In [0]:

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import math
import random

from mpl_toolkits.mplot3d import axes3d
```

In [0]:

```
def new_plot(title='', y_min=-8, y_max=8):
    fig, ax = plt.subplots(figsize=(15,10))

    ax.set(xlabel='alpha', ylabel='x',
           title=title)
    ax.grid()

    return ax


def draw_line(ax, foo, x_min=0, x_max=2, step=0.01, style='b', label=''):
    t = np.arange(x_min, x_max, step)
    s = foo(t)

    ax.plot(t, s, style, label=label)

def draw_asym(point, label=''):
    y = list(range(-8, 8))
    x = [point] * len(y)

    ax.plot(x, y, '--', label=label, linewidth=0.6)



def draw_tree(ax, f, x_min, x_max, style, step=0.01, iter=100, inf=10, inf_n=1
00, eps=10e-6, start=0):

    a = np.arange(x_min, x_max, step)

    a_ = []
    x_ = []


    for i in a:

        x_tmp = random.choice([start, -start])

        for j in range(iter):
            x_tmp = f(x_tmp, i)

    #     print(i, x_tmp)

        if not (-inf < x_tmp < inf):
```

```
            continue

        tmp = []
        for n in range(1, inf_n):
            tmp.append(x_tmp)
            x_tmp = f(x_tmp, i)

        a_.append(i)
        x_.append(tmp)


    ax.plot(a_, x_, style, ms=0.5)


def draw_line_revert(ax, foo, x_min=0, x_max=2, step=0.01, style='b', label=''
):
    t = np.arange(x_min, x_max, step)
    s = foo(t)

    ax.plot(s, t, style, label=label)

def draw_asym_revert(point, label=''):
    x = list(range(-3, 3))
    y = [point] * len(x)

    ax.plot(x, y, '--', label=label, linewidth=0.6)




def show_plot(y_min=-10, y_max=10):
    plt.ylim(y_min, y_max)     # set the ylim to bottom, top
    plt.legend()
    plt.show()



def plot_implicit(fn, bbox=(-2.5,2.5)):
    ''' create a plot of an implicit function
    fn  ...implicit function (plot where fn==0)
    bbox ..the x,y,and z limits of plotted interval'''
    xmin, xmax, ymin, ymax, zmin, zmax = bbox*3
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    A = np.linspace(xmin, xmax, 100) # resolution of the contour
    B = np.linspace(xmin, xmax, 15) # number of slices
    A1,A2 = np.meshgrid(A,A) # grid on which the contour is plotted

    for z in B: # plot contours in the XY plane
        X,Y = A1,A2
        Z = fn(X,Y,z)
        cset = ax.contour(X, Y, Z+z, [z], zdir='z',cmap='viridis')
        # [z] defines the only level to plot for this contour for this value o
f z

    for y in B: # plot contours in the XZ plane
        X,Z = A1,A2
```

```
        Y = fn(X,y,Z)
        cset = ax.contour(X, Y+y, Z, [y], zdir='y',cmap='viridis')

    for x in B: # plot contours in the YZ plane
        Y,Z = A1,A2
        X = fn(x,Y,Z)
        cset = ax.contour(X+x, Y, Z, [x], zdir='x',cmap='viridis')

    # must set plot limits because the contour will likely extend
    # way beyond the displayed level.  Otherwise matplotlib extends the plot l
imits
    # to encompass all values in the contour.
    ax.set_zlim3d(zmin,zmax)
    ax.set_xlim3d(xmin,xmax)
    ax.set_ylim3d(ymin,ymax)


    plt.show()
```

Лабораторная #2

Даны следующие отображения:

1. $x_{n+1} = a - bx + x_n^3$;
2. $x_{n+1} = a + bx - x_n^3$.

Небходимо выполнить для каждого:

- построить биффуркационную диаграму (в осях a, b)
- построить дерефо биффуркации при фиксированном a
- Построить карту режимов

Теория

Неподвижные точки

Неподвижной точкой отображения $f : X \to X$ называется такая точка $x \in X$, которую заданное отображение переводит в неё же, иными словами, неподвижная точка удовлетворяет уравнению $f(x) = x$.

Мультипликатор

Неподвижные точки могут быть *устойчивыми* и *неустойчивыми*. Данная характеристика описывает поведение точек в окрестности данной неподвижной точки. Характер устойчивости можно определить с помощью специальной характеристики – *мультипликатора*, представляющей собой производную функции, вычисленную в неподвижной точке. Действительно, в малой окрестности неподвижной точки x_0 некоторого отображения f можно считать, что $x_{n+1} = x_0 + \tilde{x}_{n+1}$ и $x_n = x_0 + \tilde{x}_n$, где $\tilde{x}_{n+1}$ и $\tilde{x}_n$ малые добавки к x_0.

Тогда, из уравнения неподвижной точки имеем:

$$x_0 + \tilde{x}_{n+1} = f(x_0 + \tilde{x}_n) \approx f(x_0) + f'(x_0)\tilde{x}_n.$$

Откуда следует, что

$$\tilde{x}_{n+1} = f'(x_0)\tilde{x}_n = \mu\tilde{x}_n.$$

Соответственно, что на каждом шаге дискретного времени (итерации) возмущение умножается на величину, равную мультипликатору. Следовательно, возмущение будет убывать и точка будет *устойчивой*, если $|\mu| < 1$. Иначе, при $|\mu| > 1$ возмущение будет нарастать, и точка окажется *неустойчивой*.

Таким образом, при $|\mu| \neq 1$ и небольшом изменении параметра отображения неподвижная точка сохранится и сохранит свой характер устойчивости - *случай общего положения*.

Существуют вырожденные случаи, при $|\mu| = 1$. Даже если изменить параметр на очень малую величину, то поведение системы изменится существенным образом. Такая ситуация называется бифуркацией.

Бифуркация

Бифуркация – эти качественная перестройка картины движения при малом изменении параметров отображения.

Значения управляющего параметра, при которых происходят бифуркации, называются **критическими** или **бифуркационными значениями**.

Качественная перестройка установившегося режима в одномерных отображениях управляется мультипликатором. Это утверждение с очевидностью носит общий характер. Действительно, пусть имеется некоторое одномерное отображение, зависящее от единственного параметра и имеющее неподвижную точку. Тогда при плавном изменении этого параметра будет изменяться взаимное расположение функции, характеризующей отображение, и биссектрисы на итерационной диаграмме. Следовательно, будет меняться мультипликатор неподвижной точки. При выполнении условия $|\mu| = 1$ точка будет терять устойчивость, а значит, будет иметь место некоторая бифуркация. Для одномерных отображений мультипликатор всегда действительная величина, поэтому бифуркации возможны в двух случаях: $\mu = 1$ и $\mu = -1$.

Бифуркация периодических циклов

Замечательное свойство отображений состоит в том, что бифуркационный анализ периодических циклов может быть построен совершенно аналогично анализу бифуркаций неподвижных точек. Обоснуем это утверждение. Пусть отображение $x_{n+1} = f(x_n)$ имеет цикл периода N. Это означает, что через N итераций последовательность x_n повторяется, то есть $x_{n+N} = x_n$.

Поскольку, $x_{n+1} = f(x_n)$, $x_{n+2} = f(f(x_n))$ и так далее, то условие реализации N-цикла можно переписать так:
$x_n = f^N(x_n)$,
где $f^N(x) = f(f\ldots(f(x_n)))$ - N-кратно проитерированная функция, задающая отображение.
Следовательно, если отображение имеет N-цикл, то отображение $f^N(x)$ имеет неподвижную точку, и наоборот.Этот результат позволяет сразу ввести определение мультипликатора N-цикла:

$\mu = (f^N(x))'$.
В частном случае для 2-цикла имеем:
$\mu = f'(x_2) f'(x_1)$.
Если мультипликатор цикла обратится в ноль, то такой цикл будет сверхустойчивым.

Отображения с двумя параметрами

Перейдем теперь к обсуждению отображений с двумя параметрами. Варьируя оба параметра, можно изменять вид функции $f(x)$, задающей отображение. Понятно, что число возможных перестроек при этом возрастает. Их удобно изучать и классифицировать, имея в виду устройство плоскости двух параметров, задающих отображение.
В данной работе проведен двухпараметрический бифуркационный анализ простейших примеров отображения с двумя параметрами- **кубических отображений**, для которых существенным является знак перед нелинейным членом. Поэтому рассматриваются два отображения:

1. $x_{n+1} = a - bx + x_n^3$;
2. $x_{n+1} = a + bx - x_n^3$.

Анализ направлен на выявление устройства плоскости (a, b) с точки зрения возможных бифуркаций. Прежде всего, отметим, что в двухпараметрических отображениях будут наблюдаться и все указанные выше бифуркации, характерные для отображений с одним параметром. Действительно, мы всегда можем выполнить однопараметрический анализ такого отображения. Самый простой вариант – зафиксировать один из параметров и варьировать другой. Несколько более сложный вариант – варьировать оба параметра одновременно, но так, чтобы этому отвечало движение по определенному маршруту на плоскости параметров. Из сказанного ясно, что обсуждавшимся выше бифуркациям будут отвечать определенные **линии** на плоскости параметров. Обратимся сначала к касательной бифуркации.

Карта динамических режимов

Поскольку число параметров больше единицы, то вместо бифуркационных диаграмм более наглядными выглядят карты динамических режимов.

Касательная бифуркация

Обсудим свойства кубического отображения. В силу свойств кубического уравнения понятно, что в системе возможно сосуществование трех неподвижных точек.
Условием касательной бифуркации является обращение мультипликатора в $+1$, откуда получаем уравнение для линии касательной бифуркации на плоскости параметров отображения. Такая линия содержит две ветви, которым отвечают два варианта знака для $a(b)$.

Точка сборки

две ветви линии касательной бифуркации сходятся в некоторой особой точке с некоторыми координатами. Это действительно особенность (понимаемая в математическом смысле), поскольку касательная в этой точке вертикальна, а характерному «клюву» отвечает степенной закон с соответствующим $a(b)$ показателем.Такую точку называют точкой сборки. Это название обусловлено следующими причинами. Если в трехмерном пространстве построить поверхность (b, a, x) заданную уравнением неподвижной точки x, то она дает зависимость координаты

неподвижной точки от параметров отображения и напоминает сборку на ткани, что и обусловило такое название. Проекция этой поверхности на плоскость (a, b) дает линии касательной бифуркации.
Имея в виду такую интерпретацию, их иногда называют еще и линиями складки. Термины «сборка» и «складка» широко распространены, поскольку соответствующие объекты появляются не только в теории бифуркаций, но и в математической теории проектирования (так называемые сборки Уитни), а также в теории катастроф, где они возникают при глубоком развитии идеи аппроксимации функций рядами Тейлора.
Для определения точки сборки необходимо выполнение сразу двух условий:

1. $f'(x) = +1$; 2.$f''(x) = 0$.

Понятия типичности и коразмерности

На примере точки сборки и отходящих от нее линий складок мы столкнулись с важным моментом в теории бифуркаций, который лежит в основе их классификации и позволяет сформулировать конструктивную стратегию исследования, восходящую к А. Пуанкаре. Ее идея состоит в том, чтобы не пытаться сразу анализировать все возможные феномены, которые реализуются в том или ином классе динамических систем, а классифицировать их по степени типичности. Или, иными словами, сначала изучить явления, представляющиеся наиболее вероятными при «случайном» выборе параметров. Такой подход приводит к важному понятию коразмерности бифуркаций. Поясним его более подробно.

Пусть система характеризуется всего двумя параметрами. Тогда при случайном выборе точки на плоскости параметров мы, скорее всего, попадем в ситуации общего положения, когда малое шевеление параметров не изменит тип динамики. Далее на плоскости параметров можно обнаружить линии, разделяющие области с разным типом поведения. Пересекая их по какой-либо траектории, мы сможем наблюдать соответствующую бифуркацию. Этот тип бифуркации можно назвать однопараметрическим, поскольку для его реализации достаточно варьировать один параметр (или одну комбинацию из двух параметров). В этом случае говорят, что мы имеем бифуркацию коразмерности один. Если теперь двигаться только вдоль этой выделенной линии, то мы будем наблюдать в определенной мере вырожденный тип поведения, при этом малые шевеления в пределах линии «скорее всего» не меняют этот тип поведения. Однако, путешествуя по этой линии, мы можем наткнуться на некоторую точку, где произойдет новая бифуркация даже в пределах этого вырожденного класса. Эта ситуация характеризуется еще большей степенью вырождения и имеет, как говорят, коразмерность два.

Эти рассуждения можно обобщить на случай трехмерного пространства параметров. Тогда бифуркациям коразмерности один будут отвечать некоторые поверхности, а коразмерности два – линии в пространстве параметров. При этом окажутся возможными бифуркации коразмерности три, которым будут отвечать некоторые избранные точки. Аналогичным образом можно ввести и бифуркации более высокой коразмерности.

Таким образом, коразмерность представляет собой минимальное число параметров, при котором тот или иной тип бифуркации является типичным. Например, для удвоений периода и касательной бифуркации это один параметр, а для сборки – два.

Под **коразмерностью** можно понимать и число дополнительных условий, которое необходимо наложить для наблюдения данной бифуркации. Например, для касательной бифуркации оно одно: равенство $f'(x) = +1$, поэтому ее коразмерность единица. Когда для сборки их два; таким

образом, коразмерность сборки равна двум. Отсюда, кстати, сразу вытекает возможность бифуркации коразмерности три, которой отвечают дополнительные условия равенства 0 третьей производной.
Коразмерность – важное понятие, которое применимо не только в теории бифуркаций или теории катастроф, но имеет гораздо более общее значение, поскольку формирует стратегию исследования сложных явлений, акцентируя программу работ на исследовании наиболее типичных и характерных эффектов, а не на поиске наиболее экзотических феноменов.

Бифуркация удвоения периода. Структуры «crossroad area»

Обсудим теперь бифуркации, связанные с мультипликатором 1 μ = – , в двухпараметрическом случае. Обратимся вновь к двум разновидностям кубического отображения, которые в этом случае приводят к более существенным различиям.
Получаем для каждой выражение для бифуркационной линии. Для первого отображения $x_{n+1} = a - bx + x_n^3$ третья производная $f'''(x) = 6$, поэтому вдоль найденной линии производная Шварца будет строго отрицательна. Таким образом, вся линия отвечает бифуркации удвоения периода, т.е. рождению устойчивого 2-цикла. Взаимное расположение этой линии и линий касательной бифуркации показано ниже. Отметим, что найденная линия не имеет самопересечений.
Продолжим исследование отображения (1) и обсудим бифуркации 2- цикла.
Условия на выполнение 2-цикла стоит дополнить условием на мультипликатор
$\mu = \pm 1.$
В этом случае следует выбирать знак «+» для поиска касательной бифуркации 2-цикла и знак «–» – для бифуркации удвоения. (Отметим, что обращение мультипликатора 2-цикла в +1 отвечает одновременно и уже найденной нами линии удвоения для неподвижной точки. Действительно, как мы видели выше для двукратно проитерированной функции, в этом случае реализуется бифуркация «вилка».)

Кроме этого, эти соотношения задают в неявной форме на плоскости (a, b) систему линий, на которой можно увидеть, что область устойчивости 2-цикла снизу ограничена линией «предыдущего» удвоения периода. В центре этой области располагается точка сборки, от которой отходят две ветви линии касательной бифуркации. По их «берегам», в свою очередь, идут линии удвоения 2-цикла (рождения 4-цикла), которые проходят мимо точки сборки, приближаясь затем к линии предыдущего удвоения. Существует область бистабильности, внутри которой сосуществуют два устойчивых 2-цикла, и изображающая точка может притянуться к какому-то одному из них в зависимости от начальных условий.
Область существования 2-цикла имеет на плоскости параметров некоторую характерную форму. Французский математик К. Мира предложил для такой структуры специальное название crossroad area. (В переводе с английского – перекресток. Некоторые исследователи называли такие образования «ласточками» из-за их характерной формы.)
Можно видеть, что при движении по плоскости параметров снизу вверх слева от точки сборки, функция отображения эволюционирует так, что складываются условия для **новой** бифуркации удвоения. Типичность же точек сборки на плоскости параметров объясняется величиной коразмерности два для этой бифуркации.

Жесткий переход через μ = –1. Структуры «spring area»

Иная ситуация возникает для второго кубического отображения.
Бифуркационная линия имеет более сложную форму, показана ниже. В частности, она имеет самопересечение в точке $a = 2, b = 0$ и охватывает точку сборки, пересекая вертикальную ось в точке $a = 0, b = -1$.
Вычислим производную Шварца в этом случае.
$S_f(x) = -f'''(x) - \frac{3}{2}[f'(x)]^2 = 6(1 - 9x^2).$

Для анализируемого отображения производная Шварца может быть как отрицательной, так и положительной. Точки смена знака производной Шварца на плоскости параметров находим из совместного решения уравнений:

1. $(1 - 9x^2) = 0;$
2. $\mu = b - 3x^2 = -1;$
3. Уравнение бифуркационной кривой.

Решение: $b = \frac{2}{3}, a = \pm\frac{16}{27}.$
Линиям, уходящим от этой точки и точки сборки вверх, отвечает отрицательная производная Шварца, т.е. это линии бифуркации удвоения периода. Отрезок, уходящий вниз и соединяющий эти точки, характеризуется положительной производной Шварца и отвечает жесткому переходу через мультипликатор $\mu = -1.$
Таким образом, наш анализ выявил новую бифуркацию коразмерности два: точку на линии $\mu = -1$, в которой линия удвоения превращается в линию жесткого переход через мультипликатор μ\-1. Можно, однако, показать, что мы установили не все бифуркационные линии, которые подходят к данной точке. Область существования устойчивого 2-цикла ограничена снизу линией касательной бифуркации, которая также подходит к исследуемой точке. В параметрической форме она ищется из условия реализации 2-цикла и обращения его мультипликатора в $+1$.
Линия жесткого перехода через мультипликатор $\mu = -1$ и линия касательной бифуркации 2-цикла подходят к обсуждаемой точке с одинаковым наклоном, т.е. линия, являющаяся их объединением и не имеет излома. Обсуждаемая бифуркация неимеет в научной литературе строго определенного названия. Один из часто используемых вариантов: вырожденная **флип-бифуркация (degenerate flip)**.
Для показанной ниже структуры Мира предложил название «spring area».
Структуры «crossroad area» и «spring area» появляются в других двухпараметрических отображениях, а также, имея коразмерность два, наблюдаются и для трехпараметрических отображений в тех или иных сечениях пространства параметров. Конечно, для трехпараметрических отображений картина бифуркаций существенно усложняется. Возможно, например, превращение структуры «crossroad area» в «spring area», и наоборот, при вариации третьего параметра (так называемый «crossroad area - spring area transition»).

1. $x_{n+1} = a - bx + x_n^3$

Неподвижные точки

Для отображения
$x_{n+1} = a - bx + x_n^3$
уравнение неподвижных точек имеет следующий вид
$x^3 - (b+1)x + a = 0.$

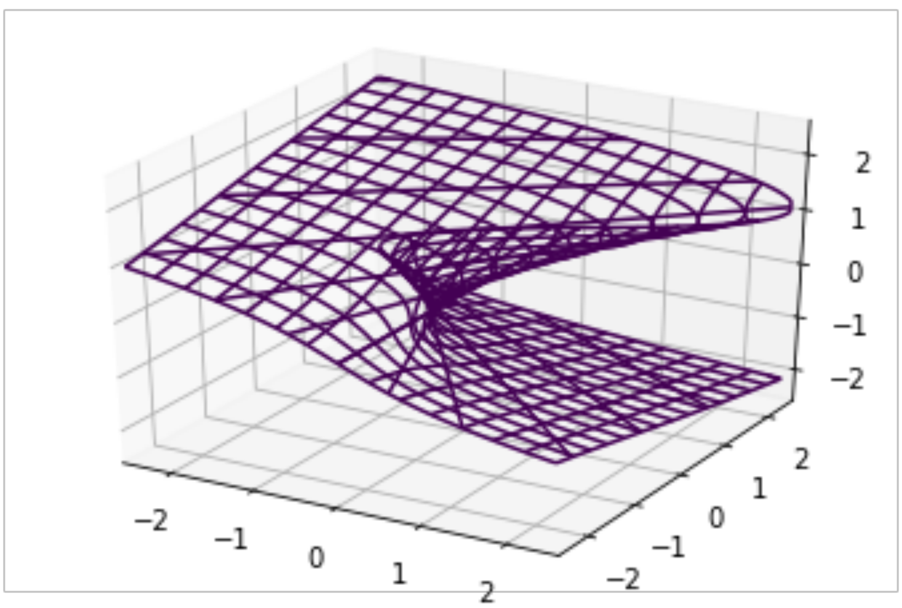
Это полином третьего порядка, который может иметь 3 корня.

Построим график $f(a,b) = x(a,b)$ для некоторых значений параметров a, b.

In [0]:

```
def f(a, b, x):  # TODO lambda
  return np.power(x, 3) - (b + 1)*x + a
def a(b, x):  # TODO lambda
  return np.power(x, 3) - (b + 1)*x

plot_implicit(f)
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/contour.py:1243:
UserWarning: No contour levels were found within the data range.
  warnings.warn("No contour levels were found"
```



Мультипликатор:

Точки бифуркации:

Вернемся к нашему отображению.

$f(x) = x^3 - (b+1)x + a$
$f'(x) = 3x^2 - b.$

- $\mu = 1$
  $3x^2 - b = 1$
  $x^2 = \frac{1+b}{3}$
  $x = \pm\sqrt{\frac{1+b}{3}}$
  Используем: $x^3 - (b+1)x - a = 0$

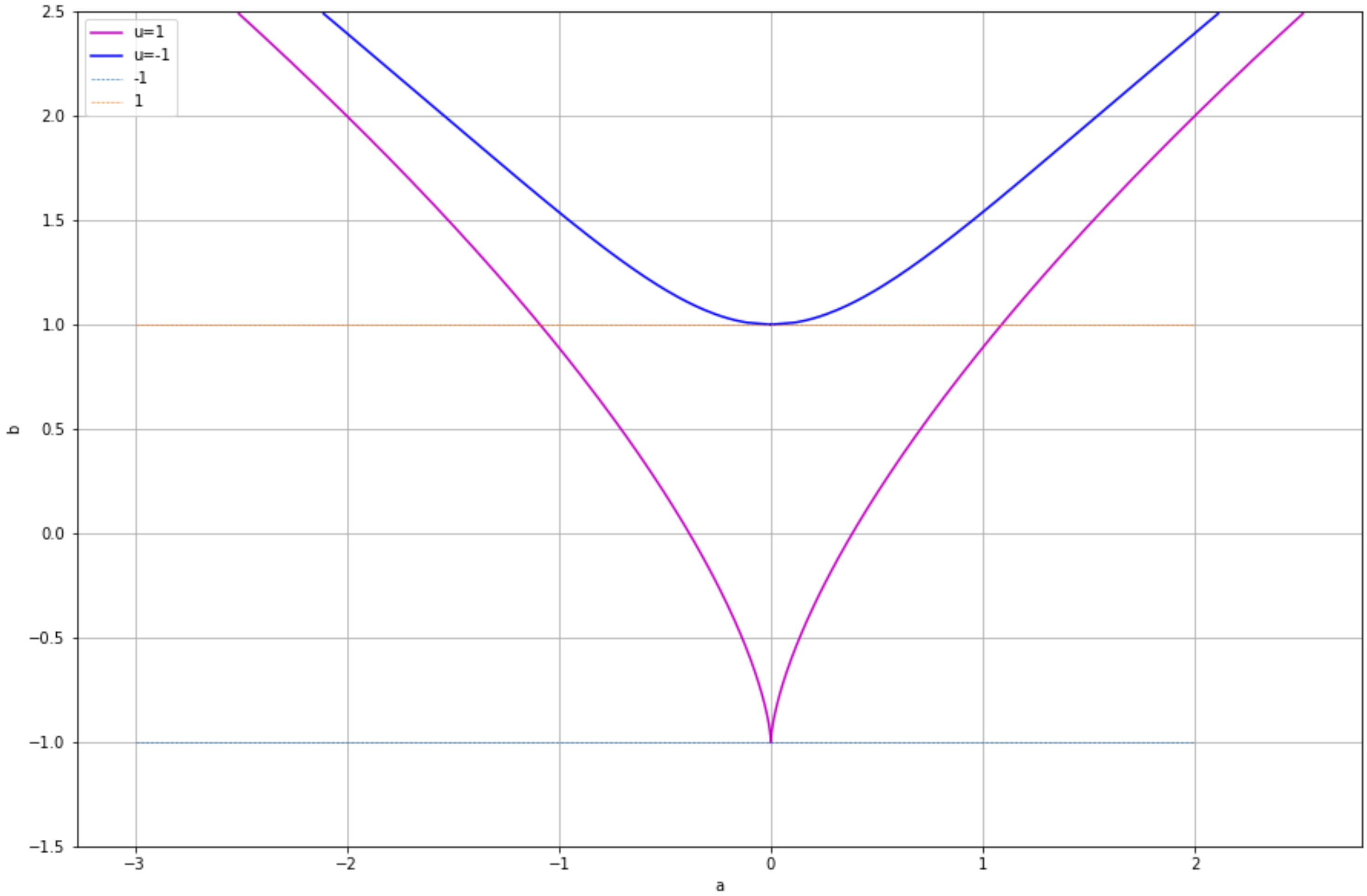
  $a = -x^3 + x(b+1)$
  $a = \pm\sqrt{\frac{b+1}{3}}(\frac{-1-b}{3} + b + 1)$
  $a = \pm\sqrt{\frac{b+1}{3}}\frac{2+2b}{3}$ - бифуркация $\mu = 1$ при $b > -1$

- $\mu = -1$
  $3x^2 - b = -1$
  $x^2 = \frac{-1+b}{3}$
  $x = \pm\sqrt{\frac{-1+b}{3}}$
  Используем: $x^3 - (b+1)x - a = 0$

  $a = -x^3 + x(b+1)$
  $a = \pm\sqrt{\frac{b-1}{3}}(\frac{1-b}{3} + b + 1)$
  $a = \pm\sqrt{\frac{b-1}{3}}\frac{4+2b}{3}$ - бифуркация $\mu = -1$ при $b > 1$

In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt((b + 1)/3)*(2 + 2*b)/3
def g(b):  # TODO lambda
  return -np.sqrt((b + 1)/3)*(2 + 2*b)/3

def h(b):  # TODO lambda
  return np.sqrt((b - 1)/3)*(4 + 2*b)/3
def t(b):  # TODO lambda
  return -np.sqrt((b - 1)/3)*(4 + 2*b)/3
ax = new_plot()

draw_line_revert(ax, f, -1, 2.5, style='m', label='u=1')
draw_line_revert(ax, g, -1, 2.5, style='m')
draw_line_revert(ax, h, 1, 2.5, style='b', label='u=-1')
draw_line_revert(ax, t, 1, 2.5, style='b')
draw_asym_revert(-1, label='-1')
draw_asym_revert(1, label='1')
ax.set(xlabel='a', ylabel='b')
show_plot(-1.5, 2.5)
```



Рассмотрим устойчивость/неустойчивость

Точка будет устойчивой, если $|\mu| < 1$ и неустойчивой, если $|\mu| > 1$.

Зафиксируем $a = 0$ и построим график x, b

$x = 0 - bx + x^3$
$x(x^2 - 1 - b) = 0$

- $x = 0$

  Мультипликатор:
  $\mu = 3x^2 - b$

  $-1 < \mu < 1$
  $-1 < 3x^2 - b < 1$
  Подставим $x = 0$

  $-1 < 3(0)^2 - b < 1$ $-1 < -b < 1$
  $-1 < b < 1$
  $-1 < b < 1$ - устойчивая точка
- $x^2 = b + 1$
  $x = \pm\sqrt{b+1}$

  Мультипликатор:
  $\mu = 3x^2 - b$

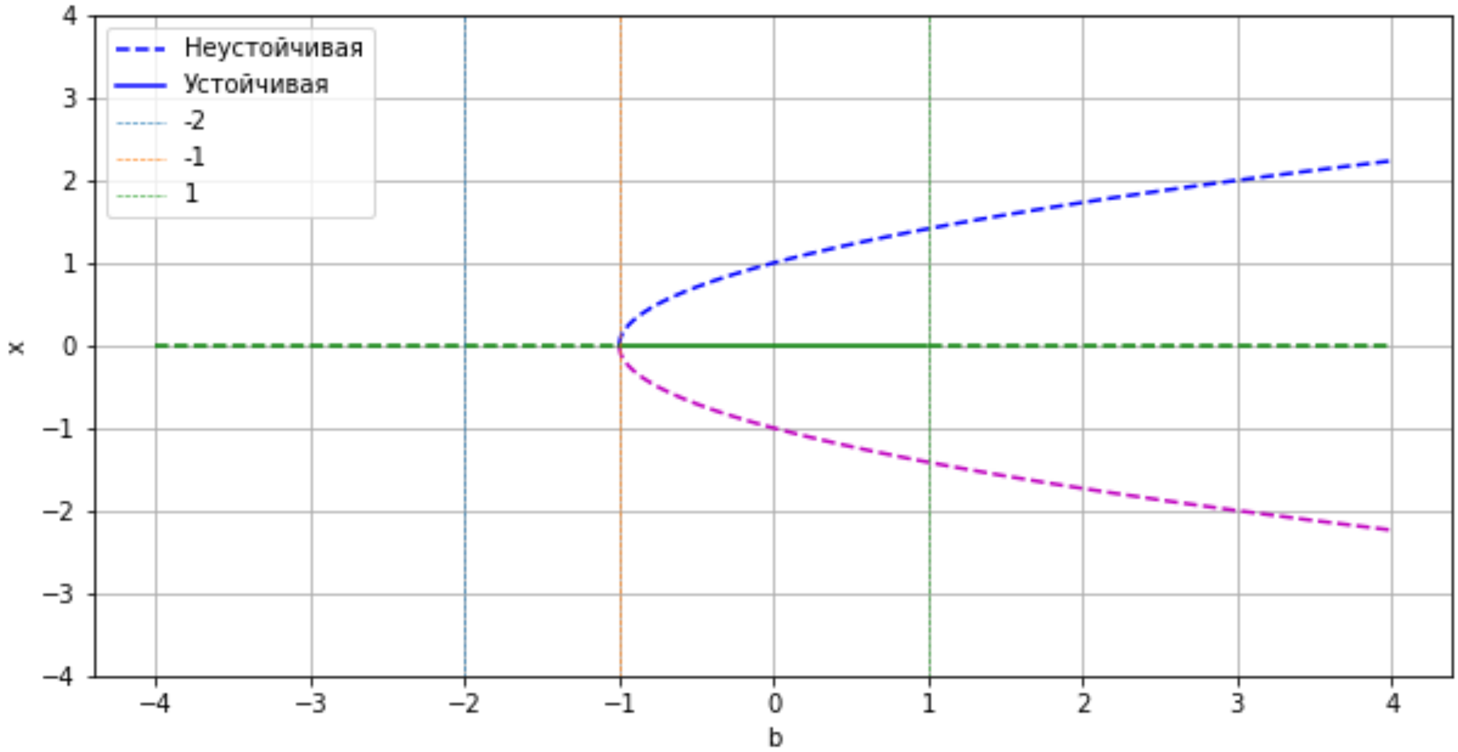
  $-1 < \mu < 1$
  $-1 < 3x^2 - b < 1$
  Подставим $x = \pm\sqrt{b+1}$

  $-1 < 3(\pm\sqrt{b+1})^2 - b < 1$
  $-1 < 3(b+1) - b < 1$
  $-1 < 2b + 3 < 1$
  $-2 < b < -1$ - устойчивая точка

In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt(b + 1)
def g(b):  # TODO lambda
  return -np.sqrt(b + 1)

def h(b):
  return b*0


ax = new_plot()

draw_line(ax, f, -4, -2, style='b--',label='Неустойчивая')
draw_line(ax, f, -2, -1, style='b',label='Устойчивая')
draw_line(ax, f, -1, 4, style='b--')

draw_line(ax, g, -4, -2, style='m--')
draw_line(ax, g, -2, -1, style='m')
draw_line(ax, g, -1, 4, style='m--')

draw_line(ax, h, -4, -1, style='g--')
draw_line(ax, h, -1, 1, style='g')
draw_line(ax, h, 1, 4, style='g--')



draw_asym(-2,label='-2')
draw_asym(-1,label='-1')
draw_asym(1,label='1')
ax.set(xlabel='b', ylabel='x')
show_plot(-4, 4)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in sqrt

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: invalid value encountered in sqrt
  after removing the cwd from sys.path.
```



Два цикл

Уравнения для поиска элементов 2-цикла имеют вид:

$x_1 = f(x_2)$
$x_2 = f(x_1) \Rightarrow$

$x_1 = a - bx_2 + x_2^3$

$x_2 = a - bx_1 + x_1^3$

Подставим одно в другое и найдем x_1, x_2

Зафиксируем $a = 0$

$x_2 = (x_2^3 - bx_2)^3 - b(x_2^3 - bx_2)$

$b^2x - bx^3 - b^3x^3 + 3b^2x^5 - 3bx^7 + x^9 - x = 0$

$x(x^2 - b + 1)(x^2 - b - 1)(x^4 - bx^2 + 1) = 0$

$x(x^2 - b - 1) = 0$

$x = 0$ $x = \pm\sqrt{b + 1}$ Заметим, что это в точности неподвижные точки (1-цикл), полученные в первом пункте. Потому что в условие, которые мы задали изначало он входит при $x_1 = x_2$

$x^2 - b + 1 = 0 => x = \pm\sqrt{b - 1}$

$x^4 - bx^2 + 1 = 0 => x = \pm\sqrt{\frac{b\pm\sqrt{b^2-4}}{2}}$

Их следует дополнить условием на мультипликатор:

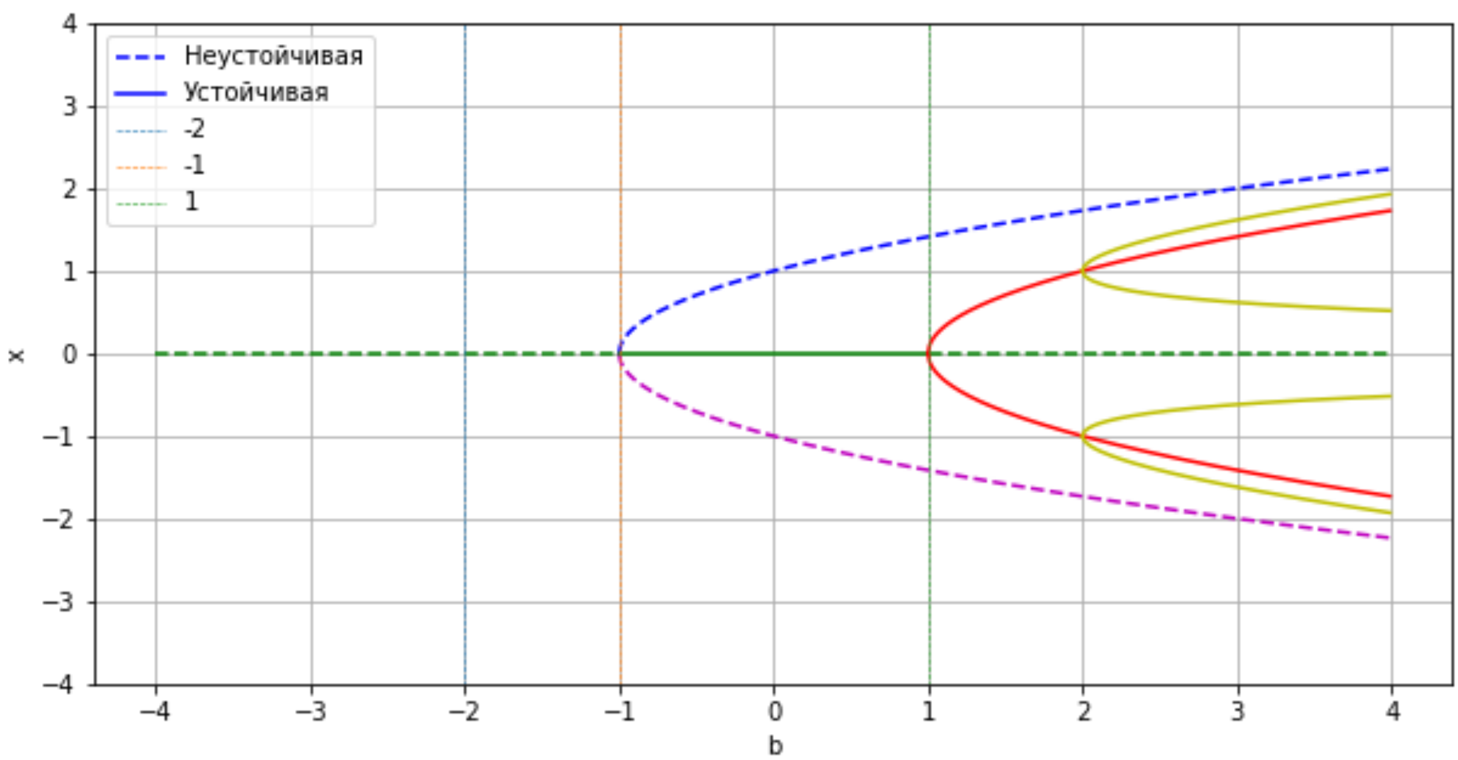
$\mu = \pm(3x_1^2 - b)(3x_2^2 - b)$

В этом случае следует выбирать знак «+» для поиска касательной бифуркации 2-цикла и знак «–» – для бифуркации удвоения.

In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt(b + 1)
def g(b):  # TODO lambda
  return -np.sqrt(b + 1)

def h(b):
  return b*0
def t1(b):
  return np.sqrt(b-1)
def t2(b):
  return -np.sqrt(b-1)
def t3(b):
  return np.sqrt((b-np.sqrt(b*b-4))/2)
def t4(b):
  return -np.sqrt((b-np.sqrt(b*b-4))/2)
def t5(b):
  return -np.sqrt((b+np.sqrt(b*b-4))/2)
def t6(b):
  return np.sqrt((b+np.sqrt(b*b-4))/2)

ax = new_plot()

draw_line(ax, f, -4, -2, style='b--',label='Неустойчивая')
draw_line(ax, f, -2, -1, style='b',label='Устойчивая')
draw_line(ax, f, -1, 4, style='b--')

draw_line(ax, g, -4, -2, style='m--')
draw_line(ax, g, -2, -1, style='m')
draw_line(ax, g, -1, 4, style='m--')

draw_line(ax, h, -4, -1, style='g--')
draw_line(ax, h, -1, 1, style='g')
draw_line(ax, h, 1, 4, style='g--')

draw_line(ax,t1,-2,4,style='r')
draw_line(ax,t2,-2,4,style='r')
draw_line(ax,t3,-2,4,style='y')
draw_line(ax,t4,-2,4,style='y')
draw_line(ax,t5,-2,4,style='y')
draw_line(ax,t6,-2,4,style='y')


draw_asym(-2,label='-2')
draw_asym(-1,label='-1')
draw_asym(1,label='1')
ax.set(xlabel='b', ylabel='x')
show_plot(-4, 4)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in sqrt

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: invalid value encountered in sqrt
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: Ru
ntimeWarning: invalid value encountered in sqrt
  if __name__ == '__main__':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:11: R
untimeWarning: invalid value encountered in sqrt
  # This is added back by InteractiveShellApp.init_path()
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:13: R
untimeWarning: invalid value encountered in sqrt
  del sys.path[0]
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:15: R
untimeWarning: invalid value encountered in sqrt
  from ipykernel import kernelapp as app
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: R
untimeWarning: invalid value encountered in sqrt
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:19: R
untimeWarning: invalid value encountered in sqrt
```



Рассмотрим устойчивость/неустойчивость

- $x_{3,4} = \pm\sqrt{b-1}$

  $\mu_{x_{3,4}} = f'(x_3) \cdot f'(x_4) = (3x_3^2 - b)(3x_4^2 - b) = 4b^2 - 12b + 9$

  $-1 < 4b^2 - 12b + 9 < 1$

  $1 < x < 2$ на промежутке - $x_3,\ x_4$ - устойчивые

- $x = \pm\sqrt{\frac{b\pm\sqrt{b^2-4}}{2}}$

  $\mu = -2b^2 + 9$

  $-1 < -2b^2 + 9 < 1$

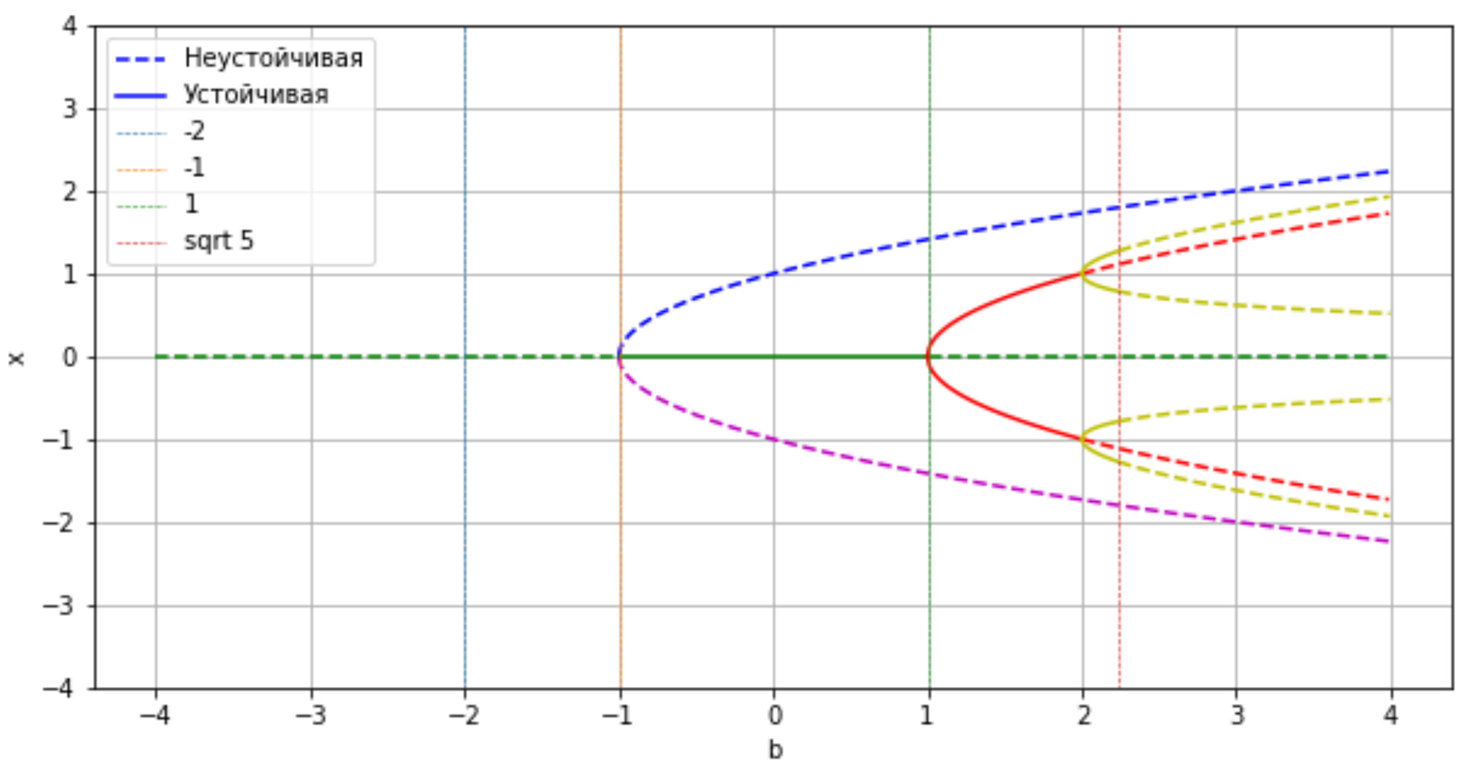
  $-\sqrt{5} < x < -2, 2 < x < \sqrt{5}$ на промежутке $x = \pm\sqrt{\frac{b\pm\sqrt{b^2-4}}{2}}$ - устойчивые

In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt(b + 1)
def g(b):  # TODO lambda
  return -np.sqrt(b + 1)

def h(b):
  return b*0
def t1(b):
  return np.sqrt(b-1)
def t2(b):
  return -np.sqrt(b-1)
def t3(b):
  return np.sqrt((b-np.sqrt(b*b-4))/2)
def t4(b):
  return -np.sqrt((b-np.sqrt(b*b-4))/2)
def t5(b):
  return -np.sqrt((b+np.sqrt(b*b-4))/2)
def t6(b):
  return np.sqrt((b+np.sqrt(b*b-4))/2)

ax = new_plot()

draw_line(ax, f, -4, -2, style='b--',label='Неустойчивая')
draw_line(ax, f, -2, -1, style='b',label='Устойчивая')
draw_line(ax, f, -1, 4, style='b--')

draw_line(ax, g, -4, -2, style='m--')
draw_line(ax, g, -2, -1, style='m')
draw_line(ax, g, -1, 4, style='m--')

draw_line(ax, h, -4, -1, style='g--')
draw_line(ax, h, -1, 1, style='g')
draw_line(ax, h, 1, 4, style='g--')
```

```
draw_line(ax,t1,1,2,style='r')
draw_line(ax,t1,2,4,style='r--')
draw_line(ax,t2,1,2,style='r')
draw_line(ax,t2,2,4,style='r--')

draw_line(ax,t3,2,np.sqrt(5),style='y')
draw_line(ax,t3,np.sqrt(5),4,style='y--')

draw_line(ax,t4,2,np.sqrt(5),style='y')
draw_line(ax,t4,np.sqrt(5),4,style='y--')

draw_line(ax,t5,2,np.sqrt(5),style='y')
draw_line(ax,t5,np.sqrt(5),4,style='y--')

draw_line(ax,t6,2,np.sqrt(5),style='y')
draw_line(ax,t6,np.sqrt(5),4,style='y--')



draw_asym(-2,label='-2')
draw_asym(-1,label='-1')
draw_asym(1,label='1')
draw_asym(np.sqrt(5),label='sqrt 5')
ax.set(xlabel='b', ylabel='x')
show_plot(-4, 4)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in sqrt

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: invalid value encountered in sqrt
  after removing the cwd from sys.path.
```



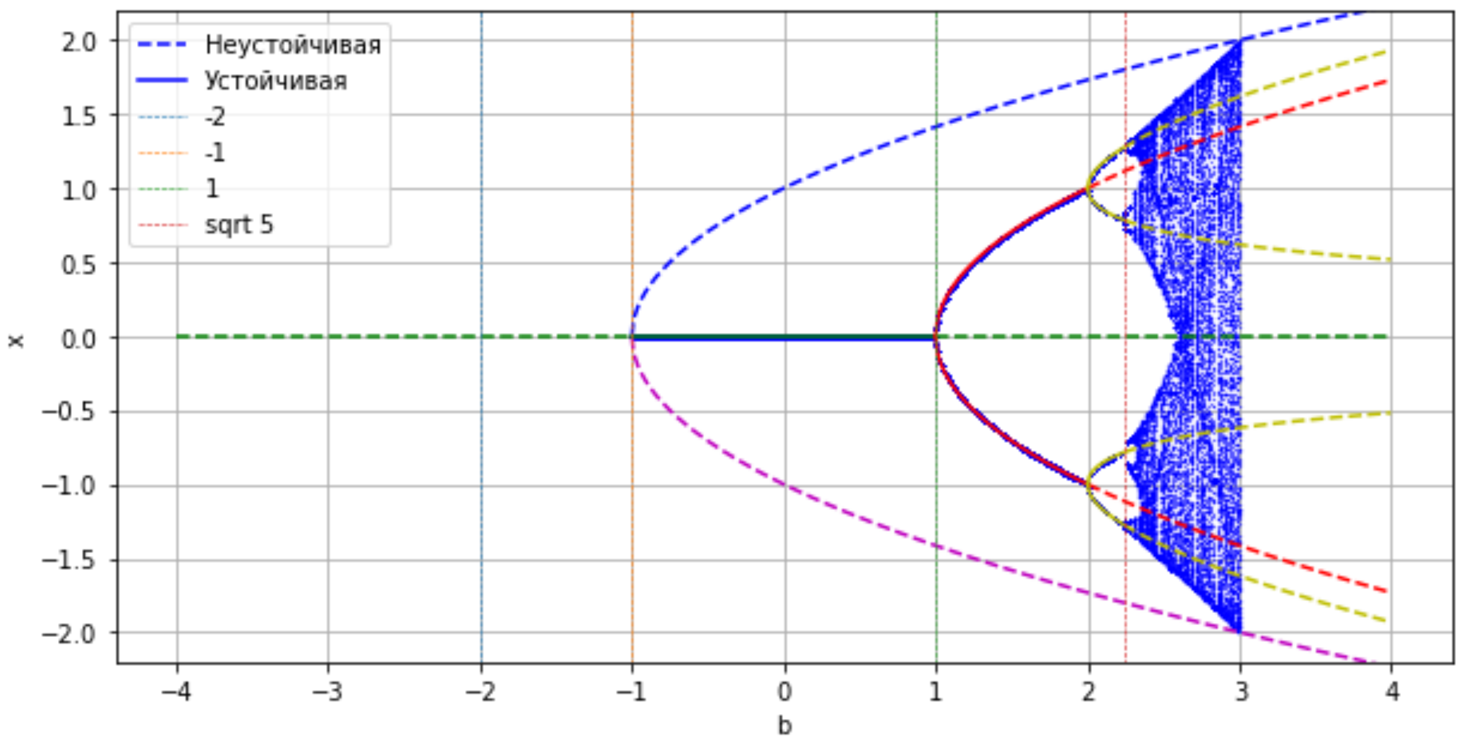
Дерево бифуркаций

In [0]:

```
def u(x, b, t=1):
    if t > 1:
        x = f(x, b, t-1)
    return (-b*x + x*x*x)

def f(b):  # TODO lambda
  return np.sqrt(b + 1)
def g(b):  # TODO lambda
  return -np.sqrt(b + 1)

def h(b):
  return b*0
def t1(b):
  return np.sqrt(b-1)
def t2(b):
  return -np.sqrt(b-1)
def t3(b):
  return np.sqrt((b-np.sqrt(b*b-4))/2)
def t4(b):
  return -np.sqrt((b-np.sqrt(b*b-4))/2)
def t5(b):
  return -np.sqrt((b+np.sqrt(b*b-4))/2)
def t6(b):
  return np.sqrt((b+np.sqrt(b*b-4))/2)



ax = new_plot()


draw_tree(ax, u, -3, -1, style='.b', start=2./3)
draw_tree(ax, u, -1, 1, style='.b', start=0)
draw_tree(ax, u, 1, 5, style='.b', start=2./3, inf_n=200)




draw_line(ax, f, -4, -2, style='b--',label='Неустойчивая')
draw_line(ax, f, -2, -1, style='b',label='Устойчивая')
draw_line(ax, f, -1, 4, style='b--')

draw_line(ax, g, -4, -2, style='m--')
draw_line(ax, g, -2, -1, style='m')
draw_line(ax, g, -1, 4, style='m--')

draw_line(ax, h, -4, -1, style='g--')
draw_line(ax, h, -1, 1, style='g')
draw_line(ax, h, 1, 4, style='g--')


draw_line(ax,t1,1,2,style='r')
draw_line(ax,t1,2,4,style='r--')
draw_line(ax,t2,1,2,style='r')
draw_line(ax,t2,2,4,style='r--')
```

```
draw_line(ax,t3,2,np.sqrt(5),style='y')
draw_line(ax,t3,np.sqrt(5),4,style='y--')

draw_line(ax,t4,2,np.sqrt(5),style='y')
draw_line(ax,t4,np.sqrt(5),4,style='y--')

draw_line(ax,t5,2,np.sqrt(5),style='y')
draw_line(ax,t5,np.sqrt(5),4,style='y--')

draw_line(ax,t6,2,np.sqrt(5),style='y')
draw_line(ax,t6,np.sqrt(5),4,style='y--')




draw_asym(-2,label='-2')
draw_asym(-1,label='-1')
draw_asym(1,label='1')
draw_asym(np.sqrt(5),label='sqrt 5')
ax.set(xlabel='b', ylabel='x')




show_plot(-2.2, 2.2)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: Ru
ntimeWarning: overflow encountered in double_scalars
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: Ru
ntimeWarning: invalid value encountered in double_scalars
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: R
untimeWarning: invalid value encountered in sqrt
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: R
untimeWarning: invalid value encountered in sqrt
  if sys.path[0] == '':
```



Дерево бифуркаций

In [0]:

```

```

In [0]:

```

```

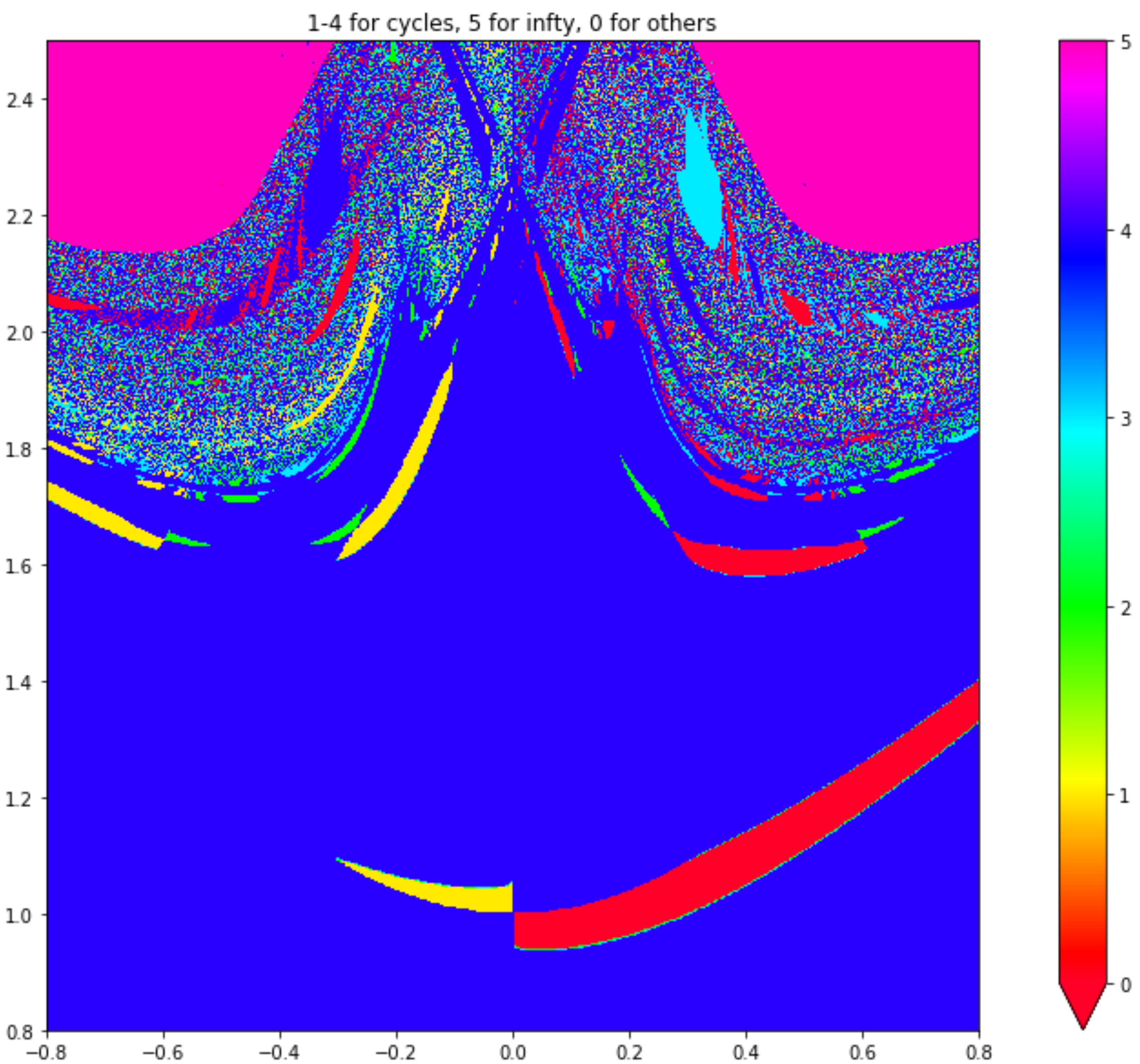
Карта режимов

In [0]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

amin, amax, bmin, bmax = -0.8, 0.8, 0.8, 2.5

apoints, bpoints = 800, 850 # число точек по горизонтали и вертикали
max_iterations = 300
infinity_border = 10000

image = np.zeros((apoints, bpoints))
# image — это двумерный массив, в котором будет записана картинка, по умолчани
ю он заполнен нулями

eps = 10e-12

a_delta = (amax - amin) / apoints
a_list = [amin + a_delta * i for i in range(apoints)]
b_delta = (bmax - bmin) / bpoints
b_list = [bmin + b_delta * i for i in range(bpoints)]

def f(x, a, b, n):
    if n == 1:
        return a - b*x + np.power(x, 3)
    else:
        x = f(x, a, b, n-1)
        return a - b*x + np.power(x, 3)

for ai, a in enumerate(a_list):
    for bi, b in enumerate(b_list):
        x_n = 0
        for i in range(max_iterations):
            x_n = f(x_n, a, b, 1)
            if np.abs(x_n) > infinity_border:
                x_n = infinity_border
                break
        # fixed point
        if f(x_n, a, b, 1) - x_n < eps:
            image[ai][len(image[ai])-1-bi] = 1
        # 2-cycle
```

```
        if f(x_n, a, b, 2) - x_n < eps:

            image[ai][len(image[ai])-1-bi] = 2
        # 4-cycle
        if f(x_n, a, b, 4) - x_n < eps:
            image[ai][len(image[ai])-1-bi] = 3
        # 8-cycle
        if f(x_n, a, b, 8) - x_n < eps:
            image[ai][len(image[ai])-1-bi] = 4
        # infinity
        if x_n == infinity_border:
            image[ai][len(image[ai])-1-bi] = 5
        # 0 for other configurations

plt.figure(figsize=(16, 10))
tmp = plt.imshow(image.T, cmap='gist_rainbow', extent=(amin, amax, bmin, bmax)
)
plt.colorbar(tmp, extend='min')
plt.title('1-4 for cycles, 5 for infty, 0 for others')
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:26: R
untimeWarning: overflow encountered in power
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:26: R
untimeWarning: invalid value encountered in double_scalars
```



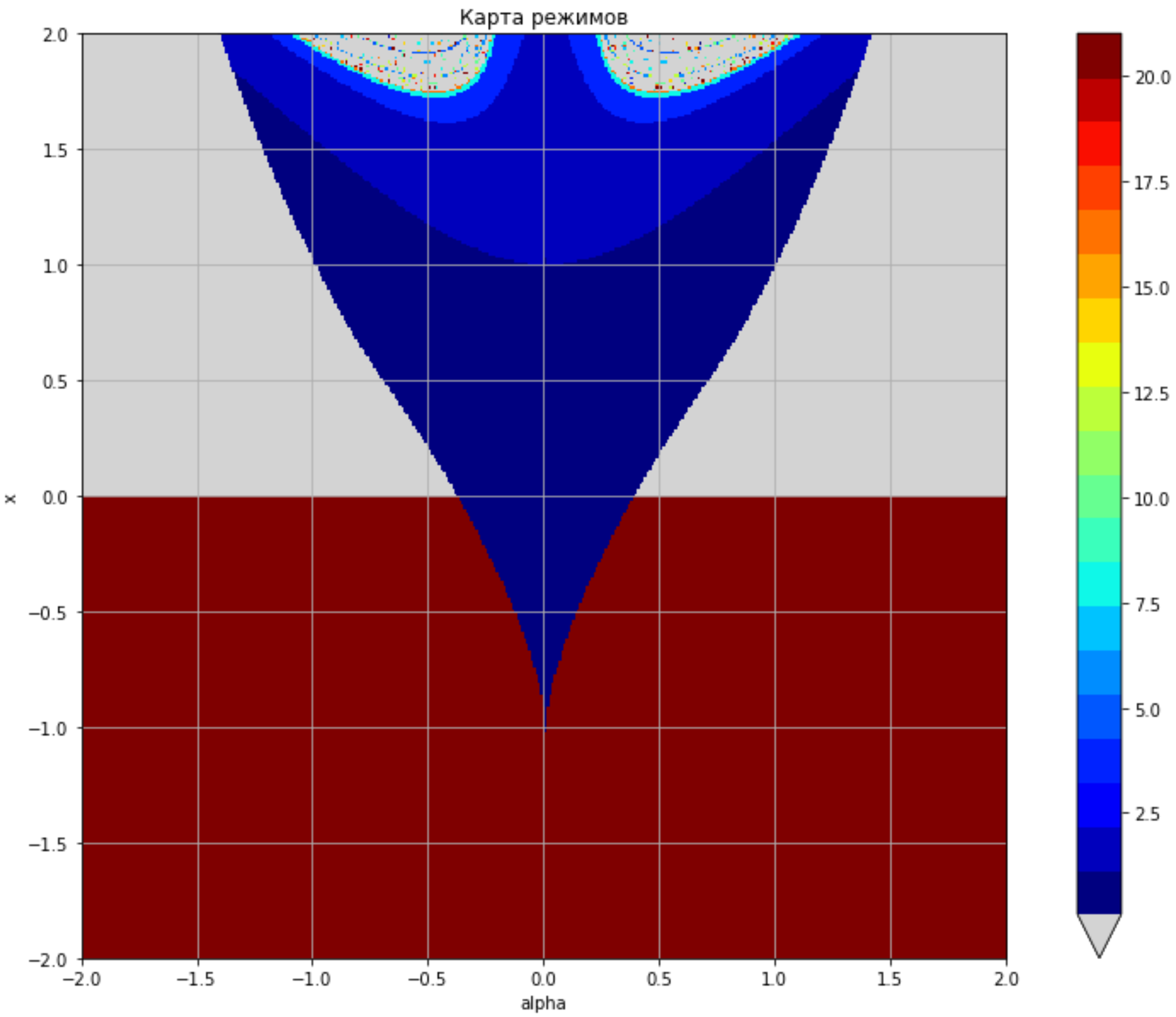
In [0]:

```
def draw_mode_map(ax, f, x_min, x_max, y_min, y_max, step=0.01, iter=1000, inf
=10, inf_n=20, eps=10e-4, start=0.1):

  b = np.arange(x_min, x_max, step)
  m = np.arange(y_min, y_max, step)


  z = np.empty((m.shape[0], b.shape[0]))

  for j, b_x in enumerate(b):
      for i, m_y in enumerate(m):

          x = start
          for k in range(iter):
              x = f(x, b_x, m_y)

          if x > inf or x < -inf:
              z[-i-1][j] = inf_n+1
              continue

          x_tmp = x
          k = 0 # 1-cicle len

          z[-i-1][j] = 0
          for k in range(inf_n):
              x_tmp = f(x_tmp, b_x, m_y)
              if -eps < x - x_tmp < eps:
                  z[-i-1][j] = k+1
                  break




  from matplotlib.colors import ListedColormap


  cmap = plt.get_cmap('jet', 20)
  cmap.set_under('lightgray')


  cax = plt.imshow(z, extent=[x_min, x_max, y_min, y_max], interpolation='none
', cmap=cmap, vmin=0.1, vmax=z.max())# plt.cm.jet)
  plt.colorbar(cax, extend='min')
```

In [0]:

```
def f(x, a, b):
  return a - b*x + x*x*x


ax = new_plot(title='Карта режимов')

draw_mode_map(ax, f, -2, 2, -2, 2)

plt.show()
```

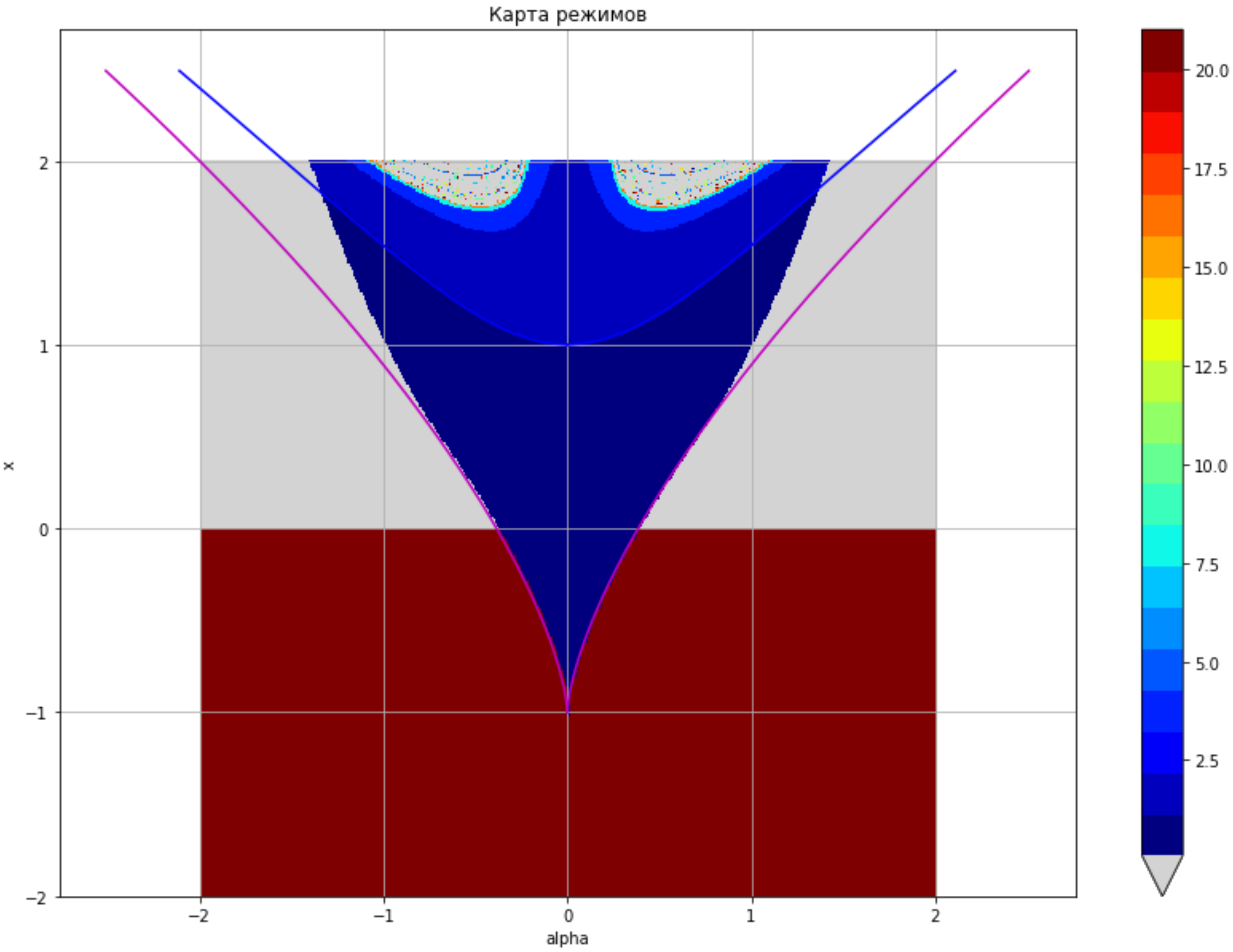
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: overflow encountered in double_scalars

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in double_scalars
```



In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt((b + 1)/3)*(2 + 2*b)/3
def g(b):  # TODO lambda
  return -np.sqrt((b + 1)/3)*(2 + 2*b)/3

def h(b):  # TODO lambda
  return np.sqrt((b - 1)/3)*(4 + 2*b)/3
def t(b):  # TODO lambda
  return -np.sqrt((b - 1)/3)*(4 + 2*b)/3

def г(x, a, b):
  return a - b*x + x*x*x


ax = new_plot(title='Карта режимов')

draw_mode_map(ax, г, -2, 2, -2, 2)



draw_line_revert(ax, f, -1, 2.5, style='m', label='u=1')
draw_line_revert(ax, g, -1, 2.5, style='m')
draw_line_revert(ax, h, 1, 2.5, style='b', label='u=-1')
draw_line_revert(ax, t, 1, 2.5, style='b')

plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: R
untimeWarning: overflow encountered in double_scalars
  if sys.path[0] == '':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: R
untimeWarning: invalid value encountered in double_scalars
  if sys.path[0] == '':
```



2. $x_{n+1} = a + bx - x_n^3$

Неподвижные точки

$x = a + bx - x^3$
$x^3 + (1 - b)x - a = 0$
$f(x) = -x^3 + bx + a.$

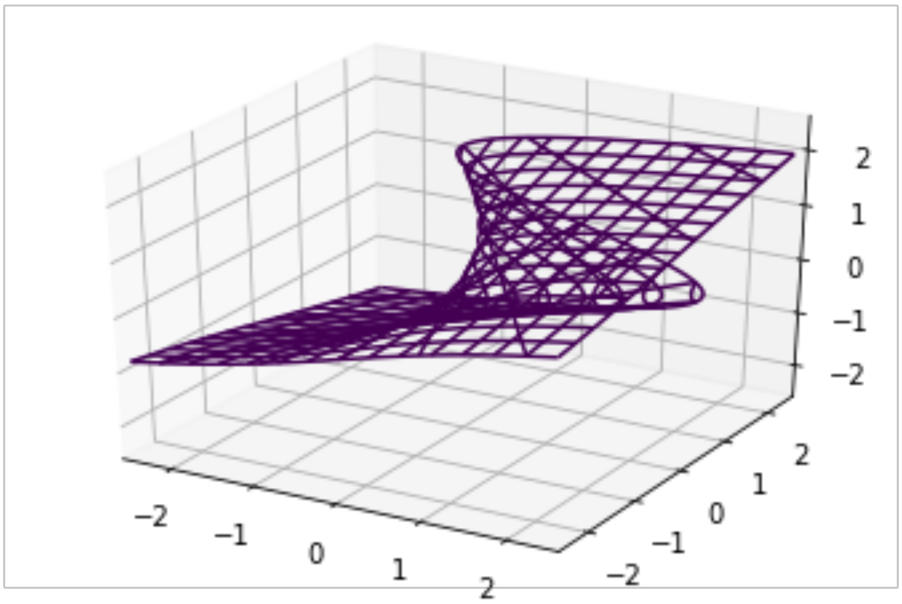
Это полином третьего порядка, который имеет 3 неподвижные точки.

Построим график $f(a, b) = x(a, b)$ для некоторых значений параметров a, b.

In [0]:

```
def f(a, b, x):  # TODO lambda
  return -np.power(x, 3) + b*x + a
def a(b, x):  # TODO lambda
  return np.power(x, 3)- b*x

plot_implicit(f)
```

```
/usr/local/lib/python3.6/dist-packages/matplotlib/contour.py:1243:
UserWarning: No contour levels were found within the data range.
  warnings.warn("No contour levels were found"
```



Точки бифуркации

Вернемся к нашему отображению.

$f(x) = x^3 + (1 - b)x - a$
$f'(x) = b - 3x^2.$

- $\mu = 1$
  $b - 3x^2 = 1$
  $x^2 = \frac{b-1}{3}$
  $x = \pm\sqrt{\frac{b-1}{3}}$
  Используем: $x^3 + (1 - b)x - a = 0$

  $a = x^3 + x(1 - b)$
  $a = \pm\sqrt{\frac{b-1}{3}}(\frac{b-1}{3} + 1 - b)$
  $a = \pm\sqrt{\frac{b-1}{3}}\frac{-2b+2}{3}$ - бифуркация $\mu = 1$ при $b \geq 1$

- $\mu = -1$
  $b - 3x^2 = -1$
  $x^2 = \frac{b+1}{3}$
  $x = \pm\sqrt{\frac{b+1}{3}}$
  Используем: $x^3 + (1 - b)x - a = 0$

  $a = x^3 + x(1 - b)$
  $a = \pm\sqrt{\frac{b+1}{3}}(\frac{b+1}{3} + 1 - b)$
  $a = \pm\sqrt{\frac{b+1}{3}}\frac{-2b+4}{3}$ - бифуркация $\mu = -1$ при $b \geq -1$

In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt((b - 1)/3)*(2 - 2*b)/3
def g(b):  # TODO lambda
  return -np.sqrt((b - 1)/3)*(2 - 2*b)/3

def h(b):  # TODO lambda
  return np.sqrt((b +1)/3)*(4 - 2*b)/3
def t(b):  # TODO lambda
  return -np.sqrt((b +1)/3)*(4 - 2*b)/3
ax = new_plot()
def tmp(b):
  return 4*b*b+12*b+9
def mtmp(b):
  return -4*b*b-12*b-9

draw_line_revert(ax, f, -2, 1, style='m', label='u=1')
draw_line_revert(ax, g,-2, 1, style='m')
draw_line_revert(ax, h, -2, 1, style='b', label='u=-1')
draw_line_revert(ax, t,-2, 1, style='b')
# draw_line_revert(ax, tmp,-2, 1, style='b')
# draw_line_revert(ax, mtmp,-2, 1, style='b')
draw_asym_revert(-1, label='-1')
draw_asym_revert(1, label='1')
ax.set(xlabel='a', ylabel='b')
show_plot(-2, 6)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-c2b61347e18c> in <module>()
      8 def t(b):  # TODO lambda
      9   return -np.sqrt((b +1)/3)*(4 - 2*b)/3
---> 10 ax = new_plot()
     11 def tmp(b):
     12   return 4*b*b+12*b+9

NameError: name 'new_plot' is not defined
```

In [0]:

Рассмотрим устойчивость/неустойчивость

Точка будет устойчивой, если $|\mu| < 1$ и неустойчивой, если $|\mu| > 1$.

Зафиксируем $a = 0$ и построим график x, b

$x = 0 + bx - x^3$
$x(x^2 + 1 - b) = 0$

- $x = 0$

  Мультипликатор:
  $\mu = 3x^2 - b$

  $-1 < \mu < 1$
  $-1 < 3x^2 - b < 1$
  Подставим $x = 0$

  $-1 < 3(0)^2 - b < 1$ $-1 < -b < 1$
  $-1 < b < 1$
  $-1 < b < 1$ - устойчивая точка
- $x^2 = b - 1$
  $x = \pm\sqrt{b - 1}$

  Мультипликатор:
  $\mu = 3x^2 - b$

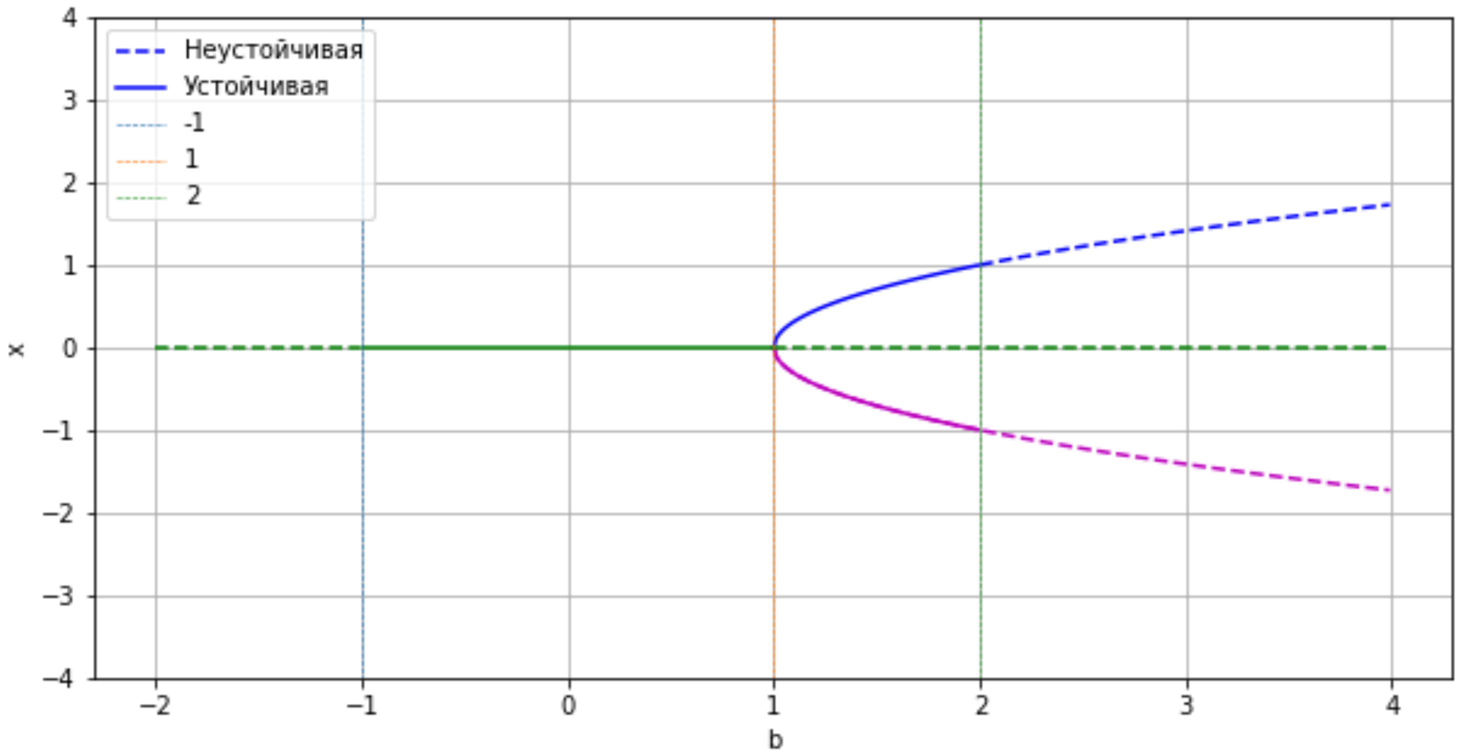
  $-1 < \mu < 1$
  $-1 < 3x^2 - b < 1$
  Подставим $x = \pm\sqrt{b - 1}$

  $-1 < 3(\pm\sqrt{b - 1})^2 - b < 1$ $-1 < 3(b - 1) - b < 1$
  $-1 < 2b - 3 < 1$
  $1 < b < 2$ - устойчивая точка

In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt(b - 1)
def g(b):  # TODO lambda
  return -np.sqrt(b - 1)

def h(b):
  return b*0


ax = new_plot()

draw_line(ax, f, -2, -1, style='b--',label='Неустойчивая')
draw_line(ax, f, 1, 2, style='b',label='Устойчивая')
draw_line(ax, f, 2, 4, style='b--')

draw_line(ax, g, -2, 1, style='m--')
draw_line(ax, g, 1, 2, style='m')
draw_line(ax, g, 1, 4, style='m--')

draw_line(ax, h, -2, -1, style='g--')
draw_line(ax, h, -1, 1, style='g')
draw_line(ax, h, 1, 4, style='g--')



draw_asym(-1,label='-1')
draw_asym(1,label='1')
draw_asym(2,label='2')
ax.set(xlabel='b', ylabel='x')
show_plot(-4, 4)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in sqrt

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: invalid value encountered in sqrt
  after removing the cwd from sys.path.
```



Два цикл

Уравнения для поиска элементов 2-цикла имеют вид:

$x_1 = f(x_2)$
$x_2 = f(x_1) \Rightarrow$

$x_1 = a + bx_2 - x_2^3$

$x_2 = a + bx_1 - x_1^3$

Подставим одно в другое и найдем x_1, x_2

Зафиксируем $a = 0$

$x_2 = b(bx_2 - x_2^3) - (bx_2 - x_2^3)^3$

$b^2x - bx^3 - b^3x^3 + 3b^2x^5 - 3bx^7 + x^9 - x = 0$

$x(x^2 - b + 1)(x^2 - b - 1)(x^4 - bx^2 + 1) = 0$

$x(x^2 - b + 1) = 0$

$x = 0$ $x = \pm\sqrt{b-1}$ Заметим, что это в точности неподвижные точки (1-цикл), полученные в первом пункте. Потому что в условие, которые мы задали изначало он входит при $x_1 = x_2$

$x^2 - b - 1 = 0 => x = \pm\sqrt{b+1}$

$x^4 - bx^2 + 1 = 0 => x = \pm\sqrt{\frac{b\pm\sqrt{b^2-4}}{2}}$

Их следует дополнить условием на мультипликатор:

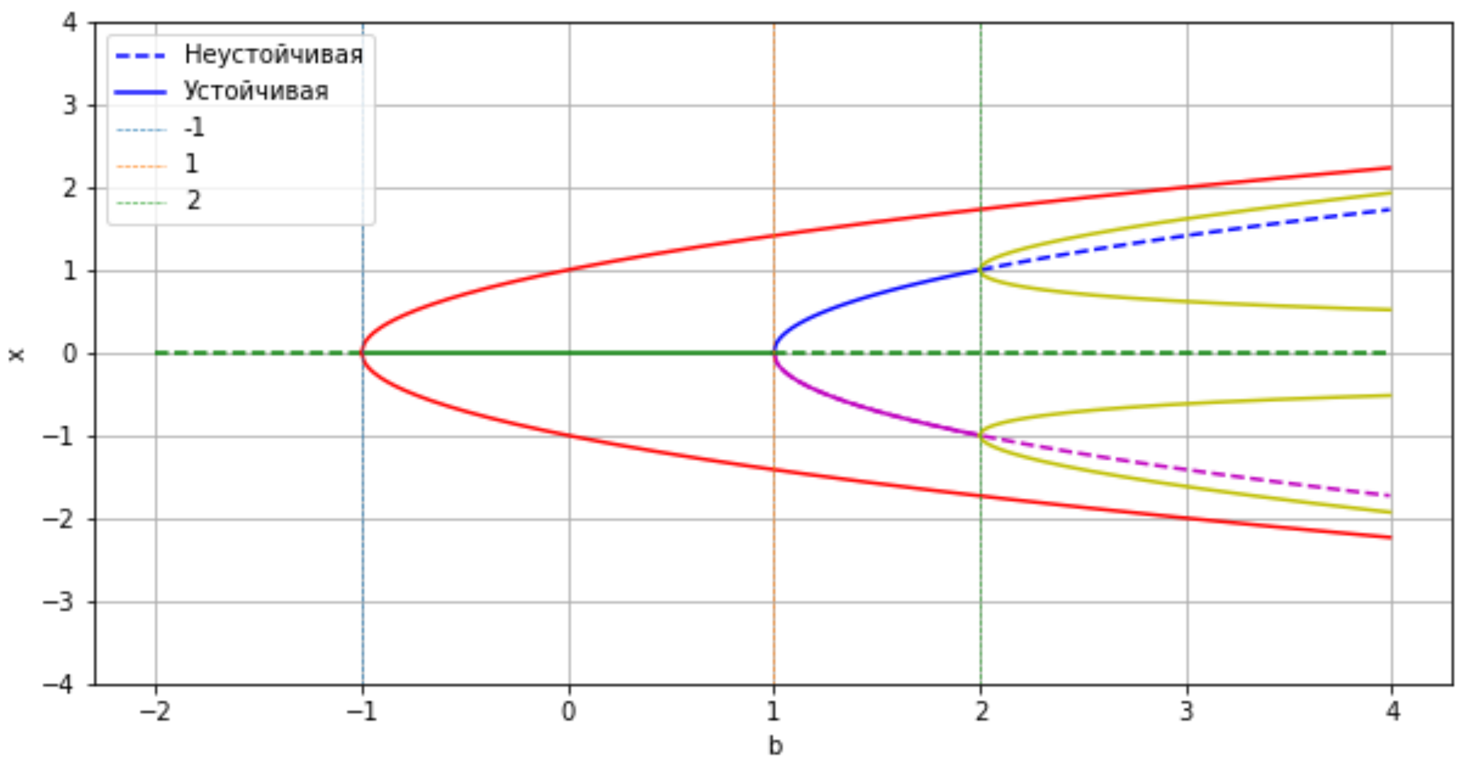
$\mu = \pm(b - 3x_1^2)(b - 3x_2^2)$

В этом случае следует выбирать знак «+» для поиска касательной бифуркации 2-цикла и знак «–» – для бифуркации удвоения.

In [0]:

```
def f(b):   # TODO lambda
  return np.sqrt(b - 1)
def g(b):   # TODO lambda
  return -np.sqrt(b - 1)

def h(b):
  return b*0
def t1(b):
  return np.sqrt(b+1)
def t2(b):
  return -np.sqrt(b+1)
def t3(b):
  return np.sqrt((b-np.sqrt(b*b-4))/2)
def t4(b):
  return -np.sqrt((b-np.sqrt(b*b-4))/2)
def t5(b):
  return -np.sqrt((b+np.sqrt(b*b-4))/2)
ax = new_plot()
def t6(b):
  return np.sqrt((b+np.sqrt(b*b-4))/2)

draw_line(ax, f, -2, -1, style='b--',label='Неустойчивая')
draw_line(ax, f, 1, 2, style='b',label='Устойчивая')
draw_line(ax, f, 2, 4, style='b--')

draw_line(ax, g, -2, 1, style='m--')
draw_line(ax, g, 1, 2, style='m')
draw_line(ax, g, 1, 4, style='m--')

draw_line(ax, h, -2, -1, style='g--')
draw_line(ax, h, -1, 1, style='g')
draw_line(ax, h, 1, 4, style='g--')

draw_line(ax,t1,-2,4,style='r')
draw_line(ax,t2,-2,4,style='r')
draw_line(ax,t3,-2,4,style='y')
draw_line(ax,t4,-2,4,style='y')
draw_line(ax,t5,-2,4,style='y')
draw_line(ax,t6,-2,4,style='y')

draw_asym(-1,label='-1')
draw_asym(1,label='1')
draw_asym(2,label='2')
ax.set(xlabel='b', ylabel='x')
show_plot(-4, 4)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in sqrt

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: invalid value encountered in sqrt
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: Ru
ntimeWarning: invalid value encountered in sqrt
  if __name__ == '__main__':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:11: R
untimeWarning: invalid value encountered in sqrt
  # This is added back by InteractiveShellApp.init_path()
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:13: R
untimeWarning: invalid value encountered in sqrt
  del sys.path[0]
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:15: R
untimeWarning: invalid value encountered in sqrt
  from ipykernel import kernelapp as app
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: R
untimeWarning: invalid value encountered in sqrt
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:20: R
untimeWarning: invalid value encountered in sqrt
```



Точки бифуркации

- $x_{3,4} = \pm\sqrt{b+1}$

  $\mu_{x_{3,4}} = f'(x_3) \cdot f'(x_4) = (b - 3x_3^2)(b - 3x_4^2) = 4b^2 + 12b + 9$

  - $1. \mu = 1$

  $\mu_{x_{3,4}} = 1$

  $4b^2 + 12b + 8 = 0$

  $b = -1; b = 2$

  - $2. \mu = -1$

  $4b^2 + 12b + 10 = 0$ нет действительных корней

- $x = \pm\sqrt{\frac{b \pm \sqrt{b^2 - 4}}{2}}$

  $\mu = -2b^2 + 9$

  - $1. \mu = 1$

  $2b^2 = 8$

  $b = \pm 2$

  - $2. \mu = -1$

  $2b^2 = 10$

  $b = \pm\sqrt{5}$

Рассмотрим устойчивость/неустойчивость

- $x_{3,4} = \pm\sqrt{b+1}$

  $\mu_{x_{3,4}} = f'(x_3) \cdot f'(x_4) = (b - 3x_3^2)(b - 3x_4^2) = 4b^2 + 12b + 9$

  $-1 < 4b^2 + 12b + 9 < 1$

  $-2 < x < -1$ на промежутке - $x_3, \ x_4$ - устойчивые

- $x = \pm\sqrt{\frac{b \pm \sqrt{b^2 - 4}}{2}}$

  $\mu = -2b^2 + 9$

  $-1 < -2b^2 + 9 < 1$

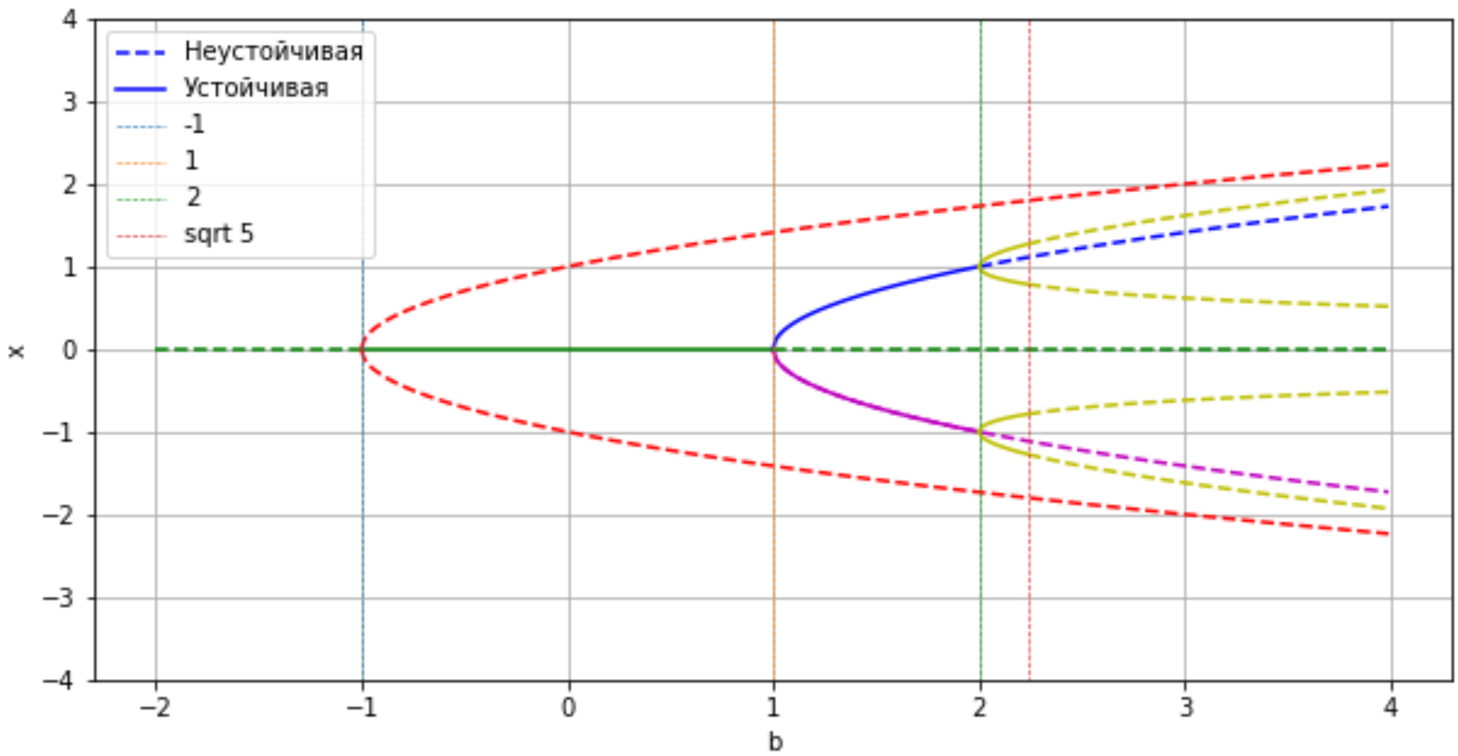
  $-\sqrt{5} < x < -2, 2 < x < \sqrt{5}$ на промежутке $x = \pm\sqrt{\frac{b \pm \sqrt{b^2 - 4}}{2}}$ - устойчивые

In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt(b - 1)
```

```
def g(b):  # TODO lambda
  return -np.sqrt(b - 1)

def h(b):
  return b*0
def t1(b):
  return np.sqrt(b+1)
def t2(b):
  return -np.sqrt(b+1)
def t3(b):
  return np.sqrt((b-np.sqrt(b*b-4))/2)
def t4(b):
  return -np.sqrt((b-np.sqrt(b*b-4))/2)
def t5(b):
  return -np.sqrt((b+np.sqrt(b*b-4))/2)

def t6(b):
  return np.sqrt((b+np.sqrt(b*b-4))/2)

ax = new_plot()
draw_line(ax, f, -2, -1, style='b--',label='Неустойчивая')
draw_line(ax, f, 1, 2, style='b',label='Устойчивая')
draw_line(ax, f, 2, 4, style='b--')

draw_line(ax, g, -2, 1, style='m--')
draw_line(ax, g, 1, 2, style='m')
draw_line(ax, g, 1, 4, style='m--')

draw_line(ax, h, -2, -1, style='g--')
draw_line(ax, h, -1, 1, style='g')
draw_line(ax, h, 1, 4, style='g--')

draw_line(ax,t1,-2,-1,style='r')
draw_line(ax,t1,-1,4,style='r--')
draw_line(ax,t2,-2,-1,style='r')
draw_line(ax,t2,-1,4,style='r--')

draw_line(ax,t3,2,np.sqrt(5),style='y')
draw_line(ax,t3,np.sqrt(5),4,style='y--')

draw_line(ax,t4,2,np.sqrt(5),style='y')
draw_line(ax,t4,np.sqrt(5),4,style='y--')

draw_line(ax,t5,2,np.sqrt(5),style='y')
draw_line(ax,t5,np.sqrt(5),4,style='y--')

draw_line(ax,t6,2,np.sqrt(5),style='y')
draw_line(ax,t6,np.sqrt(5),4,style='y--')

draw_asym(-1,label='-1')
draw_asym(1,label='1')
draw_asym(2,label='2')
draw_asym(np.sqrt(5),label='sqrt 5')
ax.set(xlabel='b', ylabel='x')
show_plot(-4, 4)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in sqrt

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: invalid value encountered in sqrt
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: Ru
ntimeWarning: invalid value encountered in sqrt
  if __name__ == '__main__':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:11: R
untimeWarning: invalid value encountered in sqrt
  # This is added back by InteractiveShellApp.init_path()
```



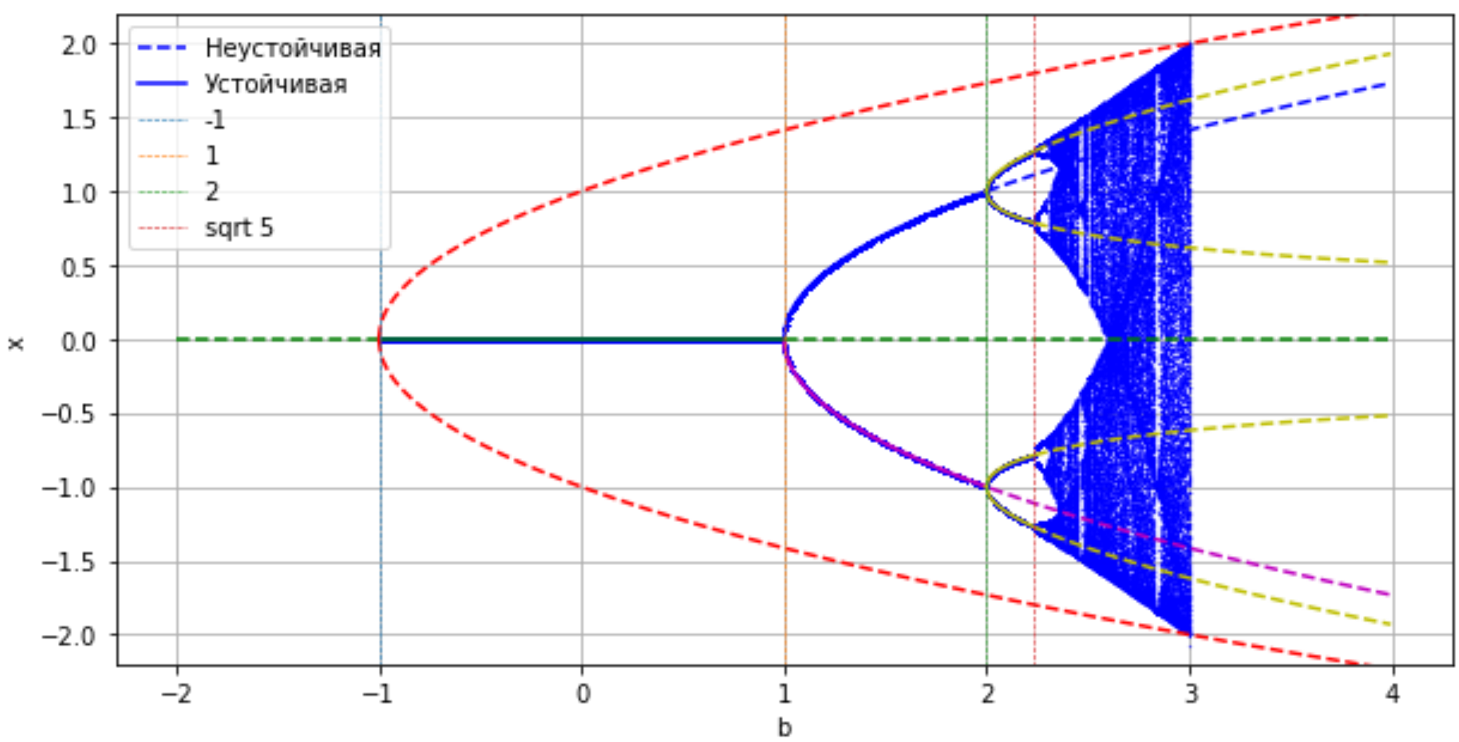
Дерево бифуркаций

In [0]:

```
def u(x, b, t=1):
    if t > 1:
        x = f(x, b, t-1)
    return (+b*x - x*x*x)

def f(b):  # TODO lambda
  return np.sqrt(b - 1)
def g(b):  # TODO lambda
  return -np.sqrt(b - 1)

def h(b):
  return b*0
def t1(b):
  return np.sqrt(b+1)
def t2(b):
  return -np.sqrt(b+1)
def t3(b):
  return np.sqrt((b-np.sqrt(b*b-4))/2)
def t4(b):
  return -np.sqrt((b-np.sqrt(b*b-4))/2)
def t5(b):
```

```
def t5(b):
  return -np.sqrt((b+np.sqrt(b*b-4))/2)
def t6(b):
  return np.sqrt((b+np.sqrt(b*b-4))/2)


ax = new_plot()


draw_tree(ax, u, -3, -1, style='.b', start=2./3)
draw_tree(ax, u, -1, 1, style='.b', start=0)
draw_tree(ax, u, 1, 5, style='.b', start=2./3, inf_n=200, step=0.003)
draw_line(ax, f, -2, -1, style='b--',label='Неустойчивая')
draw_line(ax, f, 1, 2, style='b',label='Устойчивая')
draw_line(ax, f, 2, 4, style='b--')

draw_line(ax, g, -2, 1, style='m--')
draw_line(ax, g, 1, 2, style='m')
draw_line(ax, g, 1, 4, style='m--')

draw_line(ax, h, -2, -1, style='g--')
draw_line(ax, h, -1, 1, style='g')
draw_line(ax, h, 1, 4, style='g--')

draw_line(ax,t1,-2,-1,style='r')
draw_line(ax,t1,-1,4,style='r--')
draw_line(ax,t2,-2,-1,style='r')
draw_line(ax,t2,-1,4,style='r--')

draw_line(ax,t3,2,np.sqrt(5),style='y')
draw_line(ax,t3,np.sqrt(5),4,style='y--')

draw_line(ax,t4,2,np.sqrt(5),style='y')
draw_line(ax,t4,np.sqrt(5),4,style='y--')

draw_line(ax,t5,2,np.sqrt(5),style='y')
draw_line(ax,t5,np.sqrt(5),4,style='y--')

draw_line(ax,t6,2,np.sqrt(5),style='y')
draw_line(ax,t6,np.sqrt(5),4,style='y--')

draw_asym(-1,label='-1')
draw_asym(1,label='1')
draw_asym(2,label='2')
draw_asym(np.sqrt(5),label='sqrt 5')
ax.set(xlabel='b', ylabel='x')







show_plot(-2.2, 2.2)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: Ru
ntimeWarning: overflow encountered in double_scalars
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: Ru
ntimeWarning: invalid value encountered in double_scalars
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: R
untimeWarning: invalid value encountered in sqrt
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: R
untimeWarning: invalid value encountered in sqrt
  if sys.path[0] == '':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:17: R
untimeWarning: invalid value encountered in sqrt
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:19: R
untimeWarning: invalid value encountered in sqrt
```



Карта режимов

In [0]:

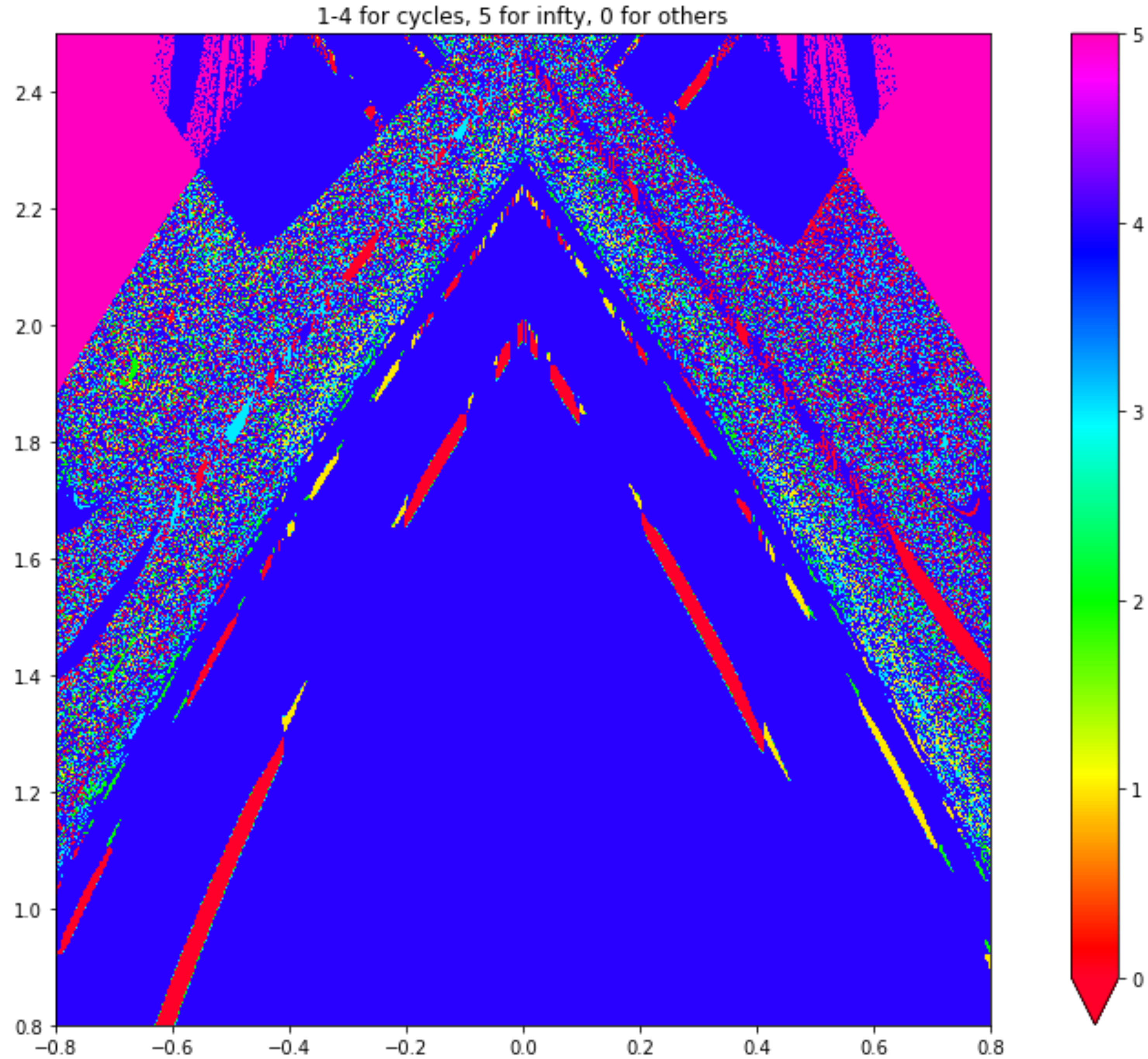
```

```

In [0]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

amin, amax, bmin, bmax = -0.8, 0.8, 0.8, 2.5

apoints, bpoints = 800, 850 # число точек по горизонтали и вертикали
max_iterations = 300
infinity_border = 10000

image = np.zeros((apoints, bpoints))
# image — это двумерный массив, в котором будет записана картинка, по умолчани
```

```
ю он заполнен нулями

eps = 10e-10

a_delta = (amax - amin) / apoints
a_list = [amin + a_delta * i for i in range(apoints)]
b_delta = (bmax - bmin) / bpoints
b_list = [bmin + b_delta * i for i in range(bpoints)]

def f(x, a, b, n):
    if n == 1:
        return a + b*x - np.power(x, 3)
    else:
        x = f(x, a, b, n-1)
        return a + b*x - np.power(x, 3)

for ai, a in enumerate(a_list):
    for bi, b in enumerate(b_list):
        x_n = 0
        for i in range(max_iterations):
            x_n = f(x_n, a, b, 1)
            if np.abs(x_n) > infinity_border:
                x_n = infinity_border
                break
        # fixed point
        if f(x_n, a, b, 1) - x_n < eps:
            image[ai][len(image[ai])-1-bi] = 1
        # 2-cycle
        if f(x_n, a, b, 2) - x_n < eps:
            image[ai][len(image[ai])-1-bi] = 2
        # 4-cycle
        if f(x_n, a, b, 4) - x_n < eps:
            image[ai][len(image[ai])-1-bi] = 3
        # 8-cycle
        if f(x_n, a, b, 8) - x_n < eps:
            image[ai][len(image[ai])-1-bi] = 4
        # infinity
        if x_n == infinity_border:
            image[ai][len(image[ai])-1-bi] = 5
        # 0 for other configurations

plt.figure(figsize=(16, 10))
tmp = plt.imshow(image.T, cmap='gist_rainbow', extent=(amin, amax, bmin, bmax)
)
plt.colorbar(tmp, extend='min')
plt.title('1-4 for cycles, 5 for infty, 0 for others')
plt.show()
```

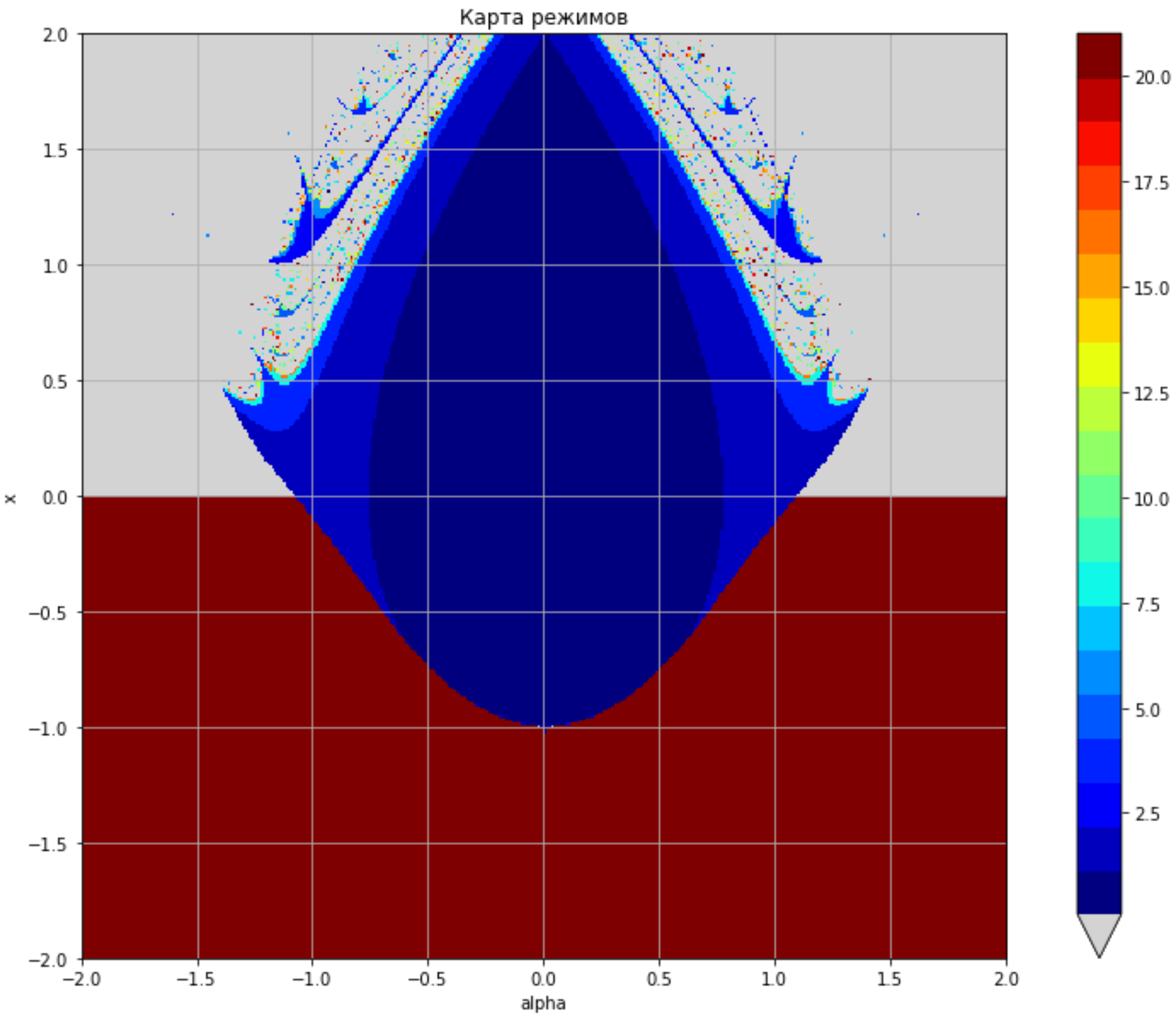
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:26: R
untimeWarning: overflow encountered in power
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:26: R
untimeWarning: invalid value encountered in double_scalars
```



In [0]:

```
def f(x, a, b):
  return a + b*x - x*x*x


ax = new_plot(title='Карта режимов')

draw_mode_map(ax, f, -2, 2, -2, 2)

plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: overflow encountered in double_scalars

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in double_scalars
```



In [0]:

```
def f(b):  # TODO lambda
  return np.sqrt((b - 1)/3)*(2 - 2*b)/3
def g(b):  # TODO lambda
  return -np.sqrt((b - 1)/3)*(2 - 2*b)/3

def h(b):  # TODO lambda
  return np.sqrt((b +1)/3)*(4 - 2*b)/3
def t(b):  # TODO lambda
  return -np.sqrt((b +1)/3)*(4 - 2*b)/3

def Γ(x, a, b):
  return a + b*x - x*x*x


ax = new_plot(title='Карта режимов')

draw_mode_map(ax, Γ, -2, 2, -2, 2)



draw_line_revert(ax, f, -5, 6, style='m', label='u=1')
draw_line_revert(ax, g,-5, 6, style='m')
draw_line_revert(ax, h, -5, 6, style='b', label='u=-1')
draw_line_revert(ax, t,-5, 6, style='b')
draw_asym_revert(-1, label='-1')
draw_asym_revert(1, label='1')
ax.set(xlabel='a', ylabel='b')
show_plot(-2, 6)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: R
untimeWarning: overflow encountered in double_scalars
  if sys.path[0] == '':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: R
untimeWarning: invalid value encountered in double_scalars
  if sys.path[0] == '':
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: Ru
ntimeWarning: invalid value encountered in sqrt

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: invalid value encountered in sqrt
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: Ru
ntimeWarning: invalid value encountered in sqrt
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9: Ru
ntimeWarning: invalid value encountered in sqrt
  if __name__ == '__main__':
```