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='', x_label='b', y_label='M'):
    fig, ax = plt.subplots(figsize=(15,10))

    ax.set(xlabel=x_label, ylabel=y_label,
           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=0.005, iter=1000, inf=200, inf_
n=100, eps=10e-3, start=0):

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

    a_ = []
    x_ = []


    for i in a:

        x_tmp = start

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

        if not (-inf < x_tmp[0] < inf and -inf < x_tmp[1] < inf):
            continue
```

```
        tmp = []

        for n in range(1, inf_n):
            tmp.append(x_tmp[0])
            x_tmp = f(x_tmp, i)

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

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


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()
```

# Лабораторная #3

Для отображения Эно
$x_{n+1} = y_n$
$y_{n+1} = M - bx - y_n^2$
def Небходимо

- Построить биффуркационные кривые ($\mu = 1, \mu = -1, |\tau| = 1$):
- построить карту режимов (в осях $M, b$)
- построить дерефо биффуркации при фиксированном $b = 0.3$
- Построить зависимость $\mu_1, \mu_2$ от параметра $M(b = 3)$

1. Для неподвижной точки
2. Для неподвижной точки периода два
   Искать методом Ньютона

# 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 $f^N(x) = f(f...(f(x_n)))$ - $N$-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.

## Отображение Эно

Отображение было предложено в 1976 году как абстрактный пример динамической системы французским математиком и астрономом Мишелем Эно. Сейчас известно, что оно может служить для описания ряда простых физических систем, таких как частица в вязкой среде под действием импульсных толчков.
Это отображение можно интерпретировать как логистическое отображение с дополнительным запаздыванием на один шаг дискретного времени.

## Параметры: след $S$ и якобиан $J$ матрицы Якоби.

Перейдем к параметрам след $S$ и якобиан $J$ матрицы Якоби. Для этого след вычисляем как сумму диагональных элементов матрицы, а якобиан – по известному правилу вычисления определителя. Тогда .$S = -2x_0\lambda, J = b$. Используя выражение для неподвижной точки , получаем связь следа и якобиана с исходными параметрами отображения.
Таким образом, отображение Эно характеризуется постоянным якобианом. Это означает, что оно будет демонстрировать не все описанные выше бифуркации, поскольку условие диссипативности отображения $|b| < 1$ в этом случае означает $|J| < 1$ и бифуркация Неймарка-Сакера невозможна. В отображении Эно будут наблюдаться только касательная бифуркация и бифуркация удвоения периода. То есть отображение Эно лишь обобщает уже известные для одномерных отображений бифуркации на случай двух измерений. Тем не менее, его бифуркационный анализ представляет интерес, как пример приложения представленного выше подхода.
Начнем анализ с касательной бифуркации, условие которой задается соотношением

## Карта динамических режимов

Как и в случае одномерных отображений, двумерные отображения допускают использование метода бифуркационных диаграмм. Однако, поскольку число параметров, как правило, больше единицы, то вместо бифуркационных диаграмм чаще более наглядными выглядят карты динамических режимов.

Этот метод представляет собой достаточно удобный способ определения типов колебательных режимов в различных точках плоскости параметров. В рамках такого представления разными цветами на плоскости параметров показывают области существования устойчивых режимов разных периодов, а также области непериодических режимов, в данном случае хаоса C (chaos) и «убегания» траекторий на бесконечность D (divergency). Область разбегания траекторий показана

серым цветом, а хаоса – белым. На карте можно видеть линии рождения 2- и 4-цикла, подтверждающие аналитическое рассмотрение в рамках бифуркационного анализа, а также области долгопериодических циклов. Внутри области хаоса обнаруживаются структуры типа «crossroad area», которые подробно рассмотрены при анализе одномерных отображений. Две такие структуры, образованные на базе

## Неподвижные точки

$$\begin{cases} x = y \\ y = M - bx - y^2 \end{cases}$$

$$y = M - by - y^2$$

$$y^2 + (b+1)y - M = 0$$

Неподвижные точки отображения Эно определяются соотношением:

$$y_{1,2} = \frac{-(1+b) \pm \sqrt{b^2+2b+1+4M}}{2}$$

$$b^2 + 2b + 1 + 4M > 0$$

$$M > \frac{-(b+1)^2}{4} \Rightarrow \exists$$ две неподвижные точки

Запишем теперь матрицу Якоби. Для этого надо найти соответствующие частные производные, так что в результате получаем:

$$J = \begin{bmatrix} 0 & 1 \\ -b & -2y \end{bmatrix}$$

$$|J| = b$$

$$S = -2y$$

## Характеристическое уравнение

$$X(\mu) = \begin{bmatrix} -\mu & 1 \\ -b & = & -2y^* - \mu \end{bmatrix} = \mu^2 + 2y^*\mu + b$$

**1) Седлоузловая биффуркация:** $\mu = 1$

$1 + 2y^* + b = 0$

$y^* = \frac{-b-1}{2}$ Подставим в $y^2 + (b+1)y - M = 0$

$\frac{(b+1)^2}{4} - \frac{(b+1)^2}{2} - M = 0$

$M = \frac{-(b+1)^2}{4}$ - кривая седлоузловой биффуркации

**2) Биффуркация удвоения периода:** $\mu = -1$

$1 + 2y^* + b = 0$

$y^* = \frac{b+1}{2}$ Подставим в $y^2 + (b+1)y - M = 0$

$\frac{(b+1)^2}{4} + \frac{(b+1)^2}{2} - M = 0$

$M = \frac{3(b+1)^2}{4}$ - кривая удвоения периода

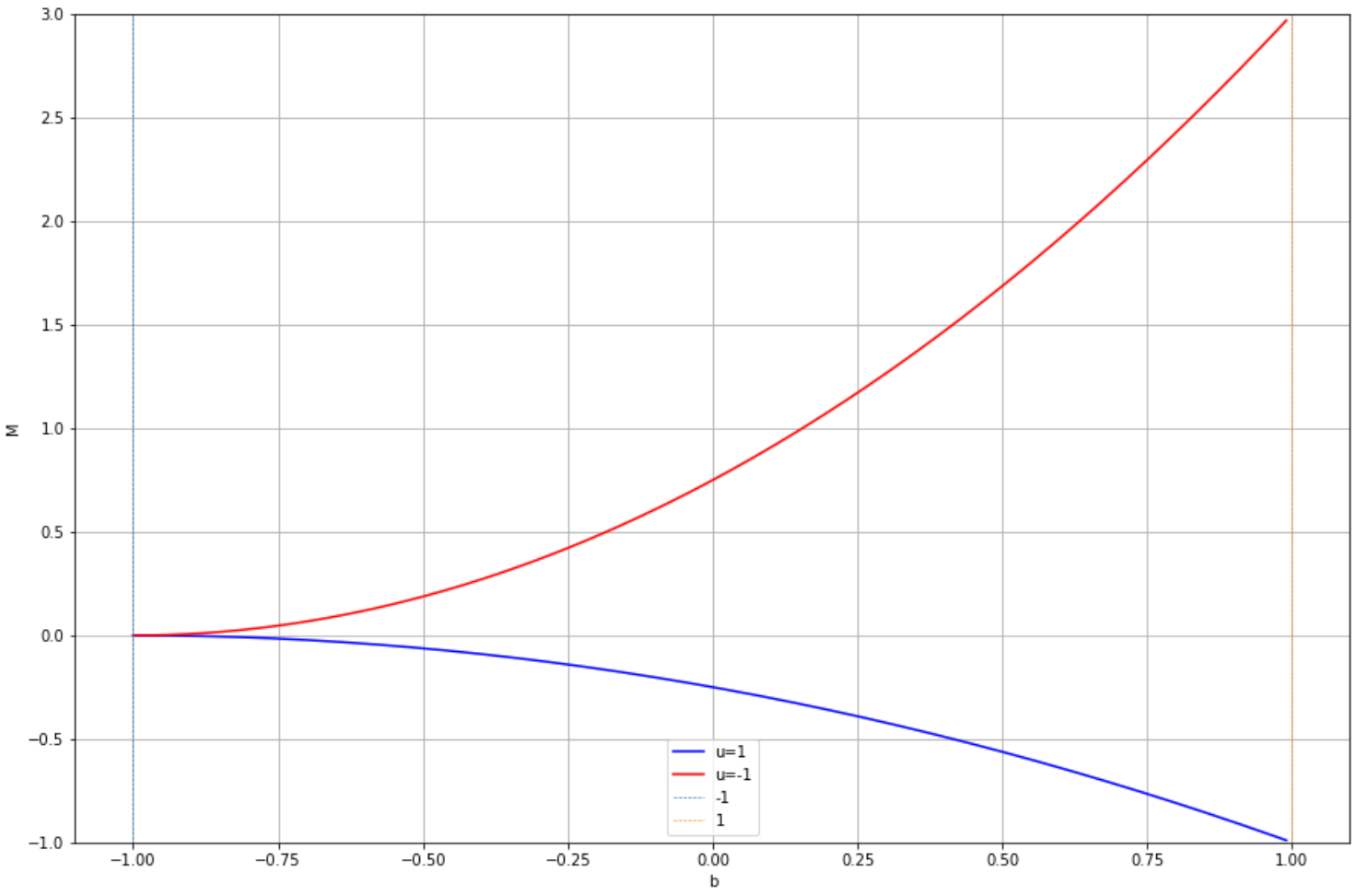
**3) Биффуркация Неймарка-Сакера**

$\mu \cdot \mu = 1 = |J| = 1$

$b = 1$

In [0]:

```
def f(b):  # TODO lambda
  return -np.square(b+1)/4
def g(b):  # TODO lambda
  return 3*np.square(b+1)/4


ax = new_plot()

draw_line(ax, f, -1, 1, style='b', label='u=1')
draw_line(ax, g, -1, 1, style='r', label='u=-1')

draw_asym(-1, label='-1')
draw_asym(1, label='1')

show_plot(-1, 3)
```




# Два цикл

$$\begin{cases} x_1 = y_2 \\ y_1 = M - bx_2 - y_2^2 \\ x_2 = y_1 \\ y_2 = M - bx_1 - y_1^2 \end{cases}$$

$$\begin{cases} y_1 = M - bx_1 - x_1^2 \\ x_1 = M - bx_1 - y_1^2 \end{cases}$$

$$x_1 = M - bx_1 - (M - bx_1 - x_1^2)^2$$

$$x_1 = M - bx_1 - M^2 - b^2x_1^2 - x_1^4 + 2Mbx_1 + 2Mx_1^2 - 2bx_1^3$$

$$x_1^4 + 2bx_1^3 + (b^2 - 2M)x_1^2 + (1 + b - 2Mb)x_1 + (M^2 - M) = 0$$

$x_{1,2} = \frac{-b-1\pm\sqrt{4M+b^2+2b+1}}{2}$ 1 цикл

$x_{3,4} = \frac{-b+1\pm\sqrt{4M+b^2+2b+1}}{2}$ 2 цикл

**Бифуркация два цикла**

Вычислим матрицу Якоби:

$$\begin{pmatrix} 0 & 1 \\ -b & -2y_4 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -b & -2y_3 \end{pmatrix} = \begin{pmatrix} -b & -2y_3 \\ 2y_4 b & -b + 4y_3 y_4 \end{pmatrix}$$

$$S = -2b + 4y_3 y_4$$

$$|J| = b^2$$

$$y_3 y_4 = -b - M => S = -6b - 4M$$

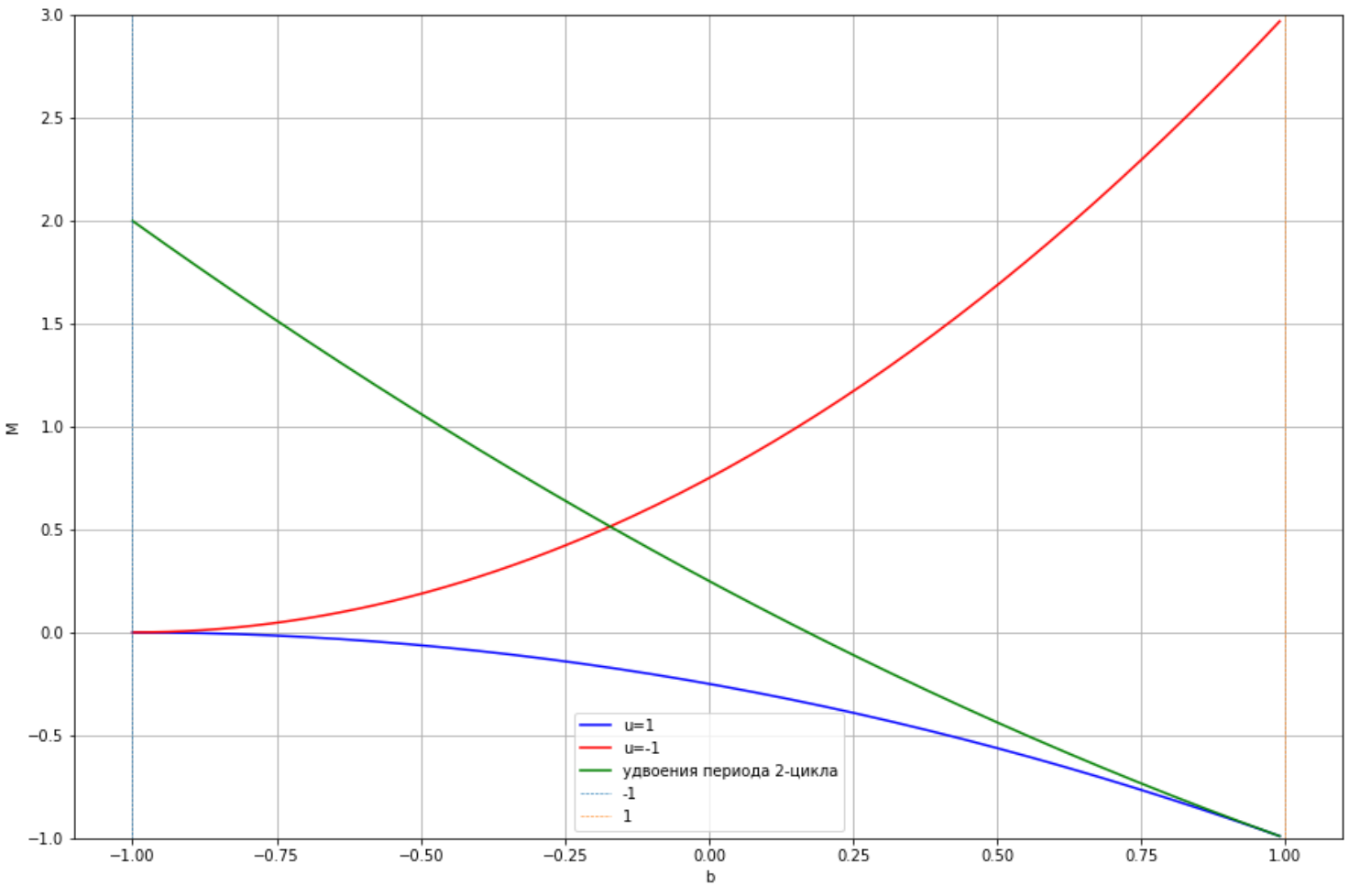
Очевидно, что касательной бифуркации 2-цикла отвечает линия удвоения неподвижной точки. Поэтому нам осталось найти линии удвоения периода 2-цикла.

Используя соотношение $1 + S + J = 0$, после некоторых преобразований получаем условие удвоения периода для 2-цикла

$$M = \frac{1-6b+b^2}{4}$$

In [0]:

```
def f(b):  # TODO lambda
  return -np.square(b+1)/4
def g(b):  # TODO lambda
  return 3*np.square(b+1)/4
def h(b):
  return (1-6*b+b*b)/4

ax = new_plot()

draw_line(ax, f, -1, 1, style='b', label='u=1')
draw_line(ax, g, -1, 1, style='r', label='u=-1')
draw_line(ax, h, -1, 1, style='g',label='удвоения периода 2-цикла')

draw_asym(-1, label='-1')
draw_asym(1, label='1')

show_plot(-1, 3)
```




# Карта режимов

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=0.5):

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

          if x > inf or x < -inf or y > inf or y < -inf:
              if (x > inf):
                  z[-i-1][j] = 5
              else:
                  z[-i-1][j] = 10
              z[-i-1][j] = inf_n+1
              continue

          x_tmp, y_tmp = x, y
          k = 0 # 1-cicle len
          z[-i-1][j] = 0

          for k in range(inf_n):
              x_tmp, y_tmp = f(x_tmp, y_tmp, b_x, m_y)
              if -eps < x - x_tmp < eps and -eps < y - y_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]:

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

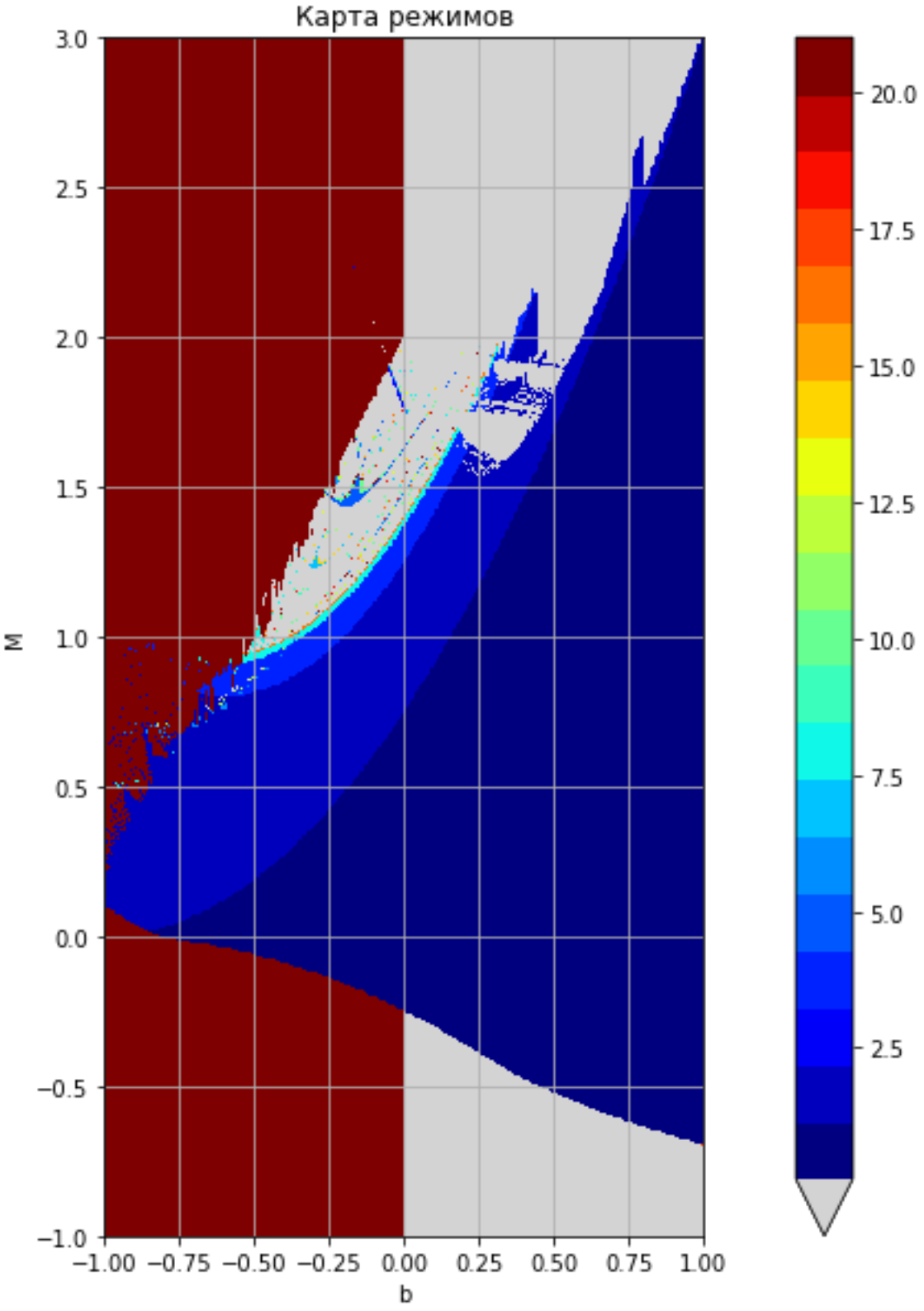
```
from mpl_toolkits.mplot3d import Axes3D

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=0.5):

  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]))

  b_template = np.arange(x_min, x_max, step)
  m_y = 0.1

  for j, b_x in enumerate(b_template):
      x, y = start, start
      for k in range(iter):
          x, y = f(x, y, b_x, m_y)
      b_template[j] = x

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

          x, y = b_template[j],  b_template[j]
          for k in range(iter):
              x, y = f(x, y, b_x, m_y)

          if x > inf or x < -inf or y > inf or y < -inf:
              if (x > inf):
                  z[-i-1][j] = 5
              else:
                  z[-i-1][j] = 10
              z[-i-1][j] = inf_n+1
              continue

          if -inf < x < inf and -inf < y < inf:
              b_template[j] = x
          else:
              b_template[j] = 0

          x_tmp, y_tmp = x, y
          k = 0 # 1-cicle len

          z[-i-1][j] = 0

          for k in range(inf_n):
              x_tmp, y_tmp = f(x_tmp, y_tmp, b_x, m_y)
              if -eps < x - x_tmp < eps and -eps < y - y_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, y, b, m):
  return (y, m - b*x - y*y)


ax = new_plot(title='Карта режимов', x_label='b', y_label='M')

draw_mode_map(ax, f, -1, 1, -1, 3, step=0.005, iter=2000)

plt.show()
```

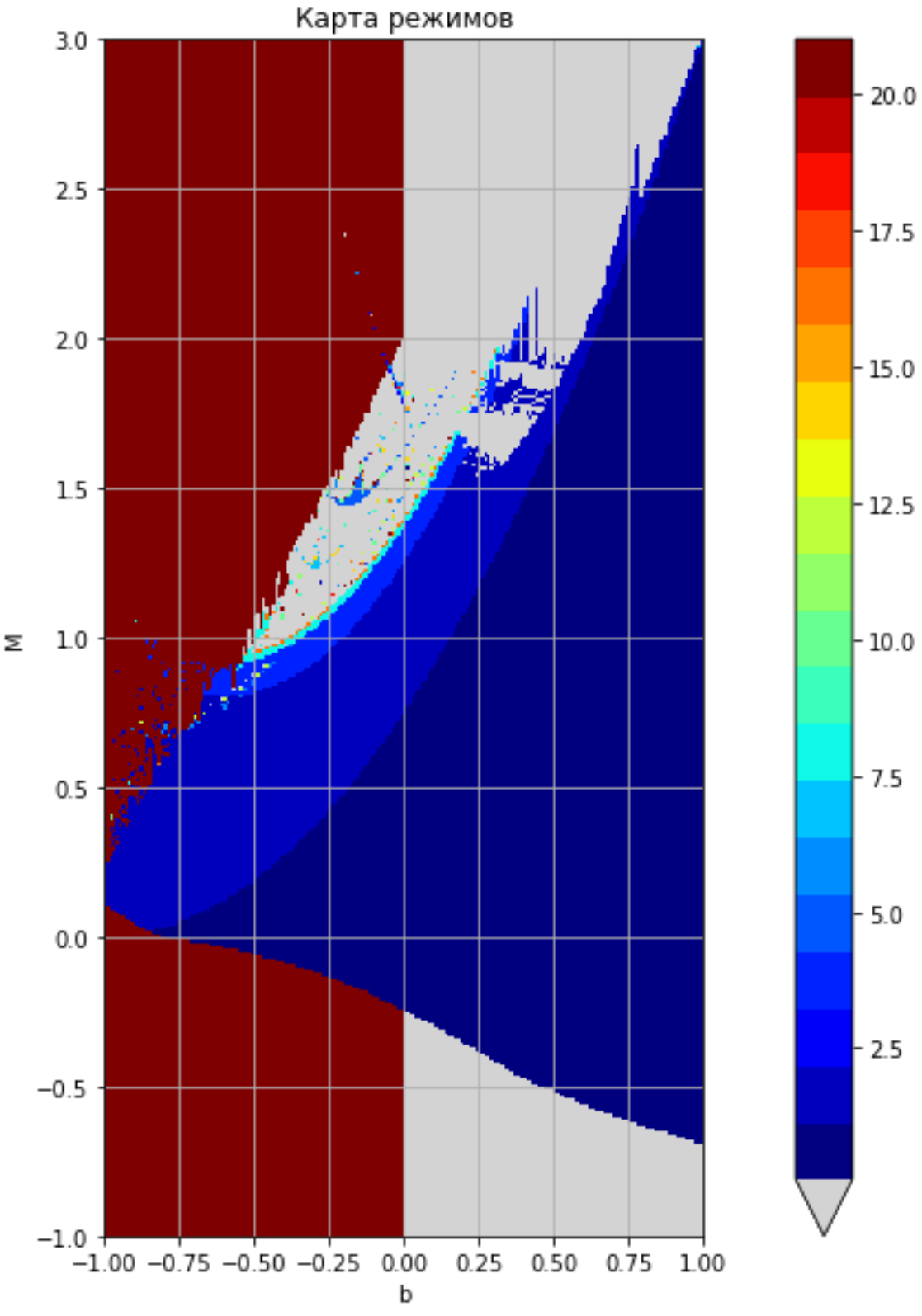
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: Ru
ntimeWarning: overflow encountered in double_scalars
  This is separate from the ipykernel package so we can avoid doin
g imports until
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: Ru
ntimeWarning: invalid value encountered in double_scalars
  This is separate from the ipykernel package so we can avoid doin
g imports until
```




In [0]:

```
def f(x, y, b, m):
  return (y, m - b*x - y*y)


ax = new_plot(title='Карта режимов', x_label='b', y_label='м')

draw_mode_map(ax, f, -1, 1, -1, 3, step=0.01, iter=5000)

plt.show()
```

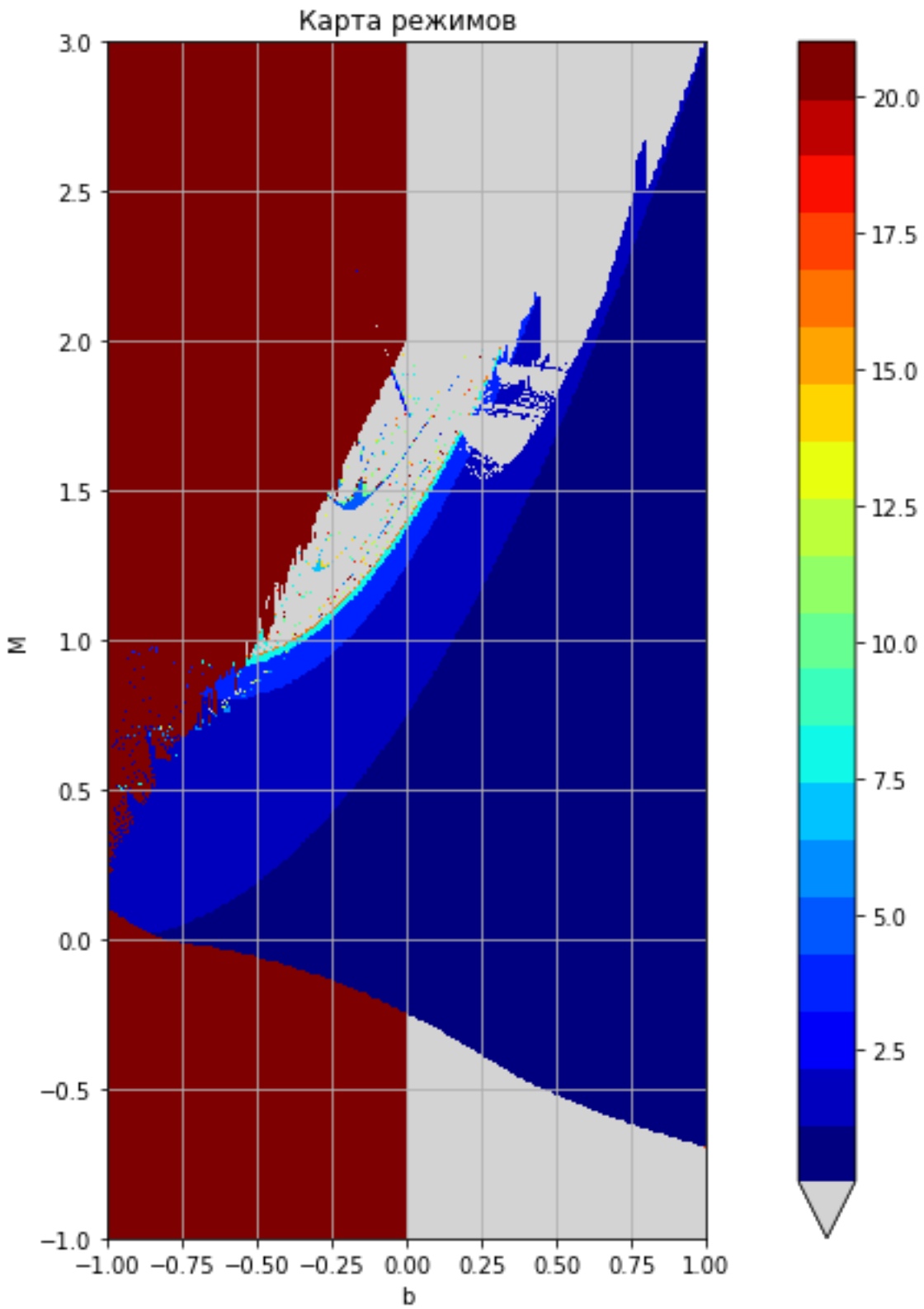
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: Ru
ntimeWarning: overflow encountered in double_scalars
  This is separate from the ipykernel package so we can avoid doin
g imports until
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: Ru
ntimeWarning: invalid value encountered in double_scalars
  This is separate from the ipykernel package so we can avoid doin
g imports until
```



In [5]:

```
def f(x, y, b, m):
  return (y, m - b*x - y*y)


ax = new_plot(title='Карта режимов', x_label='b', y_label='M')

draw_mode_map(ax, f, -1, 1, -1, 3, iter=2000, step=0.005)

plt.show()
```

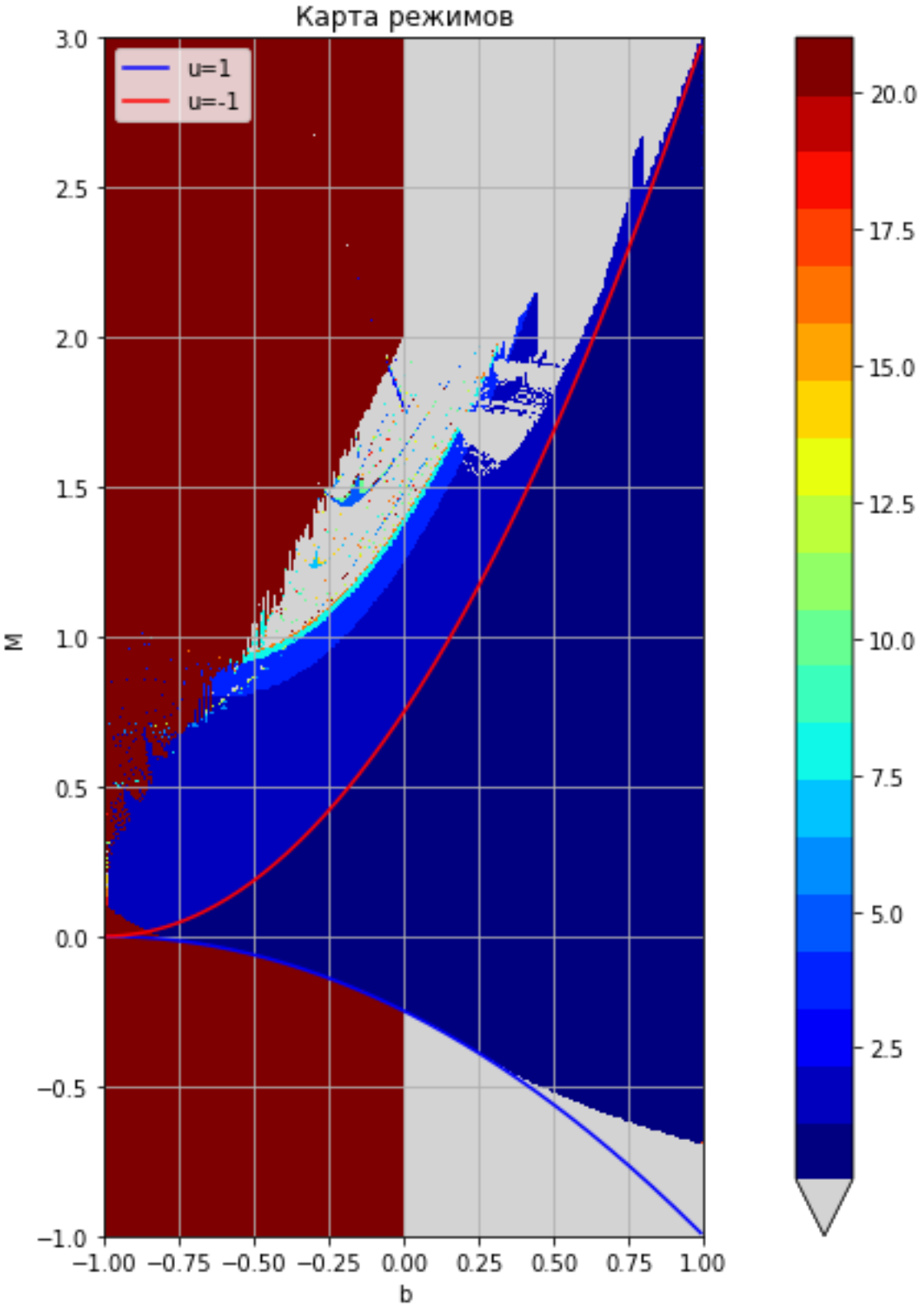
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: Ru
ntimeWarning: overflow encountered in double_scalars
  This is separate from the ipykernel package so we can avoid doin
g imports until
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: Ru
ntimeWarning: invalid value encountered in double_scalars
  This is separate from the ipykernel package so we can avoid doin
g imports until
```



In [8]:

```
def h(b):  # TODO lambda
  return -np.square(b+1)/4
def g(b):  # TODO lambda
  return 3*np.square(b+1)/4




def f(x, y, b, m):
  return (y, m - b*x - y*y)


ax = new_plot(title='Карта режимов', x_label='b', y_label='M')

draw_mode_map(ax, f, -1, 1, -1, 3, step=0.005)

draw_line(ax, h, -1, 1, style='b', label='u=1')
draw_line(ax, g, -1, 1, style='r', label='u=-1')

# draw_asym(-1, label='-1')
# draw_asym(1, label='1')

show_plot(-1, 3)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: R
untimeWarning: overflow encountered in double_scalars
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:10: R
untimeWarning: invalid value encountered in double_scalars
  # Remove the CWD from sys.path while we load stuff.
```




In [0]:

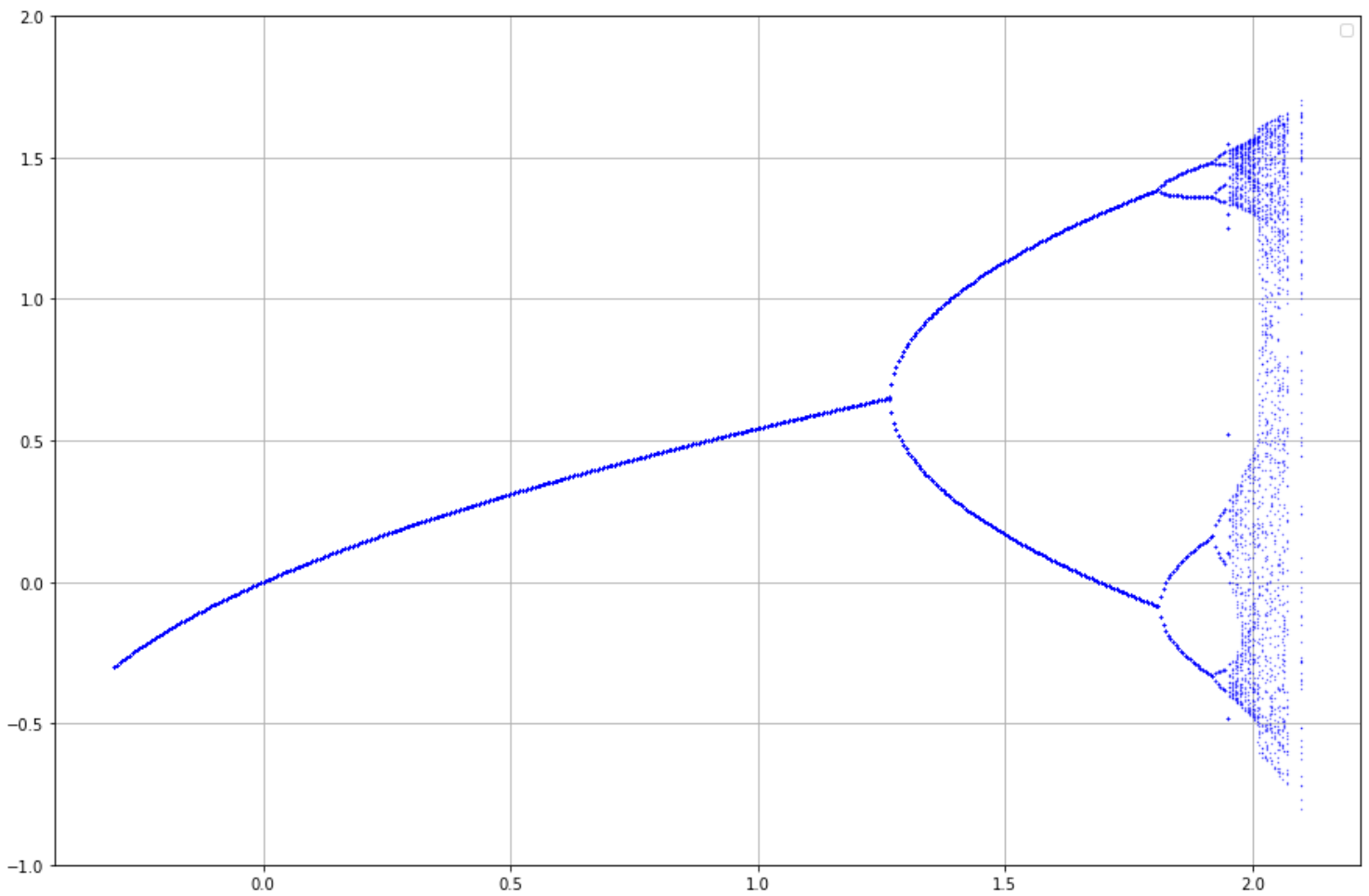
```
```

# Дерево бифуркаций

In [0]:

```
def f(cord, m, b=0.3):
  x = cord[0]
  y = cord[1]
  return (y, m - b*x - y*y)


ax = new_plot()

draw_tree(ax, f, -2, 3, '.b', start=(0.5, 0.5))

show_plot(-1, 2)
```

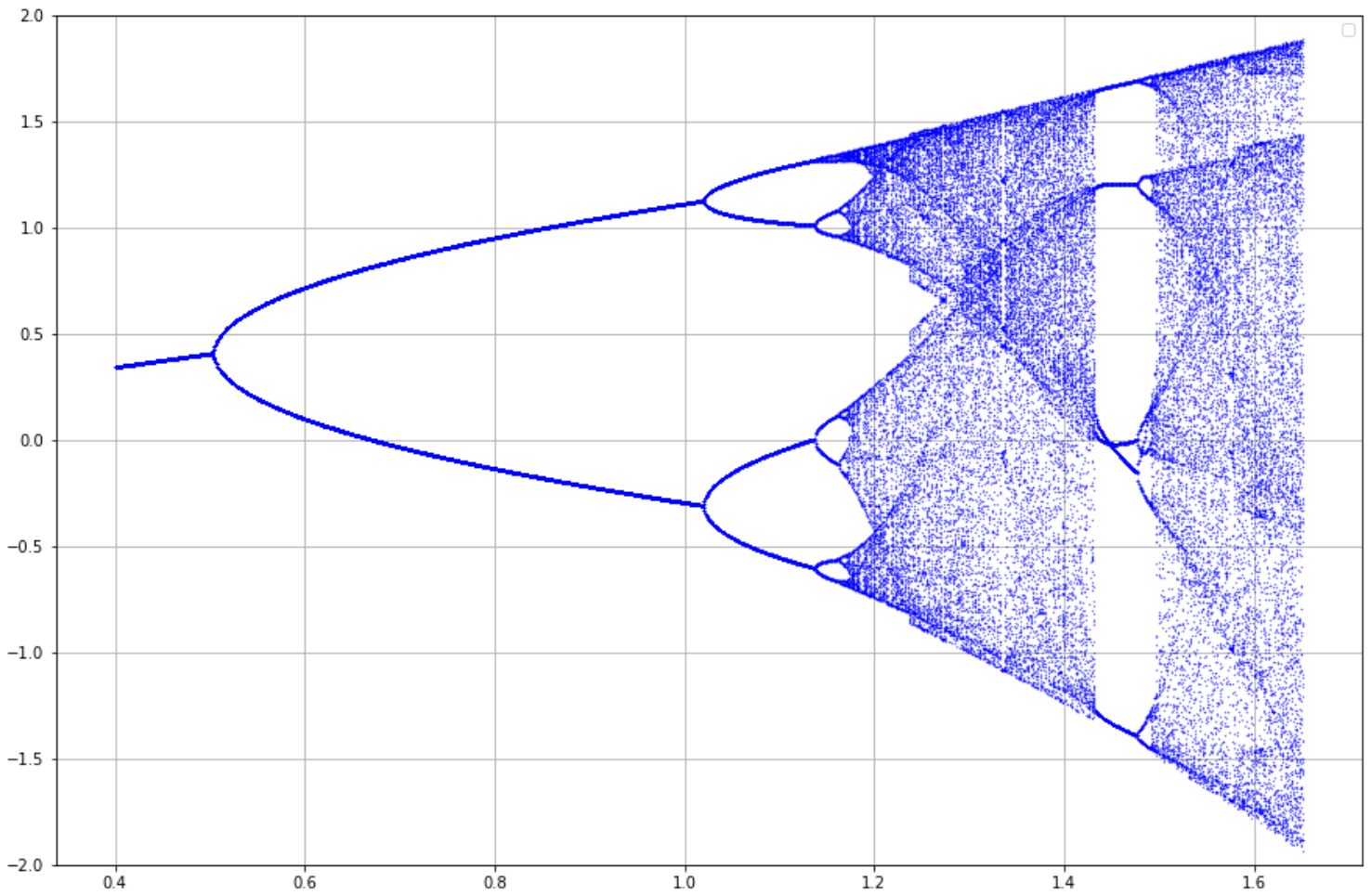
```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: overflow encountered in double_scalars
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: invalid value encountered in double_scalars
  after removing the cwd from sys.path.
No handles with labels found to put in legend.
```



In [0]:

```
def f(cord, m, b=-0.18):
  x = cord[0]
  y = cord[1]
  return (y, m - b*x - y*y)


ax = new_plot()

draw_tree(ax, f, 0.4, 1.8, '.b', step=0.001, start=(0.5, 0.5))

show_plot(-2, 2)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Ru
ntimeWarning: overflow encountered in double_scalars
  after removing the cwd from sys.path.
No handles with labels found to put in legend.
```




In [0]:

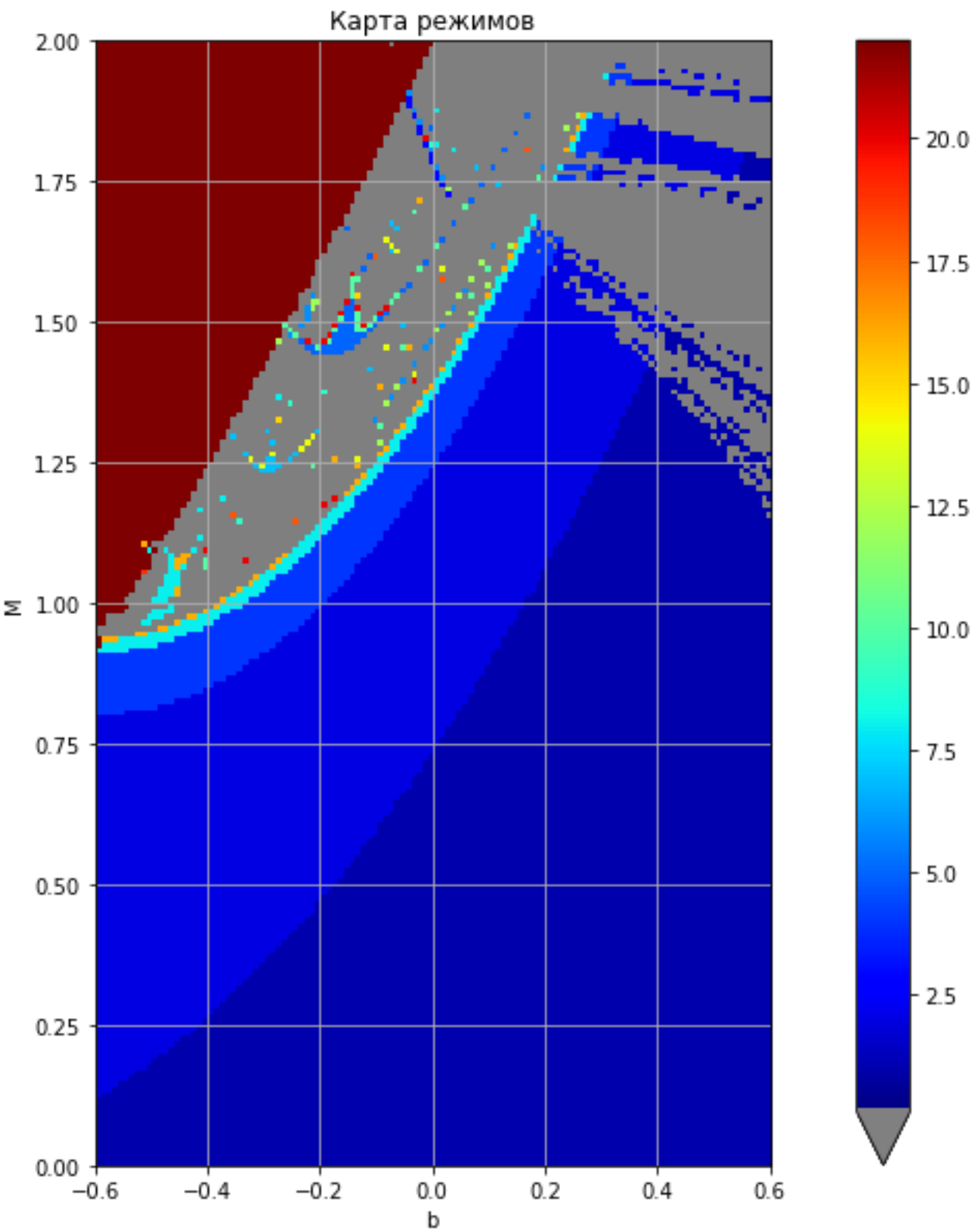
```
def f(x, y, b, m):
  return (y, m - b*x - y*y)



inf = 100
inf_n = 20
eps = 10e-5
step = 0.01
x_min, x_max = -1, 1
y_min, y_max = -1, 3
x_min, x_max = -0.6, 0.6
y_min, y_max = 0, 2
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, y = 0, 0
        for k in range(1000):
            x, y = f(x, y, b_x, m_y)

        if x > inf or x < -inf or y > inf or y < -inf:
            z[-i-1][j] = 22
            continue

        x_tmp, y_tmp = x, y
        k = 0 # 1-cicle len

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

```
from matplotlib.colors import ListedColormap

fig, ax = plt.subplots( figsize=(15, 10) )
ax.set(xlabel='b', ylabel='M',
       title='Карта режимов')
ax.grid()


cmap = plt.cm.jet  # define the colormap
# extract all colors from the .jet map
cmaplist = [cmap(i) for i in range(cmap.N)]
# force the first color entry to be grey
cmaplist[0] = (.5, .5, .5, 1.0)

# create the new map
cmap = mpl.colors.LinearSegmentedColormap.from_list(
    'Custom cmap', cmaplist, cmap.N)

# define the bins and normalize
bounds = np.linspace(0, 20, 21)
norm = mpl.colors.BoundaryNorm(bounds, cmap.N)

# create a second axes for the colorbar
#ax2 = fig.add_axes([0, 10, 1, 15])
cb = mpl.colorbar.ColorbarBase(ax2, cmap=cmap, norm=norm,
    spacing='proportional', ticks=bounds, boundaries=bounds, format='%1i')
```

```
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')
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: Ru
ntimeWarning: overflow encountered in double_scalars
  """
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: Ru
ntimeWarning: invalid value encountered in double_scalars
  """
```




In [0]:

```
x, y = 0, 0
b_x, m_y = 0.5, 1.6
for i in range(100):
  x, y = f(x, y, b_x, m_y)
  print(x, y)
```