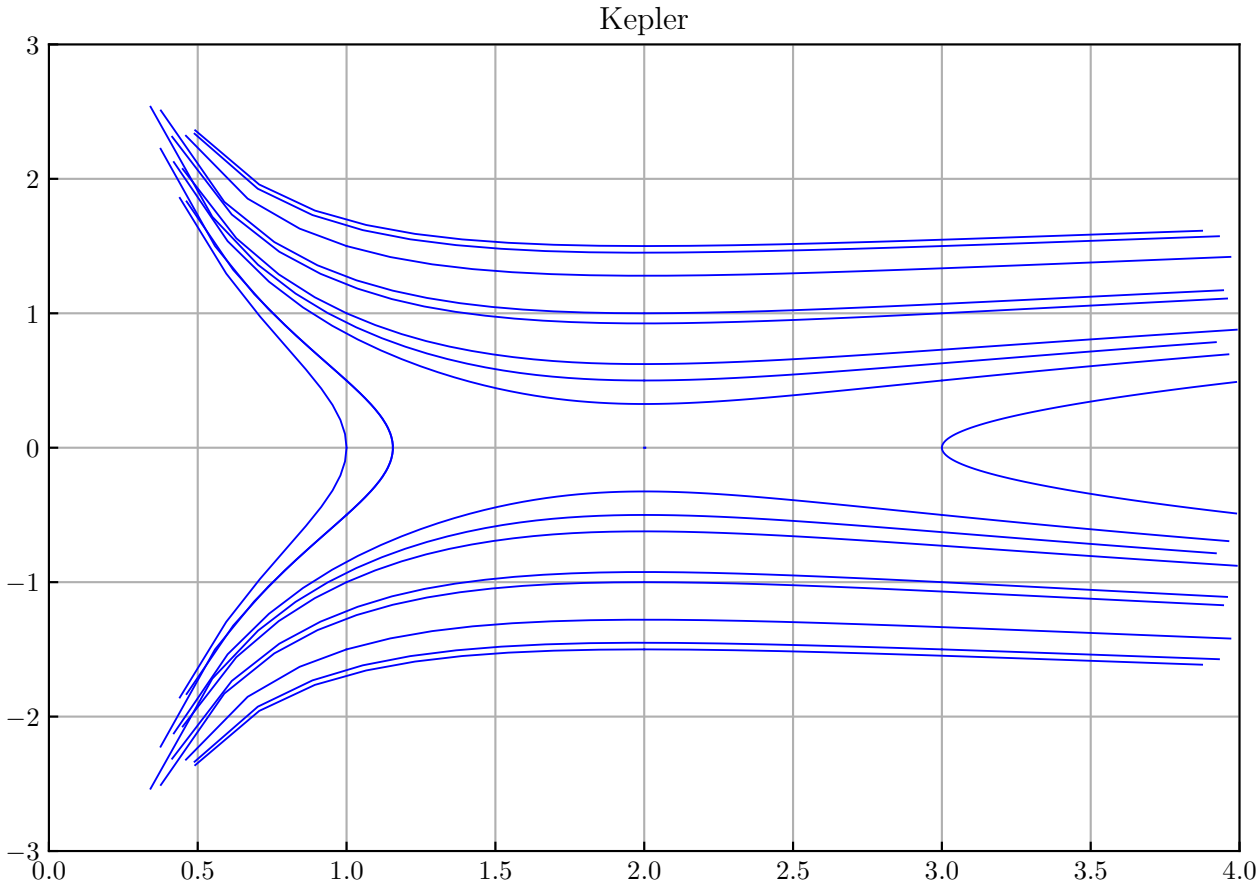
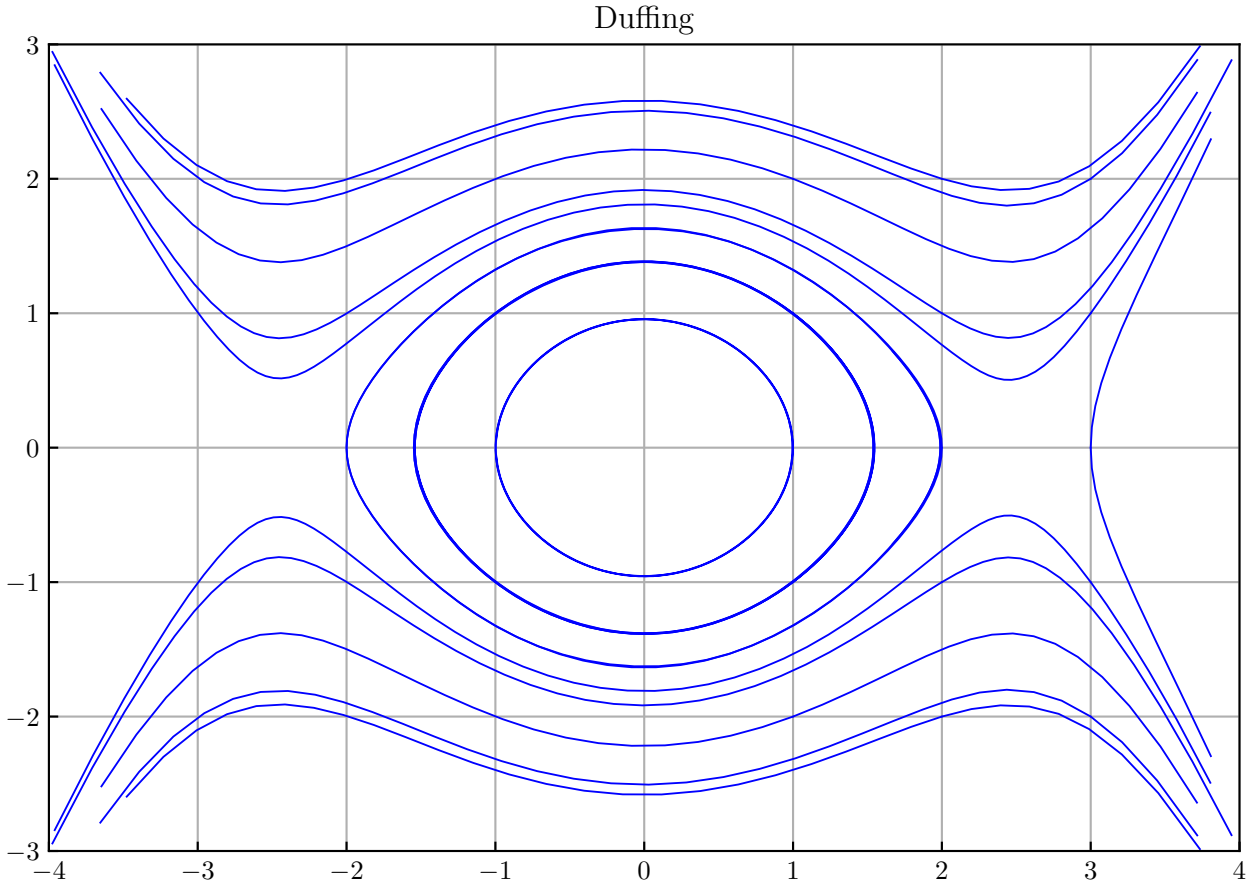

Kepler
3
2
1
0
−1
−2
−3
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


Duffing
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3
4

0.1 /code

/code

```
#coding=utf-8

import matplotlib
matplotlib.use('ps')
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import solve_ivp


def plot_phase_trajectories(f, inits, xbound, ybound, t=(0, 10), steps=100,
                            axis=None,
                            sivp_kwargs={}, plt_kwargs={'c': 'b', 'lw': .7}):
    """Plots phase trajectory of given ODE with scipy.integrate.solve_ivp
    Returns list of matplotlib-artist objects"""
    if axis is None:
        axis = plt.gca()

    def f_neg(t, x):
        # for solution backwards in time
        return -f(t, x)

    artists = []
    tt = np.linspace(*t, steps)
    for ff in (f, f_neg):
        for i in inits:
            # solve_ivp(..., dense_output=True).sol holds a OdeSolution object
            # which interpolates the solution and allows its evaluation at
            # arbitrary points
            # Returns array with shape(n,) corresponding to the RHS of the
            # given ODE
            sol = solve_ivp(ff, t, i, dense_output=1, **sivp_kwargs).sol
            sol_eval = sol(tt)
            sol_x_ma = np.ma.masked_outside(sol_eval[0], *xbound)
            sol_y_ma = np.ma.masked_outside(sol_eval[1], *ybound)
            artists.append(axis.plot(sol_x_ma, sol_y_ma, **plt_kwargs))

    axis.set_xlim(xbound)
    axis.set_ylim(ybound)
    return artists


ppt = plot_phase_trajectories

```

```
44
45  inits = lambda a, e, s: np.array([(i, j) for i in np.arange(1, 4)
46                                    for j in np.arange(a, e, s)])
47
48
49  def kepler(t, x, c=2):
50      u, v = x
51      du = v
52      dv = 1 / u - c / u**2
53      return np.array([du, dv])
54
55
56  def duffing(t, x):
57      u, v = x
58      du = v
59      dv = (u**3)/6 - u
60      return np.array([du, dv])
61
62
63  def make_plot(name='phase_plot'):
64      f, axarr = plt.subplots(2, figsize=(8, 12))
65      for f, i, b, a, t in zip((kepler, duffing),              # function
66                               ((-1.5, 2, .5), (-2, 3, 1)),    # initial values
67                                                               #  (start, stop, step)
68                               (((0, 4), (-3, 3)),             # boundaries
69                                ((-4, 4), (-3, 3))),
70                               axarr,                          # axis
71                               ('Kepler', 'Duffing')):         # title
72          ppt(f, inits(*i), b[0], b[1], axis=a)
73          a.set_title(t)
74          a.grid(True)
75      plt.savefig(name+'.ps', dpi=100, papertype='a4', orientation='portrait')
76
77
78  # ~£ python3 -c "import code_15012018 as code; code.make_plot('name')"
```