**Лабораторная работа №2**

Построение фазового портрета системы второго порядка

**Цель работы:** нахождение особых точек системы второго порядка, определение их типа и построение фазового портрета с помощью Matlab.

**Основные сведения:** данная лабораторная работа состоит из двух частей. Первая часть – нахождение особых точек системы, вторая – построение фазового портрета.

1) Исследование системы второго порядка происходит по такому же алгоритму, что и для системы первого порядка. Однако, описание системы в этом случае состоит из матрицы, полученной из двух функций двух переменных. Таким образом все операции выполняются с матричными значениями, а не со скалярными. Также для нахождения якобиана необходимо брать частные производные от каждой функции по каждой переменной.

Так как матрица Якоби для системы второго порядка будет размера 2х2, то для определения типа особой точки нужно найти собственные числа матрицы. Для этого необходимо взять определитель от матрицы и решить полученное квадратное уравнение относительно λ. При подстановке координат положения равновесия в полученные выражения собственных чисел можно определить тип особой точки. Возможны 6 вариантов:

|  |  |  |
| --- | --- | --- |
| λ1\* | λ2 | Тип особой точки |
| Re(λ1)< 0, Im(λ1)= 0 | Re(λ2)< 0, Im(λ2)= 0 | Устойчивый узел |
| Re(λ1)> 0, Im(λ1)= 0 | Re(λ2)> 0, Im(λ2)= 0 | Неустойчивый узел |
| Re(λ1)> 0, Im(λ1)= 0 | Re(λ2)< 0, Im(λ2)= 0 | Седло |
| Re(λ1)< 0, Im(λ1)≠ 0 | Re(λ2)< 0, Im(λ2)≠ 0 | Устойчивый фокус |
| Re(λ1)> 0, Im(λ1)≠ 0 | Re(λ2)> 0, Im(λ2)≠ 0 | Неустойчивый фокус |
| Re(λ1)= 0, Im(λ1)≠ 0 | Re(λ2)= 0, Im(λ2)≠ 0 | Центр |

\*Re – Real – вещественная составляющая комплексного числа; Im – Imaginary – мнимая составляющая комплексного числа

2) Для построения фазового портрета можно использовать два разных подхода (или скомбинировать оба). Первый подход заключается в том, чтобы построить фазовую траекторию при всех вариантах начальных условий. Второй – в нахождении значений градиентов (векторов изменения переменных состояния) в каждой точке фазового пространства. Рассмотрим вариант реализации каждого из двух подходов.

Для построения фазовой траектории необходимо решить систему дифференциальных уравнений. Для этого необходимо воспользоваться численным методом решения дифуров. Тогда, если построить график x2(x1), то это и будет фазовая траектория. Для получения фазового портрета необходимо повторить этот процесс при разных начальных условиях.

Для получения градиентов необходимо рассчитать значения производных в каждой точке фазового пространства. Для этого можно подставить в исходную систему уравнений все возможные начальные условия. Тогда производная каждой переменной состояния будет являться проекцией градиента на каждую ось.

**Реализация алгоритма в Matlab:** Первая часть работы не содержит новых функций, так что ее рассматривать нет необходимости, таким образом сразу переходим к построению фазового портрета. Для любого из двух подходов необходимо создать функцию, описывающую исследуемую динамическую систему. Для этого можно создать «анонимную функцию» (однострочную функцию без имени и, соответственно нигде ранее не описанную). Чтобы это сделать, необходимо присвоить переменной тело желаемой функции (обязательно строго одна строка), начинающееся с ключа @(args), где args – список всех аргументов желаемой функции, перечисленный через запятую. Например:

>> myfun = @(x) x ^ 2

myfun =

function\_handle with value:

@(x)x^2

При это здесь стоит обратить внимание, что символ в скобках не отображается в рабочей области, так как это именно аргумент функции, то есть у этой функции своя область видимости. Также не указывается тип аргумента, то есть это может быть в том числе и массив. Важно учитывать порядок аргументов, так как именно в таком порядке их необходимо будет указывать при вызове данной функции. Так как рассматривается динамическая система второго порядка, то первым аргументом должно идти время (даже если ни одна функция от него не зависит) и затем два аргумента – переменных состояния. При этом для переменных состояния можно использовать один аргумент и обращаться с ним как с матрицей 2х1:

>> sys = @(t, x) [x(1)^2+x(2)^2-17; x(1)\*x(2)+4]

sys =

function\_handle with value:

@(t,x)[x(1)^2+x(2)^2-17;x(1)\*x(2)+4]

Для того, чтобы решить полученную систему дифуров необходимо воспользоваться «решателем» - численным методом решения дифференциальных уравнений. В Matlab реализовано несколько таких алгоритмов, самым распространенным является ode45 – метод Рунге-Кутты 4-5 порядка с переменным шагом. Для настройки «солвера» можно воспользоваться функцией odeset, но в данной задаче в этом нет необходимости. У функции ode45 три обязательных аргумента – функция, которую необходимо решить, интервал времени, на котором необходимо получить решение и начальные условия для каждой переменной состояния (задаются в виде одномерного массива). Интервал времени можно задать двумя способами: массивом из двух элементов – начала и конца интервала или массивом из более чем двух значений. В первом случае полученное решение будет включать в себя значения в моменты времени, при которых производился расчет, то есть их количество будет зависеть от шага интегрирования. Во втором случае результат будет включать в себя только значения в перечисленных в массиве моментах времени, при этом алгоритм расчета и шаг интегрирования никак не изменится. Если вызвать функцию «солвера» не присваивая значения в переменную, то выводом функции будет построение графика, а для того, чтобы сохранить полученный расчет в удобном виде, можно присвоить результат в две переменные. В этом случае в первой переменной окажется массив времени, а во второй многомерный массив, где каждый столбец – соответствующая переменная состояния. Пример вызова функции ode45:

>> [t, x] = ode45(sys, [0 10], [0 0]);

В этом случае в переменной *t* окажутся все значения времени от 0 до 10 секунд, в которых производился расчет, а в переменной *x* – два столбца со значениями двух переменных состояния в эти моменты времени.

**Важное замечание:** в функцию «решателя» нельзя передавать систему, записанную через разные аргументы (x, y, …), то есть все переменные состояния должны быть элементами одного столбца – вектора состояния – как в примере выше.

Для построения фазового портрета методом градиентов можно воспользоваться функцией quiver. Данная функция принимает в себя четыре аргумента: координату по оси *x,* координату по оси *y*, проекцию вектора по оси *x*, проекцию вектора по оси *y*. Результатом функции является график, изображающий заданный вектор. Например:

>> quiver(0, 0, 2, -1)

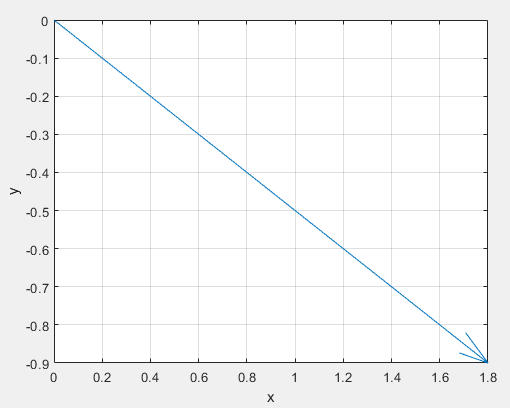


Рисунок 1. – отображение вектора с помощью функции quiver

Таким образом, для построения фазового портрета необходимо передать функции quiver множества начальных значений и множества проекций векторов. Для того, чтобы не собирать данные множества вручную, можно воспользоваться функцией meshgrid, которая принимает в качестве аргументов два одномерных массива, а возвращает два двумерных массива, которые содержат координаты сетки, полученной на основе исходных массивов. Например:

>> [x, y] = meshgrid([1, 2, 3], [2, 3, 4])

|  |  |
| --- | --- |
| x =  1 2 3  1 2 3  1 2 3 | y =  2 2 2  3 3 3  4 4 4 |

Для того, чтобы получить значения проекций градиента, необходимо рассчитать значения функций исходной системы во всех координатах полученной сетки начальных условий. Так как система уже записана в виде функции, то ее можно просто вызвать, при этом значение первого аргумента (времени) может быть любым, так как функции от него не зависят. Например:

>> sys(0, [x(3, 2) y(3, 2)])

ans =

3

12

То есть значение производных исходной системы в точке с координатами (2, 4) принимают значения 3 и 12 для первой и второй переменных состояния соответственно.

**Задание на лабораторную работу:**

* необходимо реализовать алгоритм поиска особых точек системы второго порядка и определения их типа
* программу необходимо оформить как функцию, которая на вход принимает массив функций в символьном виде, а возвращает все найденные особые точки с указанием их типа
* формат вывода необходимо придумать самостоятельно. Так как в последствии полученный вывод придется использовать, рекомендуется тип особой точки кодировать численным значением. Пример удобного вывода: двумерный массив, где первые два столбца — это координаты особой точки, а третий столбец – код типа особой точки.
* необходимо построить фазовый портрет исследуемой системы. Решение можно разбить на две независимых программы.