test_Fe_vac

March 30, 2018

In [35]:
```
import logging
import numpy as np

import matplotlib.pyplot as plt

from ase.io import read
from pathlib import Path

from m_ff.interpolation import Spline1D, Spline3D
from m_ff.calculators import TwoBodySingleSpecies, ThreeBodySingleSpecies

logging.basicConfig(level=logging.INFO)
```

In [36]:
```
directory = Path('test/data/Fe_vac')

print('========== Load trajectory ==========')

filename = directory / 'movie.xyz'
traj = read(str(filename), index=slice(0, 5))
```

```
========== Load trajectory ==========
```

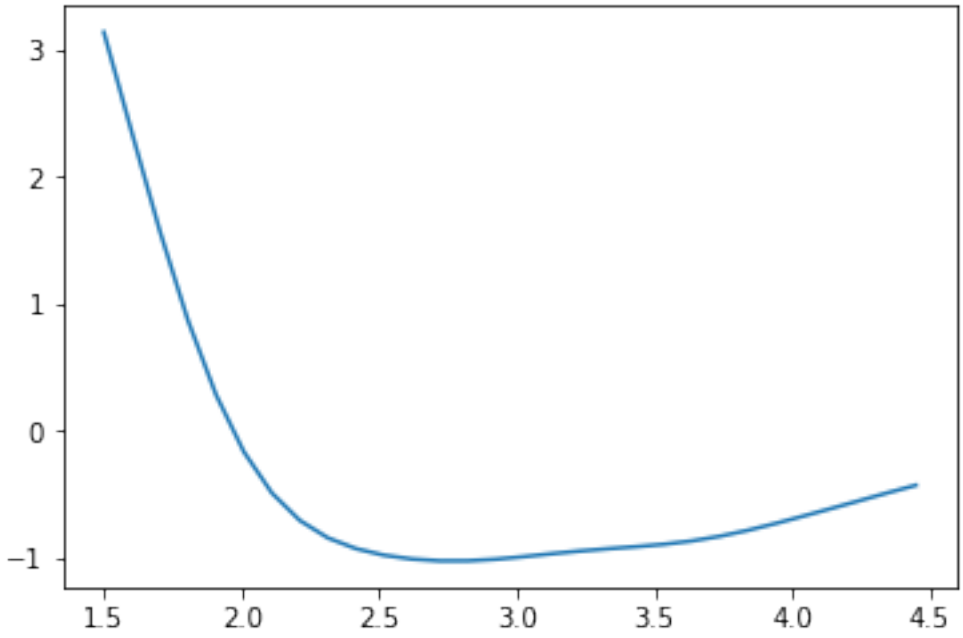
In [40]:
```
print('========== TwoBodySingleSpecies ==========')

# future: TwoBodySingleSpecies.from_json(directory / 'test.json')
rs, element1, _, grid_data_1_1, _ = np.load(str(directory / 'MFF_2b_ntr_10_sig_1.00_cu
grid_1_1 = Spline1D(rs, grid_data_1_1)

calc = TwoBodySingleSpecies(r_cut=3.7, grid_1_1=grid_1_1)

atoms = traj[0]
atoms.set_calculator(calc)

rms = np.sqrt(np.sum(np.square(atoms.arrays['force'] - atoms.get_forces()), axis=1))
print('MAEF on forces: {:.4f} +- {:.4f}'.format(np.mean(rms), np.std(rms)))
print(atoms.get_potential_energy())
```

```
        print('========== Calculate MAEF for each steps ==========')

        for atoms in traj:
            atoms.set_calculator(calc)

            rms = np.sqrt(np.sum(np.square(atoms.arrays['force'] - atoms.get_forces()), axis=
            print('MAEF on forces: {:.4f} +- {:.4f}'.format(np.mean(rms), np.std(rms)))
            print('energy: {}'.format(atoms.get_potential_energy()))
```

```
INFO:m_ff.calculators:numbers is in system_changes
INFO:m_ff.calculators:initialize


========== TwoBodySingleSpecies ==========
MAEF on forces: 0.9893 +- 0.4852
860.9655406389996
========== Calculate MAEF for each steps ==========
MAEF on forces: 0.9893 +- 0.4852
energy: 860.9655406389996
MAEF on forces: 0.9742 +- 0.4746
energy: 861.0317531962813
MAEF on forces: 0.9606 +- 0.4629
energy: 861.0882201884568
MAEF on forces: 0.9491 +- 0.4495
energy: 861.125175615536
MAEF on forces: 0.9418 +- 0.4332
energy: 861.1524326380268
```

```
In [43]: plt.plot(rs, -grid_data_1_1)
```

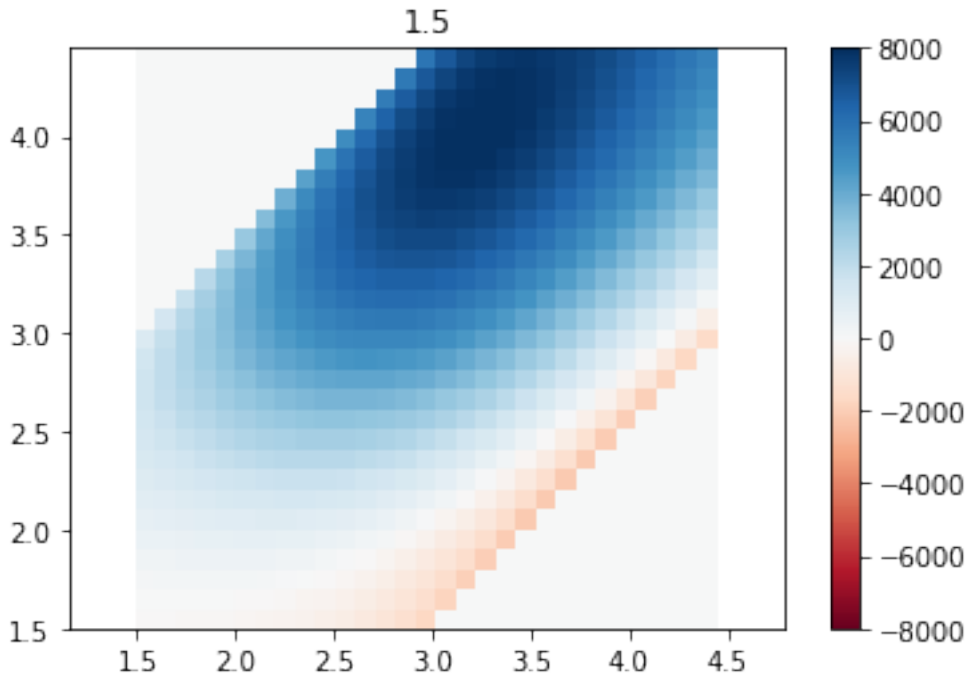
```
Out[43]: [<matplotlib.lines.Line2D at 0x18151b3a58>]
```

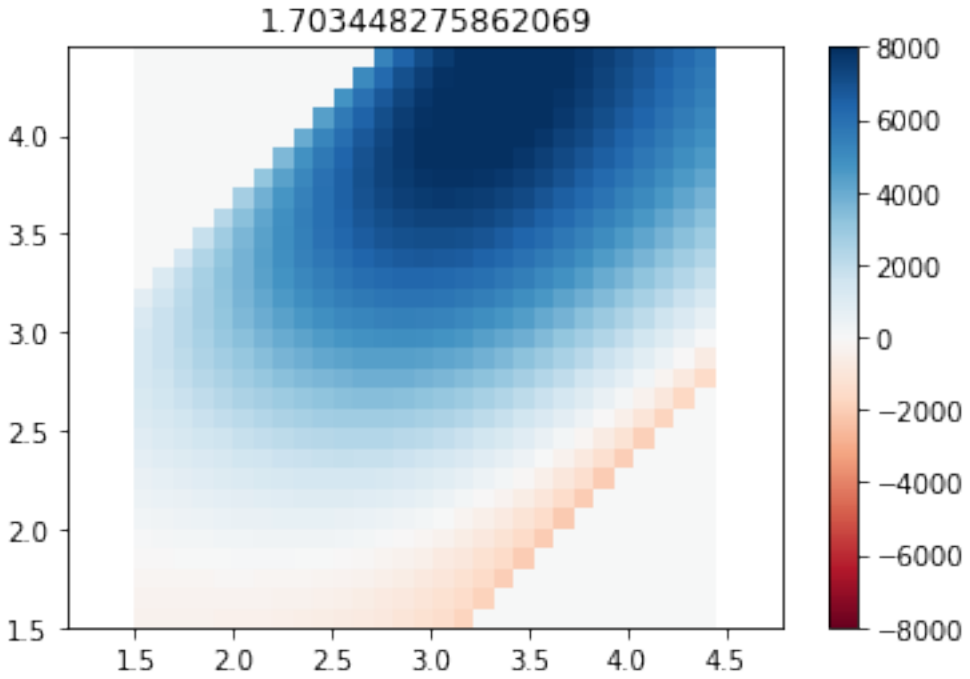


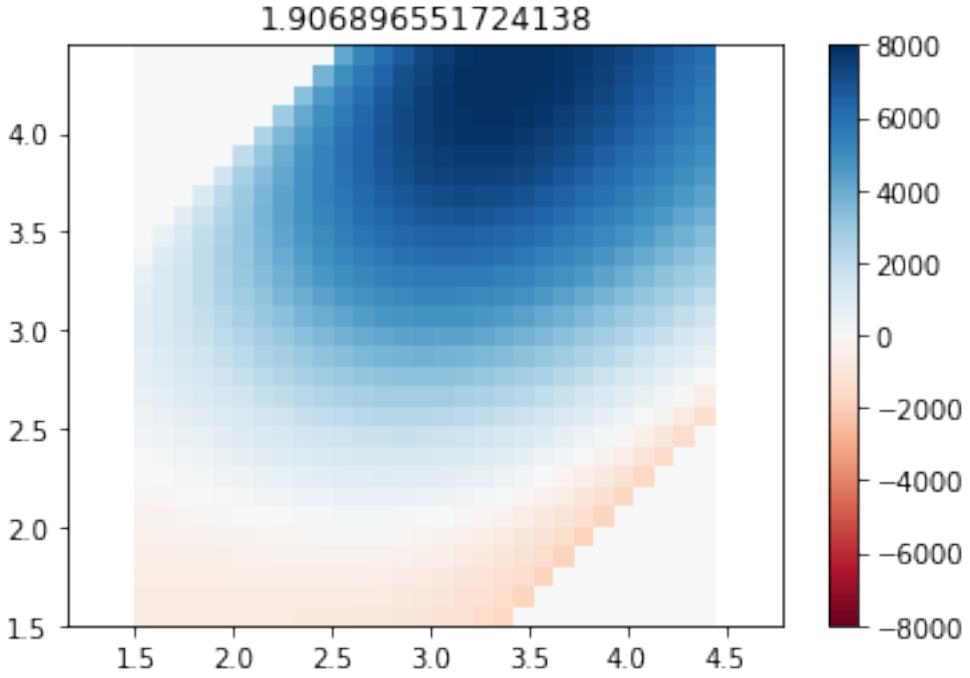

```
In [51]: print('========== ThreeBodySingleSpecies ==========')

         # future: TwoBodySingleSpecies.from_json(directory / 'test.json')
         # rs, element1, _, _, grid_data_1_1_1 = np.load(str(directory / 'MFF_3b_ntr_10_sig_1.
         rs, element1, _, _, grid_data_1_1_1 = np.load(str(directory / 'MFF_3b_ntr_20_sig_1.00_
         grid_1_1_1 = Spline3D(rs, rs, rs, grid_data_1_1_1)

         calc = ThreeBodySingleSpecies(r_cut=3.7, grid_1_1_1=grid_1_1_1)

         atoms = traj[0]
         atoms.set_calculator(calc)

         rms = np.sqrt(np.sum(np.square(atoms.arrays['force'] - atoms.get_forces()), axis=1))
         print('MAEF on forces: {:.4f} +- {:.4f}'.format(np.mean(rms), np.std(rms)))
         print(atoms.get_potential_energy())

         print('========== Calculate MAEF for each steps ==========')

         for atoms in traj:
             atoms.set_calculator(calc)

             rms = np.sqrt(np.sum(np.square(atoms.arrays['force'] - atoms.get_forces()), axis=
             print('MAEF on forces: {:.4f} +- {:.4f}'.format(np.mean(rms), np.std(rms)))
             print('energy: {}'.format(atoms.get_potential_energy()))

INFO:m_ff.calculators:numbers is in system_changes
```

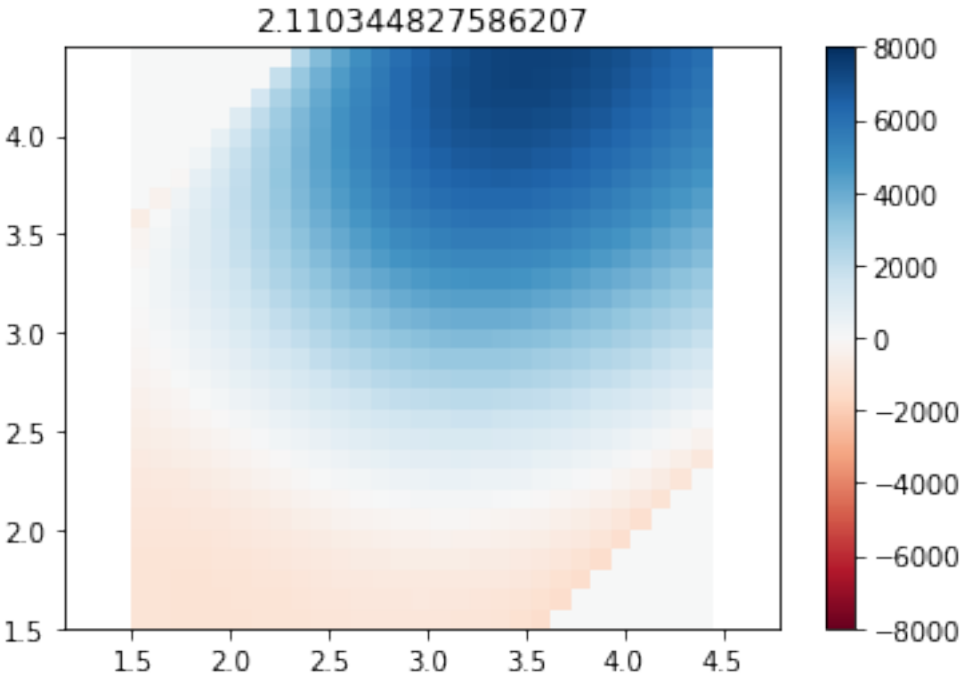
```
INFO:m_ff.calculators:initialize


========== ThreeBodySingleSpecies ==========
MAEF on forces: 82460.7472 +- 35603.8817
2905106.355425914
========== Calculate MAEF for each steps ==========
MAEF on forces: 82460.7472 +- 35603.8817
energy: 2905106.355425914
MAEF on forces: 81603.0771 +- 38472.0366
energy: 2916482.086216846
MAEF on forces: 81517.2225 +- 38519.9173
energy: 2913256.3958221003
MAEF on forces: 81418.0889 +- 38584.2406
energy: 2910011.553464438
MAEF on forces: 81310.2322 +- 38647.8092
energy: 2906773.0471959733
```

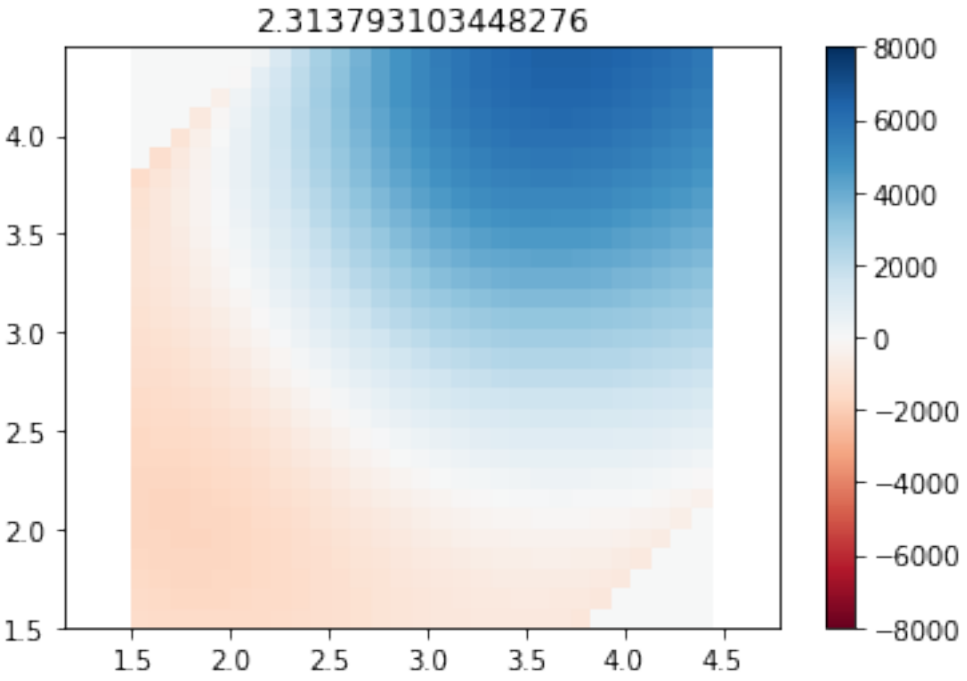
```
In [52]: z_min, z_max = -8000, 8000
         for i in range(0, len(rs), 2):
             plt.title(rs[i])
             plt.pcolor(rs, rs, -grid_data_1_1_1[i,:,:], cmap='RdBu', vmin=z_min, vmax=z_max)
             plt.colorbar()
             plt.axis('equal')

             plt.show()
```

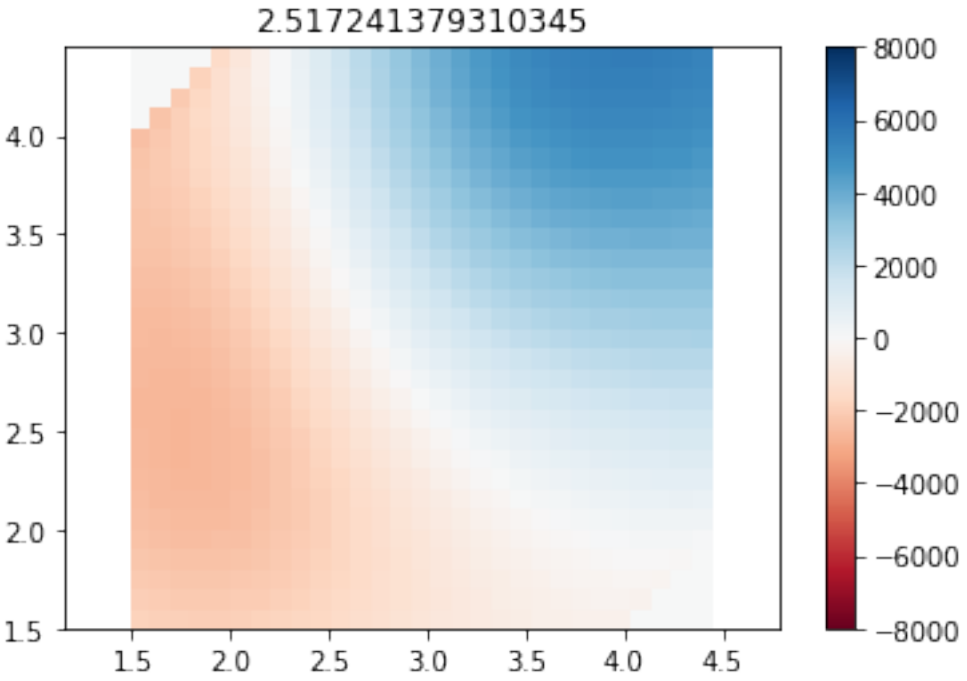


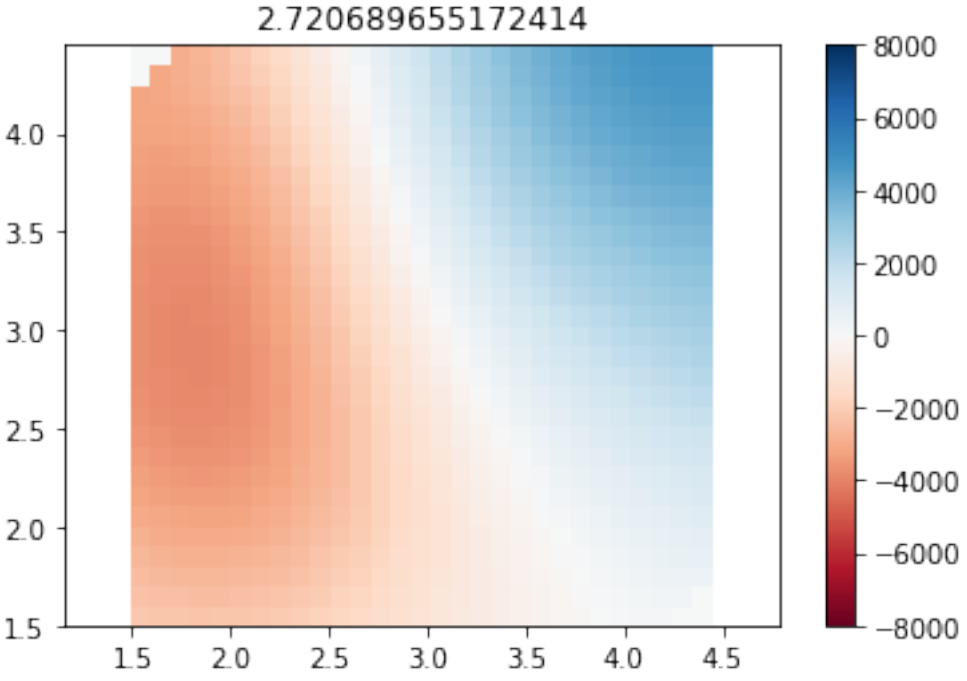

1.703448275862069
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000

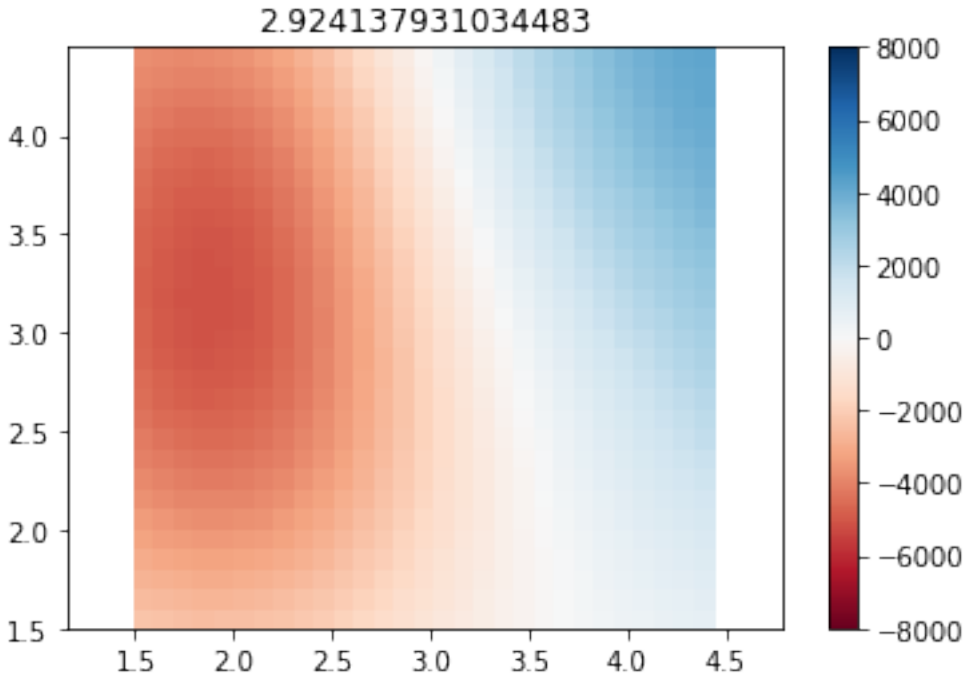

1.906896551724138
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000

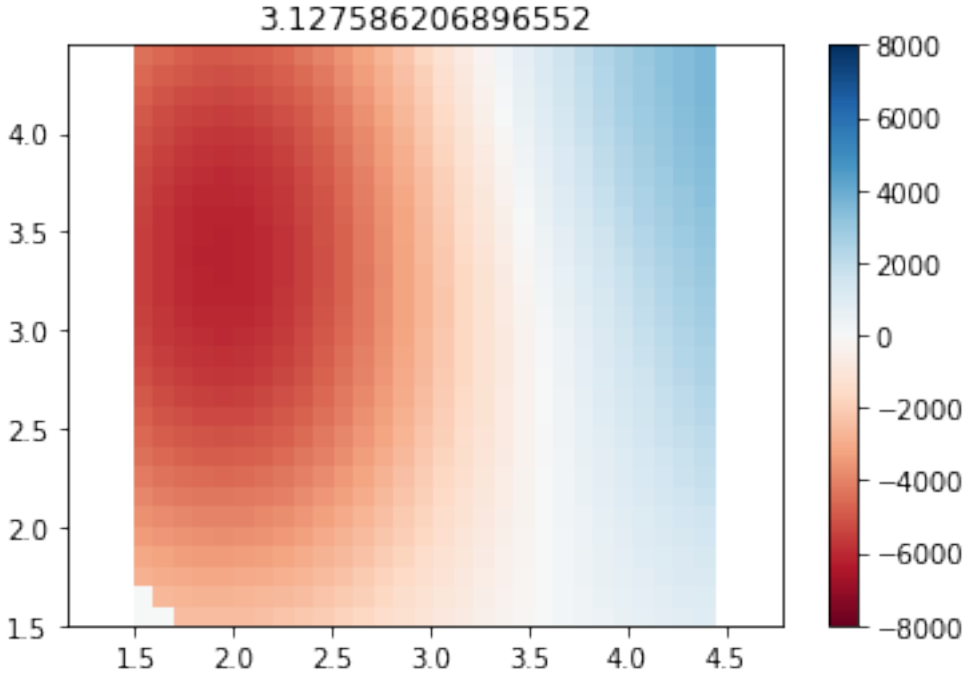

2.110344827586207
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000

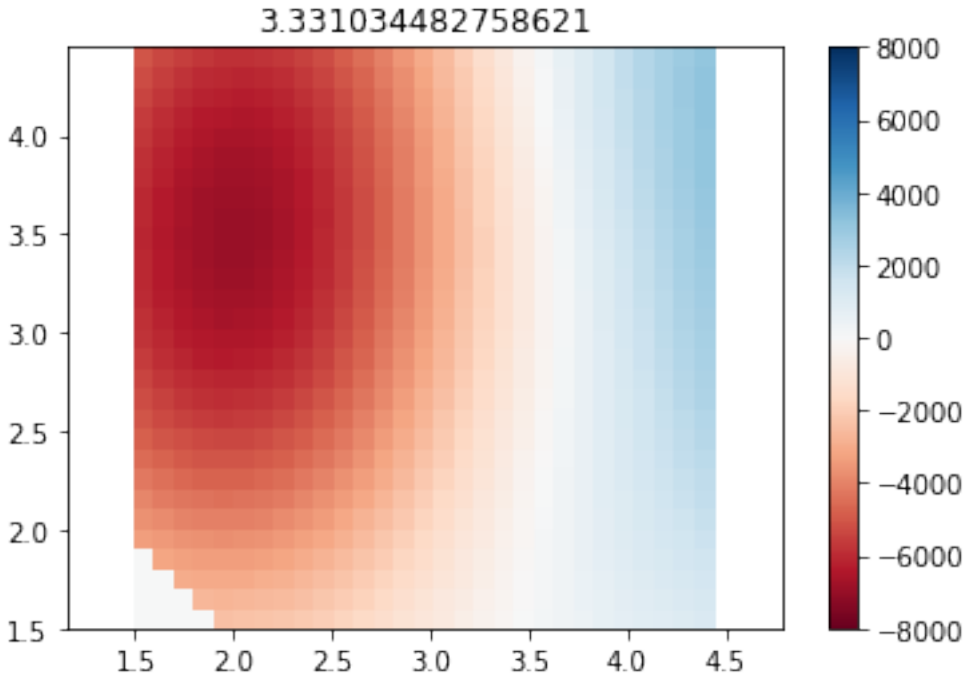

2.313793103448276
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000

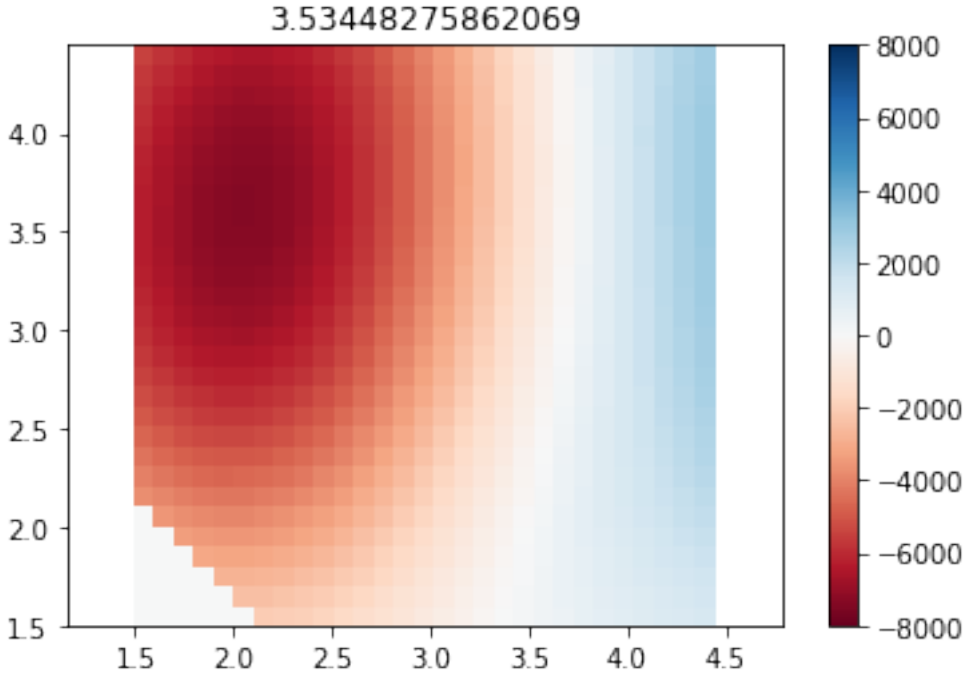

2.517241379310345
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000

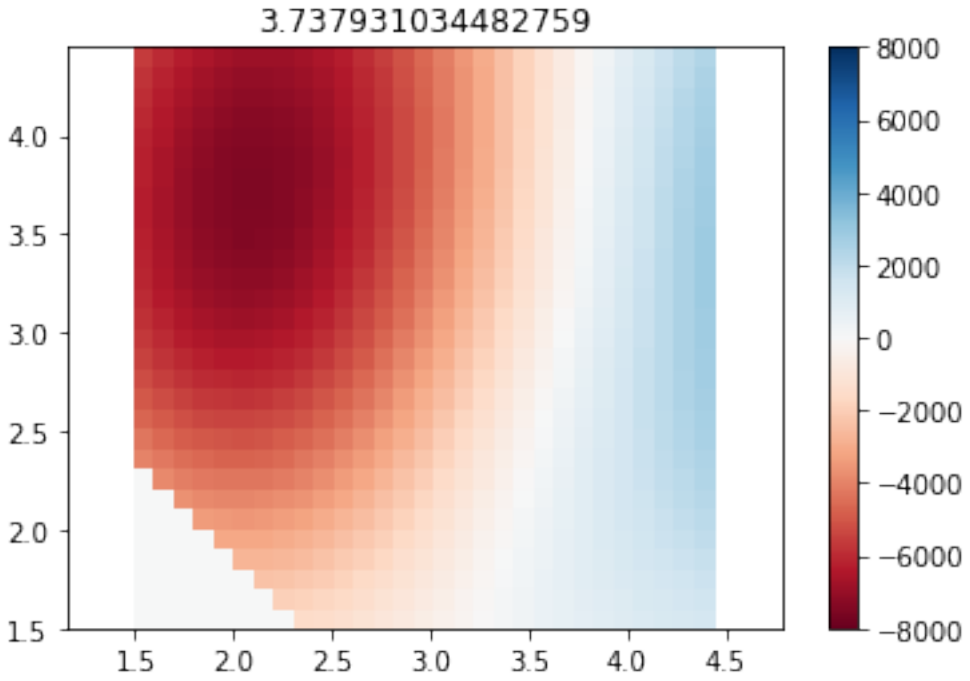

2.720689655172414
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000

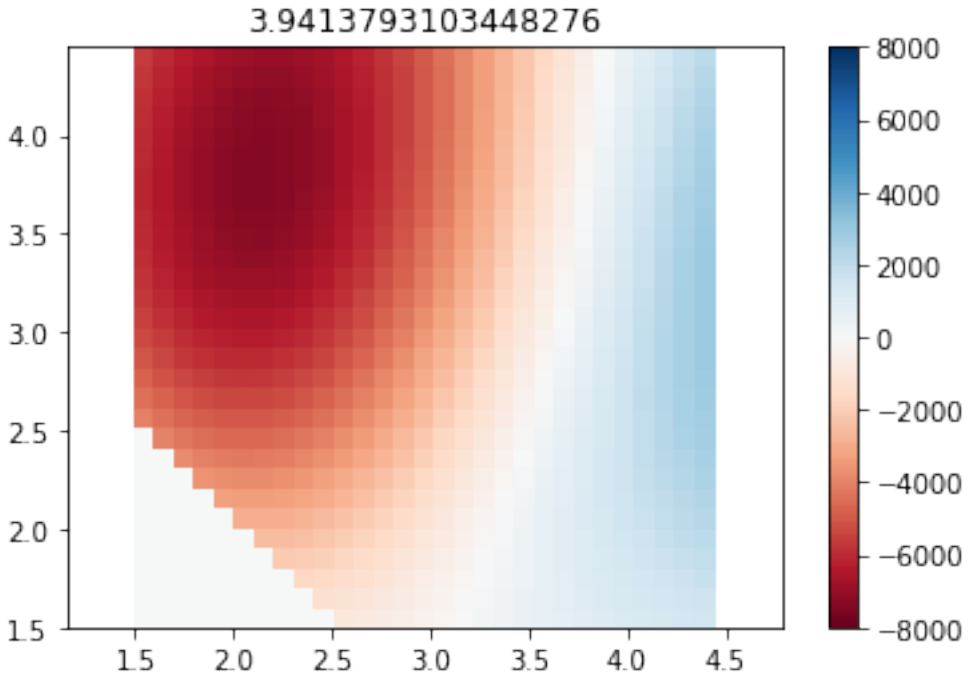

2.924137931034483
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000

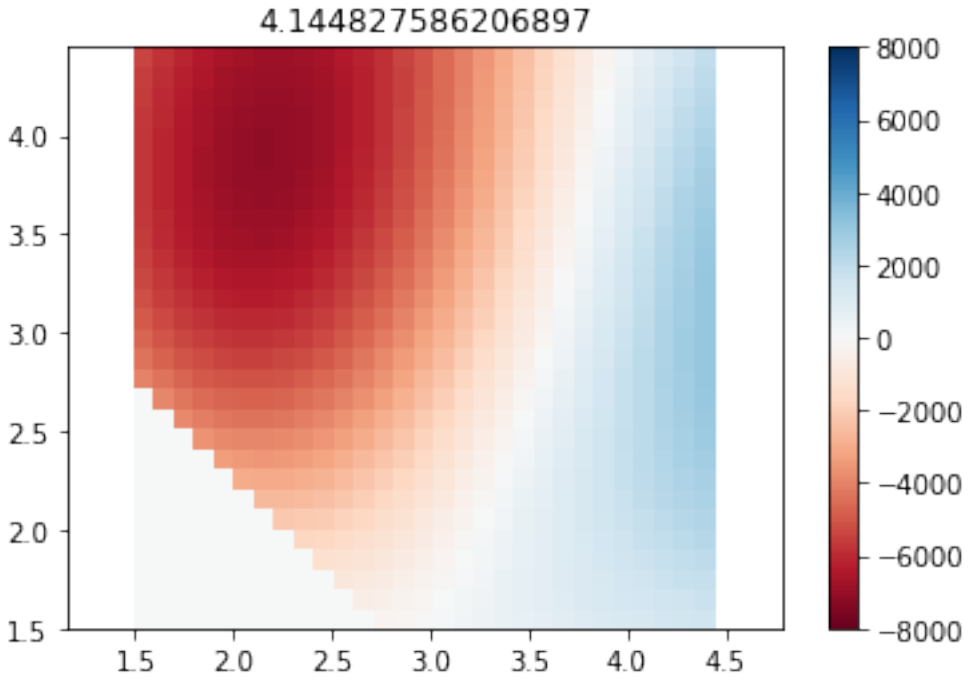

3.127586206896552
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000

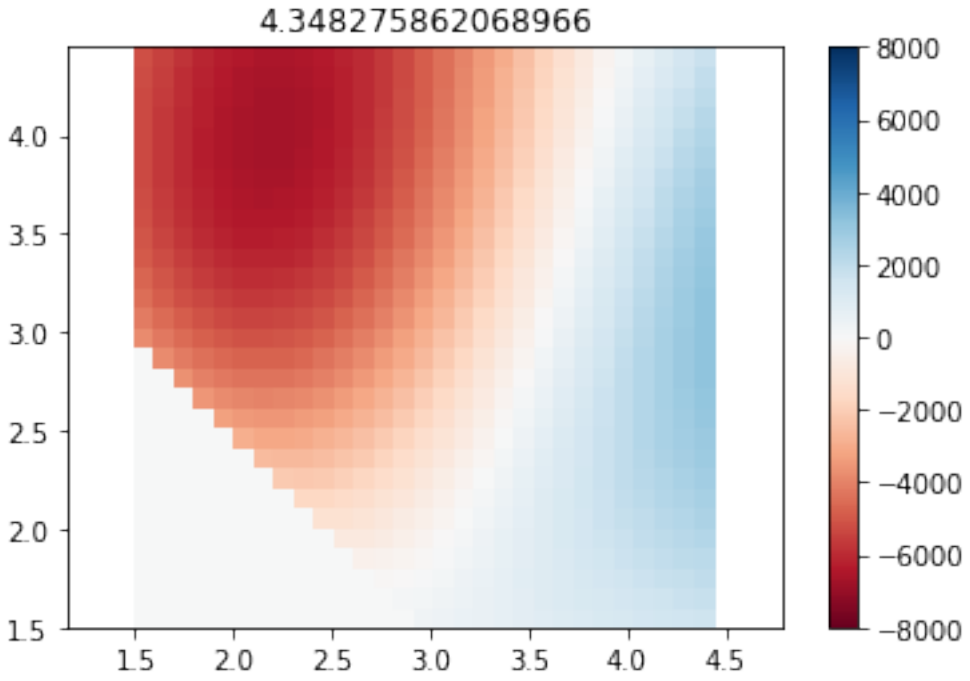

3.331034482758621
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000


3.53448275862069
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000


3.737931034482759
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000


3.9413793103448276
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000


4.144827586206897
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000


4.348275862068966
4.0
3.5
3.0
2.5
2.0
1.5
1.5
2.0
2.5
3.0
3.5
4.0
4.5
8000
6000
4000
2000
0
−2000
−4000
−6000
−8000