|  |  |
| --- | --- |
| 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 5

Выполнил: Садовец Роман

Группа: СМ7-62Б

Проверил(а):

Москва, 2024 г.

1. **Работа с фазовыми портретами двумерной систем**
   1. **Построение фазового портера системы**

В данном пункте мы будем строить фазовый портрет функции, представленный в таблице 1.



Табл 1. Дифференциальное уравнение системы для 5-го варианта

Зададим для данного уравнения фазовые переменные:

Подставив переменные в исходное уравнение, а также переписав уравнение в виде системы (т.е. переписываем систему естественным образом), получим:

(1)

Следующим пунктом является отображение фазового портрета системы. Для этого потребуется адаптация прилагаемого к лабораторной работе файлов. В первую очередь запишем систему (1) в виде функции Matlab.

|  |
| --- |
| function dxdt = model(t, x)  % Define the nonlinear, continious-time, state-space model  % t is the time;  % x it the vector of state coordinates  dxdt(1,:) = x(2,:);  dxdt(2,:) = x(2,:) + x(1,:).^2 - 5 .\* sin( 0.3 \* x(1,:).^2);  end |

Воспользуемся готовыми скриптами отрисовки системы дифференциального уравнения, а именно файлами **прорисовки системы дифференциальных уравнений (plotLocus.m), прорисовки направлений векторов (направление передвижения) фазового портрета (plotQuiver.m), а также отслеживание выходов за пределы сетки (outOfBounds.m).** Все указанные файлы и наработки можно просмотреть на удаленном репозитории GitHub, директория Lab-2/2D-plot/Code (см. прил. 1). Важное замечание – файлы были немного скорректированы для более удобного отображения информации.

Назначаем размер сетки 5x5 с шагом 0.8. Далее запускаем программу для системы (1). Получим фазовый портрет, представленный на рисунке 1. Время симуляции составило

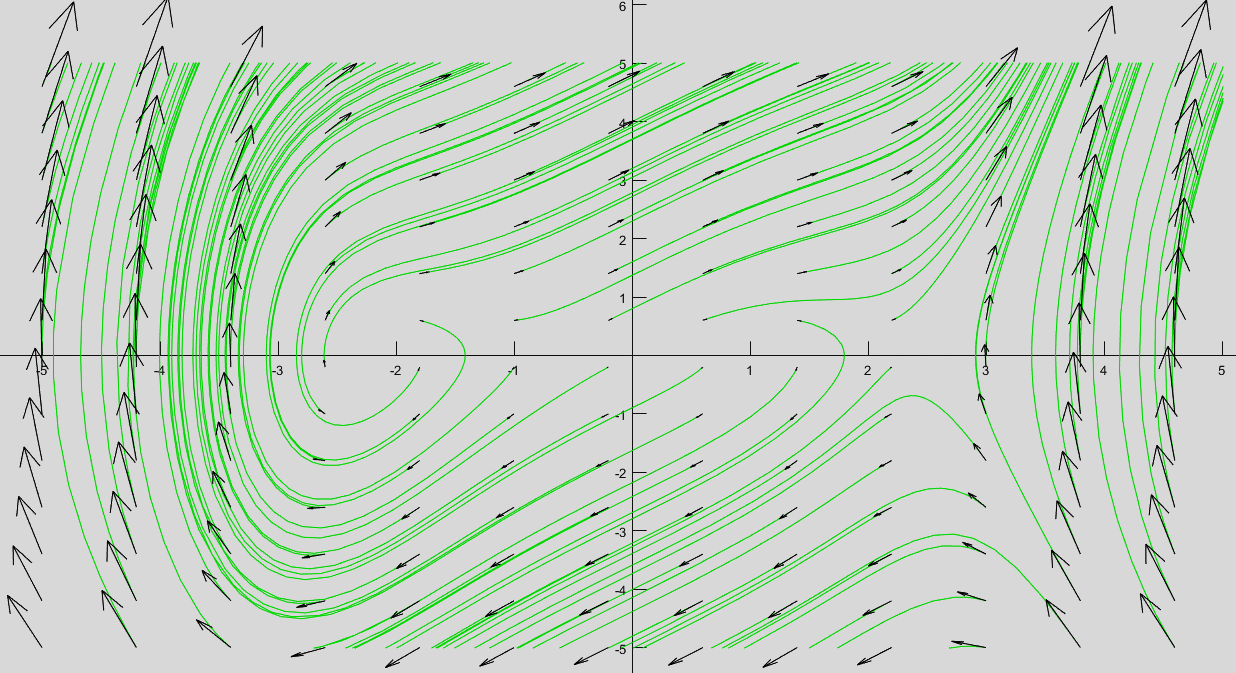


Рис 1. Фазовый портрет системы дифференциальных уравнений (1), построенный с помощью кода

Второй способ построения фазового портрета системы – это использование средств Simulink. Все разрабатываемые файлами будем располагать в директории Lab-2/2D-plot/Simulink.

Вновь воспользуемся прилагаемыми к лабораторной работе файлами, а именно **файлами отрисовки (plotLocus.m), файлами инициализации системы (simInitSet.m), файлом запуска прорисовки (main.m), а также отладочную модель в среде Simulink (SimulinkMain.slx).**

Основная задача – создание подсистемы, описывающую систему дифференциальных уравнений (1). Такая система имеет вид, представленный на рисунке 2.

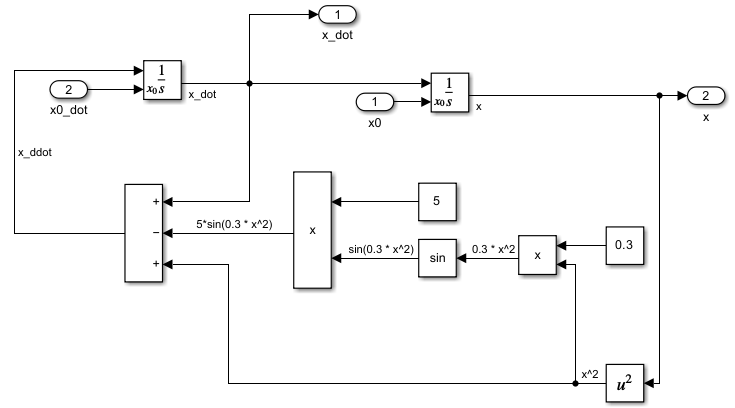


Рис 2. Представление системы дифференциальных уравнений (1) в виде подсистемы в среде разработки Simulink

Назначаем размер сетки 8 Х 8 с шагом 0.5, максимальное время моделирования 10 секунд. Время симуляции составило

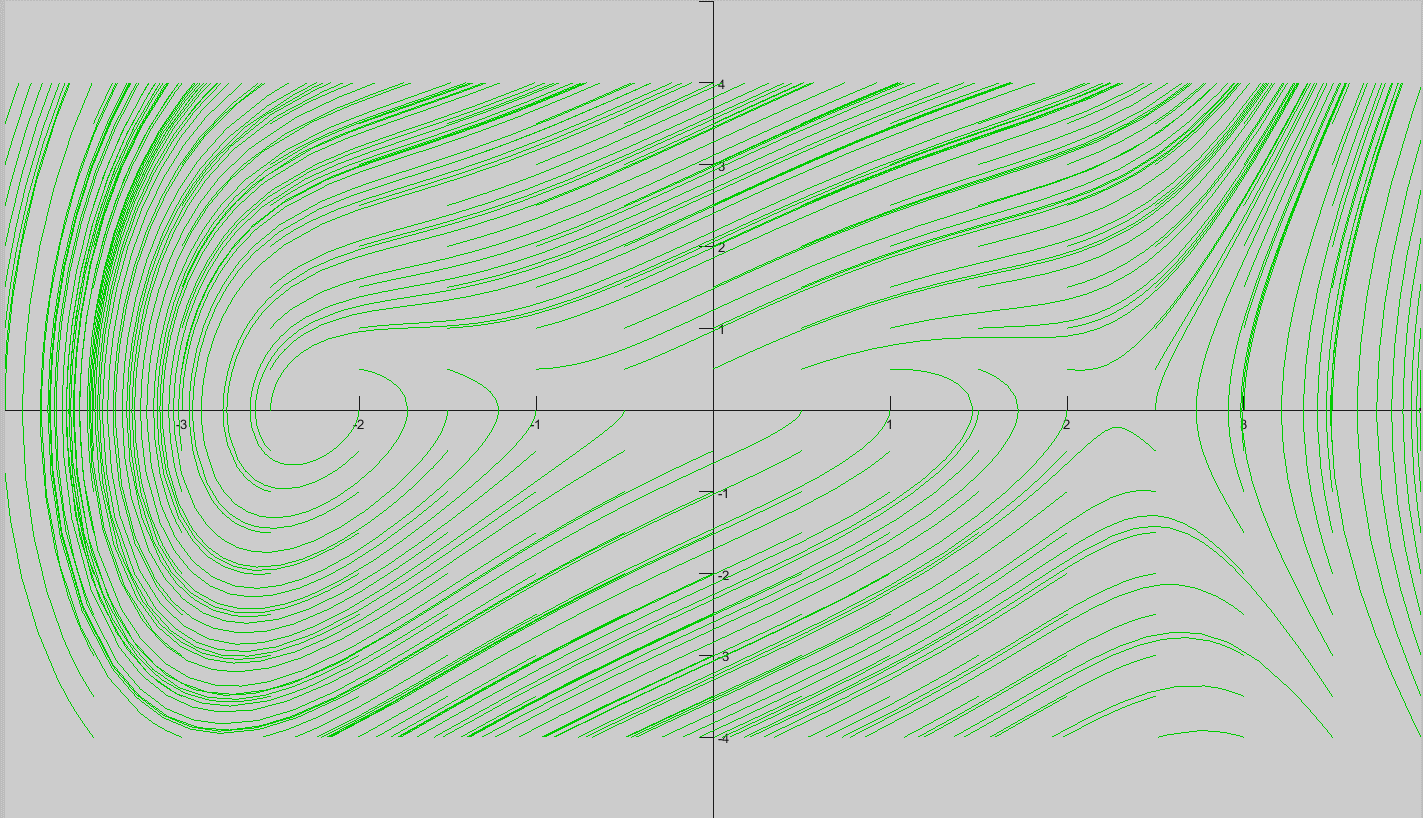


Рис 3. Фазовый портрет системы дифференциальных уравнений (1), построенный с помощью Simulink модели

Определим особые точки дифференциальной системы уравнений (1) и их тип.

Для нахождения особых точек приравниваем производные к нулю. Тогда получаем:

(2)

Из системы (2) явно можно выделить одну особую точку – (0; 0). Однако остальные определить затруднительно - необходимо найти решение уравнения , что является нетривиальной задачей. Для упрощения можно заметить, что все особые точки обязаны лежать на оси абсцисс

Воспользуемся средствами MatLab и фазовым портретом (рис. 1) для нахождения особых точек. Заметим по рисунку 1, что существуют ещё как минимум две особые точки – около координат (-2.5; 0) и (2.5;). Для более точного решения воспользуемся функцией **fsolve(fun, x0),** предназначенную для решения систем нелинейных уравнений, где задачей является нахождение координат (значений переменных) при равенстве функции нулю fun(x) = 0. Реализация представлена в скрипте **FindZeros.m**

|  |
| --- |
| options = optimoptions('fsolve','Display','none');  [z, fval] = fsolve(@model, [, ], options); % Search zeros of function  Out = ['x1 = ', num2str( z(1,1) ), '; x2 = ', num2str( z(1,2) )];  disp(Out) % Output to command window  %% User functions  function f = model( x )  % Searching zeros of system with non-linear equations  f(1) = x(2);  f(2) = x(2) + x(1)^2 - 5 \* sin( 0.3 \* x(1)^2);  end |

Подставляем наши примерные точки в данный код и получаем более точные координаты особых точек. Тогда имеем 3 особые точек, имеющие координаты:

(3)

По фазовому портрету (рис. 1) можно четко определить и тип особых точек: – **неустойчивый фокус**; – **седло**. Для точки так легко сказать нельзя, поэтому для этой точки произведем линеаризацию:

(4)

Поскольку , а также собственные корни действительны, положительны, то точка – **неустойчивый узел**.

Получим локальные фазовые портреты для каждое из особых точек (рис. 4 – 6).

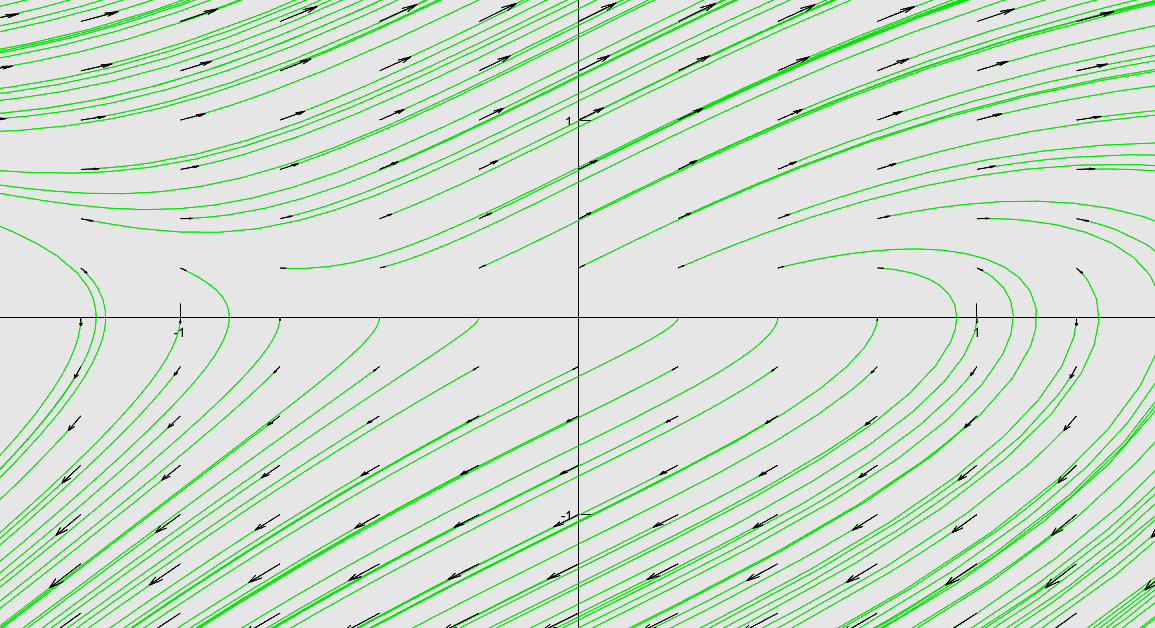


Рис 4. Локальный фазовый портрет для особой точки

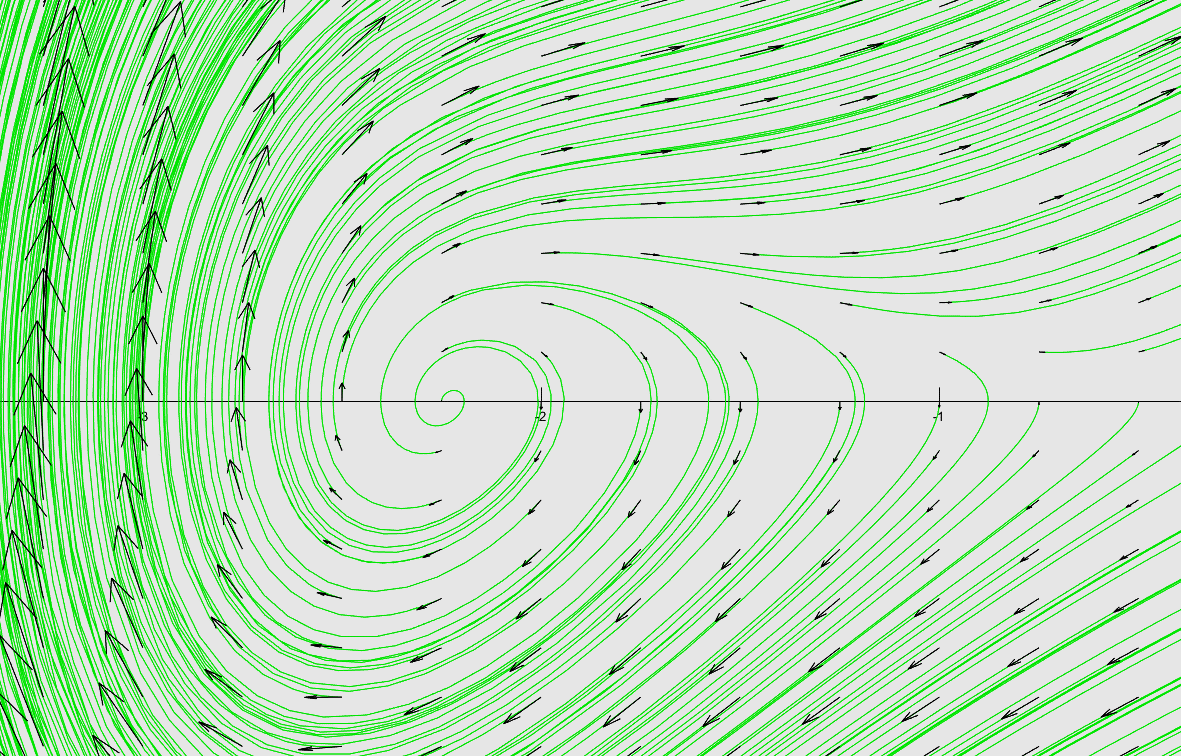


Рис 5. Локальный фазовый портрет для особой точки

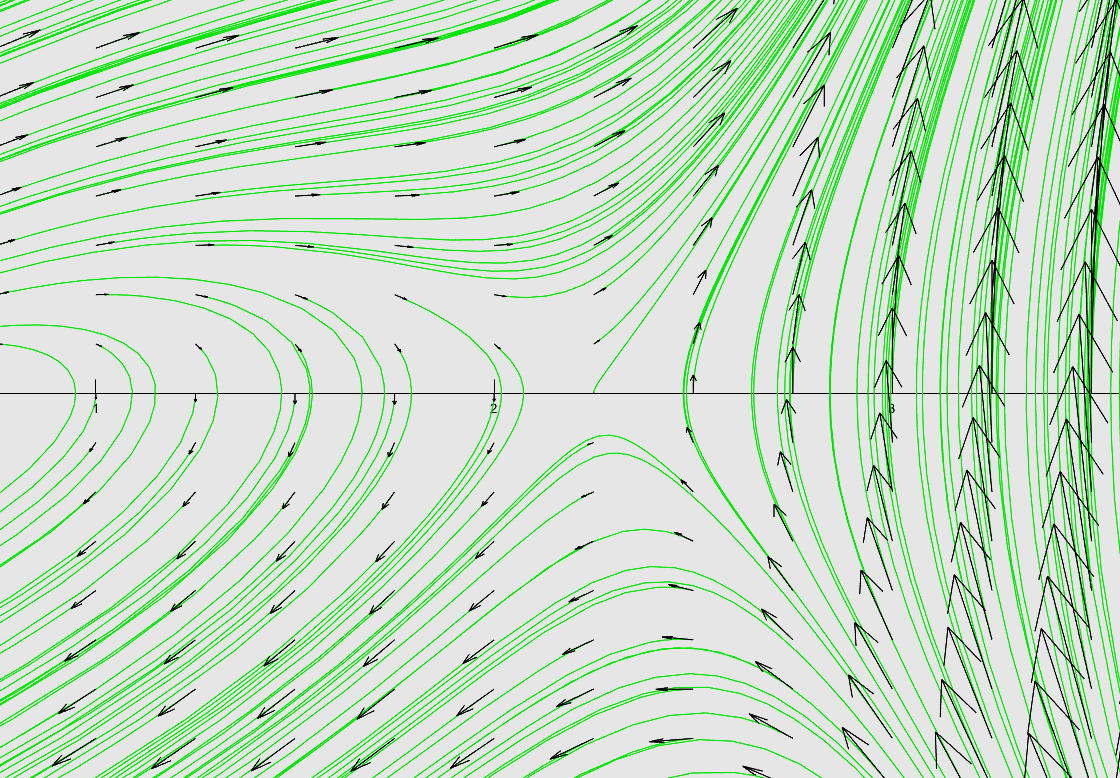


Рис 6. Локальный фазовый портрет для особой точки

* 1. **Построение фазового портрета системы с вырожденной особой точкой**

В данном пункте мы будем строить фазовый портрет функции с вырожденной особой точкой, представленной в таблице 2.



Табл 2. Дифференциальное уравнение системы с вырожденной точкой

Зададим для данного уравнения фазовые переменные:

Подставив переменные в исходное уравнение, а также переписав уравнение в виде системы (т.е. переписываем систему естественным образом), получим:

(5)

Следующим пунктом является отображение фазового портрета системы (рис. 7, 8). Для этого потребуется адаптация прилагаемого к лабораторной работе файлов. В первую очередь запишем систему (5) в виде функции Matlab.

|  |
| --- |
| function dxdt = FuncWithDegeneratePoints(t, x)  % Define function with degenerate points  dxdt(1,:) = x(2,:);  dxdt(2,:) = sin( x(1,:) ) .\* x(2,:) + log( 1 + x(1,:) .^ 2);  end |

Вырожденной особой точкой называется такая точка такой системы, собственный значения которой совпадают между собой (кратны), т.е. в линеаризованной системе у особой точки присутствует лишь **один собственный вектор**. Попробуем найти особые точки системы (5).

(6)

Получим, что

Попробуем линеаризовать систему:

(7)

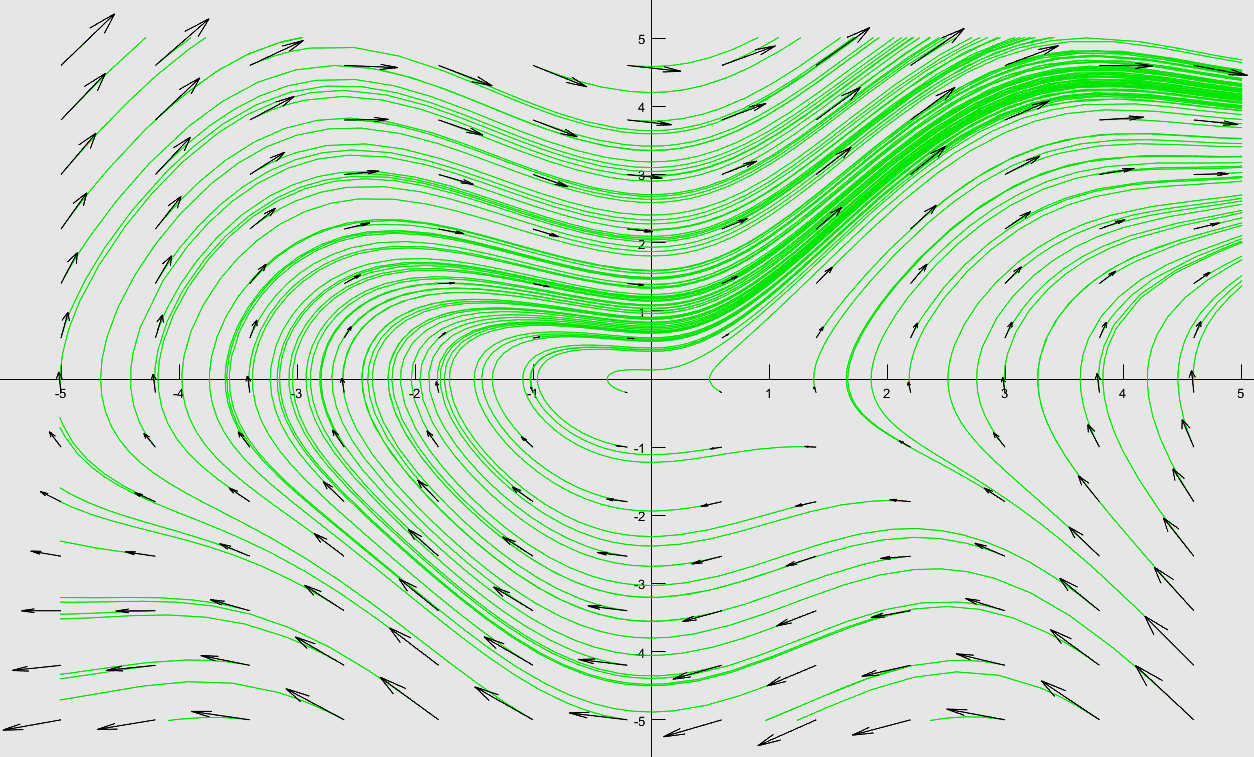


Рис 7. Фазовый портрет системы с вырожденной особой точкой

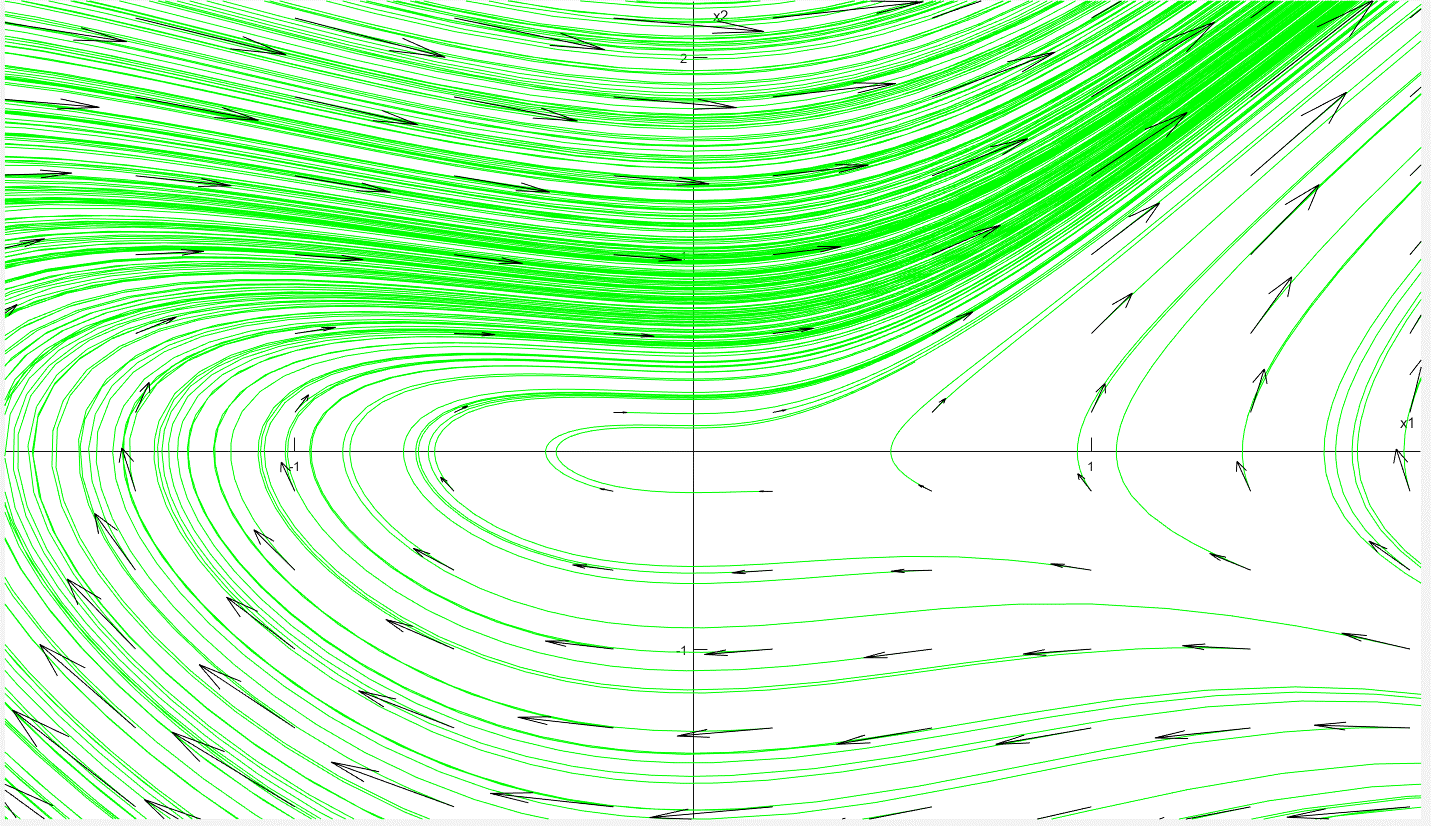


Рис 8. Фазовый портрет системы с вырожденной особой точкой в окрестности точки .

* 1. **Построение фазового портрета системы с континуумом особых точек**

В данном пункте мы будем строить фазовый портрет функции с континуумом особых точек, представленной в таблице 3.



Табл 3. Дифференциальное уравнение системы с вырожденной точкой

Зададим для данного уравнения фазовые переменные:

Подставив переменные в исходное уравнение, а также переписав уравнение в виде системы (т.е. переписываем систему естественным образом), получим:

(8)

Следующим пунктом является отображение фазового портрета системы (рис. 9). Для этого потребуется адаптация прилагаемого к лабораторной работе файлов. В первую очередь запишем систему (5) в виде функции Matlab.

|  |
| --- |
| function dxdt = FuncWithSingularPoints(t, x)  % Define function wit singular points  dxdt(1,:) = x(2,:);  dxdt(2,:) = x(2,:) .^ 4 .\* x(1,:) + x(2,:);  end |

Попробуем найти особые точки системы (8).

(9)

Получим, что может принимать любое значение, лежащее в области действительных чисел. На основе этого можно сказать, что все особые точки системы (8) лежат на оси , а значит вся ось является «особенной». Это же можно увидеть и по фазовому портрету систему (рис.9)

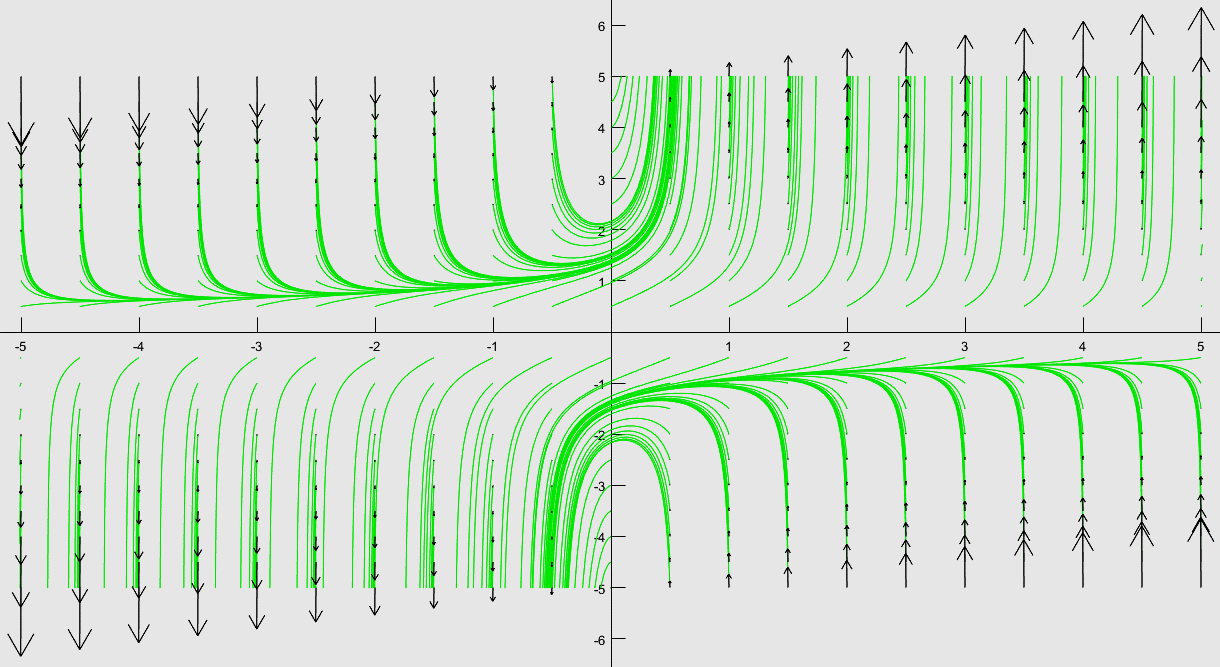


Рис 9. Фазовый портрет системы с континуумом особых точек

1. **Построение фазовых траекторий трехмерных систем**

В данной главе необходимо построить фазовое представление автономной системы дифференциальных уравнений третьего порядка. Рассматриваемая система называется **уравнениями** **Рабиновича-Фабриканта** и выглядит следующий образом:

, (10)

где – параметры системы.

Необходимо построить график данной системы дифференциальных уравнений в трехмерном пространстве со следующими начальными условиями:

А)

Б)

Перенесем уравнения (10) в MatLab

|  |
| --- |
| function dxdt = model(t, x, ALPHA, GAMMA)  % Rabinovich–Fabrikant equations  dxdt(1,:) = x(2,:) .\* ( x(3,:) - 1 + x(1,:).^2 ) + GAMMA .\* x(1,:);  dxdt(2,:) = x(1,:) .\* (3 \* x(3,:) + 1 - x(1,:).^2 ) + GAMMA .\* x(2,:);  dxdt(3,:) = -2 \* x(3,:) .\* ( ALPHA + x(1,:) .\* x(2,:) );  end |

Также внесем основные константы и начальные условия для системы.

|  |
| --- |
| tic; % Start timer  v = [0.1; -0.1; 0.1]; % Starting position  GAMMA = 0.1; ALPHA = 0.05; % Parameters of equation  TMAX = 100; % Time of modeling  dxdt = @(t, x) model(t, x, ALPHA, GAMMA);  plotLocus(v, dxdt, TMAX); % Plot portrait  toc; % Stop timer |

Скрипт **plotLocus.m** для системы выглядит следующим образом

|  |
| --- |
| function plotLocus(v0, dxdt, tmax)  % Function that sketches a phase portrait of a dynamical system  figure("Name","Rabinovich–Fabrikant");  hold on;  title('Rabinovich–Fabrikant equations')  xlabel('x');  ylabel('y');  zlabel('z');  colororder(["#8040E6";"#1AA640";"#E68000"]);  tspan = [0, tmax]; % Time of modeling    % Solver parameters  [~, z] = ode23t(dxdt, tspan, v0, odeset('RelTol',1e-3));  plot3( z(:, 1), z(:, 2), z(:, 3)); % Plotting  hold off  end |

Выведем полученные графики (рис. 10, 11).



Рис 10. График системы дифференциальных уравнений Рабиновича-Фабриканта (3-ех координатная) со значением переменных из пункта А

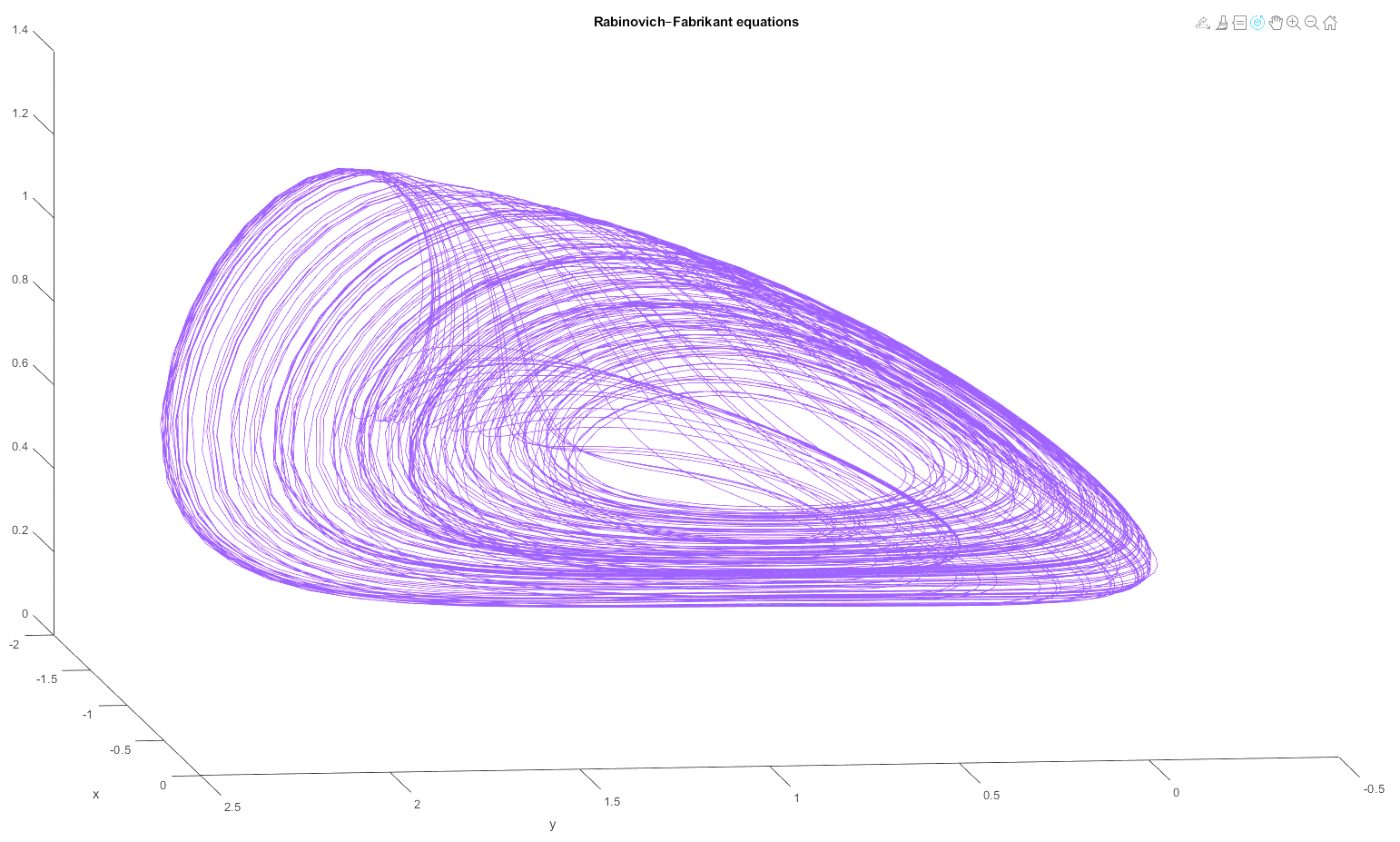


Рис 11. График системы дифференциальных уравнений Рабиновича-Фабриканта (3-ех координатная) со значением переменных из пункта Б

Попробуем построить график системы (рис. 12) со следующими параметрами системы:

В)

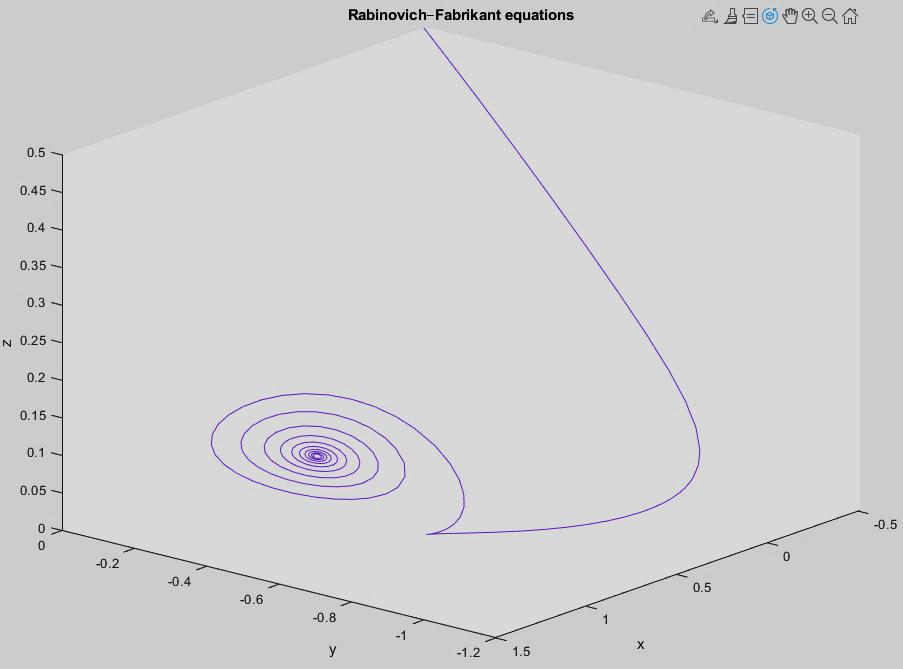


Рис 12. График системы дифференциальных уравнений Рабиновича-Фабриканта (3-ех координатная) со значением переменных из пункта В

1. **Построение фазовых траекторий нестационарных систем**
   1. **Хаотичность уравнений Рабиновича-Фабриканта**

В этом пункте необходимо определить хаотичность системы дифференциальных уравнений Рабиновича-Фабриканта (10), рассмотренных пунктом ранее.

Перед началом моделирования определим понятие нестационарной системы. **Нестационарная система** — это система с перемен­ными параметрами или даже структурой. При математическом описании не­стационарной системы это проявляется в том, что некоторые коэффициенты описывающего ее дифференциального уравнения являются функциями вре­мени.

В соответствии с данным определением, **реакция нестационарной системы на одно и то же воздействие за­висит от момента времени приложения этого воздействия.**

В качестве начальных условия положим:

Для определения хаотичности системы, введём некоторую сходящуюся последовательность чисел, которая будет определять радиус , на поверхности которой мы будем выбирать начальные условия для нашей системы и далее искать пределе расстояния между исходным графиком и совокупностью системы с различными начальными условиями.

В первую очередь напишем программу, которая строит массив расстояний между различными точками траектории

|  |
| --- |
| function ro = RangeBetweenPoints( x, y )  % RANGEBETWEENPOINTS calculate distance between two points on graph  rows = min( size(x,1), size(y,1) );    ro = eye(size(x,1), 1);  for i = 1:rows  sum = 0;  for j = 1:size(x, 2)  sum = sum + ( y(i, j) - x(i, j) ).^2;  end  ro(i, 1) = sqrt( sum );  end    end |

Вторым шагом – напишем программу, которая считает хаотичность внутри и выводит массив хаотичности для различных радиусов.

|  |
| --- |
| function H = Chaotic( n, int, v0, dxdt, TMAX, z0)  % CHAOTIC in search chaotic parameters of system with  % differential equations  %% Input  % n - order of convergent sequence ( d = { 1/10^1, 1/10^2 ... 1/10^n } )  % int - quantity of points around start point ( we will be  % calculate other phase trajectories based on this points)  % v0 - initial point  % dxdt - system of differential equations;  % TMAX - ending time  % z0 - (x1, x2, x3) coordinates of initial phase trajectory  %% Output  % H - massiv with maximal distances between starting phase trajectory  % and secondary trajectory on the delta-sphere    tspan = [0, TMAX]; % Time of modeling    seq = eye(n,1);  for i = 1:n  seq(i,1) = 1/10^(i-1); % Generate convergent  % sequence delta^n  end  H = eye(1, n);  for i = 1:n  ro = 0; % Maximal distance on delta sphere  %Works in sphrecil coordinates system  theta = transpose( linspace(0, 2 \* pi, int) ); % Angle  phi = transpose( linspace(0, 2 \* pi, int) );  % 0x coordinates on the delta-sphere  x\_resh = v0(1,1) + seq(i,1) .\* sin(theta) .\* cos(phi);  % 0y coordinates on the delta-sphere  y\_resh = v0(1,2) + seq(i,1) .\* sin(theta) .\* sin(phi);  % 0z coordinates on the delta-sphere  z\_resh = v0(1,3) + seq(i,1) .\* cos(theta);    %Reference points (mas of intermediate initial values)  refPoints = [ x\_resh, y\_resh, z\_resh ];    for j = 1:size(refPoints, 2)  [~, z] = ode23t(dxdt, tspan, refPoints(j,:), ...  odeset('RelTol',1e-3));  % RangeBetweenPoints( x1, x2 ) - calculating  % distance between points of x1 and x2 massive with coordinates  % dist - maximal range between starting trajectory  % and intermediate trajectory  dist = max( RangeBetweenPoints( z, z0 ) );  % Search max range on the delta-sphere  if dist > ro  ro = dist;  end  end    H(1, i) = ro;  end  end |

Запустим нашу прогрумму и выведем полученные значения хаотичности в некоторой малой окрестности начальных условий (рис. 13). Как видно на рисунке, наша последовательность сходиться, т.е. при стремлении функции к заданным начальным условиям хаотичность уменьшается. Это значит, что наша система является **нехаотичной**. Это же подтверждается и тем фактом, что система уравнений Рабиновича-Фабриканта является по определению автономной, т.е. стационарной





Рис 13. Хаотичность системы Рабиновича-Фабриканта при стремлении к начальному условию

**Вывод**: система уравнений Рабиновича-Фабриканта является стационарной, **система нехаотична**

Решение находится в директории: Non-stationary/First\_part

* 1. **Определение хаотичности произвольной нестационарной системы**

На входе имеем следующее дифференциальное уравнение:

, (11)

где – некоторая функция, изменяющая своё значение в завимости от времени

Зададим для данного уравнения фазовые переменные:

Подставив переменные в исходное уравнение, а также переписав уравнение в виде системы, получим:

(12)

Предлагается рассмотреть следующие функции:

*А) Противоположные импульсы*

(13)

*Б) Противоположные линейные кусочки*

(14)

*В) Веселый синус*

(15)

Перепишем систему (12) с учетом уравнений (13) – (15) в MatLab

|  |
| --- |
| function dxdt = opposingImpulse(t, x)  % Define opposing impulse function    if mod( round( t - 0.5 ) , 2) == 0 || t == 0  dxdt(1,:) = x(2,:);  dxdt(2,:) = 1 - x(2,:) - x(1,:);  else  dxdt(1,:) = x(2,:);  dxdt(2,:) = -1 - x(2,:) - x(1,:);  end  end  function dxdt = opposingLines(t, x)  % Define opposing linear lines function  if mod( round( t - 0.5 ) , 2) == 0 || t == 0  dxdt(1,:) = x(2,:);  dxdt(2,:) = t - x(2,:) - x(1,:);  else  dxdt(1,:) = x(2,:);  dxdt(2,:) = -t - x(2,:) - x(1,:);  end  end  function dxdt = funnySinus(t, x)  % Define funnySinus function  dxdt(1,:) = x(2,:);  dxdt(2,:) = sin(1/(t+10^(-7))) - x(2,:) - x(1,:);  end |

В дальнейшем с этими уравнениями необходимо произвести III действия. Будем идти по порядку с каждой из вариантов функции f(t)

1. **Построить фазовую траекторию для нулевых начальных условий**

Начальные условия:

Построим графики для каждой из функций (рис. 14 – 16)

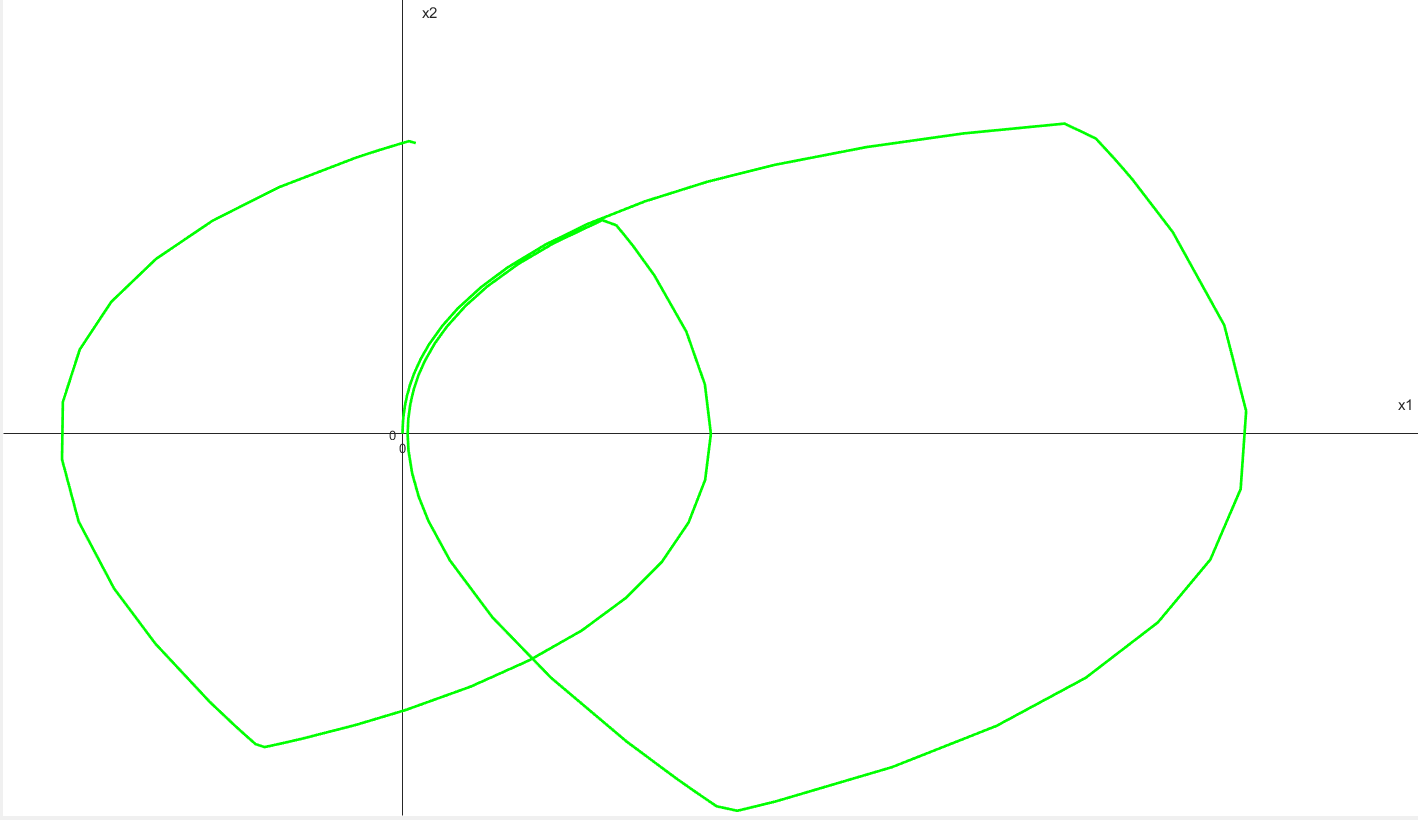


Рис 14. График противоположных импульсов (А)

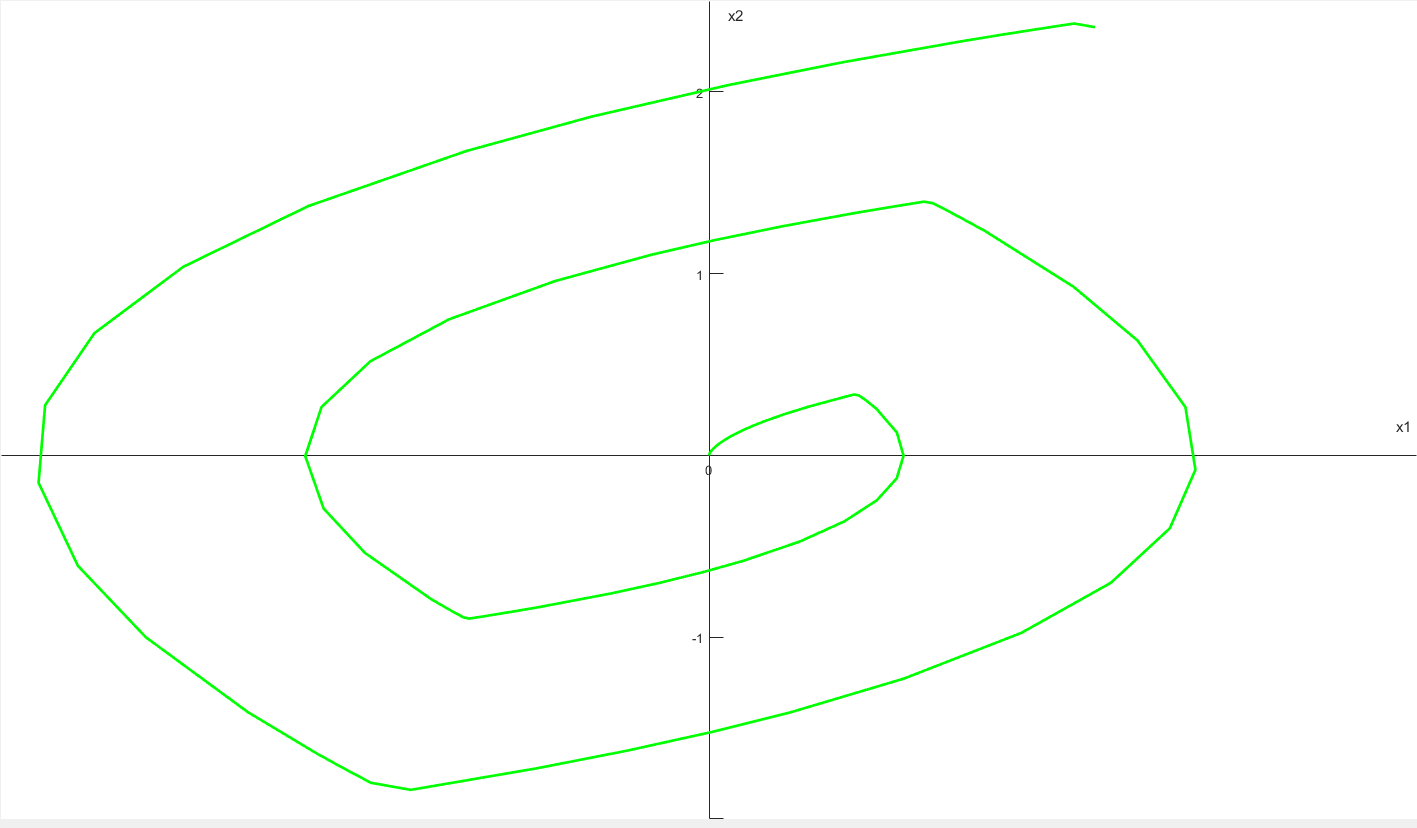


Рис 15. График противоположных линейных кусочков (Б)



Рис 16. График веселого синуса (В)

1. **Нарисовать quiver (локальные направления градиента фазового вектора) в окрестности фазовой траектории**

Начальные условия оставляем неизменными.

Будем рисовать 5 локальных направлений вокруг каждой точки траектории, которую выводит солвер.

Вновь выведем три графика (рис. 17 – 19), но уже с направлениями локального градиента

|  |
| --- |
| function plotQuivers( t, z0, dzdt)  % Define the direction of phase lines and quiver near main line  % t - time of modeling  % z0 - (x1, x2) coordinates of base phase trajectory  % dzdt - system of differential equations  hold on    tspan = transpose(t);  z = transpose(z0);    for i = 1:size( tspan, 2 )  for n = 1:5  % Generate 5 random points around based coordinates  v = z(:, i) + (rand( size(z,1), 1) - 0.5)./ 10;  % Calculate dz/dt in this points  u = transpose( dzdt( tspan(i), v(:, 1) ) ) ./ 15;  % Plotting  quiver( v(1, 1), v(2, 1), u(1, 1), u(1, 2), 'k');  end  end    hold off  end |

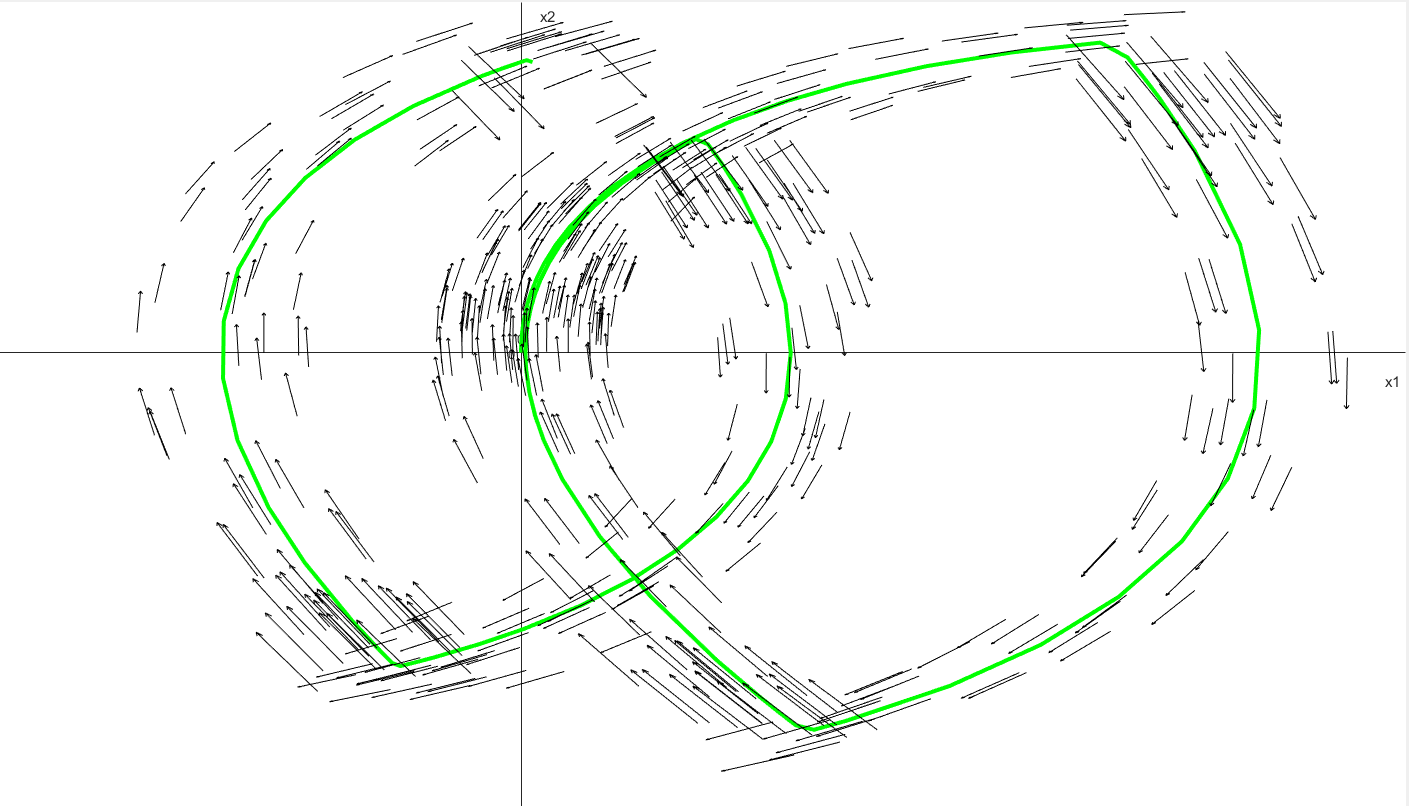


Рис 17. График противоположных импульсов с локальными направлениями (А)

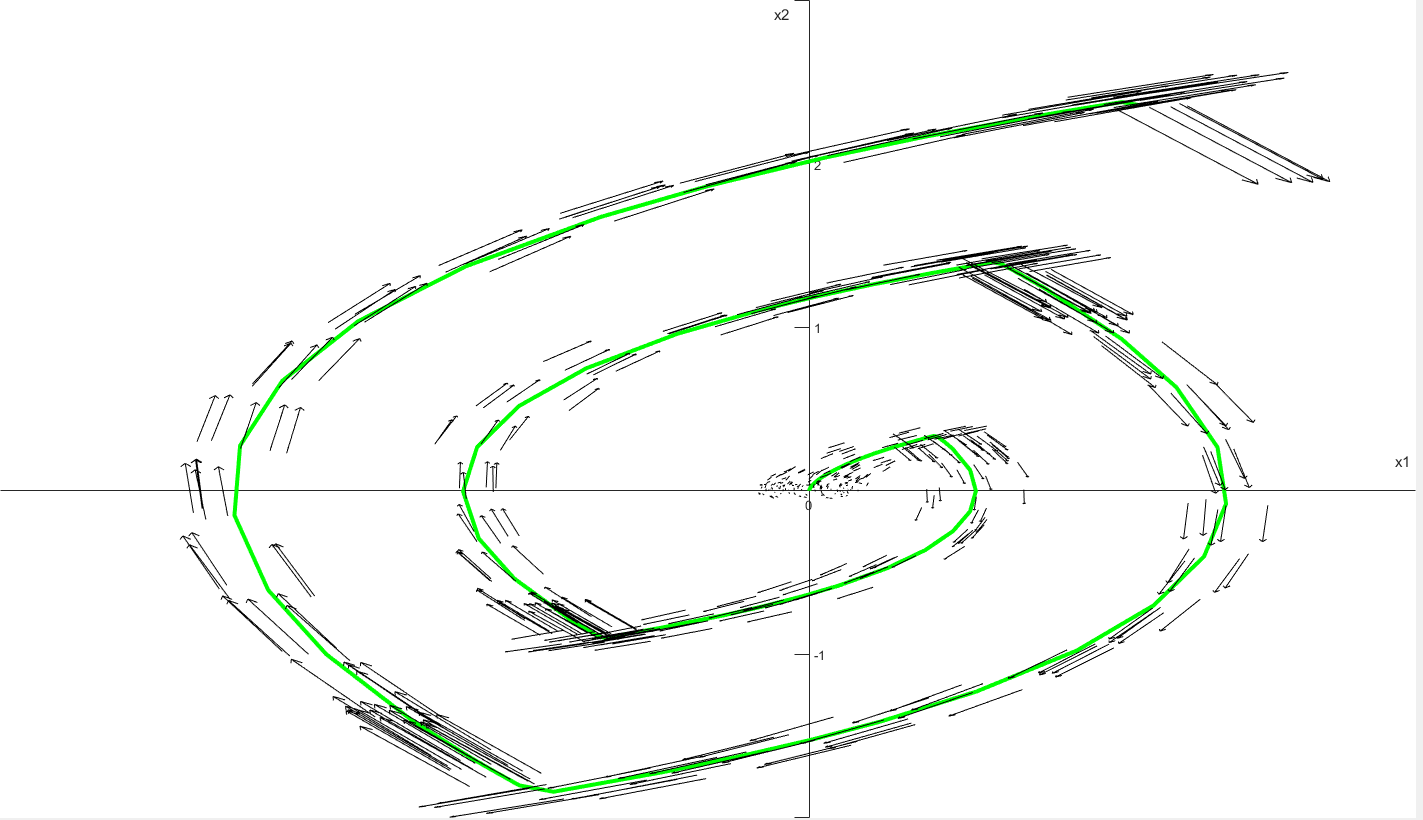


Рис 18. График противоположных линейных кусочков с локальными направлениями (Б)

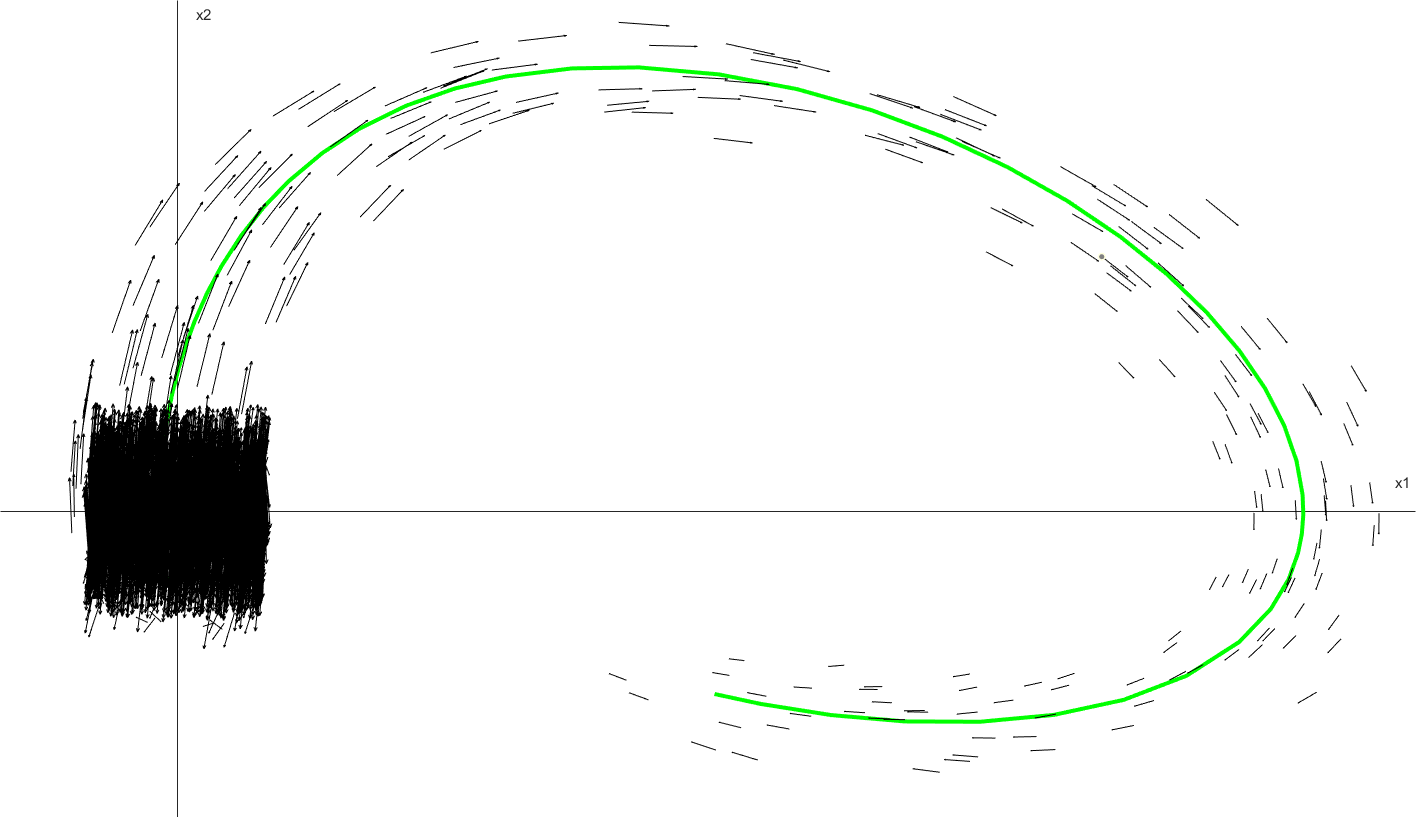


Рис 19. График веселого синуса с локальными направлениями (В)

1. **Расчёт хаотичности систем**

Для расчёта хаотичности системы напишем отдельную функцию, внутри которой создадим и сходящуюся последовательность с размерностью n.

|  |
| --- |
| function H = Chaotic( n, int, v0, dxdt, TMAX, z0)  % CHAOTIC in search chaotic parameters of system with  % differential equations  %% Input  % n - order of convergent sequence ( d = { 1/10^1, 1/10^2 ... 1/10^n })  % int - quantity of points around start point ( we will be  % calculate other phase trajectories based on this points)  % v0 - initial point  % dxdt - system of differential equations;  % TMAX - ending time  % z0 - (x1, x2) coordinates of phase trajectory  %% Output  % H - mas with maximal distances between starting phase trajectory  % and secondary trajectory on the delta-sphere    tspan = [0, TMAX]; % Time of modeling    seq = eye(n,1);  for i = 1:n  seq(i,1) = 1/10^(i-1); % Generate convergent  % sequence delta^n  end  H = eye(1, n);  for i = 1:n  ro = 0; % Maximal distance on delta sphere  %Works in polar coordinates system  theta = transpose( linspace(0, 2 \* pi, int) ); % Angle  % 0x coordinates on the delta-sphere  x\_resh = v0(1,1) + seq(i,1) .\* cos(theta);  % 0y coordinates on the delta-sphere  y\_resh = v0(1,2) + seq(i,1) .\* sin(theta);    %Reference points (mas of intermediate initial values)  refPoints = [ x\_resh, y\_resh ];    for j = 1:size(refPoints, 2)  [~, z] = ode23t(dxdt, tspan, refPoints(j,:), ...  odeset('RelTol',1e-3));  % RangeBetweenPoints( x1, x2 ) - calculating  % distance between points of x1 and x2  % dist - maximal range between starting trajectory  % and intermediate trajectory  dist = max( RangeBetweenPoints( z, z0 ) );  % Search max range on the delta-sphere  if dist > ro  ro = dist;  end  end    H(1, i) = ro;  end  end |

Тогда для систем получим следующие значения (рис. 20-22).

Первый строчка определяется номер числа в последовательности, второй – меру хаотичности для заданной



Рис 20. Таблица значений хаотичности системы (А) при разных значениях



Рис 21. Таблица значений хаотичности системы (Б) при разных значениях



Рис 22. Таблица значений хаотичности системы (В) при разных значениях

На основе приведенных выше таблиц можно сделать вывод о том, что системы **противоположных импульсов** и **противоположных линейных кусочков** являются **не хаотическими в нулевой точке**, поскольку параметр хаотичности *H* стремиться к нулевому значению. В случае **веселого синуса** можно говорить об **ограниченной хаотичности**, поскольку её мера в случае системы (В) стремится к значению **0.65**

**Приложение**

1. Публичный репозиторий для лабораторных по ТАУ // GitHub URL: <https://github.com/RiXenGC/Theory-of-Automatic-Control>