|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

Факультет: «Специальное машиностроение»

Кафедра: «Робототехнические системы и мехатроника»

**Лабораторная работа № 2**

по курсу «Теория автоматического управления»

Вариант 4

Выполнил: Давыдов В.Ю.

Группа: СМ7-62Б

Проверил:

*Москва, 2024г.*

Обязательная часть.

1. Работа с фазовыми портретами двумерных систем.

Построение фазового портрета варианта mod((4+3), 6) + 1 = 2.

Уравнение вида: 

Приведём его к виду:

Данные уравнения реализованы в функции MATLAB:

|  |
| --- |
| dynamics.m  function dxdt = dynamics(t,x)  dxdt = [x(2); 5 \* x(2) ^ 2 + x(1) \* (x(1) - 1)];  end |

Файл функции обнаружения выхода за пределы 2D поля, взят из программы, показанной на лабораторной работе.

Функция рисования стрелок и фазовых траекторий реализована в виде кода:

|  |
| --- |
| function phasePortrait(f, XMAX, YMAX, STEP, TMAX, event\_fnc)  [x, y] = meshgrid(-XMAX:STEP:XMAX, -YMAX:STEP:YMAX);    [d1, d2] = size(x);  u = zeros(d1);  v = zeros(d1);  t = 0;  for i = 1:d1  for j = 1:d2  YPrime = f(t, [x(i,j), y(i,j)]);  u(i, j) = YPrime(1);  v(i, j) = YPrime(2);  end  end    % Рисование стрелок  for i = 1:d1  for j = 1:d2  Vmod = sqrt(u(i, j)^2 + v(i,j)^2);  u(i, j) = u(i, j) / Vmod;  v(i, j) = v(i, j) / Vmod;  end  end    figure(1);  hold on;  quiver(x,y,u,v,0.5);    % Рисование фазовых траекторий  for i = 1:d1  for j = 1:d2  x0 = [x(i,j), y(i,j)];  [~,z, ~,~,~] = ode23t(f,[0, TMAX],x0, odeset('RelTol',1e-3,'Events', event\_fnc));  plot(z(:,1),z(:,2));  end  end  end |

Время моделирования с использованием моего кода:



С использованием кода лабы:



Вывод: вычисления с использованием векторной алгебры проходят быстрее.

Полученный результат:

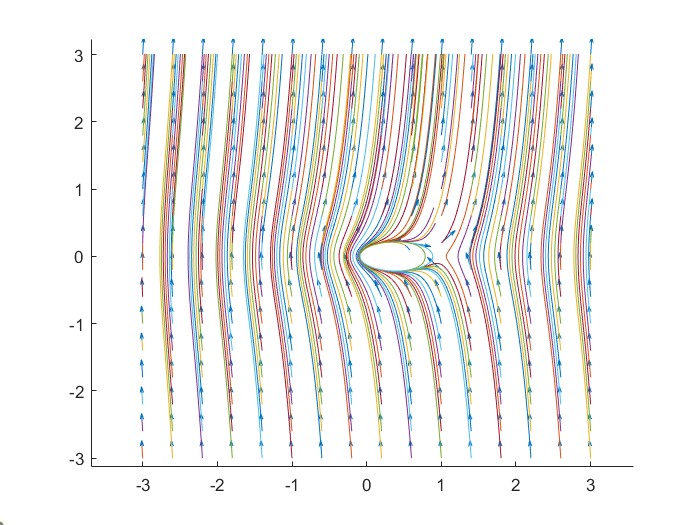


Рисунок 1, фазовый портрет исходного дифференциального уравнения

Построение фазового портрета с использованием Simulink модели: будет использоваться код лабораторной работы.

Для запуска модели был изменён код запуска систем, а именно поменялось название субсистемы.



А также дифференциальное уравнение было представлено в виде структурной схемы.

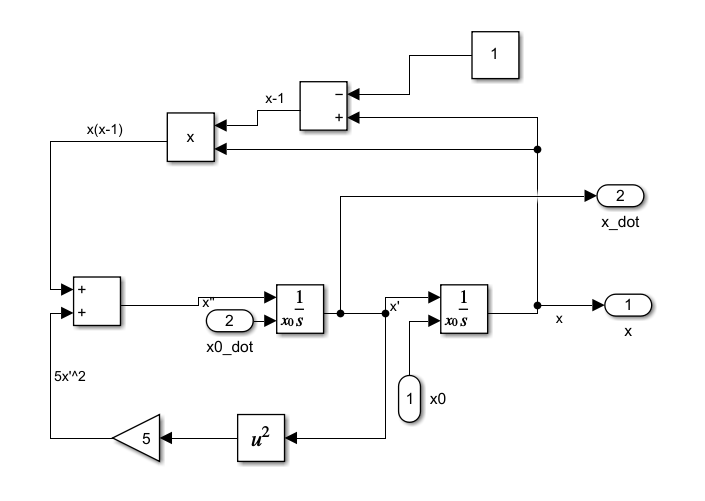


Рисунок 2, дифференциальное уравнение, представленное в виде блоков Simulink

В итоге был получен фазовый портрет:

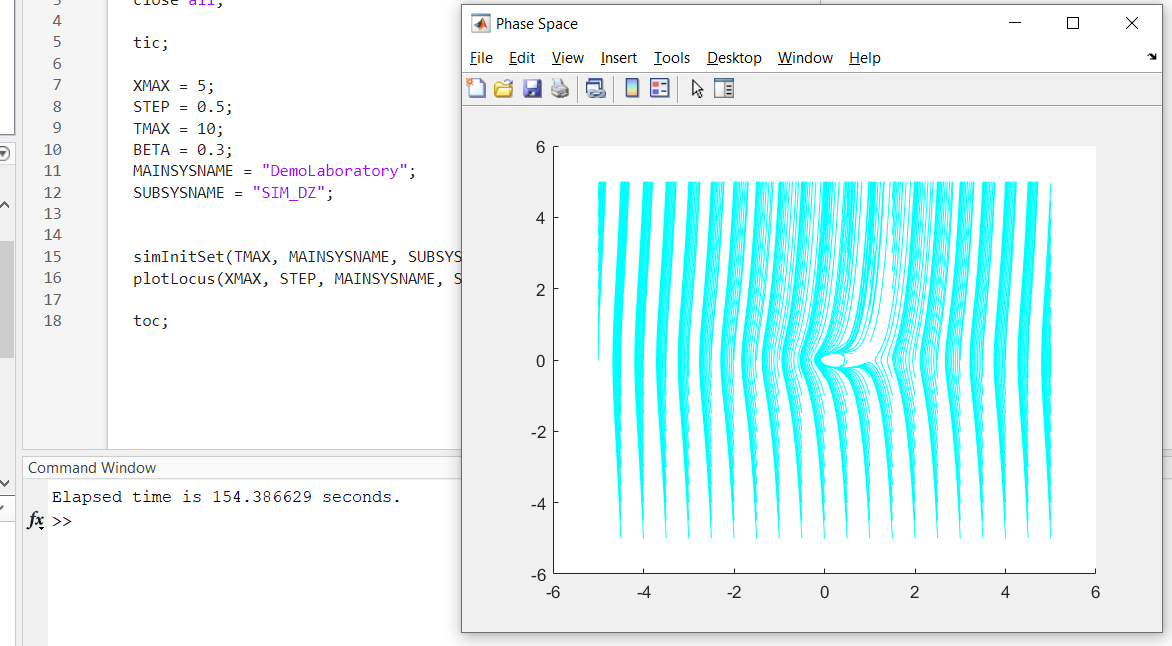


Рисунок 3, полученные графики

Он очень похож на полученные фазовые портреты, другим способом.

Особый точки можно найти из дифференциальных уравнений, приравняв производные x1 и x2 к 0. Полученная точка (0, 0) - центр, а также (1, 0) - седло

В окрестности особой точки фазовый портрет:

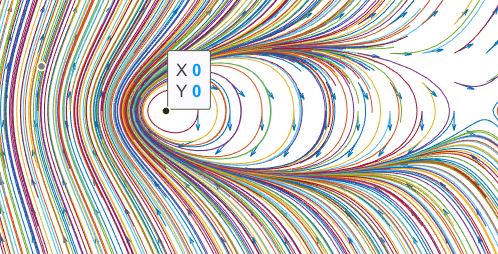


Рисунок 4, фазовый портрет в окрестности точки (0, 0)

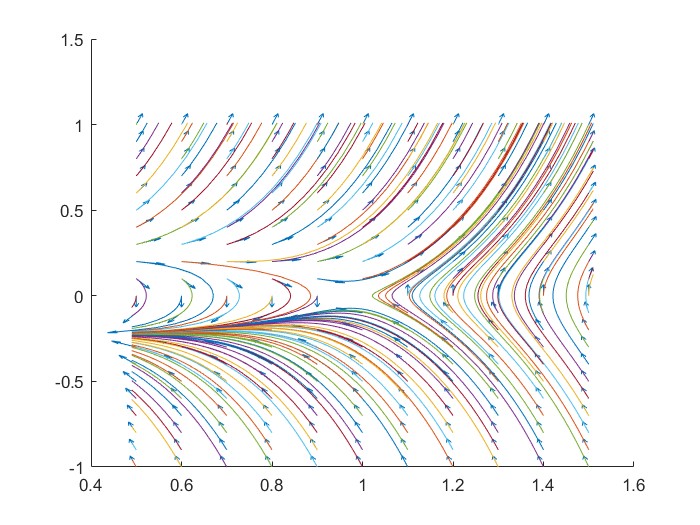


Рисунок 5, фазовый портрет в окрестности точки (1, 0)

Также были построены фазовые траектории с вырожденной особой точкой

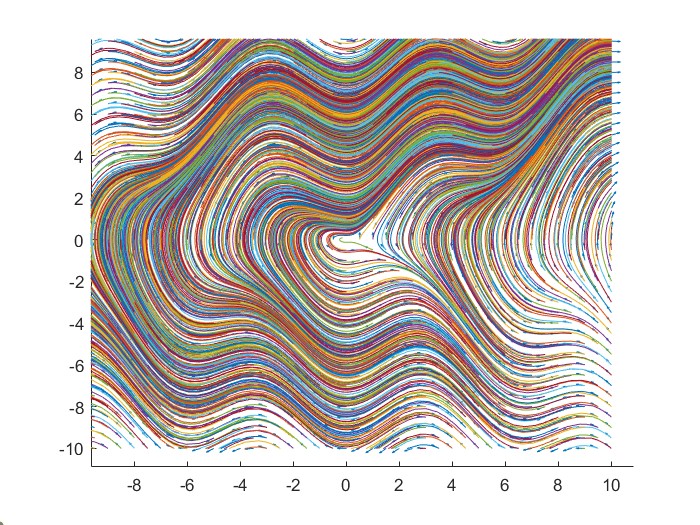


Рисунок 6, фазовый портрет с вырожденной особой точкой

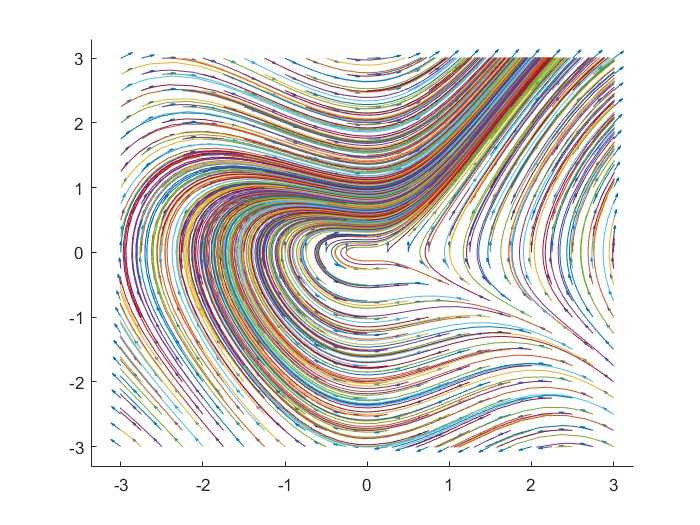


Рисунок 7, фазовый портрет с вырожденной особой точкой

И с континуумом особых точек

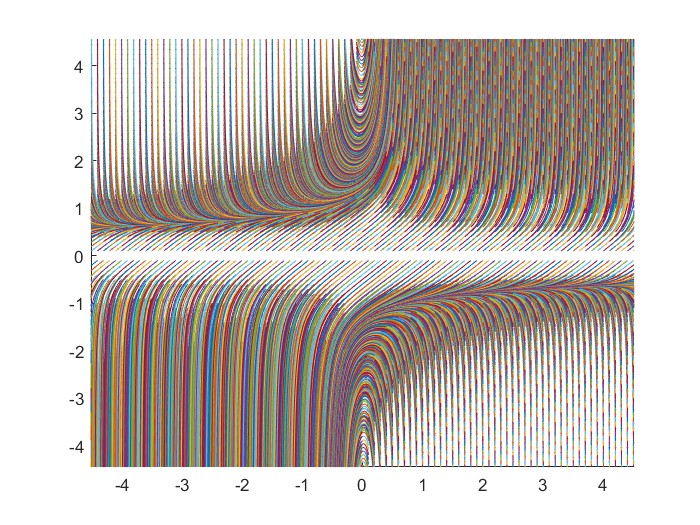


Рисунок 8, фазовый портрет с континуумом особых точек

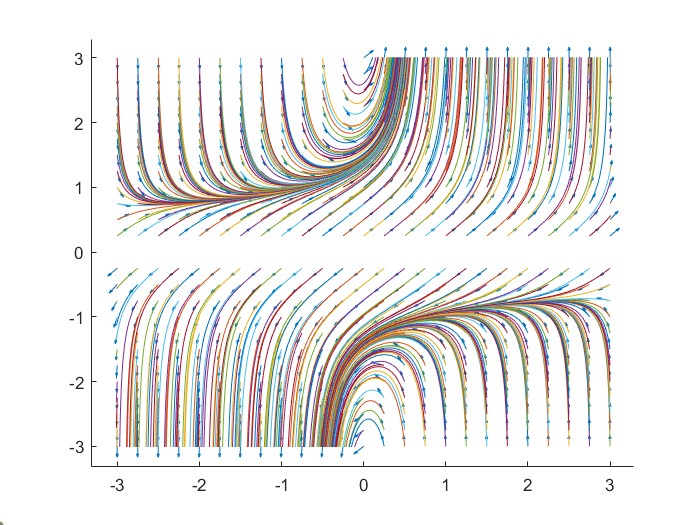


Рисунок 9, фазовый портрет с континуумом особых точек

Часть 2. Построение фазовых траекторий 3-х мерных систем

Ниже в таблице представлен код MATLAB реализуешь построение фазовой траектории 3-х мерной системы.

|  |
| --- |
| clc;  clear all;  close all;  %% Constants  TMAX = 500;  prov = 'B';  if prov == 'A'  %Var A;  gamma = 0.1;  alpha = 0.05;    X0 = 0.1;  Y0 = -0.1;  Z0 = 0.1;  else  % Var B  gamma = 0.87;  alpha = 1.1;    X0 = -1;  Y0 = 0;  Z0 = 0.5;  end  tspan = [0 TMAX];  xyz\_0 = [X0;Y0;Z0];  [t, v] = ode45(@(t, v) func\_V(t, v, gamma, alpha), tspan, xyz\_0, odeset('RelTol',1e-3));  plot3(v(:,1), v(:,2), v(:,3), LineWidth=1);  hold on;  plot3(X0,Y0,Z0, 'o');  xlabel('X');  ylabel('Y');  zlabel("Z");  function dVdt = func\_V(t, v, gamma, alpha)  dVdt = [v(2,:) \* (v(3, :) - 1 + v(1,:)^2) + gamma \* v(1,:); ....  v(1,:) \* (3 \* v(3,:) + 1 - v(1,:)^2) + gamma \* v(2,:); ...  -2 \* v(3,:) \* (alpha + v(1,:) \* v(2,:))];  end |

Решатель ode45 построил фазовые траектории, представленные ниже:

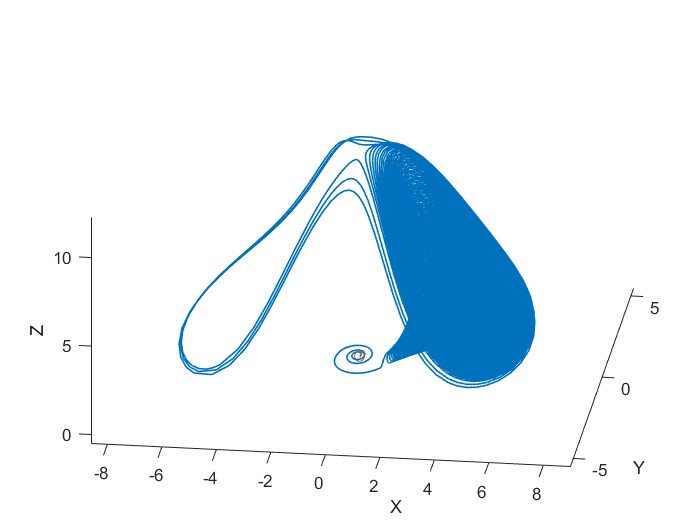


Рисунок 10, решатель ode45, 1-ые условия

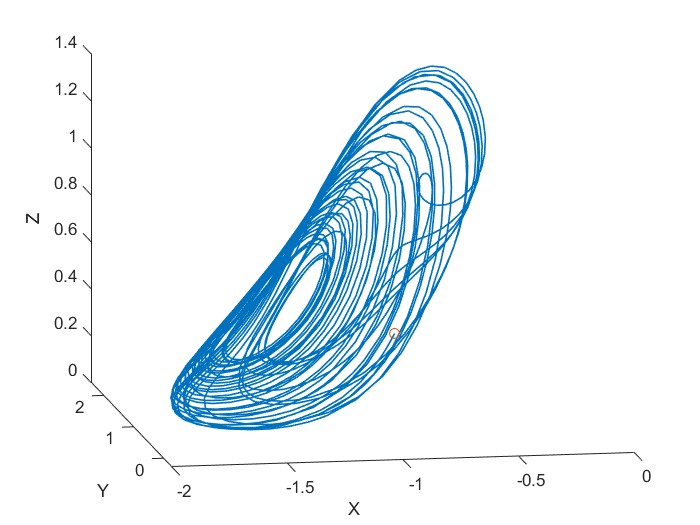


Рисунок 11, ode45, вторые начальные условия

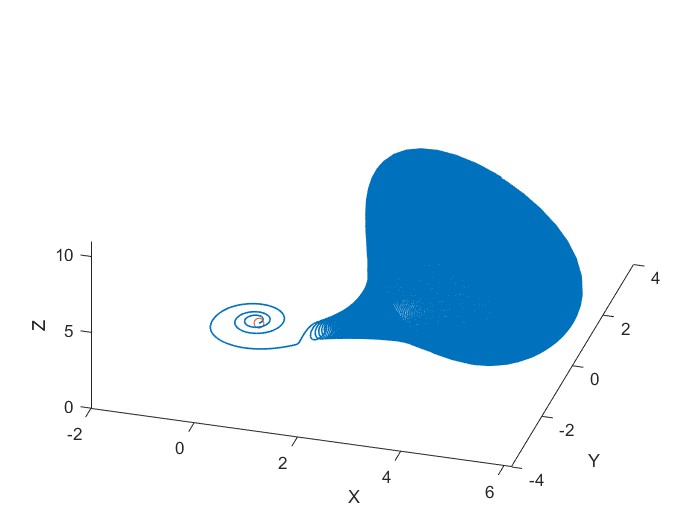


Рисунок 12, ode23t, первые начальные условия

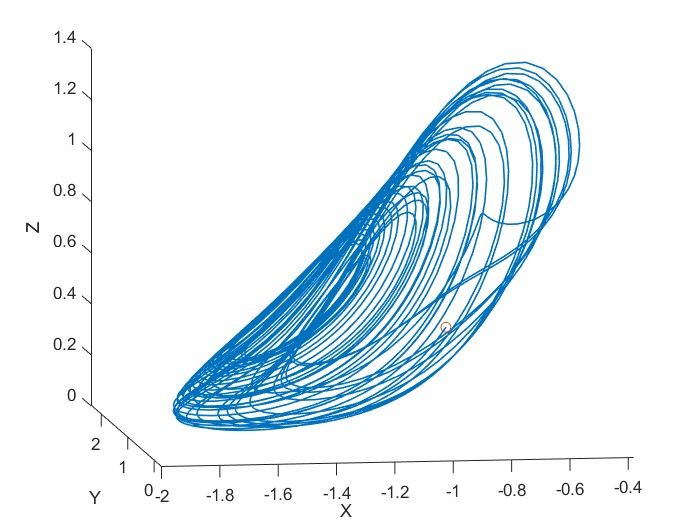


Рисунок 13, ode23t, вторые начальные условия

Как видно из графиков разные решатели примерно одинаково решают данную систему уравнений, но свойства системы, очень сильно зависят от начальных условий.

Часть 3. Построение фазовых траекторий нестационарных систем.

Исходя из результатов предыдущей задачи, хотелось бы сказать, что система хаотична из-за того что, при изменении натальных условий решение получается совершенно другим, с другой стороны при небольшом отклонении начальных условий решение получается практически идентичным.

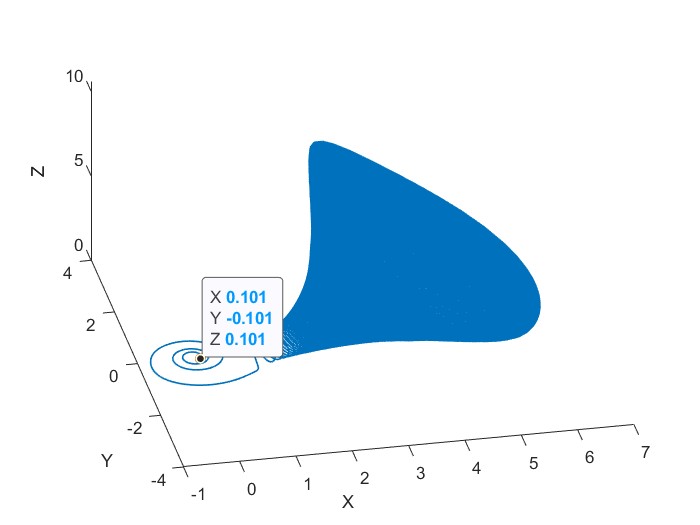


Рисунок 14, построение фазовой траектории при небольшом изменении начальных условий

Код, строящий решение дифференциального уравнения и направление поля, в окрестности точки решения представлен ниже:

|  |
| --- |
| Main.m % задаёт дифференциальные уравнения и вызывает функцию построения портрета  clc;  clear all;  close all;  XMAX = 5;  YMAX = 5;  TMAX = 10;  STEP = 0.5;  tic;  phasePortrait(@varC, TMAX);  toc;  function dxdt = var(t, x)  if mod(round(t - 0.5),2) == 0  dxdt = [x(2); -x(2) - x(1) + 1];  else  dxdt = [x(2); -x(2) - x(1) - 1];  end  end  function dxdt = varB(t, x)  if mod(round(t - 0.5),2) == 0  dxdt = [x(2,:); -x(2,:) - x(1,:) + t];  else  dxdt = [x(2,:); -x(2,:) - x(1,:) - t];  end  end  function dxdt = varC(t, x)  dxdt = [x(2,:); -x(2,:) - x(1,:) + sin(1 ./ (t + 10^-7))];  end |

Также ниже представлен код построения решения дифференциального уравнения.

|  |
| --- |
| function phasePortrait(f, TMAX)    figure(1);  hold on;    x0 = [0; 0];  [t,z] = ode23t(f,[0, TMAX],x0);  plot(z(:,1),z(:,2));  quiv = zeros(10,4);  for i = 1:size(z(:,1))  for j = 1:5  radx = rand / 1000 - 0.0005;  rady = rand / 1000 - 0.0005;  mas = f(t(i), [z(i,1) + radx; z(i,2) + rady]);  nm = sqrt(mas(1)^2 + mas(2)^2);  quiv(j, 1) = z(i,1) + radx;  quiv(j, 2) = z(i,2) + rady;  quiv(j, 3) = mas(1) / nm;  quiv(j, 4) = mas(2) / nm;  quiver(quiv(:,1), quiv(:,2), quiv(:,3), quiv(:,4), 0.5);  end  end    end |

Получившиеся графики:

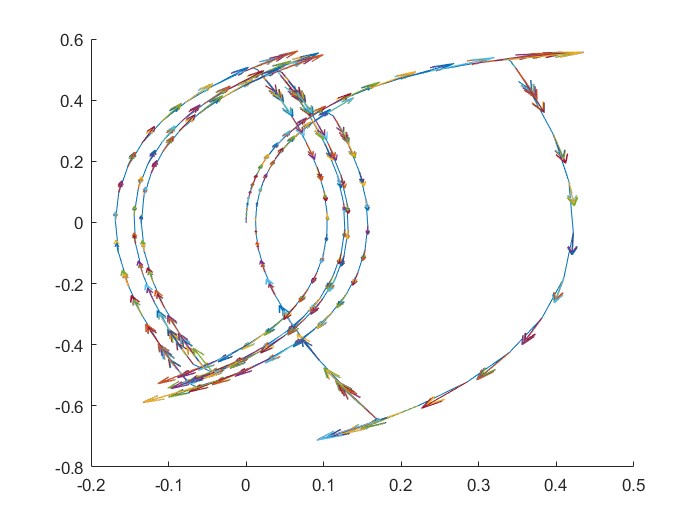


Рисунок 15, функция 1

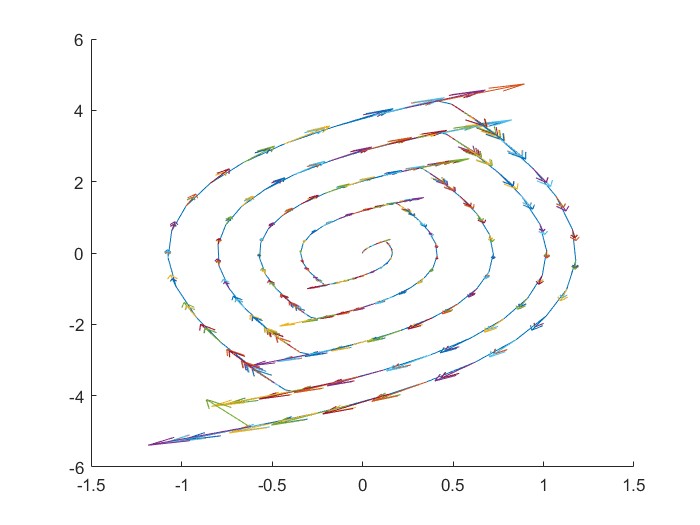


Рисунок 16, функция 2

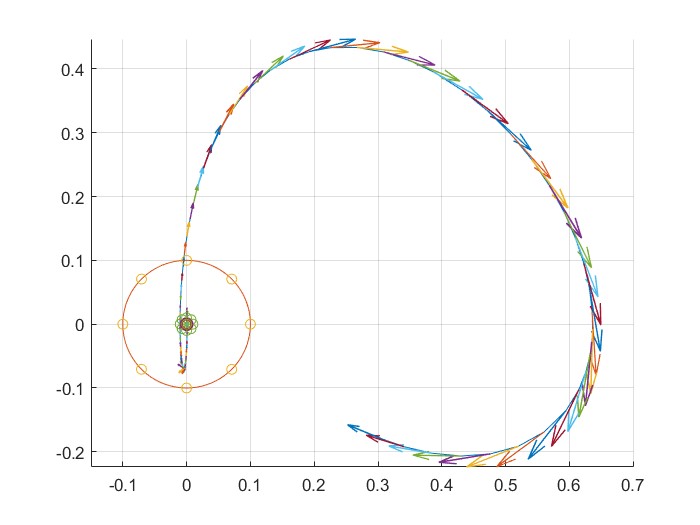


Рисунок 17, функция 3

Как видно из рисунков, при отклонении точки интегрирования от решения на небольшое расстояние, состояние системы остаётся практически таким же. То есть система не хаотична, либо она не сильно хаотична, относительно изменения координат начала интегрирования.

Расчёт хаотичности системы:

|  |
| --- |
| Haos.m  function otvet = Haos(f, x0, time, phase\_coord)    N0 = 8;  otvet = zeros(1, N0 + 1);  for n = 0:N0 % schet posled  delta\_posled = 0.1 / (10 ^ n); %Задание последовательности    theta = linspace(0,2 \* pi);  x = delta\_posled \* cos(theta) + x0(1);  y = delta\_posled \* sin(theta) + x0(2);  plot(x, y);  axis equal;  grid on;    theta = linspace(0, 2 \* pi, 9);  x\_resh = delta\_posled .\* cos(theta);  y\_resh = delta\_posled .\* sin(theta);  plot(x\_resh + x0(1), y\_resh + x0(2), 'o'); % построение окружностей  max\_znac = zeros(1, 8);  for j = 1:8  tmp = phase\_coord;  len1 = size(tmp(:,1));  init\_cond = [x\_resh(j) y\_resh(j)];  [~, v] = ode23t(f, time, init\_cond); %Решение диффуров относительно точки на окружности  len2 = size(v(:,1));  if len1(1) > len2(1)  tmp((len2(1) + 1):len1(1), :) = [];  elseif len2(1) > len1(1)  v((len1(1) + 1):len2(1), :) = []; %изменение размерности массива, для того, чтобы возможно было посчитать промежуточные значения  end  prom = (v - tmp) .^ 2;  max\_znac(j) = max(sqrt(prom(:,1) + prom(:,2)));  end  otvet(1, n + 1) = max(max\_znac);  end  end |

В результате моделирования противоположных импульсов получилась последовательность:

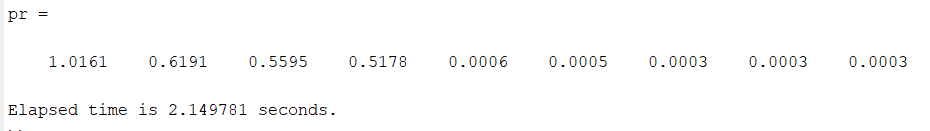


Рисунок 18, результат для противоположных импульсов

Как видно из рисунка, последовательность стремиться к 0, а значит система не хаотична.

Система, противоположные линейные кусочки:

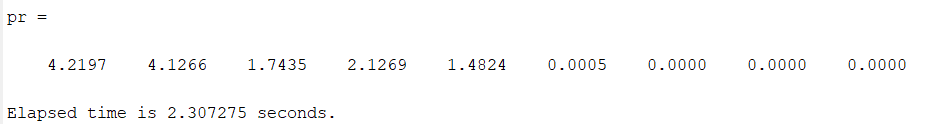


Рисунок 19, результат для противоположных линейных кусочков

Как видно из рисунка, последовательность стремиться к 0, а значит система не хаотична.

Система, весёлый синус:

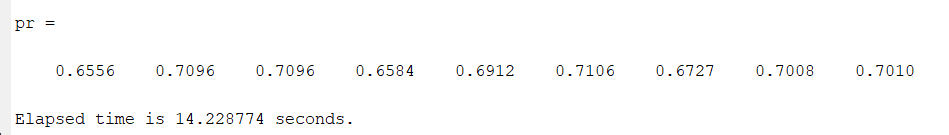


Рисунок 20, результаты весёлый синус

Как видно из рисунка, последовательность стремиться к 0.7, а значит система ограничено хаотичная.