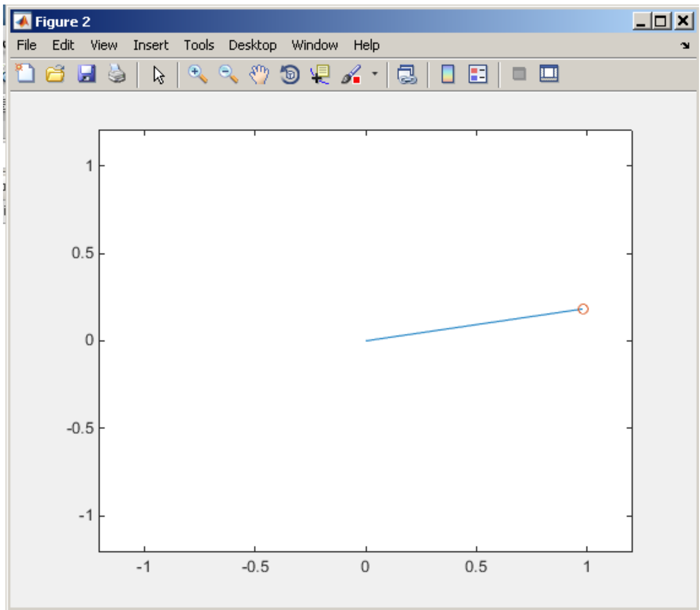

Рисунок 2.3.3. – Зависимости x=x(t), y= y(t)

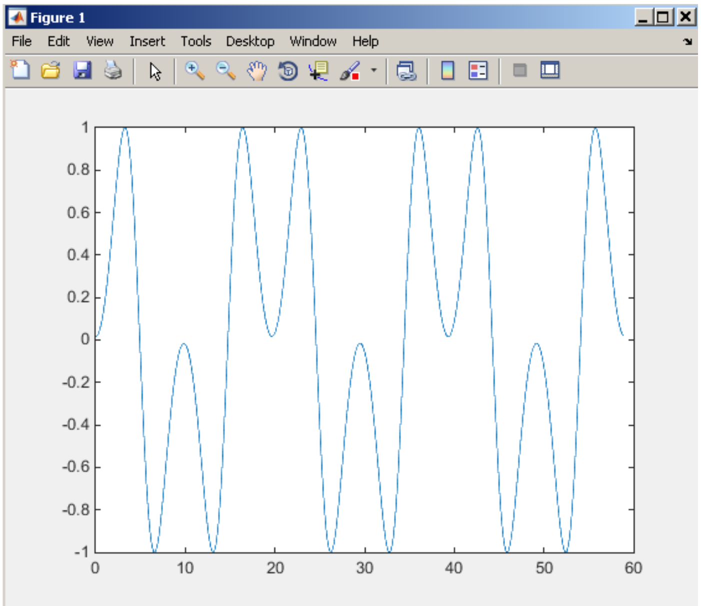

Рисунок 2.3.4. – Зависимости x=x(t), y= y(t)

ЗАКЛЮЧЕНИЕ

В основном преподавание физики складывается из теоретического курса, излагаемого в виде лекций, семинарский занятий, на которых проводиться решение задач без использования ПК, лабораторных занятий. В результате чего не удается сформировать умение дать физически правильную формулировку поставленной задачи; умение создать математическую модель изучаемого процесса; умение разработки и отладки компьютерной реализации математической модели; умение проводить анализ и оценивать адекватность получаемых результатов.

Выбор среды MATLAB был обусловлен тем, что это высокоуровневый язык технических расчетов, интерактивная среда разработки алгоритмов и современный инструмент анализа данных.

В работе рассмотрены задачи: моделирование движения тела, брошенного под углом к горизонту; численное моделирование орбиты (Задача Кеплера); моделирование движения математического маятника. Все эти задачи были реализованы в среде MATLAB. Моделирование движения тела брошенного под углом к горизонту производилось с использованием GUI.

Все поставленные задачи решены.

Созданные приложение помогут более глубоко проникнуть в суть изучаемой задачи, поспособствуют закреплению физического материала и развитию физической интуиции.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дьяконов В.П. Полный самоучитель - Издательство "ДМК Пресс", Москва, 2012. - 362-365с.
2. Поршнев С.В. Компьютерное моделирование физических процессов в пакете MATLAB - Издательство "Лань", Санкт-Петербург, 2011 - 68-75с., 257- 261с.
3. Иродов И.Е. 1. Механика. Основные законы - Издательство "Бином. Лаборатория знаний", Москва, 2010. - 9-12с.
4. Ландау Л. Д., Лифшиц Е.М. - Теоретическая физика. Том 1 из 10. Механика, 5.изд. - Издательская фирма "Физико-математическая литература", Москва,2004, - 51с.
5. Пивоварук Т.В., Ткач С.Н. - Инструкция по подготовке, оформлению и защите курсовых работ и проектов, Брест, 2008