МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ М.В.ЛОМОНОСОВА
ФАККУЛЬТЕТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И КИБЕРНЕТИКИ

Кафедра Оптимального управления

Отчет по практикуму
Численные методы решения задач оптимального управления в среде Matlab

Выполнил
студент 413 группы
Поликарпов А.М.

Руководители семинара:
Будак Борис Александрович
Артемьева Людмила Анатольевна
Ничипорчук Анастасия Владимировна

Москва 2013

Содержание

1 Постановка задачи **2**

1.1 Задача оптимального управления 2

1.2 Задача оптимального управления с фиксированным правым концом 2

2 Алгоритмы решения задачи **2**

2.1 Метод проекции градиента 2

2.1.1 Описание метода 2

2.1.2 Пошаговая схема 3

2.2 Метод последовательных приближений 4

2.2.1 Описание метода 4

2.2.2 Пошаговая схема 4

3 Общее описание программммы **6**

4 Скриншоты программы **7**

5 Исходный код программы **11**

6 Полученные результаты **43**

1 Постановка задачи

В рамках практикума на ЭВМ IV курса осеннего семестра кафедры ОУ факультета ВМК МГУ было необходимо составить программу в среде Matlab реализующую решение задачи оптимального управления методами:

1. Проекции градиента.
2. Последовательных приближений.

1.1 Задача оптимального управления

$$\begin{cases} \dot{x} = f(x,t,u), \\ x(t_0) = x_0 \in R^n, \\ u \in U \subseteq L_2^r[t_0,T], \\ J(u) \to min_{u \in U}, \end{cases} \tag{1}$$

где

1. $t \in [t_0, T]$, где t_0, T - моменты времени, T не задано.
2. U - класс допустимых управлений.
3. $J(u) = \int\limits_{t_0}^{T} f^0(x(t), u(t), t)dt + \Phi(x(T))$, где $f^0(x,u,t), \Phi(x(T))$ - известные функции своих аргументов.

1.2 Задача оптимального управления с фиксированным правым концом

$$\begin{cases} \dot{x} = f(x,t,u), \\ x(t_0) = x_0 \in R^n, \\ x(T) = x_1 \in R^n u \in U \subseteq L_2^r[t_0,T], \\ J(u) \to min_{u \in U}, \end{cases} \tag{2}$$

где

1. $t \in [t_0, T]$, где t_0, T - моменты времени, T задано.
2. U - класс допустимых управлений.
3. $J(u) = \int\limits_{t_0}^{T} f^0(x(t), u(t), t)dt + \beta * ||x - x_1||^2$, где $f^0(x,u,t)$ - известная функция своих аргументов, β - число - штрафной коэффициент, x_1 - конечное состояние.

2 Алгоритмы решения задачи

2.1 Метод проекции градиента

2.1.1 Описание метода

Рассмотрим следующую задачу оптимального управления:

Минимизировать функцию $J(u) = \int\limits_{t_0}^{T} f_0(x(t), u(t), t)dt + \Phi(x(T))$ при условиях $\dot{x} = f(x(t), u(t), t)$, $t_0 \leq t \leq T$; $x(t_0) = x_0$, $u = u(t) \in U \subseteq L_2^r[t_0, T]$, где $x = (x^1, ..., x^n)$, $u = (u^1, ..., u^r)$, функции $f^0(x, u, t)$, $f(x, u, t) = (f^1(x, u, t), ..., f^n(x, u, t))$, $\Phi(x)$ переменных $(x, u, t) \in E^n \times E^r \times [t_0, T]$ считаются известными, U - заданное множество из $L_2^r[t_0, T]$, моменты t_0, T и начальная точка x_0 заданы.

Далее будут сформулированы достаточные условия дифференцируемости функции $J(u)$ на $L_r^2[t_0, T]$, и получена форма для ее градиента. Примем обозначения:

$f_x = \frac{\partial f}{\partial x} = (f_x^1, ..., f_x^n)^T$

$f_u = \frac{\partial f}{\partial u} = (f_u^1, ..., f_u^n)^T, i = 0, ..., n$ $\Phi_x = (\Phi_{x^1} ... \Phi_{x^n})^T$. Здесь $f_{x^i}^i = \frac{\partial f^i}{\partial x^j}$ частная производная функции f^i по переменной x^j, T - знак транспонирования матрицы. Введем функцию Гамильтона-Понтрягина

$H(x, u, t, \psi) = -f^0(x, u, t) + < f(x, u, t), \psi >, (\psi)^T = (\psi_1, ..., \psi_n)$.

Обозначим $H_x = (H_{x^1}, ..., H_{x^n})^T$, $H_u = (H_{u^1}, ..., H_{u^n})^T$.

Теорема 1 *Пусть функции f^0, f, Φ непрерывны по совокупности своих аргументов вместе со своими частными производными по переменным x, и при $(x, u, t) \in E^n \times E^r \times [t_0, T]$ и, кроме того, выполнены следующие условия $|f(x + \Delta x, u + h, t) - f(x, u, t)| \leq L(|\Delta x + |h||)$,*

$||f_x(x + \Delta x, u + h, t) - f_x(x, u, t)| \leq L(|\Delta x + |h||)$,

$|f_x^0(x + \Delta x, u + h, t) - f_x^0(x, u, t)| \leq L(|\Delta x + |h||)$,

$||f_u(x + \Delta x, u + h, t) - f_u(x, u, t)|| \leq L(|\Delta x + |h||)$,

$|f_u^0(x + \Delta x, u + h, t) - f_u(x, u, t)| \leq L(|\Delta x + |h||)$,

$|\Phi_x(x + \Delta x) - \Phi_x(x)| \leq L|\Delta x|$

при всех $(x + \Delta x, u + h, t), (x, u, t) \in E^n \times E^r \times [t_0, T]$, где $L = const \geq 0$.

Тогда функция $J(u)$ при указанных условиях непрерывна и дифференцируема по $u = u(t)$ в норме $L_2^r[t_0, T]$ всюду на $L_2^r[t_0, T]$, причем ее градиент $J'(u) = J'(u, t) \in L_2^r[t_0, T]$ в точке $u = u(t)$ представим в виде:

$J'(u) = -H_u(x, u, t, \psi)|_{x=x(t,u), u=u(t), \psi=\psi(t,u)} = f_u^0(x(t, u), u(t), t) - (f_u(x(t, u), u(t), t))^T \psi(t, u)$, $t_0 \leq t \leq T$, где $x(t) = x(t, u), t_0 \leq T$ является решением задачи оптимального управления, соответствующее управлению $u = u(t)$, а $\psi(t) = \psi(t, u), t_0 \leq t \leq T$, является решением сопряженной системы $\dot{\psi} = -H_x(x, u, t, \psi(t))|_{x=x(t,u), u=u(t), \psi=\psi(t,u)} = f_u^0(x(t, u), u(t), t) - (f_x(x(t, u), u(t), t))^T \psi(t), t_0 \leq t \leq T$ при начальных условиях $\psi(T) = -\Phi_x(x)|_{x=x(T,u)}$.

Теорема 2 *Пусть выполнены все условия теоремы 1 и $U = \{u = u(t) \in L_2^r[t_0, T] : u(t) \in V(t) p.v[t_0, T]\}$, где $V(t)$ заданные множества из E^r, причем*

$\sup_{t_0 \leq t \leq T} \sup_{u \in V(t)} |u| \leq R < \infty$

Тогда $||J'(u) - J'(v)|| \leq L_1 ||u - v||_{L_2}, L_1 = const \geq 0$ при любых $u, v \in U$.

2.1.2 Пошаговая схема

I Начало. Выбрать начальное приближение $u_0(t)$

II Основной цикл. Найти $x_k(t)$ решение системы

$$\begin{cases} \dot{x} = f(x(t), u_k(t), t), \\ \qquad x(t_0) = 0 \end{cases} \tag{3}$$

III. Найти $\psi_k(t)$ решение сопряженной системы.

$$\begin{cases} \dot{\psi} = -H_x(x_k(t), t, u_k(t), \psi(t)) \\ \qquad \psi(T) = -\Phi_x(x(T)). \end{cases} \tag{4}$$

IV. Найти $u_{k+1}(t)$:

$u_{k+1} = pr_U(u_k + H_u(x_k, t, u_k, \psi_k))$, где $pr_U(u)$ - оператор проектирования вектора u на U.

IV. Если выполнен критерий останова, то положить $u_* = u_k$ и прекратить вычисления, иначе к шагу VI. *VI.* $k = k + 1$ и перейти к шагу II.

Замечания

1. H - функция Гамильтона Понтрягина:
$$H(x, u, t, \psi) = -f^0(x, u, t) + < f(x, u, t), \psi >$$

2. Если условие argmax выполняется не единственным способом, то выбираем u_{k+1} любое из возможных.

3. В качестве критерия останова могут быть выбраны следующие

   (a) $u_{k+1} = u_k$

   (b) $||J'(u)|| < eps$

   (c) Количество итераций $k = N$

2.2 Метод последовательных приближений

2.2.1 Описание метода

Пусть управляемая система описывается уравнениями с начальными условиями и ограничениями $\frac{dx}{dt} = f(x, t, u)$, $x(t_0) = x^0$, $u(t) \in U$, $t \geq t_0$. Здесь $x = (x_1, ..., x_n)$ - n-мерный вектор фазовых координат, $u = (u_1, ..., u_m)$ - m-мерный вектор управляющих функций, t - время, $f = (f_1, ..., f_n)$ -заданная вектор функция, x^0 постоянный вектор, t_0 - начальный момент времени, U - замкнутое множество m-мерного пространства. Допустимым управлением будет называть кусочно-непрерывные функции удовлетворяющие $u(t) \in U$.

Поставим задачу об определении допустимого управления $u(t)$ минимизирующего функционал:

$J = (c, X(T)), T > t_0$.

Здесь T заданный момент времени, $c = (c_1, ..., c_n)$ - ненулевой постоянный вектор. Скобками обозначено скалярное произведение векторов. Будем предполагать, что поставленная задача имеет решение в классе допустимых управлений $u(t)$, это решение будем называть оптимальным управлением. Введением дополнительных фазовых координат широкий класс функционалов сводится к такому виду.

Введем n-мерный вектор $p = (p_1, ..., p_n)$ сопряженных переменных (импульсов) и функцию Гамильтона H, запишем сопряженную систему и условия трансверсальности:

$H(t, x, p, u) = (p, f(x, t, u))$ $dp_j/dt = -\frac{\partial H}{\partial x_i} = -\sum_{j=1}^{n} p_j \partial f_j / \partial x_i, \quad p(T) = -c.$

Согласно принципу максимума, искомое оптимальное управление доставляет функции H максимум по $u \in U$ при любом $t \in [t_0, T]$, если x и p удовлетворяют условиям, указаным выше.

2.2.2 Пошаговая схема

I Начало. Выбрать начальное приближение $u_0(t)$ положить $k = 0$

II Основной цикл. Найти $x_k(t)$ решение системы

$$\begin{cases} \dot{x} = f(x(t), u_k(t), t), \\ x(t_0) = 0 \end{cases} \tag{5}$$

III. Найти $\psi_k(t)$ решение сопряженной системы.

$$\begin{cases} \dot{\psi} = -H_x(x_k(t), t, u_k(t), \psi(t)) \\ \psi(T) = -\Phi_x(x(T)). \end{cases} \tag{6}$$

IV. Найти $u_{k+1}(t)$:

$$u_{k+1} = argmax_{u \in U} H(x_k, t, u_k(t), \psi(k)).$$

IV. Если выполнен критерий останова, то положить $u_* = u_k$ и прекратить вычисления, иначе к шагу VI. *VI.* $k = k + 1$ и перейти к шагу II.

Замечания

1. H - функция Гамильтона Понтрягина:

$$H(x, u, t, \psi) = -f^0(x, u, t) + < f(x, u, t), \psi >$$

2. В качестве критерия останова могут быть выбраны следующие

   (a) $u_{k+1} = u_k$

   (b) $||J'(u)|| < eps$

   (c) Количество итераций $k = N$

3 Общее описание программы

Структура исходного кода:

1. Скрипт связывающий расчеты и визуализацию.
2. Модуль реализующий расчеты.
3. Модуль реализующий визуализацию.
4. Различные подфункции, в частности, нужные для вычисления ode.

Описание интерфейса программы:

1. Окно для ввода задачи. Включает в себя:
   (a) Поля для ввода интервала времени с заданием шага.
   (b) Поля для ввода размерности системы, самой системы и начального условия.
   (c) Поля для ввода размерности управления и начального управления.
   (d) Поле для ввода интегрального функционала
   (e) Поле для ввода терминального функционала
   (f) Вызов диалогового окна(по нажатию кнопки) для задания закрепленного правого конца.
   (g) Выбор критерия останова.
   (h) Поле для ввода количества итераций.
   (i) Выбор метода решения задачи (реализовано с помощью диалоговых окон).
   (j) Выбор множества U (реализовано с помощью диалоговых окон).
2. Окно для просмотра результатов.
3. Возможность сохранения/загрузки примеров.
4. Пункты меню "Help"и "About".

4 Скриншоты программы

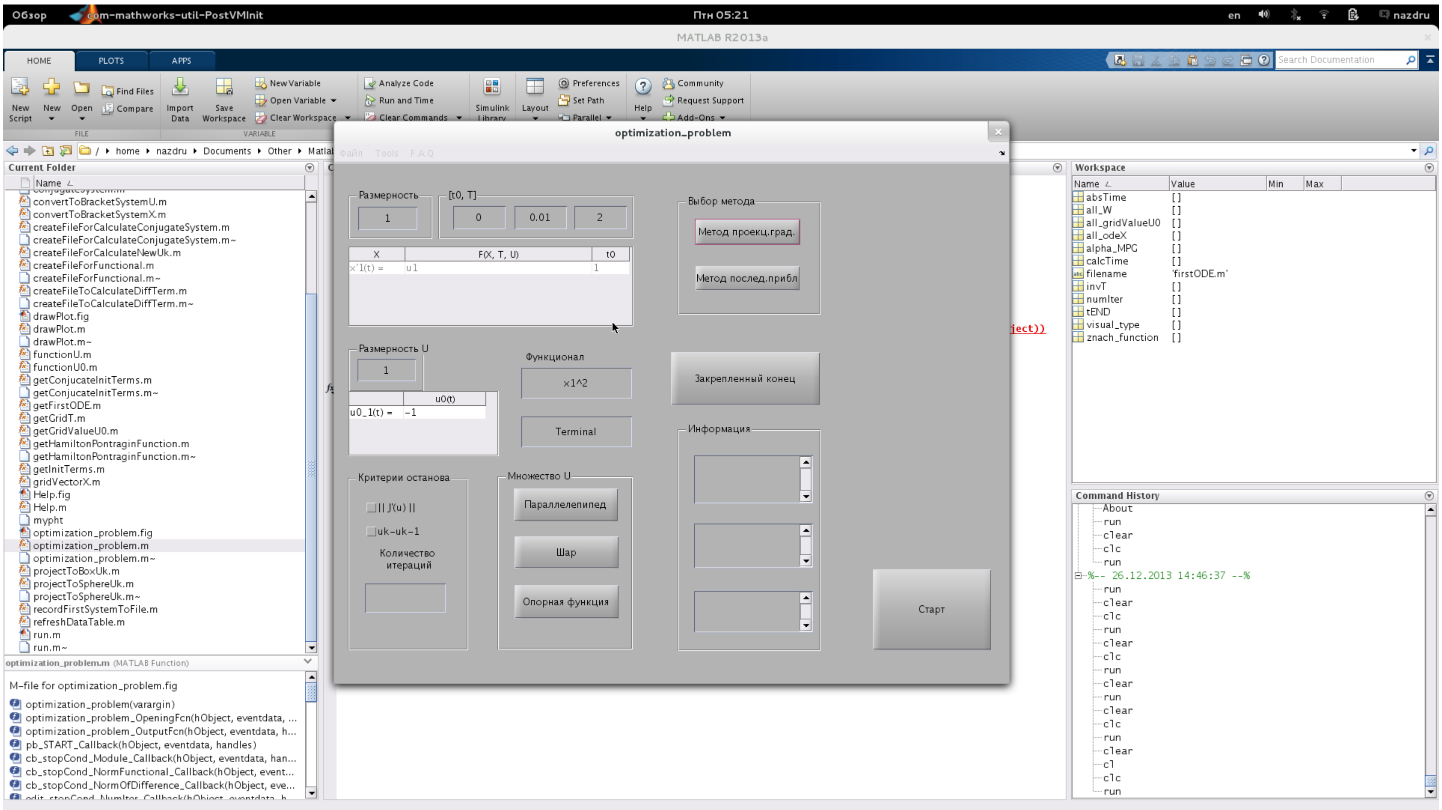


Рис. 1: Главное окно

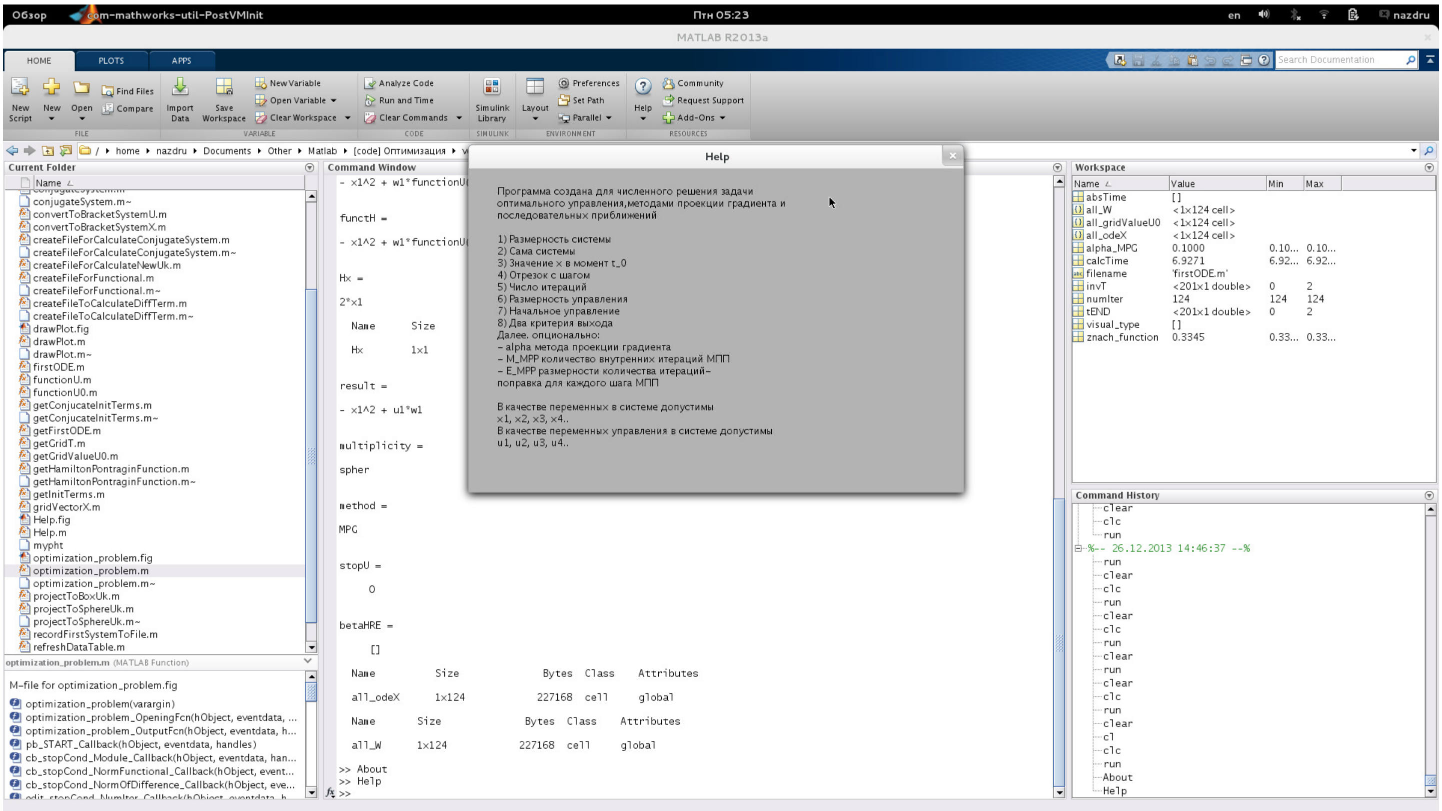

MATLAB R2013a
Help
Программа создана для численного решения задачи
оптимального управления,методами проекции градиента и
последовательных приближений
1) Размерность системы
2) Сама системы
3) Значение x в момент t_0
4) Отрезок с шагом
5) Число итераций
6) Размерность управления
7) Начальное управление
8) Два критерия выхода
Далее. опционально:
– alpha метода проекции градиента
– M_MPP количество внутренних итераций МПП
– E_MPP размерности количества итераций–
поправка для каждого шага МПП
В качестве переменных в системе допустимы
x1, x2, x3, x4..
В качестве переменных управления в системе допустимы
u1, u2, u3, u4..

Рис. 2: Помощь

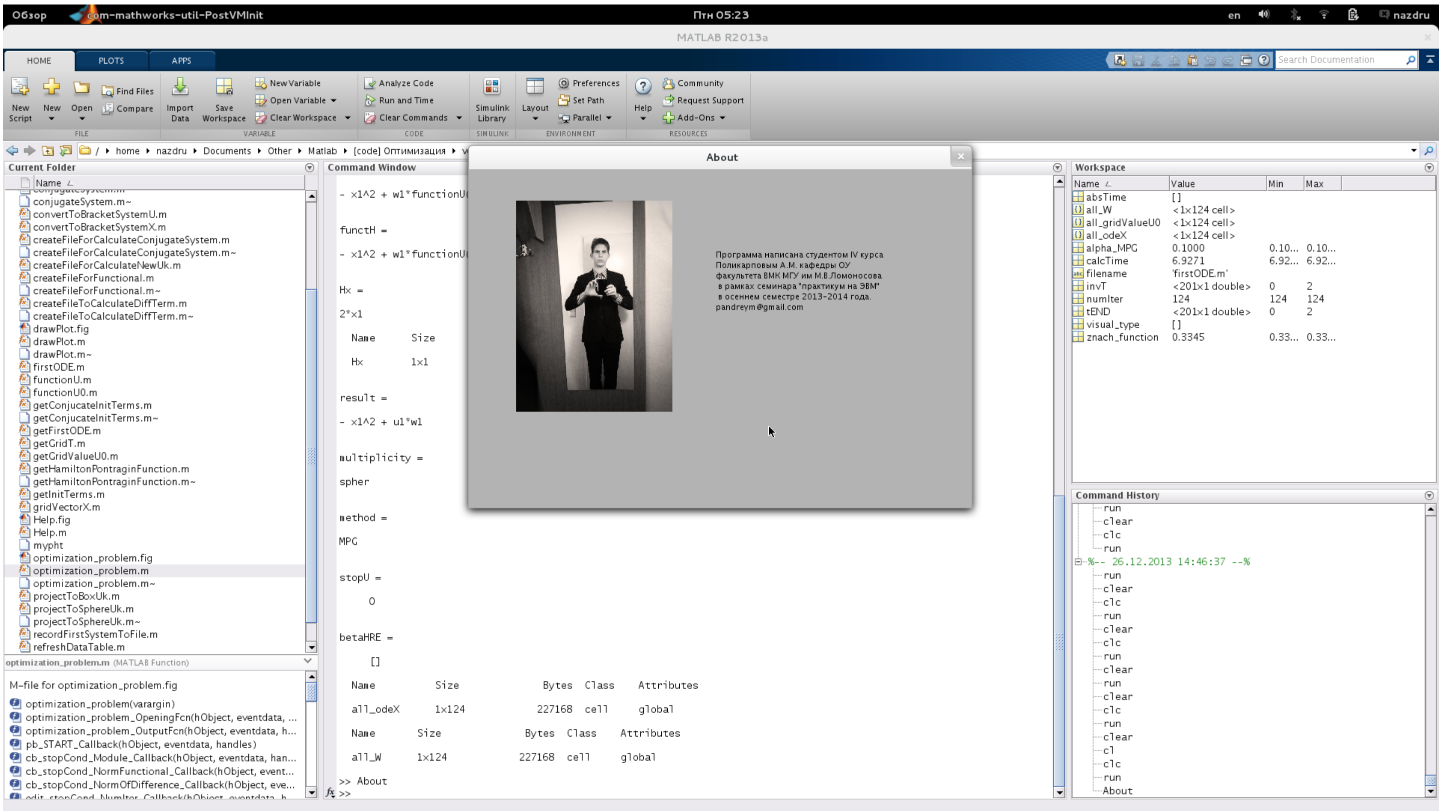


Рис. 3: Об Авторе

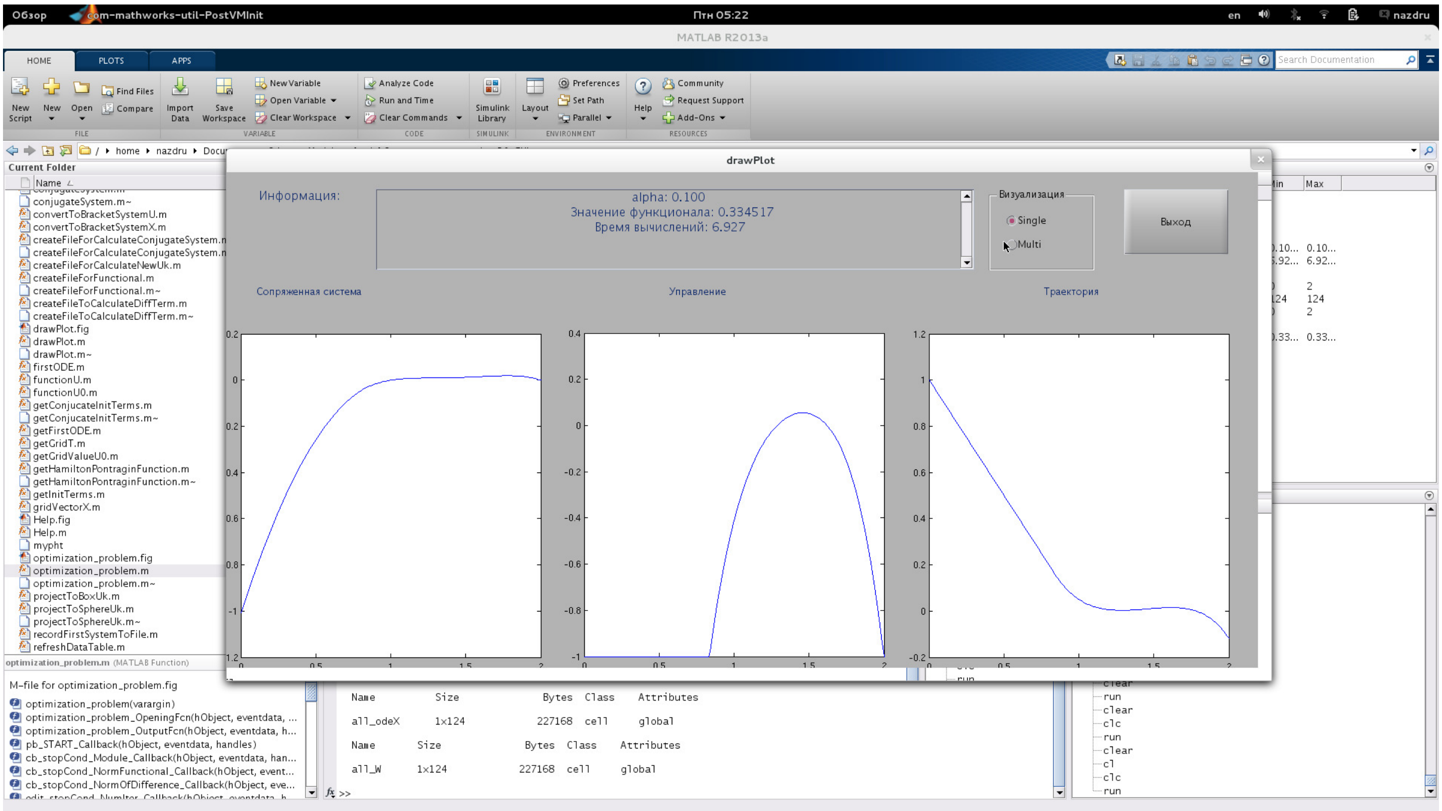


Рис. 4: Визуализация

5 Исходный код программы

Ниже приведен исходный код основных трех частей программы.

Исходный код скрипта:

```
clc
clear
clear all
warning off
try
  matlabpool close
catch
end
  global all_odeX;
  global all_W;
  global tEND;
  global all_gridValueU0;
  global invT; global alpha_MPG; global absTime; global calcTime;
  global numIter; global znach_function;

  global visual_type;
  filename = sprintf('firstODE.m');
if exist(filename) ~ 0
  delete firstODE.m;
end
if exist('calculateNewUk.m') ~ 0
  delete calculateNewUk.m;
end

optimization_problem
```

Исходный код основной части:

```
function varargout = optimization_problem(varargin)
%OPTIMIZATION_PROBLEM M-file for optimization_problem.fig
%      OPTIMIZATION_PROBLEM, by itself, creates a new OPTIMIZATION_PROBLEM or raises the exis
%      singleton*.
%
%      H = OPTIMIZATION_PROBLEM returns the handle to a new OPTIMIZATION_PROBLEM or the handl
%      the existing singleton*.
%
%      OPTIMIZATION_PROBLEM('Property','Value',...) creates a new OPTIMIZATION_PROBLEM using
%      given property value pairs. Unrecognized properties are passed via
%      varargin to optimization_problem_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      OPTIMIZATION_PROBLEM('CALLBACK') and OPTIMIZATION_PROBLEM('CALLBACK',hObject,...) cal
%      local function named CALLBACK in OPTIMIZATION_PROBLEM.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help optimization_problem

% Last Modified by GUIDE v2.5 26-Dec-2013 12:00:42

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @optimization_problem_OpeningFcn, ...
                   'gui_OutputFcn',  @optimization_problem_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
% End initialization code - DO NOT EDIT


% --- Executes just before optimization_problem is made visible.
function optimization_problem_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for optimization_problem
  handles.output = hObject;
  guidata(hObject, handles);

  global dimension; global functional;
  global left; global right; global step;
      global U_dimension; global dataU0;  global gridValueU0;
      global odeX; global tEND;

  global HoldRightEnd;
  global boolHRE;


global stopJ;
global stopU;

boolHRE = false;

stopJ = false;
stopU = false;

  %[dimension, U_dimension, dataFX, dataU0] = getInitTerms();



 % set(handles.ut_FunctionAndT, 'Data', dataFX);
 % set(handles.ut_functionU0,   'Data', dataU0);
 % set(handles.edit_Dimension, 'String', num2str(dimension));
 % set(handles.ed_U_dimension, 'String', num2str(U_dimension));




% UIWAIT makes optimization_problem wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = optimization_problem_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% eventdata  reserved - to be defined in a future version of MATLAB
% hObject    handle to figure
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pb_START.
function pb_START_Callback(hObject, eventdata, handles)
% hObject    handle to pb_START (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


global all_odeX;
  global all_W;
  global all_gridValueU0;

%размерность левый конец, правый конец, шаг, ГЛОБАЛЬНОЕ значение текущего U
  global dimension; global left; global right; global step; global gridValueU0;
% функционал - интегральный + количество итераций
  global numIter;
  %размерность управления
  global U_dimension; global odeX; global tEND; global W;
% начальные значения
  global invT;

%%%%
  global stopJ;
global stopU;
%%%%
%интегральный + функциональный
  global znach_function;
  global term_functional;

  global functional;


  global centerSphere;

%%
  global multiplicity;
  global method;

  global calcTime;
%%

  %%%%%
  %----MPG-param
    global alpha_MPG;
    global M_MPP; global E_MPP;
  %----MPG-param
```

```
%%%%%%%%%%%%
  global HoldRightEnd; global betaHRE;
  global boolHRE;

  global eps_stopU;

%%%%%%%%%%%%

   symbolicX = sym('x', [1 dimension]);
if exist('firstODE.m' ) ~ 0
  delete firstODE.m;
end

if exist('calculateNewUk.m')
  delete calculateNewUk.m;
end
  %%%%%%%%%%%
    step = str2double(get(handles.ed_TimeStep,    'String'));
    left = str2double(get(handles.ed_LeftEnd,     'String'));
    right = str2double(get(handles.ed_rightEnd,     'String'));
  gridSize = (right - left)/step + 1 ;
  %%%%%%%%%%%

  dataFX = get(handles.ut_FunctionAndT, 'Data');
  vIC    = str2double(dataFX(:, 3));   vIC = vIC.';
  dataFX = dataFX(:, 2);

  %%%%
    dataFXdup = dataFX; %сохранение для след
  %%%%
  dataU0 = get(handles.ut_functionU0, 'Data');
  dataU0 = dataU0(:, 2);

  h = waitbar(0, 'Please wait...');



%замена на скобки в интегральном функционале



boolHRE
   new_functional = sprintf('%s', functional)
   integr_funct = new_functional

  for j = 1 : dimension
     integr_funct = strrep(integr_funct, sprintf('x%d', j), sprintf('x(i, %d)', j));
  end

  for j = 1 : U_dimension
```

```
      integr_funct = strrep(integr_funct, sprintf('u%d', j), sprintf('u(i, %d)', j));
  end
%для начальных условий сопряженной системы

if (boolHRE == true)
  tmpe = sym('0');
  for i = 1 : 1 : dimension
    tmpe = tmpe + sym(sprintf('(x%d - %d)^2', i, HoldRightEnd(i)));
  end
  term_functional = betaHRE * tmpe
end

    proizvd_term = -jacobian(term_functional, symbolicX)
%замена на скобки в терминальном функционале
  for i = 1 : 1 : dimension
    term_functional = subs((term_functional), sprintf('x%d', i), sprintf('x(%d)', i));
  end
%замена на скобки в производной sтерминального функционала

for i = 1 : 1 : dimension
    proizvd_term = subs((proizvd_term), sprintf('x%d', i), sprintf('x(%d)', i));
  end


  tmpZet = zeros(1, dimension);
  if isequal(tmpZet, proizvd_term)
    tmp = 'zero'
    proizvd_term = sym(tmpZet)
  end



whos proizvd_term
tmp_size_tmp = (size(proizvd_term));
tmp_size_tmp = tmp_size_tmp(2)



tmp_func_prx = cell(tmp_size_tmp, 1);
  for i = 1 : 1 : tmp_size_tmp
   tmp_func_prx{i} = char(proizvd_term(1, i));
  end

%char(proizvd_term)
  createFileForFunctional(integr_funct, dimension,gridSize, U_dimension);
  createFileToCalculateDiffTerm(tmp_func_prx, dimension, gridSize);

  %functionU0 - для значения начального U0
  %functionU - для значений U потом
  %gridValueU0 - получить массив значений U0 на сетке
  %----------------------------
```

```
  %получить значения начального U - U0 - на сетке
  gridValueU0 = getGridValueU0(dataU0, gridSize, U_dimension, left, step);
  % заменить в системе u0 на реальные значения


  for i = 1 : 1 : dimension
    for j = 1 : 1 : U_dimension
      dataFX{i} = strrep(dataFX{i}, sprintf('u%d', j), sprintf('functionU(t, %d)', j));
    end
  end



%скобки + запись в файл
  dataFX_brc = convertToBracketSystemX(dataFX, dimension, 0);
  recordFirstSystemToFile(dataFX_brc);
%скобки для U0

  tr = 'please wait';
  while exist('firstODE.m') == 0
    tr
  end


%получаем функция ГП

  functH = getHamiltonPontraginFunction(new_functional, dataFX)


%Производная по x функции ГП

  Hx = -jacobian(functH, symbolicX)
  whos Hx

  tmpdim = dimension;
  tmpUdim = U_dimension;
 %надо заменить на скобки
  for i = 1 : 1 : tmpUdim
    for j = 1 : 1 : tmpdim
      Hx(i) = subs(Hx(i), sprintf('w%d', j), sprintf('w(%d)', j));
      Hx(i) = subs (Hx(i), sprintf('x%d', j), sprintf('gridVectorX(t, %d, odeX)', j));
    end
  end


  %функция ГП + производная по u
  functHnotU = getHamiltonPontraginFunction(new_functional, dataFXdup) ;
  symbolicU = sym('u', [1 dimension]);
  Hu = -jacobian( functHnotU, symbolicU)  ;
```

```
  % Замена на скобки Hu

  for i = tmpUdim : -1 : 1
    for j = tmpdim : -1 : 1
      Hu(i) = subs (Hu(i), sprintf('x%d', j), sprintf('x(i, %d)', j));
      Hu(i) = subs (Hu(i), sprintf('w%d', j), sprintf('w(i, %d)', j));
    end
     for j = tmpdim : -1 : 1
      Hu(i) = subs (Hu(i), sprintf('u%d', j), sprintf('functionU(t, %d)', j));
    end

  end


  %запись в файл сопряженной системы
  createFileForCalculateConjugateSystem(dimension, Hx);

  %запись в файл Uk
  if isequal(method, 'MPP')
    alpha_MPG = 1;
  end
  createFileForCalculateNewUk(Hu, U_dimension, gridSize, alpha_MPG);
  tr = 'please wait';
  while exist('calculateNewUk.m') == 0
    tr
  end


  initT = zeros(1, dimension);

%%%%%%%%%%%%%%%%%%%%%%%
%  W это сопряженная переменная для системы, не PSI
%%%%%%%%%%%%%%%%%%%%%%%
  %
  prcODE = odeset('AbsTol', 0.001, 'RelTol', 0.001);

tic
multiplicity
method
stopU
betaHRE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


previous_funct = 100000000;
previous_U0    = getGridValueU0(dataU0, gridSize, U_dimension, left, step);
```

```
TT = [left : step : right];

if(isequal(method, 'MPG'))
    if isequal(multiplicity, 'spher')
      for i = 1 : 1 : numIter
        [tEND, odeX] = ode23(@firstODE, left:step:right, vIC, prcODE);
        all_odeX{i} = {odeX};
         initT = calculatePrzvdTermFunctional(odeX(gridSize, :));
        [invT, W] = ode23(@conjugateSystem, right:-step:left, initT, prcODE);

        W = W(end: -1: 1, : );
        all_W{i} = {W};
        gridValueU0 = calculateNewUk(left, step, W, odeX);
        gridValueU0 = projectToSphereUk(gridValueU0);
        all_gridValueU0{i} = {gridValueU0};
        waitbar(i/numIter,h)
        if boolHRE == true
          znach_function = trapz(TT, calculateFunctional(left, step, odeX, gridValueU0)) + b
        else
          znach_function = trapz(TT, calculateFunctional(left, step, odeX, gridValueU0)) + c
        end

       previous_funct = znach_function;
        previous_U0 = gridValueU0;
       if stopJ == true
          if previous_funct < znach_function
            break;
          end
        end

        if stopU == true
            tmp =gridValueU0 - previous_U0;

      if (norm(tmp) < eps_stopU)
                break;
             end
        end



      end
    end
  %%%%тут функционал..
    if isequal(multiplicity, 'paral')
        for i = 1 : 1 : numIter
        [tEND, odeX] = ode23(@firstODE, left:step:right, vIC, prcODE);
        all_odeX{i} = {odeX};
        initT = calculatePrzvdTermFunctional(odeX(gridSize, :));
       [invT, W] = ode23(@conjugateSystem, right:-step:left, initT, prcODE);
```

```
      W = W(end: -1: 1, : );
      all_W{i} = {W};
      gridValueU0 = calculateNewUk(left, step, W, odeX);

      gridValueU0 = projectToBoxUk(gridValueU0);
      all_gridValueU0{i} = {gridValueU0};
      waitbar(i/numIter,h)
      if boolHRE == true
        znach_function  = trapz(TT, calculateFunctional(left, step, odeX, gridValueU0)) + 
      else
        znach_function = trapz(TT, calculateFunctional(left, step, odeX, gridValueU0)) + c
      end

     previous_funct = znach_function;
      previous_U0 = gridValueU0;
     if stopJ == true
        if previous_funct < znach_function
          break;
        end
      end

      if stopU == true
          tmp =gridValueU0 - previous_U0;

    if (norm(tmp) < eps_stopU)
          break;
        end
      end


    end
  end
end


if(isequal(method, 'MPP'))

  if isequal(multiplicity, 'spher')
   for i = 1 : 1 : numIter
      for j = 1 : 1 : M_MPP
        [tEND, odeX] = ode23(@firstODE, left:step:right, vIC, prcODE);
        all_odeX{i} = {odeX};
        initT = calculatePrzvdTermFunctional(odeX(gridSize, :));
        [invT, W] = ode23(@conjugateSystem, right:-step:left, initT, prcODE);

        W = W(end: -1: 1, : );
        all_W{i} = {W};
        gridValueU0 = calculateNewUk(left, step,  W, odeX);
```

```
      gridValueU0 = projectToSphereUk(gridValueU0);

      all_gridValueU0{i} = {gridValueU0};
       if boolHRE == true
      znach_function = trapz(TT, calculateFunctional(left, step, odeX, gridValueU0)) + b
    else
      znach_function = trapz(TT, calculateFunctional(left, step, odeX, gridValueU0)) + c
    end
    previous_funct = znach_function;
    previous_U0 = gridValueU0;

    end



   if stopJ == true
      if previous_funct < znach_function
        break;
      end
    end

    if stopU == true
        tmp =gridValueU0 - previous_U0;

  if (norm(tmp) < eps_stopU)
        break;
      end
    end
    gridValueU0 = E_MPP(i) * gridValueU0;
    waitbar(i/numIter,h)
  end


end

if isequal(multiplicity, 'paral')
  k = 1;
  for i = 1 : 1 : numIter
    for j = 1 : 1 : M_MPP
      [tEND, odeX] = ode23(@firstODE, left:step:right, vIC, prcODE);
      all_odeX{i} = {odeX};
      initT = calculatePrzvdTermFunctional(odeX(gridSize, :));
      [invT, W] = ode23(@conjugateSystem, right:-step:left, initT, prcODE);
      W = W(end: -1: 1, : );
      all_W{i} = {W};
      gridValueU0 = calculateNewUk(left, step,  W, odeX);
      gridValueU0 = projectToBoxUk(gridValueU0);
      all_gridValueU0{i} = {gridValueU0};
       if boolHRE == true
      znach_function = trapz(TT, calculateFunctional(left, step, odeX, gridValueU0)) + b
```

```
        else
          znach_function = trapz(TT, calculateFunctional(left, step, odeX, gridValueU0)) + ca
        end

       if stopJ == true
          if previous_funct < znach_function
            break;
          end
        end

        if stopU == true
            tmp =gridValueU0 - previous_U0;

      if (norm(tmp) < eps_stopU)
            break;
          end
        end

        end
        gridValueU0 = E_MPP(k) * gridValueU0;
        \newpage

\begin{thebibliography}{0}
\bibitem{bey:meth}
{\bf Бейко~И.В., Бублик~Б.Н., Зинько~П.Н.}\
Методы и алгоритмы решения задач оптимизации.\
Вища школа, 1983.

\bibitem{rov:opt}
{\bf Rovenskaya~E.A., Kamzolkin~D.V.}\ Infinite Horizon Optimal Control with Applications in
MSU CMC Publication Department, MAKS Press, Moscow, Russia, 2009.

\bibitem{kis:av}
{\bf Киселев~Ю.Н., Аввакумов~С.Н., Орлов~М.В.}\
Оптимальное управление. Линейная теория и приложения. Учебное пособие для студентов ВМиК МГУ

\bibitem{Ponrt}
{\bf Понтрягин~Л.С., Болтянский~В.Г.,  Гамкрелидзе~Р.В., Мищенко~Е.Ф.} \ Математическая теор
\bibitem{Vasiliev}
{\bf Васильев~Ф.П.} Методы оптимизации.М.Факториал Пресс: 2002.

\end{thebibliography}

        k = k + 1;
        waitbar(i/numIter,h)
      end

   end
  end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  whos all_odeX;
  whos all_W;


  calcTime = toc;
  close(h);


  drawPlot

% --- Executes on btton press in cb_stopCond_Module.
function cb_stopCond_Module_Callback(hObject, eventdata, handles)
% hObject    handle to cb_stopCond_Module (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA 0.4995

% Hint: get(hObject,'Value') returns toggle state of cb_stopCond_Module


% --- Executes on button press in cb_stopCond_NormFunctional.
function cb_stopCond_NormFunctional_Callback(hObject, eventdata, handles)
% hObject    handle to cb_stopCond_NormFunctional (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cb_stopCond_NormFunctional


global stopJ;
stopJ = get(hObject,'Value')


% --- Executes on button press in cb_stopCond_NormOfDifference.
function cb_stopCond_NormOfDifference_Callback(hObject, eventdata, handles)
% hObject    handle to cb_stopCond_NormOfDifference (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cb_stopCond_NormOfDifference



function edit_stopCond_NumIter_Callback(hObject, eventdata, handles)
% hObject    handle to edit_stopCond_NumIter (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_stopCond_NumIter as text
%        str2double(get(hObject,'String')) returns contents of edit_stopCond_NumIter as a dou
  global numIter;
  numIter = str2double(get(hObject,'String'))
  if (isnan(numIter))
    numIter = 10;
  end

  set(hObject,'String', num2str(numIter));

% --- Executes during object creation, after setting all properties.
function edit_stopCond_NumIter_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_stopCond_NumIter (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_Dimension_Callback(hObject, eventdata, handles)
% hObject    handle to edit_Dimension (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_Dimension as text
%        str2double(get(hObject,'String')) returns contents of edit_Dimension as a double
  global dimension; global U_dimension;
  dimension = str2double(get(hObject,'String'));

  if (isnan(dimension))
    dimension = 2;
  end
  [dataFX, dataU0] = refreshDataTable(dimension, U_dimension);

  set(handles.edit_Dimension, 'String', num2str(dimension));
  set(handles.ut_FunctionAndT,   'data', dataFX)


% --- Executes during object creation, after setting all properties.
function edit_Dimension_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_Dimension (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pb_UOprFunct.
function pb_UOprFunct_Callback(hObject, eventdata, handles)
% hObject    handle to pb_UOprFunct (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pb_UBox.
function pb_UBox_Callback(hObject, eventdata, handles)
% hObject    handle to pb_UBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

  global U_dimension; global box_alpha; global box_beta;
    global multiplicity;


  info_str = sprintf('U_Dimension = %s', num2str(U_dimension));
  option.Resize = 'on';
  option.WindowStyle = 'normal';
  option.Interpreter = 'tex';


  for i = 1 : 1 : U_dimension
    dialogProperties{i} = sprintf('alpha_%d < x', i);
    defaultAnswer{i} = sprintf('%d', -1);
  end

  %title = 'Control area - globe';
  num_lines = 1;
  box_alpha = inputdlg(dialogProperties, info_str,  num_lines, defaultAnswer, option);


  if (~isempty(box_alpha))
    for i = 1 : 1 : U_dimension
      dialogProperties{i} = sprintf('x < beta_%d', i);
      defaultAnswer{i} = sprintf('%d', 1);

    end
    num_lines = 1;
    box_beta = inputdlg(dialogProperties, info_str,  num_lines, defaultAnswer, option);
```

```
  end



  tmp_alpha = str2double(box_alpha);
  tmp_beta = str2double(box_beta);
  tmpstr = {sprintf('Параллелепипед ')};
  for i = 1 : 1 : U_dimension
    str = sprintf('%.2f < u%d < %.2f ', tmp_alpha(i), i, tmp_beta(i));
    tmpstr{i+1} = str;
  end

   multiplicity = 'paral';
  set(handles.ed_info_USet, 'String', tmpstr);

  %title = 'Control area - globe';






% --- Executes on button press in pb_USphere.
function pb_USphere_Callback(hObject, eventdata, handles)
% hObject    handle to pb_USphere (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
  global U_dimension;
  global radiusSphere;
  global centerSphere;

    global multiplicity;
  info_str = sprintf('Dimension %s', num2str(U_dimension));
  option.Resize = 'off';
  option.WindowStyle = 'normal';
  option.Interpreter = 'tex';
  dialogProperties = {'Input coordinate:', 'Input radius:'};
  tmp = '';
  for i = 1 : 1 : U_dimension
    tmp = strcat(tmp, sprintf('%d', 0));
  end

  defaultAnswer = {tmp, '1'};
  %title = 'Control area - globe';
  numlines = 2;
  answer = inputdlg(dialogProperties, info_str, numlines, defaultAnswer, option);
  if (~isempty(answer))
    [sphere_coord, status1] = str2num(answer{1});
    [sphere_radius, status2] = str2num(answer{2});
```

```
    input_size = size(sphere_coord);

    if ((~status1) || (input_size(2) ~= U_dimension))
      errordlg('Input error or error dimension', 'Coordinates of the ball')
    elseif ((~status2))
      errordlg('Input radius error', 'Radius of the ball')
    else
      radiusSphere = sphere_radius
      centerSphere = sphere_coord;


    end


    tmpstr = {sprintf('Сфера\nРадиус сферы: %.2f\n Центр:', radiusSphere )};
    for i = 1 : 1 : U_dimension
      str = sprintf('%.2f, ', sphere_coord(i));
      tmpstr{i+1} = str;
    end

    set(handles.ed_info_USet, 'String', tmpstr)
    multiplicity = 'spher';
  end



function ed_LeftEnd_Callback(hObject, eventdata, handles)
% hObject    handle to ed_LeftEnd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

  global left;
  left = str2double(get(hObject,'String'));
  if (isnan(left))
    left = 0;
  end
  set(handles.ed_LeftEnd, 'String', num2str(left));


% --- Executes during object creation, after setting all properties.
function ed_LeftEnd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ed_LeftEnd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function ed_rightEnd_Callback(hObject, eventdata, handles)
% hObject    handle to ed_rightEnd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

  global right;
  right = str2double(get(hObject,'String'));
  if (isnan(right))
    right = 1;
  end

  set(handles.ed_rightEnd, 'String', num2str(right));


% --- Executes during object creation, after setting all properties.
function ed_rightEnd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ed_rightEnd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ed_TimeStep_Callback(hObject, eventdata, handles)
% hObject    handle to ed_TimeStep (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

  global step;
  step = str2double(get(hObject,'String'));
  if (isnan(step))
    step = 0;
  end

  set(handles.ed_TimeStep, 'String', num2str(step));



% --- Executes during object creation, after setting all properties.
function ed_TimeStep_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ed_TimeStep (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_Functional_Callback(hObject, eventdata, handles)
% hObject    handle to edit_Functional (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints:  returns contents of edit_Functional as text
%        str2double(get(hObject,'String')) returns contents of edit_Functional as a double


  global functional;
  functional = get(hObject,'String');


% --- Executes during object creation, after setting all properties.
function edit_Functional_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_Functional (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pb_ChooseMethod_ProjectionGrad.
function pb_ChooseMethod_ProjectionGrad_Callback(hObject, eventdata, handles)
% hObject    handle to pb_ChooseMethod_ProjectionGrad (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% A lot of methods

% ----------------------------------------------------------------------------------method p

  global method;
global alpha_MPG;
```

```
  infoStr  = {'Input alpha', 'Input lambda'};
  dlg_title = 'Parametrs alpha and lambda';
  num_lines = 1;

  def = {'0.1', '10'};

  answer = inputdlg(infoStr, dlg_title, num_lines, def);
  alpha_MPG = str2num(answer{1});


  tmpstr = sprintf('alpha метода проекции градиента= %.2f', alpha_MPG);
  set(handles.ed_info_Method, 'String', tmpstr);
  method = 'MPG';






% --- Executes on button press in pb_ChooseMethod_Approxim.
function pb_ChooseMethod_Approxim_Callback(hObject, eventdata, handles)
% hObject    handle to pb_ChooseMethod_Approxim (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global method; global numIter;
  global M_MPP; global E_MPP;

infoStr  = {'Input M', 'Input eps'};
  dlg_title = 'Parametrs M and epsilon';
  num_lines = 1;

  def = {'10', '0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1,'};

  answer = inputdlg(infoStr, dlg_title, num_lines, def);
  M_MPP = str2num(answer{1});
  E_MPP = ones(1, numIter);
  E_MPP = str2num(answer{2});


  tmpstr = {sprintf('M (количество итераций) метода последовательных приближений = %d\nEk:\n
    for i = 1 : 1 : numIter
      str = sprintf('%.2f, ', E_MPP(i));
      tmpstr{i+1} = str;
    end

    set(handles.ed_info_Method, 'String', tmpstr)

  method = 'MPP';
```

```
% -------------------------------------------------------------------------------------method p

% --- Executes on button press in pb_HoldRightEnd.
function pb_HoldRightEnd_Callback(hObject, eventdata, handles)
% hObject    handle to pb_HoldRightEnd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
  global HoldRightEnd; global betaHRE;
  global boolHRE; global dimension;


  info_str = sprintf('Dimension = %s', num2str(dimension));
  option.Resize = 'on';
  option.WindowStyle = 'normal';
  option.Interpreter = 'tex';
  HoldRightEnd = zeros(1, dimension);

  dialogProperties{1} = sprintf('x(T)=');
  defaultAnswer = {'1'};


  num_lines = 1;
  answer = inputdlg(dialogProperties, info_str,  num_lines, defaultAnswer, option);
  HoldRightEnd = str2num(answer{1})


  if (~isempty(answer))
    dial_Prop{1} = sprintf('Beta'); dial_def{1} = sprintf('%f', 10);
    num_lines = 1;
    answer = inputdlg(dial_Prop, info_str,  num_lines, dial_def, option);
    betaHRE = str2double(answer{1})
    boolHRE = true;
  else
    boolHRE = false;
  end


% --------------------------------------------------------------------------
function menu_File_Callback(hObject, eventdata, handles)
% hObject    handle to menu_File (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% ----------------------------------------------------------------------
function menu_Tools_Callback(hObject, eventdata, handles)
% hObject    handle to menu_Tools (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
function menu_FAQ_Callback(hObject, eventdata, handles)
% hObject    handle to menu_FAQ (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function menu_Help_Callback(hObject, eventdata, handles)
% hObject    handle to menu_Help (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Help


% --------------------------------------------------------------------
function menu_About_Callback(hObject, eventdata, handles)
% hObject    handle to menu_About (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

About


% --------------------------------------------------------------------
function menu_Start_Callback(hObject, eventdata, handles)
% hObject    handle to menu_Start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function menu_Stop_Callback(hObject, eventdata, handles)
% hObject    handle to menu_Stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function menu_Open_Callback(hObject, eventdata, handles)
% hObject    handle to menu_Open (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


  global dataFX; global dataU0; global dimension; global U_dimension;
  global step; global right; global left; global functional;
```

```
    [FName, PName] = uigetfile;
    if ~ isequal(FName, 0)
      FullName = strcat(PName, FName);
      S = load(FullName);
      f = fieldnames(S);
dimension =  str2num(S.(f{3}));
      set(handles.edit_Dimension, 'String', S.(f{3}));
U_dimension =  str2num(S.(f{4}));
      set(handles.ed_U_dimension, 'String', S.(f{4}));
left =  str2double(S.(f{6}));
      set(handles.ed_LeftEnd,     'String', S.(f{6}));
step =  str2double(S.(f{5}));
      set(handles.ed_TimeStep,    'String', S.(f{5}));
right = str2double( S.(f{7}));
      set(handles.ed_rightEnd,    'String', S.(f{7}));
dataFX =  S.(f{1});
      set(handles.ut_FunctionAndT,  'Data', S.(f{1}));
dataU0 =  S.(f{2});
      set(handles.ut_functionU0,    'Data', S.(f{2}));

     functional = S.(f{8});
     set(handles.edit_Functional, 'String',  (S.(f{8})));

      dataU0 = dataU0(:, 2);
    dataFX = dataFX(:, 2);


    end

% --------------------------------------------------------------------
function menu_Save_Callback(hObject, eventdata, handles)
% hObject    handle to menu_Save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function menu_SaveAs_Callback(hObject, eventdata, handles)
% hObject    handle to menu_SaveAs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


  dataFX = get(handles.ut_FunctionAndT, 'Data');
  dataU0 = get(handles.ut_functionU0, 'Data');
  dimension = get(handles.edit_Dimension, 'String');
  U_dimension = get(handles.ed_U_dimension, 'String');
  step = get(handles.ed_TimeStep,    'String');
  left = get(handles.ed_LeftEnd,     'String');
```

```
  right = get(handles.ed_rightEnd,     'String');
  functional = get(handles.edit_Functional, 'String');
  uisave({'dimension', 'U_dimension', 'left', 'step', 'right', 'dataFX', 'dataU0', 'functiona

% ----------------------------------------------------------------------
function menu_Exit_Callback(hObject, eventdata, handles)
% hObject    handle to menu_Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(0,'ShowHiddenHandles','on')
delete(get(0,'Children'))
close all;



function ed_U_dimension_Callback(hObject, eventdata, handles)
% hObject    handle to ed_U_dimension (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ed_U_dimension as text
%        str2double(get(hObject,'String')) returns contents of ed_U_dimension as a double

  global U_dimension; global dimension;
  U_dimension = str2double(get(hObject,'String'));

  if (isnan(U_dimension))
    U_dimension = 2;
  end
  [dataFX, dataU0] = refreshDataTable(dimension, U_dimension);

  set(handles.ed_U_dimension, 'String', num2str(U_dimension));
  set(handles.ut_functionU0,   'Data', dataU0);


% --- Executes during object creation, after setting all properties.
function ed_U_dimension_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ed_U_dimension (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


%-------------/// INFO /// ----------------/// INFO /// ----------------
function ed_info_StopCondition_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to ed_info_StopCondition (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ed_info_StopCondition as text
%        str2double(get(hObject,'String')) returns contents of ed_info_StopCondition as a dou


% --- Executes during object creation, after setting all properties.
function ed_info_StopCondition_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ed_info_StopCondition (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function ed_info_USet_Callback(hObject, eventdata, handles)
% hObject    handle to ed_info_USet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ed_info_USet as text
%        str2double(get(hObject,'String')) returns contents of ed_info_USet as a double


% --- Executes during object creation, after setting all properties.
function ed_info_USet_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ed_info_USet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor')
    set(hObject,'BackgroundColor','white');
end


function ed_info_Method_Callback(hObject, eventdata, handles)
% hObject    handle to ed_info_Method (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ed_info_Method as text
%        str2double(get(hObject,'String')) returns contents of ed_info_Method as a double
```

```
% --- Executes during object creation, after setting all properties.
function ed_info_Method_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit_term_functional_Callback(hObject, eventdata, handles)

  global term_functional;
  term_functional = get(hObject,'String');


% --- Executes during object creation, after setting all properties.
function edit_term_functional_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in cb_StopCond_UK.
function cb_StopCond_UK_Callback(hObject, eventdata, handles)
% hObject    handle to cb_StopCond_UK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cb_StopCond_UK



global stopJ;
global stopU;

global eps_stopU;

stopU =  get(hObject,'Value');
  info_str = sprintf('Epsilon');

  dialogProperties{1} = sprintf('eps=');
  defaultAnswer = {'0.01'};


  num_lines = 1;
  get(hObject,'Value')

  if stopU == 1
    answer = inputdlg(dialogProperties, info_str,  num_lines, defaultAnswer);
```

```
  eps_stopU = str2num(answer{1})
end
```

Исходный код части, отвечающей за визуализацию:

```
function varargout = drawPlot(varargin)
% DRAWPLOT MATLAB code for drawPlot.fig
%      DRAWPLOT, by itself, creates a new DRAWPLOT or raises the existing
%      singleton*.
%
%      H = DRAWPLOT returns the handle to a new DRAWPLOT or the handle to
%      the existing singleton*.
%
%      DRAWPLOT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DRAWPLOT.M with the given input arguments.
%
%      DRAWPLOT('Property','Value',...) creates a new DRAWPLOT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before drawPlot_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to drawPlot_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help drawPlot

% Last Modified by GUIDE v2.5 03-Dec-2013 02:58:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @drawPlot_OpeningFcn, ...
                   'gui_OutputFcn',  @drawPlot_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
% End initialization code - DO NOT EDIT


% --- Executes just before drawPlot is made visible.
function drawPlot_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to drawPlot (see VARARGIN)

% Choose default command line output for drawPlot
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes drawPlot wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = drawPlot_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

  global all_odeX;
  global all_W;
  global tEND;
  global all_gridValueU0;
  global invT; global alpha_MPG;

  global calcTime;
 global znach_function;
  set(handles.rb_Single, 'Value', 1);

  axes(handles.conjucatePlot);
  plot(tEND, cell2mat(all_W{end}));

  axes(handles.ukPlot);
  plot(tEND, cell2mat(all_gridValueU0{end}));

  axes(handles.odexPlot);
  plot(tEND, cell2mat(all_odeX{end}));


  resFunct = 0;%trapz(tEND, cell2mat(all_odeX{end}

  str = sprintf('alpha: %.3f\n  Значение функционала: %.6f\n Время вычислений: %.3f', alpha_I
  set(handles.editInfo, 'String', str);
```

```
function editInfo_Callback(hObject, eventdata, handles)
% hObject    handle to editInfo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editInfo as text
%        str2double(get(hObject,'String')) returns contents of editInfo as a double


% --- Executes during object creation, after setting all properties.
function editInfo_CreateFcn(hObject, eventdata, handles)
% hObject    handle to editInfo (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_exit.
function PB_exit_Callback(hObject, eventdata, handles)
% hObject    handle to PB_exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


close (handles.figure1);


% --- Executes on button press in rb_Multi.
function rb_Multi_Callback(hObject, eventdata, handles)
% hObject    handle to rb_Multi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of rb_Multi

 multi = get(hObject,'Value');

   global all_odeX;
  global all_W;
  global tEND;
  global all_gridValueU0;
  global invT; global alpha_MPG;
```

```
 global numIter; global calcTime;

  numIter = size(all_W);
  numIter = numIter(2)
 if multi == 1
set(handles.rb_Single, 'Value', 0);
 %  axes(handles.conjucatePlot);
 %  cla;
 %  axes(handles.ukPlot);
 %  cla;
 %  axes(handles.odexPlot)
 %  cla;

    set(handles.conjucatePlot, 'NextPlot', 'add');
    axes(handles.conjucatePlot);


    for i = 1 : 1 : numIter
       plot(tEND, cell2mat(all_W{1, i}));
     end

     set(handles.ukPlot, 'NextPlot', 'add');
     axes(handles.ukPlot);

     for i = 1 : 1 : numIter
       plot(tEND, cell2mat(all_gridValueU0{1, i}));
     end

     set(handles.odexPlot, 'NextPlot', 'add');
     axes(handles.odexPlot)
     for i = 1 : 1 : numIter
       plot(tEND, cell2mat(all_odeX{1, i}));
     end

  resFunct = trapz(tEND, cell2mat(all_odeX{end}));



 end



% --- Executes on button press in rb_Single.
function rb_Single_Callback(hObject, eventdata, handles)
% hObject    handle to rb_Single (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of rb_Single
```

```
global all_odeX;
  global all_W;
  global tEND;
  global all_gridValueU0;
  global invT; global alpha_MPG;

 global numIter; global calcTime;

  single = get(hObject,'Value') ;
  if single == 1

set(handles.rb_Multi, 'Value', 0);


 set(handles.conjucatePlot, 'NextPlot', 'replace');
  axes(handles.conjucatePlot);
  plot(tEND, cell2mat(all_W{end}));

   set(handles.ukPlot, 'NextPlot', 'replace');
  axes(handles.ukPlot);
  plot(tEND, cell2mat(all_gridValueU0{end}));

   set(handles.odexPlot, 'NextPlot', 'replace');
  axes(handles.odexPlot);
  plot(tEND, cell2mat(all_odeX{end}));
  resFunct = trapz(tEND, cell2mat(all_odeX{end}));

  end
```

6 Полученные результаты

В этом разделе описаны 7 результатов для 5 задач, решенных с помощью написанной программе.

Задачи 4 и 5 взяты из [2].

Для всех дифференциальных уравнений, получавшихся на разных этапа работы программы, был использован метод ode23. Также, если не указано иное, то шаг сетки разбиения временного интервала step = 0.01.

Задача 1

$$\begin{cases} \dot{x_1} = u_1 & t_0 = 0 \\ x_1(t_0) = 1 & T = 2 \\ u_0 = -1 \\ J(u) = x_1^2 \end{cases}$$

Задача 2 Задача с закрепленным правым концом.

$$\begin{cases} \dot{x_1} = u_1 & t_0 = 0 \\ x_1(t_0) = 1 & T = 2 \\ x(T) = 1 \\ u_0 = -1 \\ J(u) = x_1^2 \end{cases}$$

Задача 3 Трехмерная задача.

$$\begin{cases} \dot{x_1} = u_1 & t_0 = 0 \\ \dot{x_2} = u_2 \\ \dot{x_3} = u_3 \\ x_1(t_0) = 1 & T = 2 \\ x_2(t_0) = 1 \\ x_3(t_0) = 1 \\ J(u) = x_1^2 + x_2^2 + x_3^2 \end{cases}$$

Задача 4 Задача с интегральным и терминальным функционалом.

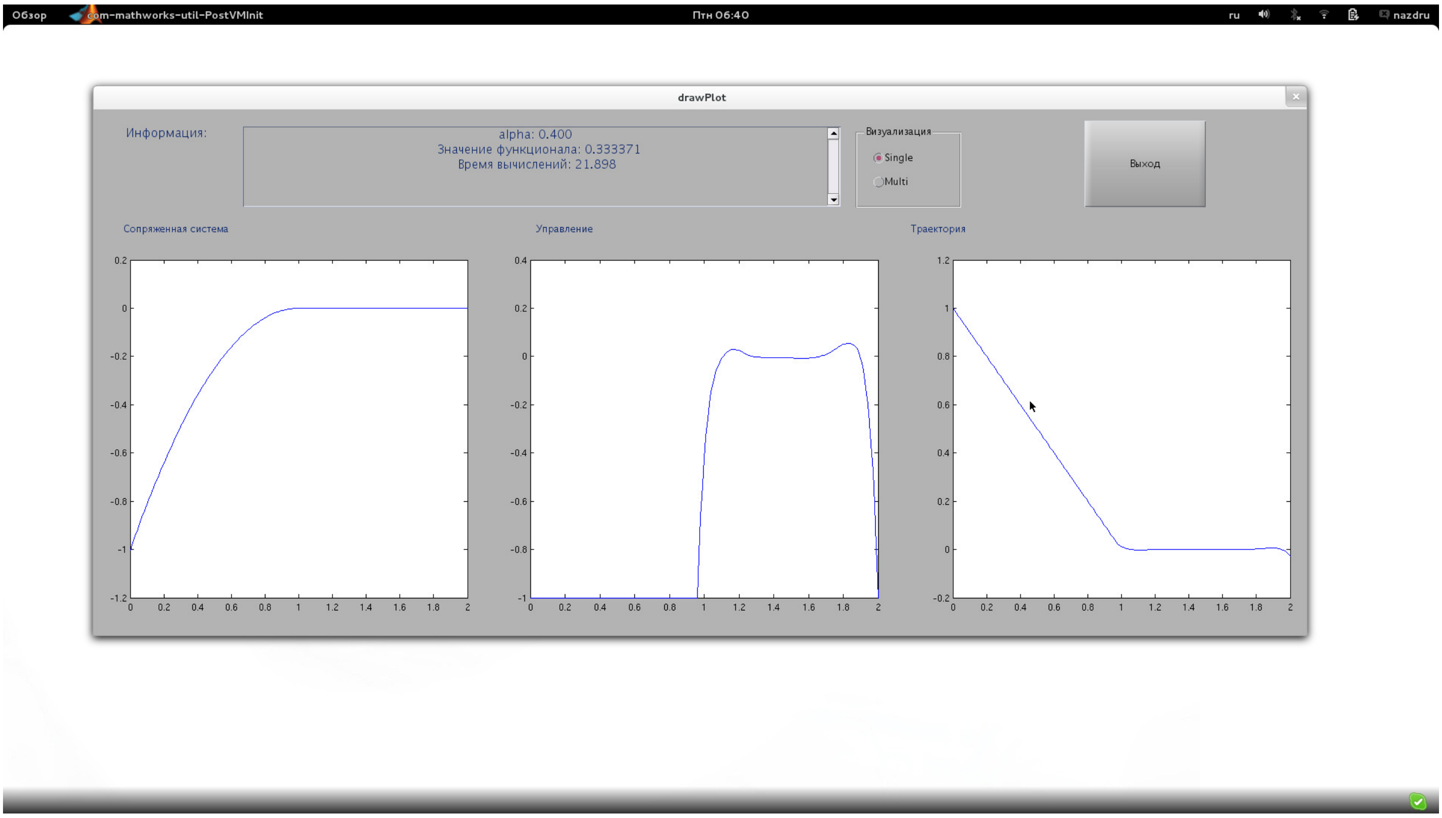
$$\begin{cases} \dot{x_1} = u_1 & t_0 = 0 \\ x_1(t_0) = 0 & T = 1 \\ J(u) = \int\limits_0^1 -x_1(t) + u_1^2(t)dt + x_1^2(1) \end{cases}$$

Задача 5 Задача с интегральным функционалом

$$\begin{cases} \dot{x_1} = u_1 & t_0 = 0 \\ x_1(t_0) = 0 & T = 1 \\ J(u) = \int\limits_0^4 -x_1(t) + u_1^2(t)dt \end{cases}$$

Метод проекции градиента для Задачи 1

(∗) $\alpha = 0.4$
(∗) *количество итераций 500*
(∗) $-1 \leq u \leq 1$
(∗) *полученное* $J(u) = 0.33337$



Метод последовательных приближений для Задачи 1

Параметры:

(∗) $\alpha = 1$

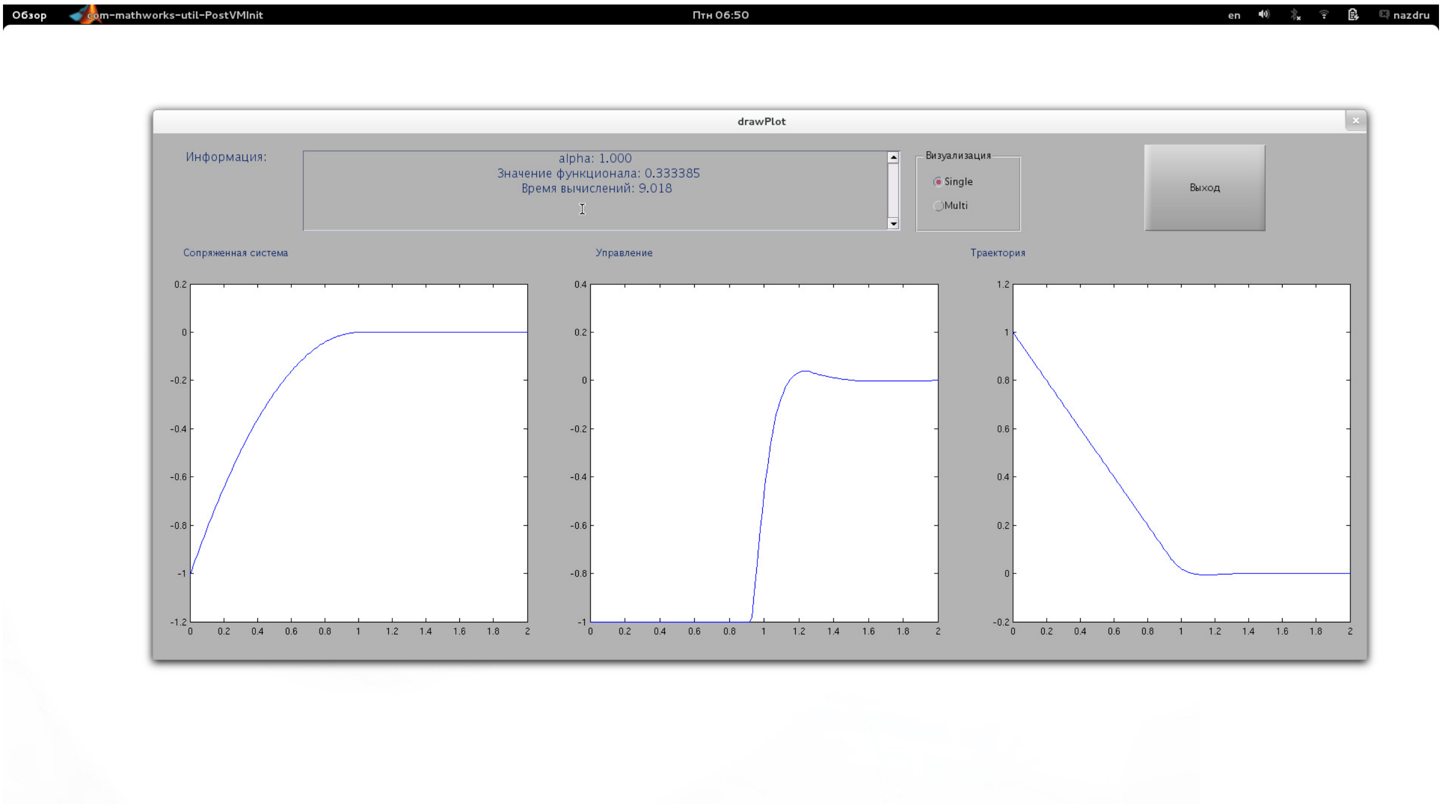
(∗) *количество внешних итераций 10*

(∗) *количество внутренних итераций 25*

(∗) *последовательность* $eps = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1$

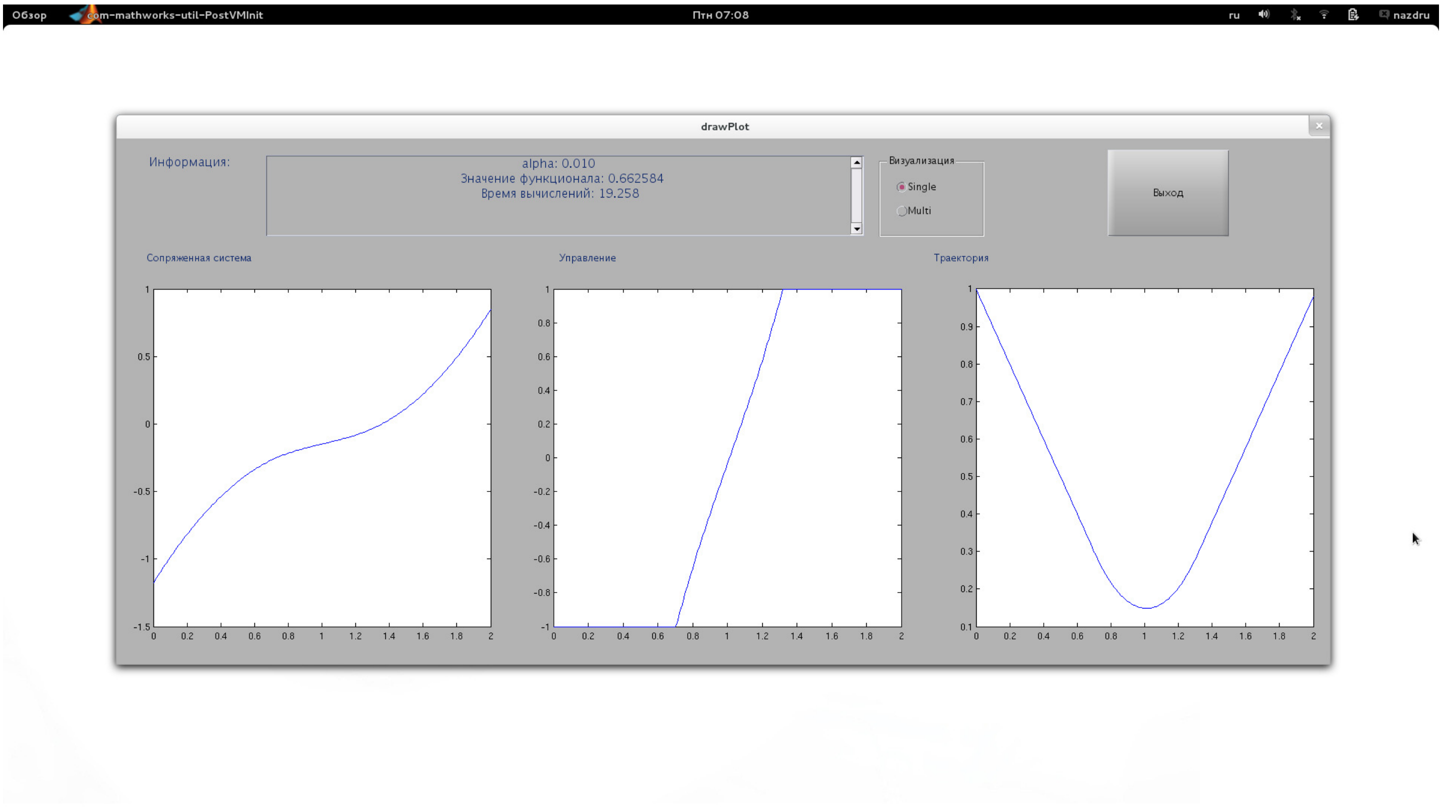
(∗) $-1 \le u \le 1$

(∗) $J(u) = 0.333385$



Метод проекции градиента для Задачи 2

(∗) $\alpha = 0.01$
(∗) $\beta = 20$
(∗) *количество итераций 500*
(∗) $-1 \le u \le 1$



Метод последовательных приближений для Задачи 2

Параметры:

(∗) $\alpha = 0.01$

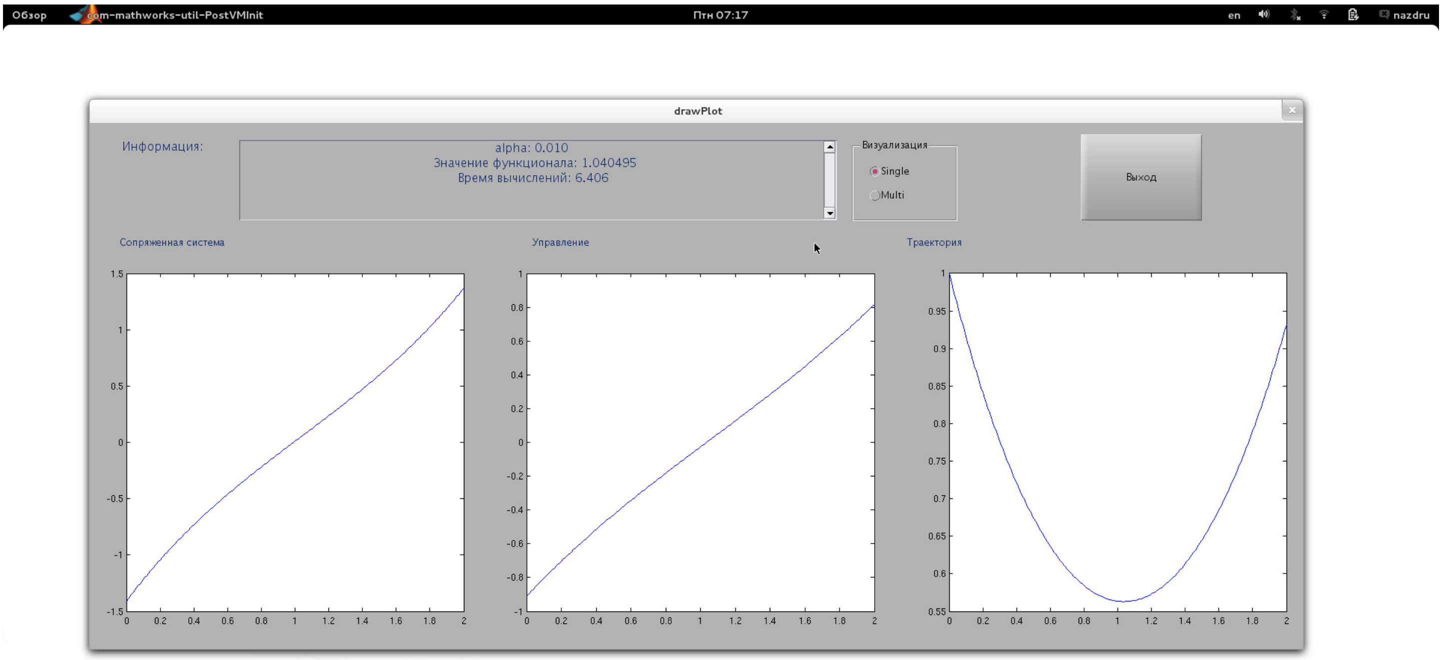
(∗) *количество внешних итераций 10*

(∗) *количество внутренних итераций 25*

(∗) $\beta = 10$

(∗) *последовательность* $eps = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1$

(∗) $-1 \leq u \leq 1$



Метод проекции градиента для Задачи 3

$(*)\ \alpha = 0.4$

$(*)$ *количество итераций 450*

$(*)\ -1 \leq u_1 \leq 1$

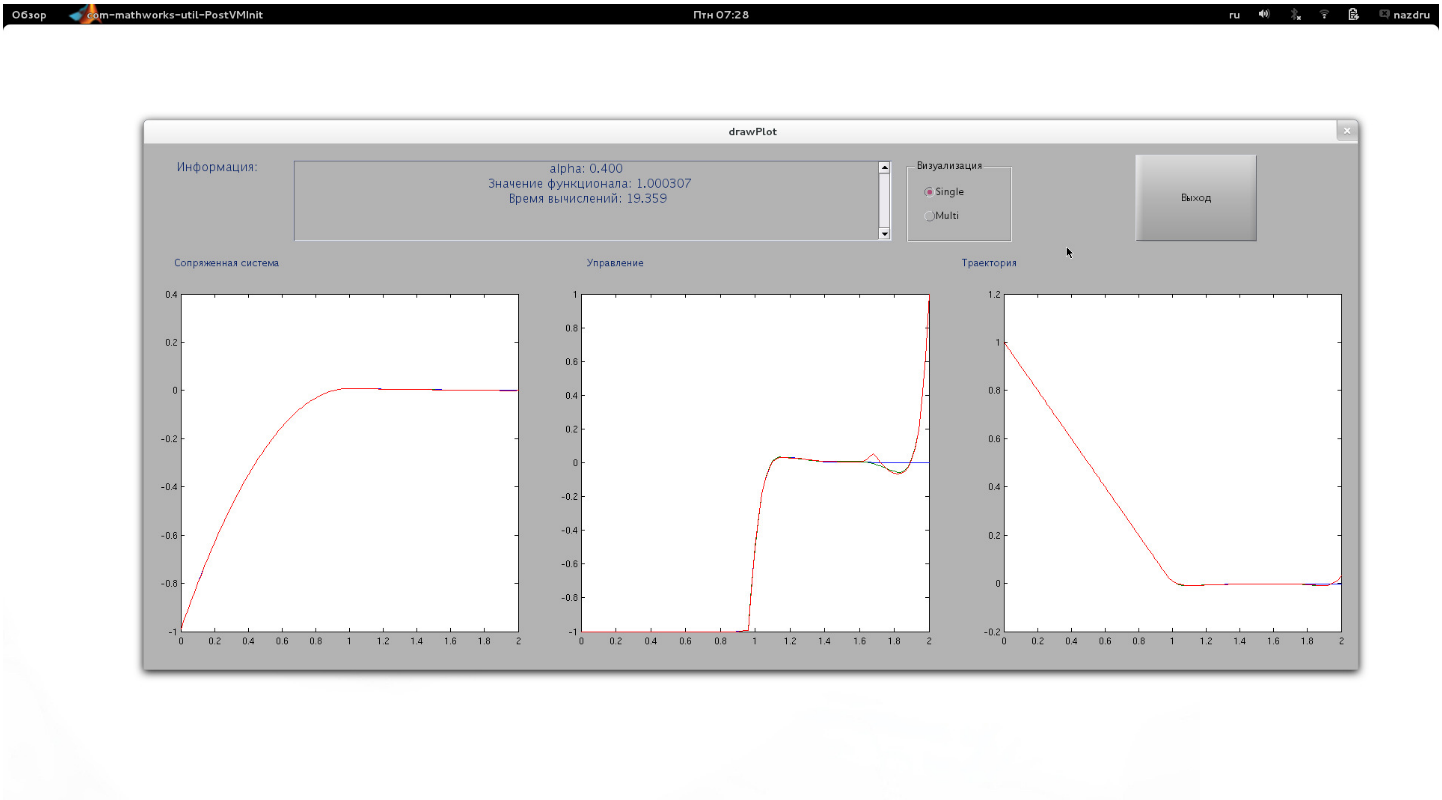
$(*)\ -1 \leq u_2 \leq 1$

$(*)\ -1 \leq u_3 \leq 1$

$u_1 0 = 0$

$u_2 0 = 1$

$u_3 0 = t$

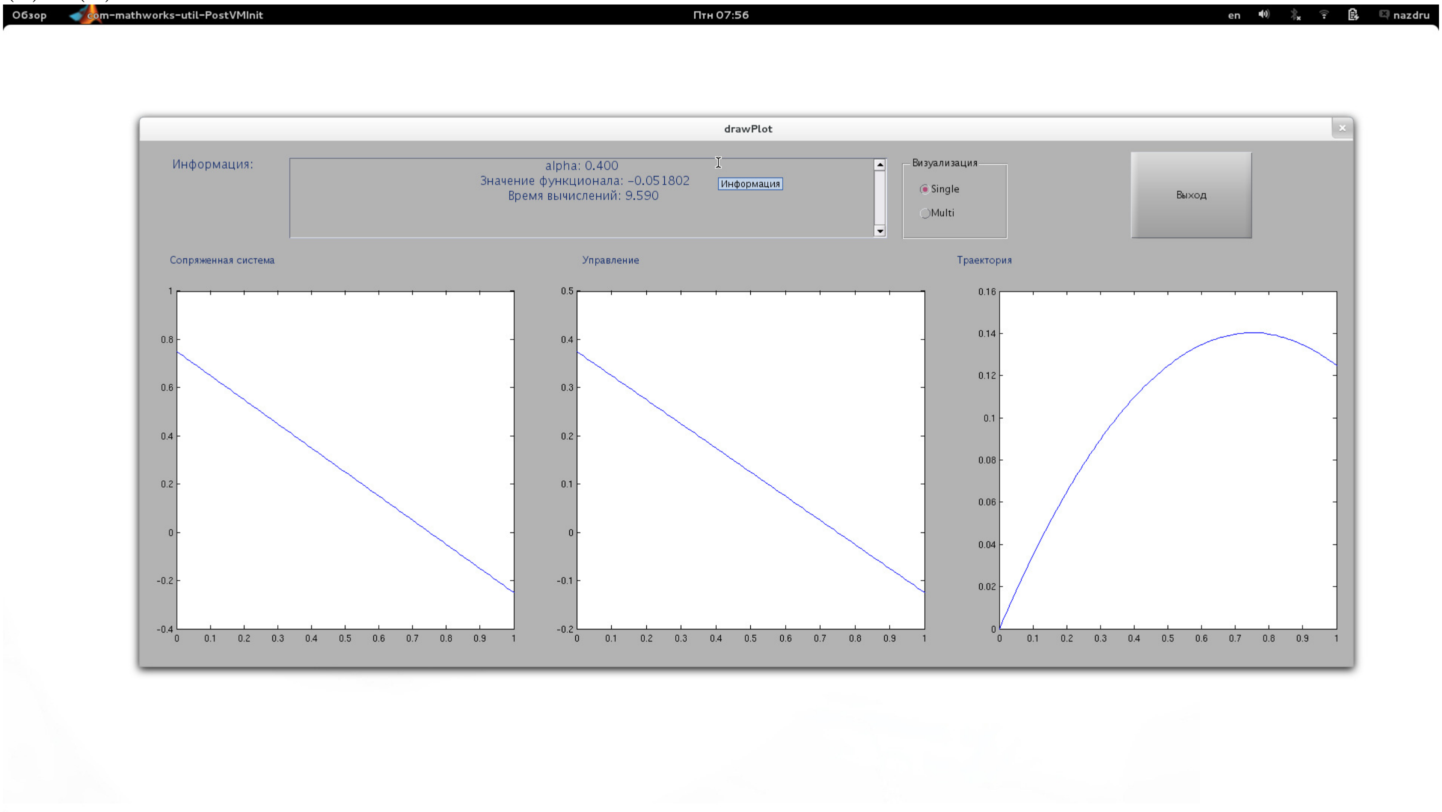


Метод проекции градиента для Задачи 4

(∗) $\alpha = 0.4$

(∗) *количество итераций 450*

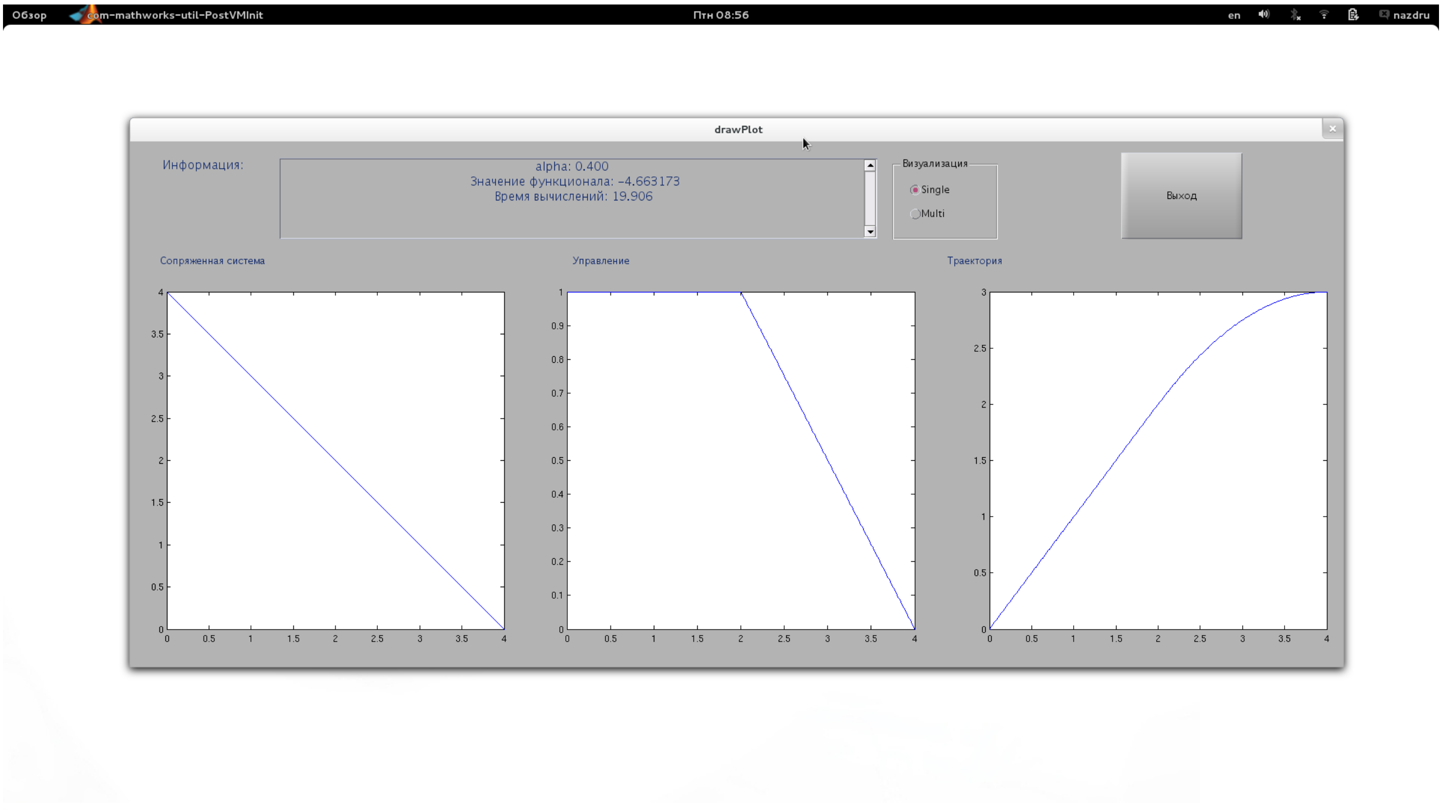
(∗) $J(u) = -0.05$



Метод проекции градиента для Задачи 5

(∗) $\alpha = 0.4$

(∗) *количество итераций 245*



Список литературы

[1] **Бейко И.В., Бублик Б.Н., Зинько П.Н.** Методы и алгоритмы решения задач оптимизации. Вища школа, 1983.

[2] **Rovenskaya E.A., Kamzolkin D.V.** Infinite Horizon Optimal Control with Applications in Growth Theory: Practical Guide. MSU CMC Publication Department, MAKS Press, Moscow, Russia, 2009.

[3] **Киселев Ю.Н., Аввакумов С.Н., Орлов М.В.** Оптимальное управление. Линейная теория и приложения. Учебное пособие для студентов ВМиК МГУ, Москва 2007

[4] **Понтрягин Л.С., Болтянский В.Г., Гамкрелидзе Р.В., Мищенко Е.Ф.** Математическая теория оптимальных процессов . — М.:Наука, 1961.

[5] **Васильев Ф.П.** Методы оптимизации.М.Факториал Пресс: 2002.