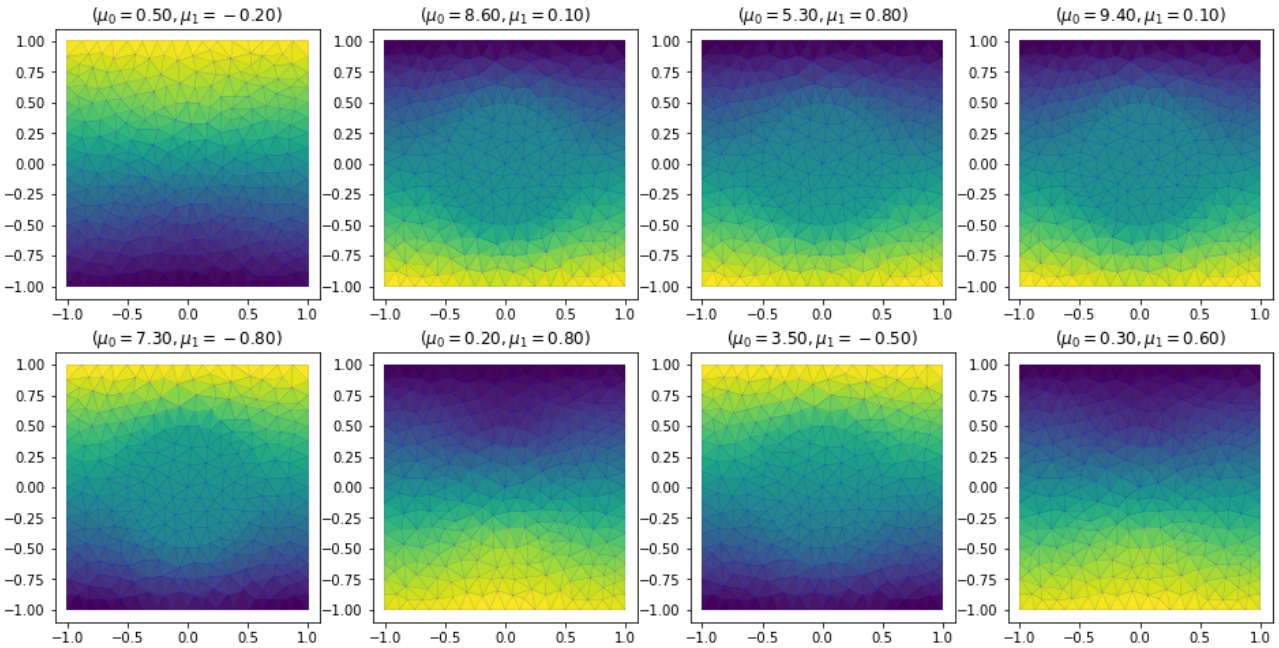
Prove EZyRB (POD+ANN)

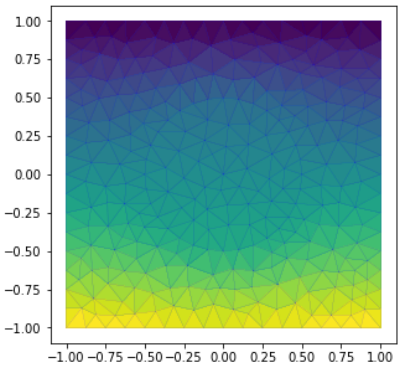
Prova Tutorial (https://github.com/mathLab/EZyRB/blob/master/tutorials/tutorial-1.ipynb) con metodo ANN al posto di RBF.

```
>>import numpy as np
>>import matplotlib.tri as mtri
>>import matplotlib.pyplot as plt

>>from pod import POD
>>from database import Database
>>from reducedordermodel import ReducedOrderModel as ROM
>>from ann import ANN

>>snapshots = np.load('tut1_snapshots.npy')
>>param = np.load('tut1_mu.npy')
>>print(snapshots.shape, param.shape)

(8, 304) (8, 2)

>>tri = np.load('tut1_triangles.npy')
>>coord = np.load('tut1_coord.npy')
>>triang = mtri.Triangulation(coord[0],coord[1],tri)

>>fig, ax = plt.subplots(nrows=2, ncols=4, figsize=(16, 8))
>>ax = ax.flatten()
>>for i in range(8):
>>    ax[i].triplot(triang, 'b-', lw=0.1)
>>    ax[i].tripcolor(triang, snapshots[i])
>>    ax[i].set_title('($\mu_0={:5.2f}, \mu_1={:5.2f})$'.format(*param[i]))
>>plt.show()
```




```
>>db = Database(param, snapshots)
>>pod = POD('svd')
>>ann=ANN()
>>rom = ROM(db, pod, ann)
>>rom.fit();
```

```
>>new_mu = [8,1]
>>pred_sol = rom.predict(new_mu)
>>plt.figure(figsize=(5, 5))
>>plt.triplot(triang, 'b-', lw=0.1)
>>plt.tripcolor(triang, pred_sol);
>>plt.show()
```




```
>>for pt, error in zip(rom.database.parameters, rom.loo_error()):
>>    print(pt, error)

[ 0.5 -0.2] 17.635544982576203
[8.6 0.1] 0.12007101548509831
[5.3 0.8] 5.5987467404017774
[9.4 0.1] 0.26921660448149615
[ 7.3 -0.8] 2.9681464165793514
[0.2 0.8] 6.530223896707341
[ 3.5 -0.5] 4.0078763716508625
[0.3 0.6] 5.241983607145813

>>print(rom.optimal_mu())

[array([2.9101239 , 0.54092062])]
```

Prova alternativa con dati inventati (come nel commento file ann.py ma con POD)

```
>>import numpy as np
>>from pod import POD
>>from database import Database
>>from reducedordermodel import ReducedOrderModel as ROM
>>from ann import ANN
>>x = np.random.uniform(-1, 1, size=(4, 2))
>>y = np.array([np.sin(x[:, 0]), np.cos(x[:, 1]**3)]).T
>>db = Database(x,y)
>>pod = POD('svd')
>>ann=ANN()
>>rom = ROM(db, pod, ann)
>>rom.fit();
>>y_pred=rom.predict(x)
>>print(y_pred) # valori predetti

[[-0.35667669  0.99398365]
 [ 0.600165    0.98064295]
 [ 0.08926728  1.00172569]
 [ 0.27144866  0.99035989]]

>>print(y) # valori veri (training)

[[-0.35634118  0.99758056]
 [ 0.60101312  0.98678394]
```

```
 [ 0.08903828  0.9999996 ]
 [ 0.27047773  0.98245491]]
```

Prova come nel commento del file ann.py

```
>>import numpy as np
>>from ann import ANN
>>x = np.random.uniform(-1, 1, size=(4, 2))
>>y = np.array([np.sin(x[:, 0]), np.cos(x[:, 1]**3)]).T
>>ann=ANN()
>>ann.fit(x,y) # restituisce valore training loss a terminazione
Out[1]: 5.482215328811435e-06
>>y_pred = ann.predict(x)
>>print(y)

[[-0.50516199  1.        ]
 [-0.72991689  0.99996131]
 [-0.32313993  0.99999838]
 [-0.28573522  0.99732108]]

>>print(y_pred)
[[-0.50097495  1.0032822 ]
 [-0.7317835   0.9985058 ]
 [-0.32590324  0.998602  ]
 [-0.2854157   0.9968272 ]]
```