POD-NN

POD-NN (usefull when you do not have affinity, so you cannot exploite offline-online phase) is a strategy that allows not to rely on affinity on the online stage: the projection stage is not performed and thus the speedup is guaranteed, yet having accurate solutions.

The POD-NN algorithm relies on two stages:

1. a POD,
2. a training of a Feed-forward Neural Network that predicts the entries of the reduced vector u_{rb}.

As usual, we need **a lot of FOM simulations**. Let us import gedim!

In [1]:
```python
import sys
sys.path.append('../../CppToPython')
```

In [2]:
```python
import numpy as np
import GeDiM4Py as gedim
```

In [3]:
```python
lib = gedim.ImportLibrary("../../CppToPython/release/GeDiM4Py.so")

config = { 'GeometricTolerance': 1.0e-8 }
gedim.Initialize(config, lib)
```

The parametric version of the heat conductivity equation

Solving the following equation on square $\bar{\Omega} = [-1, +1] \times [-1, +1]$, studying a parametric diffiusion coefficient

$$\begin{cases} \nabla \cdot (k_\mu \nabla u) = 0 & \text{in } \Omega \\ k_\mu \nabla u \cdot n_1 = \mu_2 & \text{in } \Gamma_{down} \\ u = \sin(\mu_3 \pi x) & \text{in } \Gamma_{up} \text{ - here we can see that we cannot use the affinity: no separation of variable} \\ k_\mu \nabla u \cdot n_2 = 0 & \text{otherwise } \text{ Omogeneuson Neumann} \end{cases}$$

where $k = \mu_1$ if $x^2 + y^2 \leq R^2$ and $k = 1$ otherwise. The parametric space is $\mathcal{P} = [0.1, 10] \times [0, 1] \times [-1, 1]$.

The problem is *standard*. However, we note a nonlinear dependency of the Dirichlet boundary term over Γ_{up} --> here is the problem, the sin!

In [4]:
```python
def Heat_R():
        return 0.5 # Take the radius

def Domain(numPoints, points): # Put 1 allover the points to consider all the domain
        matPoints = gedim.make_nd_matrix(points, (3, numPoints), np.double)
        values = np.ones(numPoints)
        return values.ctypes.data

############## DIRICHLET VARYING WRT mu_3 ####################
def Dirichlet_Term(numPoints, points): # We do not put here another depencency (mu_3) because Gedim does not support this
    # So we will define mu_3 and after call this function
        matPoints = gedim.make_nd_matrix(points, (3, numPoints), np.double)
        values = np.ones(numPoints)
        for p in range(0, numPoints):
        # The values on the Dirichlet boundary I set the value sin(mu_3 * pi * x)
                values[p] = np.sin(mu_3*np.pi*matPoints[0,p])  ### mu_3 is not defined, but not a problem
        return values.ctypes.data
###################

def Circle(numPoints, points): # In the circle
        matPoints = gedim.make_nd_matrix(points, (3, numPoints), np.double)
        values = np.ones(numPoints)
        for p in range(0, numPoints):
                if (matPoints[0,p] * matPoints[0,p] + matPoints[1,p] * matPoints[1,p]) > (Heat_R() * Heat_R() + 1.0e-16):
                        values[p] = 0.
        return values.ctypes.data

def NotCircle(numPoints, points): # Out of the circle
        matPoints = gedim.make_nd_matrix(points, (3, numPoints), np.double)
        values = np.ones(numPoints)
        for p in range(0, numPoints):
                if (matPoints[0,p] * matPoints[0,p] + matPoints[1,p] * matPoints[1,p]) <= (Heat_R() * Heat_R() + 1.0e-16):
                        values[p] = 0.
        return values.ctypes.data

def Heat_weakTerm_down(numPoints, points):
        values = np.ones(numPoints)
        return values.ctypes.data
```

Let us define the High Fidelity Simulation Parameters and import the mesh.

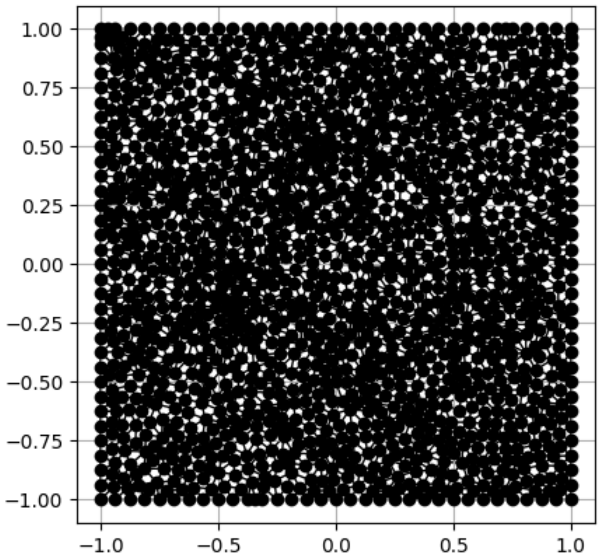
In [5]:
```python
order = 1
```

In [6]:
```python
%%writefile ImportMesh.csv
InputFolderPath
../../CppToPython/Meshes/Mesh3
```

```
Overwriting ImportMesh.csv
```

In [7]:
```
[meshInfo, mesh] = gedim.ImportDomainMesh2D(lib)
```

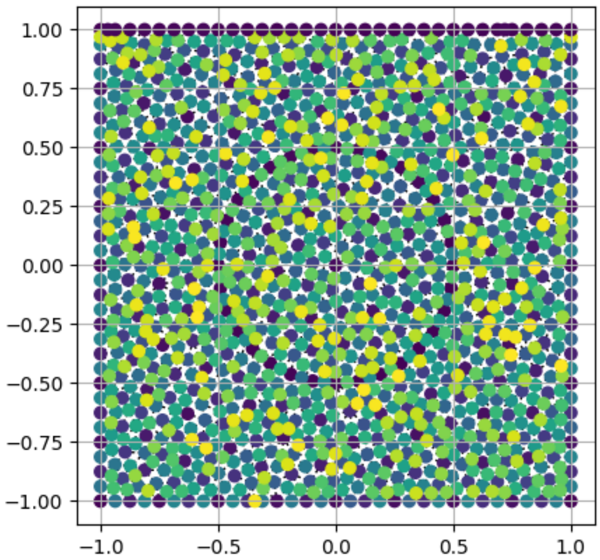
In [8]:
```
gedim.PlotMesh(mesh)
```



Let us create the space

In [9]:
```
discreteSpace = { 'Order': order, 'Type': 1, 'BoundaryConditionsType': [1, 3, 3, 2] }
[problemData, dofs, strongs] = gedim.Discretize(discreteSpace, lib)
```

In [10]:
```
gedim.PlotDofs(mesh, dofs, strongs)
```



Assemble the system

We can assemble only the parts that are $\mu-$ independent (together with the inner product matrix!). Namely, the Dirichlet term needs to be assembled later on. It is non-affine and nonlinear w.r.t to the parameter $\boldsymbol{\mu}$!

In [11]:
```
[stiffness1, stiffnessStrong1] = gedim.AssembleStiffnessMatrix(NotCircle, problemData, lib)

# Moltiply to the inner prodocond (as lab 2), to deal with non omogenoeus condition
[stiffness2, stiffnessStrong2] = gedim.AssembleStiffnessMatrix(Circle, problemData, lib)

weakTerm_down1 = gedim.AssembleWeakTerm(Heat_weakTerm_down, 1, problemData, lib)

#### inner product
# ||grad(u)||^2
inner_product = stiffness1 + stiffness2 # Inner product matrix (same notation of the theory stiffness_matrix = A)

######## DIRICHLET CANNOT BE ASSEMBLED NOW #########################
# We want to performe simulation for different parameters --> so every time that we performe this operation,
# before to doing that, we have to define the Dirichlet condition, compute everything, and definte it again
```

Let us define the training set for the POD

In [12]:
```
### define the training set

snapshot_num = 300
mu1_range = [0.1, 10.]
mu2_range = [-1., 1.]
mu3_range = [-1., 1.]
P = np.array([mu1_range, mu2_range, mu3_range])
```

```python
training_set = np.random.uniform(low=P[:, 0], high=P[:, 1], size=(snapshot_num, P.shape[0]))
```

We can now proceed with the snapshot matrix creation. However, we need to be careful: the problem is not affine in the parameters and we need to assemble the Dirichlet term for each parametric instance.

In [13]:
```python
#### snapshot matrix creation
thetaA1 = 1
snapshot_matrix = []

tol = 1. - 1e-7
N_max = 10

for mu in training_set: # All the parameters that I have to use
  thetaA2 = mu[0]
  thetaf1 = mu[1]
  mu_3 = mu[2]

  #### the problem is not affine: I have to assemble in this stage!! ###
  ## label --> in that way assemble the Dirichlet term non-omogeneous
  Dirichlet_top = gedim.AssembleStrongSolution(Dirichlet_Term, 3, problemData, lib)
  f1_D = stiffnessStrong1 @ Dirichlet_top # Change the value of the forcing term considering the Dirichlet condition
  f2_D = stiffnessStrong2 @ Dirichlet_top

  stiffness = thetaA1*stiffness1 + thetaA2*stiffness2
  weakTerm_down = thetaf1*weakTerm_down1 # Forcing to the Neumann boundary condition
  Dirichlet_contribution = thetaA1*f1_D + thetaA2*f2_D # Contribution on the Dirichlet boundary condition

  f = weakTerm_down - Dirichlet_contribution

  snapshot = gedim.LUSolver(stiffness, f, lib)

  # if you do not want to plot uncomment
  # gedim.PlotSolution(mesh, dofs, strongs, snapshot, Dirichlet_top)
  snapshot_matrix.append(np.copy(snapshot))

snapshot_matrix = np.array(snapshot_matrix)
```

Let us build and analyze the covariance matrix.

In [14]:
```python
### covariance matrix

C = snapshot_matrix @ inner_product @ np.transpose(snapshot_matrix) # Build covariance wrt the inner product

# VM, L, VMt = np.linalg.svd((C))

# Look for eigenvalue (as to be real and non-negative) and eigenvector
L_e, VM_e = np.linalg.eig(C)
eigenvalues = []
eigenvectors = []


#### check
for i in range(len(L_e)):
  eig_real = L_e[i].real
  eig_complex = L_e[i].imag
  assert np.isclose(eig_complex, 0.) # Check if the eigenvalue are non-negative
  eigenvalues.append(eig_real)
  eigenvectors.append(VM_e[i].real)


total_energy = sum(eigenvalues)
retained_energy_vector = np.cumsum(eigenvalues)
relative_retained_energy = retained_energy_vector/total_energy # To check the torelance (as always)


if all(flag==False for flag in relative_retained_energy>= tol):
  N = N_max
else:
  N = np.argmax(relative_retained_energy >= tol) + 1

print(N)
print(relative_retained_energy) # In 9 basis function we reach a good approximation
# --> here that we have non-linearity, the number of basis function that we have to use
# is increased (in the other lab was only 3)
```

```
9
[0.92034006 0.97085465 0.99625438 0.99909937 0.99997566 0.99999823
 0.99999961 0.99999985 1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.        ]
```

And now let us build the basis functions and $\mathbb{B}$.

In [15]:
```python
# Create the basis function matrix
basis_functions = []
for n in range(N):
  eigenvector =  eigenvectors[n]
  basis = np.transpose(snapshot_matrix)@eigenvector
  norm = np.sqrt(np.transpose(basis) @ inner_product @ basis) ## inner product
  basis /= norm # To have more stability on the bases --> Good operation that we have to do!
  basis_functions.append(np.copy(basis))

basis_functions = np.transpose(np.array(basis_functions))

# FINISH THE OFFLINE PHASE, now consider another parameter and compute the online phase
```

If we want to perform standard ROMs we still need to assemble the system.

During offline phase you have to perform

- POD
- Project: $Au = f$, where f is the inner condition and compute

$$\sum_{q_A} \Theta_A^{q_A}(\mu) A^{q_A} = \sum_{q_f} \Theta_f^{q_f}(\mu) f^{q_f}$$

- Store $A_N^{q_A}$ & $f_N^{q_f}$ so that during the online phase you have only to compute the $\mu :=$ the weight

Can we asseble it?

In [16]:
```python
########## ASSEMBLE WHAT I CAN ##### STILL OFFLINE
# I can assemble the reduced stiffness matrix
# Online I'll assemble the Dirichlet term
reduced_stiff1 = np.transpose(basis_functions) @ stiffness1 @ basis_functions
reduced_stiff2 = np.transpose(basis_functions) @ stiffness2 @ basis_functions
reduced_w =  np.transpose(basis_functions) @ weakTerm_down1
```

For each new parameter I have to assemble the Dirichlet term, once again.

In [17]:
```
########### I CANNOT DO THAT ################ STILL ONLINE??? WE NEED THE PARAMETER
thetaA2 = 2.
thetaf1 = 0.8
mu_3 = 1.
```

In [18]:
```
#### the problem is not affine: I have to assemble in this stage!! ###

Dirichlet_top = gedim.AssembleStrongSolution(Dirichlet_Term, 3, problemData, lib) ## label
f1_D = stiffnessStrong1 @ Dirichlet_top
f2_D = stiffnessStrong2 @ Dirichlet_top
r_f1_D = np.transpose(basis_functions) @ (stiffnessStrong1 @ Dirichlet_top)
r_f2_D = np.transpose(basis_functions) @ (stiffnessStrong2 @ Dirichlet_top)
```

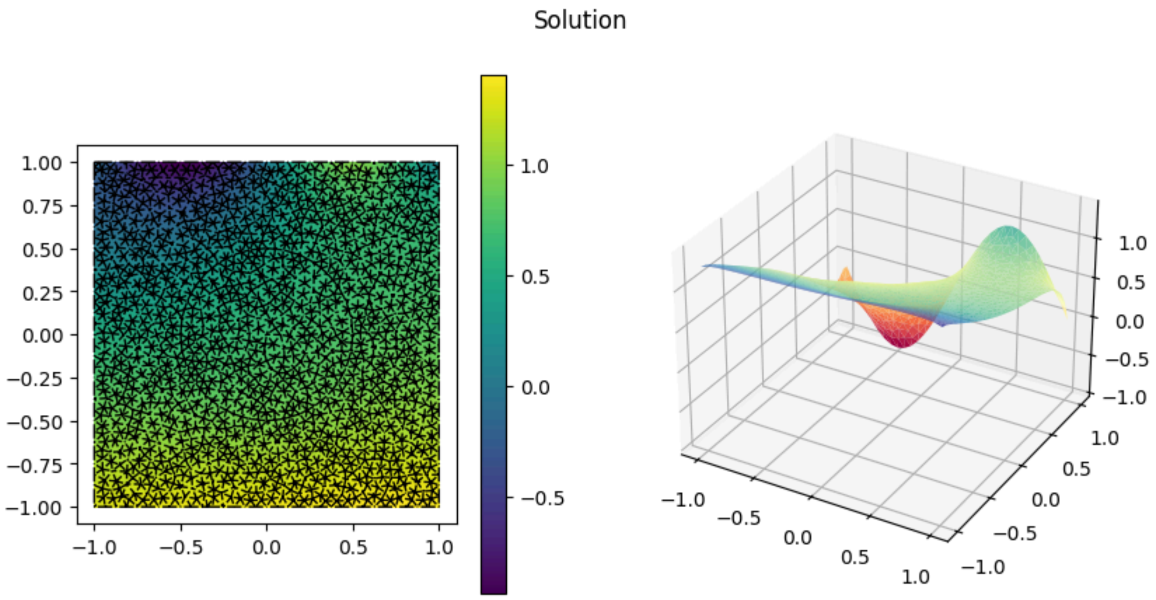
Solve linear system for a new μ

In [19]:
```
reduced_rhs = thetaA1*reduced_stiff1 + thetaA2*reduced_stiff2
reduced_lhs = thetaf1*reduced_w - (thetaA1*r_f1_D + thetaA2*r_f2_D)
```

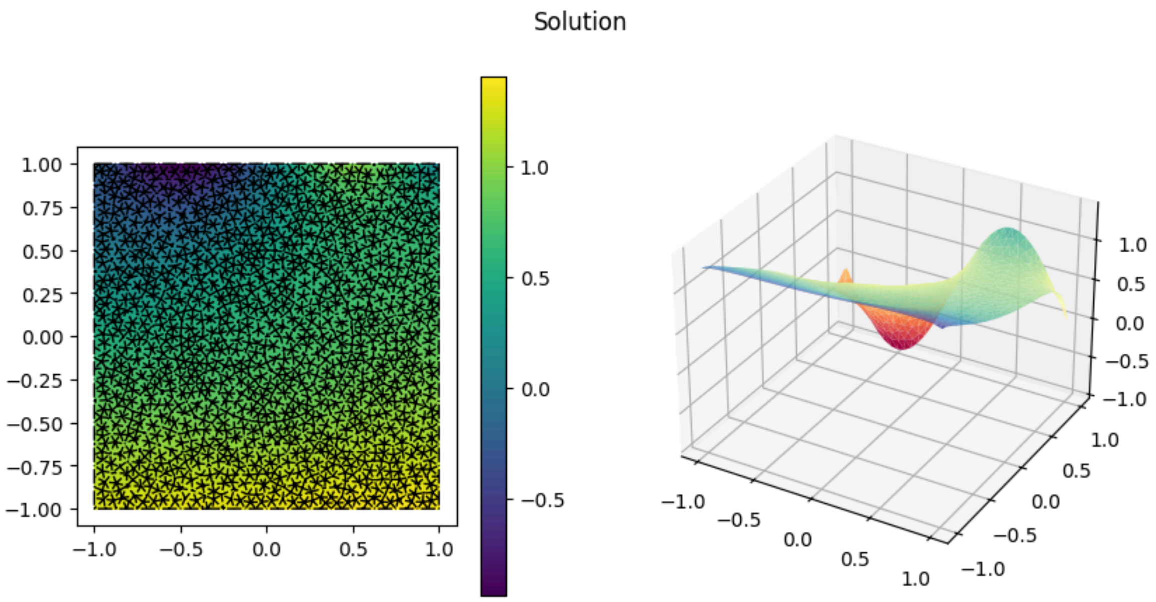
In [20]:
```
#####solve
reduced_solution = np.linalg.solve(reduced_rhs, reduced_lhs)
print(reduced_solution)
```

```
[ -6.03581072   0.71357335  -2.71814498  23.38612387 -22.62647494
   0.54477816  -0.73222505  15.39373308   1.74032697]
```

In [21]:
```
###### plot #######
proj_reduced_solution = basis_functions @ reduced_solution
gedim.PlotSolution(mesh, dofs, strongs, proj_reduced_solution, Dirichlet_top)

stiffness = thetaA1*stiffness1 + thetaA2*stiffness2
weakTerm_down = thetaf1*weakTerm_down1
f = weakTerm_down - (thetaA1*stiffnessStrong1 + thetaA2*stiffnessStrong2) @ Dirichlet_top

full_solution = gedim.LUSolver(stiffness, f, lib)
```



In [22]:
```
gedim.PlotSolution(mesh, dofs, strongs, full_solution, Dirichlet_top)
```



Let us comment a bit on the error analysis and the *speed up*.

In [23]:
```
### compute error
import time

abs_err = []
rel_err = []
testing_set = np.random.uniform(low=P[:, 0], high=P[:, 1], size=(100, P.shape[0]))
speed_up = []

print("Computing error and speedup analysis") # Same block that we already see to compute the error

for mu in testing_set:
  thetaA2 = mu[0]
  thetaf1 = mu[1]
  mu_3 = mu[2]

  #### the problem is not affine: I have to assemble in this stage!! ###
  start_assembling = time.time()
  Dirichlet_top = gedim.AssembleStrongSolution(Dirichlet_Term, 3, problemData, lib) ## label
  f1_D = stiffnessStrong1 @ Dirichlet_top
  f2_D = stiffnessStrong2 @ Dirichlet_top
  r_f1_D = np.transpose(basis_functions) @ (stiffnessStrong1 @ Dirichlet_top)
  r_f2_D = np.transpose(basis_functions) @ (stiffnessStrong2 @ Dirichlet_top)
  time_assembling =  time.time() - start_assembling # Time of assemble the part of the problem -> everytime you have to assemble the problem

  ##### full #####
  stiffness = thetaA1*stiffness1 + thetaA2*stiffness2
  weakTerm_down = thetaf1*weakTerm_down1
  f = weakTerm_down - (thetaA1*stiffnessStrong1 + thetaA2*stiffnessStrong2) @ Dirichlet_top


  start_fom = time.time()
  full_solution = gedim.LUSolver(stiffness, f, lib)
  time_fom = time.time() - start_fom # Here not adding the time for assembling

  #### reduced #####
  reduced_rhs = thetaA1*reduced_stiff1 + thetaA2*reduced_stiff2
  reduced_lhs = thetaf1*reduced_w - (thetaA1*r_f1_D + thetaA2*r_f2_D)

  start_rom = time.time()
  reduced_solution = np.linalg.solve(reduced_rhs, reduced_lhs)
  time_rom = time.time() - start_rom

  speed_up.append(time_fom/(time_rom + time_assembling)) # Time assembling is for ROM and FOM so, NOT like here,
                                                        # you have to take in account in all the two part

  proj_reduced_solution = basis_functions@reduced_solution

  ### computing error
  error_function = full_solution - proj_reduced_solution
  error_norm_squared_component = np.transpose(error_function) @ inner_product @ error_function
  absolute_error = np.sqrt(abs(error_norm_squared_component))
  abs_err.append(absolute_error)

  full_solution_norm_squared_component = np.transpose(full_solution) @  inner_product @ full_solution
  relative_error = absolute_error/np.sqrt(abs(full_solution_norm_squared_component))
  rel_err.append(relative_error)
```

```
Computing error and speedup analysis
```

In [24]:
```
print("avarege relative error = ", np.mean(rel_err) )
print("avarege absolute error = ", np.mean(abs_err) )
print("avarege speed_up = ", np.mean(speed_up) )
```

```
avarege relative error =  0.00045595117863909046
avarege absolute error =  0.0016646749673025001
avarege speed_up =  12.673577305539716
```

The speed up is quite small for a linear problem. Let understand the role of POD-NN in this setting.

We want to use a feed-forward NN. Let us define the Class Net with pytorch.

POD NN

- Apply POD
- Train $\Pi^{NN}(\mu, \mu \rightarrow u(\mu)$ where $\mu \in \mathcal{R}^3$ and $u(\mu) \in \mathcal{R}^N$, that in this case $N = 9$, chosen by the POD (tol that we select before)

The dimension N is fixed before with the value of the tollerance.

NB More parameters, more basis function, the number of the basis function is related to the dinamics complete of the parameters, where they are, which dimension have

In [25]:
```
# Define the Net
import torch
import torch.nn as nn
import torch.nn.functional as F

mu_dim = P.shape[0]
basis_dim = N

# I need this dimension
input_dim = mu_dim
output_dim = basis_dim
```

```
nodes = 30

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # Starting layer
        self.fc1 = nn.Linear(input_dim, nodes)

        # Hidden layer
        self.fc2 = nn.Linear(nodes, nodes)
        self.fc3 = nn.Linear(nodes, nodes)
        self.fc4 = nn.Linear(nodes, nodes)
        # Finish hidden layer

        # Last layer
        self.fc5 = nn.Linear(nodes, output_dim)

        self.tanh = nn.Tanh()
        # self.apply(self._init_weights)


    def forward(self, x):  ### Forward law ----> prediction
        # Activation function
        x = self.tanh(self.fc1(x)) # Iperbolic tangent [-1, 1]
        x = self.tanh(self.fc2(x))
        x = self.tanh(self.fc3(x))
        x = self.tanh(self.fc4(x))

        x = self.fc5(x) # This is the output, we do not change the range of the solution
        # It is better to leave the final solution in the space where it is to do not modify it

        return x
```

In [26]:
```
seed_num = 31 # You can change this, different seed == different initialization of weight --> maybe better performance

torch.manual_seed(seed_num)
net = Net()
torch.set_default_dtype(torch.float32)

my_loss = nn.MSELoss() # MSE loss
optimizer = torch.optim.Adam(net.parameters(), lr=0.001)

epoch_max = 500000
epoch = 0
tol = 1e-5
loss = 1. # To start the optimization
```

We need to prepare the outputs to train the NN. Indeed, our goal is to define

$$\boldsymbol{\pi}(\boldsymbol{\mu}) = \underline{u}_{\text{rb}}^{NN}(\boldsymbol{\mu}).$$

Namely, our inputs are the parameters of the training set and the output is the Galerkin projection of the snapshots of the training set. The output is of the form $\underline{u}_{\text{rb}}$ where:

$$\mathbb{B}\underline{u}_{\text{rb}}(\boldsymbol{\mu}) = \mathbb{P}^{\boldsymbol{\mu}} u_\delta(\boldsymbol{\mu}), \qquad (1)$$

where $\mathbb{P}^{\boldsymbol{\mu}} = \mathbb{B}\mathbb{X}_N^{-1}\mathbb{B}^T\mathbb{X}_{N_\delta}$ (direct projection, projecjtion matrix along the line in which I project, this is not a change of variable) is the reduced vector related to the Galrkin projector, i.e. the best approximation of u_δ in V_N w.r.t. the inner-product defined by the matrix X_δ.

Instead of computing the inverse of $\mathbb{X}_N = \mathbb{B}^T\mathbb{X}_\delta\mathbb{B}$ we solve the following system:

$$\mathbb{B}^T\mathbb{X}_\delta\mathbb{B}\underline{u}_{\text{rb}}(\boldsymbol{\mu}) = \mathbb{B}^T\mathbb{X}_\delta u_\delta(\boldsymbol{\mu})$$

to find $u_{\text{rb}}(\boldsymbol{\mu})$ for each snapshot.

In this way we are taking the vector of the reduced solution related to the parameter $\boldsymbol{\mu}$ **without solving the reduced system**, thanks to the relation (1). This element is the closest element (the best choice) to u_δ in the norm of the problem.

Computation performe during the theory (check lesson 10 or 11)

$$\mathbb{B}^T\mathbb{X}_\delta\mathbb{B}\underline{u}_{\text{rb}} = \mathbb{B}^T\mathbb{X}_\delta u_\delta(\boldsymbol{\mu})$$

$$\underline{u}_{\text{rb}} = \mathbb{X}_N^{-1}\mathbb{B}^T\mathbb{X}_\delta u_\delta$$

Inverte the matrix is NOT a good idea, so we can solve the linear sisteme to take the solution.

In [27]:
```
####### training set ########
reduced_inner_product = np.transpose(basis_functions) @ inner_product @ basis_functions
x_train = torch.tensor(np.float32(training_set)) # The input are all the parameters that I have
y_train = []


for i in range(snapshot_matrix.shape[0]):
  snapshot_to_project = snapshot_matrix[i]

  projected_snapshot = np.linalg.solve(reduced_inner_product, np.transpose(basis_functions)@inner_product@snapshot_to_project)
    # == B X snapshots
    # We have to solve X_N u_rb = BT X_delta u_delta
```

```
  y_train.append(projected_snapshot) # On dimension N = 9

y_train = np.float32(y_train)
y_train = torch.tensor(y_train) # To np object to tensor
```

Let us train our neural network!

The loss is

$$\sum_{i=1}^{300} = \frac{1}{300}||\pi(\mu)^{NN} - u_\delta^P||_{l_2}^2$$

where u_δ^P is the projected u_δ.

During the epochs we

- Change tha learning rate
- The loss fluctuated

In [28]:
```
while loss >= tol and epoch < epoch_max:
  epoch = epoch + 1
  optimizer.zero_grad()

  ## compute output
  output = net(x_train) # Net apply to my parameter --> to each parameter I have a solution in dimension 9 that are the projection

  loss = my_loss(output, y_train) # Computing the loss

  if epoch >= 20000:
    optimizer.param_groups[0]['lr'] = 0.0001 # To change the learning rate during the epoch with a dictionary

  #compute the gradients
  loss.backward()

  # optimizer update
  optimizer.step()

  if epoch % 600 == 199:
    print("epoch", epoch, 'loss', loss.item(), 'lr', optimizer.param_groups[0]['lr'] )
```

```
epoch 199 loss 22.54425621032715 lr 0.001
epoch 799 loss 7.93619441986084 lr 0.001
epoch 1399 loss 5.095813751220703 lr 0.001
epoch 1999 loss 3.3643243312835693 lr 0.001
epoch 2599 loss 2.026888370513916 lr 0.001
epoch 3199 loss 1.184842586517334 lr 0.001
epoch 3799 loss 0.6669406890869141 lr 0.001
epoch 4399 loss 0.3686889708042145 lr 0.001
epoch 4999 loss 0.21508687734603882 lr 0.001
epoch 5599 loss 0.13974961638450623 lr 0.001
epoch 6199 loss 0.09046071022748947 lr 0.001
epoch 6799 loss 0.06065932661294937 lr 0.001
epoch 7399 loss 0.045128095895051956 lr 0.001
epoch 7999 loss 0.03695325180888176 lr 0.001
epoch 8599 loss 0.031973566859960556 lr 0.001
epoch 9199 loss 0.028423696756362915 lr 0.001
epoch 9799 loss 0.025726882740855217 lr 0.001
epoch 10399 loss 0.023443803191184998 lr 0.001
epoch 10999 loss 0.02149907313287258 lr 0.001
epoch 11599 loss 0.019777318462729454 lr 0.001
epoch 12199 loss 0.018219957128167152 lr 0.001
epoch 12799 loss 0.01690034568309784 lr 0.001
epoch 13399 loss 0.01578049547970295 lr 0.001
epoch 13999 loss 0.014831479638814926 lr 0.001
epoch 14599 loss 0.0139616709202528 lr 0.001
epoch 15199 loss 0.013202089816331863 lr 0.001
epoch 15799 loss 0.012594764120876789 lr 0.001
epoch 16399 loss 0.012102173641324043 lr 0.001
epoch 16999 loss 0.011321942321956158 lr 0.001
epoch 17599 loss 0.011109759099781513 lr 0.001
epoch 18199 loss 0.010327382013201714 lr 0.001
epoch 18799 loss 0.010159967467188835 lr 0.001
epoch 19399 loss 0.009460605680942535 lr 0.001
epoch 19999 loss 0.00907201785594225 lr 0.001
epoch 20599 loss 0.008995935320854187 lr 0.0001
epoch 21199 loss 0.008909512311220169 lr 0.0001
epoch 21799 loss 0.008802286349236965 lr 0.0001
epoch 22399 loss 0.008667339570820332 lr 0.0001
epoch 22999 loss 0.008497134782373905 lr 0.0001
epoch 23599 loss 0.008283249102532864 lr 0.0001
epoch 24199 loss 0.008016826584935188 lr 0.0001
epoch 24799 loss 0.007704297546297312 lr 0.0001
epoch 25399 loss 0.007407039869576693 lr 0.0001
epoch 25999 loss 0.007136390078812838 lr 0.0001
epoch 26599 loss 0.006887876894325018 lr 0.0001
epoch 27199 loss 0.006662232335656881 lr 0.0001
epoch 27799 loss 0.006449060048907995 lr 0.0001
epoch 28399 loss 0.006251634564250708 lr 0.0001
epoch 28999 loss 0.006068953778594732 lr 0.0001
epoch 29599 loss 0.005904466845095158 lr 0.0001
epoch 30199 loss 0.005737600848078728 lr 0.0001
epoch 30799 loss 0.005588515195995569 lr 0.0001
epoch 31399 loss 0.005452996119856834 lr 0.0001
epoch 31999 loss 0.005317409988492727 lr 0.0001
epoch 32599 loss 0.005189369898289442 lr 0.0001
epoch 33199 loss 0.005069057922810316 lr 0.0001
epoch 33799 loss 0.005036523099988699 lr 0.0001
epoch 34399 loss 0.004849950782954693 lr 0.0001
epoch 34999 loss 0.004748696461319923 lr 0.0001
epoch 35599 loss 0.004653992131352425 lr 0.0001
epoch 36199 loss 0.004558723419904709 lr 0.0001
epoch 36799 loss 0.0044708289206027985 lr 0.0001
epoch 37399 loss 0.004387640859931707 lr 0.0001
epoch 37999 loss 0.004742261487990618 lr 0.0001
epoch 38599 loss 0.004397101700305939 lr 0.0001
epoch 39199 loss 0.004158081021159887 lr 0.0001
epoch 39799 loss 0.004088208079338074 lr 0.0001
epoch 40399 loss 0.004020282998681068 lr 0.0001
epoch 40999 loss 0.003955028485506773 lr 0.0001
epoch 41599 loss 0.0038940017111599445 lr 0.0001
epoch 42199 loss 0.003937260247766972 lr 0.0001
epoch 42799 loss 0.003773100906983018 lr 0.0001
epoch 43399 loss 0.0037163300439715385 lr 0.0001
epoch 43999 loss 0.004133789800107479 lr 0.0001
epoch 44599 loss 0.0036063441075384617 lr 0.0001
epoch 45199 loss 0.003555083880200982 lr 0.0001
epoch 45799 loss 0.0035998958628624678 lr 0.0001
epoch 46399 loss 0.0034557448234409094 lr 0.0001
epoch 46999 loss 0.003413899103179574 lr 0.0001
epoch 47599 loss 0.0033636237494647503 lr 0.0001
epoch 48199 loss 0.0033196844160556793 lr 0.0001
epoch 48799 loss 0.0032772067934274673 lr 0.0001
epoch 49399 loss 0.005071789026260376 lr 0.0001
epoch 49999 loss 0.003195305122062564 lr 0.0001
epoch 50599 loss 0.0031561932992190123 lr 0.0001
epoch 51199 loss 0.0031237697694450617 lr 0.0001
epoch 51799 loss 0.0030810420867055655 lr 0.0001
epoch 52399 loss 0.0030450019985437393 lr 0.0001
epoch 52999 loss 0.0032224052120000124 lr 0.0001
epoch 53599 loss 0.0029758100863546133 lr 0.0001
epoch 54199 loss 0.0029426165856420994 lr 0.0001
epoch 54799 loss 0.0029218210838735104 lr 0.0001
epoch 55399 loss 0.0030868349131196737 lr 0.0001
epoch 55999 loss 0.0028474030550569296 lr 0.0001
```

```
epoch 56599 loss 0.00293141626752913 lr 0.0001
epoch 57199 loss 0.0027881755959242582 lr 0.0001
epoch 57799 loss 0.0027593588456511497 lr 0.0001
epoch 58399 loss 0.0035875362809747458 lr 0.0001
epoch 58999 loss 0.0027041074354201555 lr 0.0001
epoch 59599 loss 0.002677340991795063 lr 0.0001
epoch 60199 loss 0.002651260467246175 lr 0.0001
epoch 60799 loss 0.0026256274431943893 lr 0.0001
epoch 61399 loss 0.0026487053837627172 lr 0.0001
epoch 61999 loss 0.002610068768262863 lr 0.0001
epoch 62599 loss 0.0025829379446804523 lr 0.0001
epoch 63199 loss 0.0025297575630247593 lr 0.0001
epoch 63799 loss 0.0025054726283997297 lr 0.0001
epoch 64399 loss 0.002494345884770155 lr 0.0001
epoch 64999 loss 0.00246035004965961 lr 0.0001
epoch 65599 loss 0.002438570139929652 lr 0.0001
epoch 66199 loss 0.002427855972200632 lr 0.0001
epoch 66799 loss 0.00239601731300354 lr 0.0001
epoch 67399 loss 0.00237540272064507 lr 0.0001
epoch 67999 loss 0.0023564747534692287 lr 0.0001
epoch 68599 loss 0.0023353302385658026 lr 0.0001
epoch 69199 loss 0.002316216239705682 lr 0.0001
epoch 69799 loss 0.0023100716061890125 lr 0.0001
epoch 70399 loss 0.0022778287529945374 lr 0.0001
epoch 70999 loss 0.002310990821570158 lr 0.0001
epoch 71599 loss 0.0022413653787225485 lr 0.0001
epoch 72199 loss 0.002300693653523922 lr 0.0001
epoch 72799 loss 0.002206094330176711 lr 0.0001
epoch 73399 loss 0.002788717160001397 lr 0.0001
epoch 73999 loss 0.0021718856878578663 lr 0.0001
epoch 74599 loss 0.0021554045379161835 lr 0.0001
epoch 75199 loss 0.0021391520276665688 lr 0.0001
epoch 75799 loss 0.0021236229222267866 lr 0.0001
epoch 76399 loss 0.0021071420051157475 lr 0.0001
epoch 76999 loss 0.0020916739013046026 lr 0.0001
epoch 77599 loss 0.002076211152598262 lr 0.0001
epoch 78199 loss 0.0020615295507013798 lr 0.0001
epoch 78799 loss 0.002046682871878147 lr 0.0001
epoch 79399 loss 0.0020320676267147064 lr 0.0001
epoch 79999 loss 0.0020177513360977173 lr 0.0001
epoch 80599 loss 0.0020032981410622597 lr 0.0001
epoch 81199 loss 0.0019920000340789557 lr 0.0001
epoch 81799 loss 0.0019756434485316277 lr 0.0001
epoch 82399 loss 0.0019624591805040836 lr 0.0001
epoch 82999 loss 0.0019493341678753495 lr 0.0001
epoch 83599 loss 0.0020454435143619776 lr 0.0001
epoch 84199 loss 0.001923150266520679 lr 0.0001
epoch 84799 loss 0.0019667502492666245 lr 0.0001
epoch 85399 loss 0.0018977278377860785 lr 0.0001
epoch 85999 loss 0.001885262201540172 lr 0.0001
epoch 86599 loss 0.0018732005264610052 lr 0.0001
epoch 87199 loss 0.001861050259321928 lr 0.0001
epoch 87799 loss 0.0018542443867772818 lr 0.0001
epoch 88399 loss 0.0018375176005065441 lr 0.0001
epoch 88999 loss 0.001825875835493207 lr 0.0001
epoch 89599 loss 0.0022001394536346197 lr 0.0001
epoch 90199 loss 0.0018039249116554856 lr 0.0001
epoch 90799 loss 0.0017920236568897963 lr 0.0001
epoch 91399 loss 0.0017810099525377154 lr 0.0001
epoch 91999 loss 0.0017700957832857966 lr 0.0001
epoch 92599 loss 0.0017601593863219023 lr 0.0001
epoch 93199 loss 0.0017967153107747436 lr 0.0001
epoch 93799 loss 0.001738276332616806 lr 0.0001
epoch 94399 loss 0.001730455318465829 lr 0.0001
epoch 94999 loss 0.0017177388072013855 lr 0.0001
epoch 95599 loss 0.0017143116565421224 lr 0.0001
epoch 96199 loss 0.0016976107144728303 lr 0.0001
epoch 96799 loss 0.0017003813991323113 lr 0.0001
epoch 97399 loss 0.0016780077712610364 lr 0.0001
epoch 97999 loss 0.0016686932649463415 lr 0.0001
epoch 98599 loss 0.001739339786581695 lr 0.0001
epoch 99199 loss 0.0016495477175340056 lr 0.0001
epoch 99799 loss 0.0016402378678321838 lr 0.0001
epoch 100399 loss 0.0016310494393110275 lr 0.0001
epoch 100999 loss 0.001622187439352274 lr 0.0001
epoch 101599 loss 0.001612956402823329 lr 0.0001
epoch 102199 loss 0.0016041112830862403 lr 0.0001
epoch 102799 loss 0.0016163036925718188 lr 0.0001
epoch 103399 loss 0.0015866777393966913 lr 0.0001
epoch 103999 loss 0.0015781496185809374 lr 0.0001
epoch 104599 loss 0.001569615793414414 lr 0.0001
epoch 105199 loss 0.0015646298415958881 lr 0.0001
epoch 105799 loss 0.0015531186945736408 lr 0.0001
epoch 106399 loss 0.0023520218674093485 lr 0.0001
epoch 106999 loss 0.0015385414008051157 lr 0.0001
epoch 107599 loss 0.0015288959257304668 lr 0.0001
epoch 108199 loss 0.0015866667963564396 lr 0.0001
epoch 108799 loss 0.0015131245600059628 lr 0.0001
epoch 109399 loss 0.0015053896931931376 lr 0.0001
epoch 109999 loss 0.001941040507517755 lr 0.0001
epoch 110599 loss 0.0014916375512257218 lr 0.0001
epoch 111199 loss 0.001482621300965547 6 lr 0.0001
epoch 111799 loss 0.0014751895796507597 lr 0.0001
epoch 112399 loss 0.0014680471504107118 lr 0.0001
```

```
epoch 112999 loss 0.001464011613279581 lr 0.0001
epoch 113599 loss 0.0014547195751219988 lr 0.0001
epoch 114199 loss 0.0014463659608736634 lr 0.0001
epoch 114799 loss 0.0014866904821246862 lr 0.0001
epoch 115399 loss 0.0014324000803753734 lr 0.0001
epoch 115999 loss 0.0014254350680857897 lr 0.0001
epoch 116599 loss 0.0014198088319972157 lr 0.0001
epoch 117199 loss 0.002009305637329817 lr 0.0001
epoch 117799 loss 0.0014051321195438504 lr 0.0001
epoch 118399 loss 0.0013985410332679749 lr 0.0001
epoch 118999 loss 0.0013918845215812325 lr 0.0001
epoch 119599 loss 0.0013854358112439513 lr 0.0001
epoch 120199 loss 0.0013788789510726929 lr 0.0001
epoch 120799 loss 0.0013732619117945433 lr 0.0001
epoch 121399 loss 0.0013662598794326186 lr 0.0001
epoch 121999 loss 0.0013599700760096312 lr 0.0001
epoch 122599 loss 0.0013555503683164716 lr 0.0001
epoch 123199 loss 0.001347511773929 lr 0.0001
epoch 123799 loss 0.0013416698202490807 lr 0.0001
epoch 124399 loss 0.0013355627888813615 lr 0.0001
epoch 124999 loss 0.0013294455129653215 lr 0.0001
epoch 125599 loss 0.0013234803918749094 lr 0.0001
epoch 126199 loss 0.0013383653713390231 lr 0.0001
epoch 126799 loss 0.0013117885682731867 lr 0.0001
epoch 127399 loss 0.0013060809578746557 lr 0.0001
epoch 127999 loss 0.001300376490689814 lr 0.0001
epoch 128599 loss 0.001294737565331161 lr 0.0001
epoch 129199 loss 0.0012890665093436837 lr 0.0001
epoch 129799 loss 0.0013688361505046487 lr 0.0001
epoch 130399 loss 0.0012779932003468275 lr 0.0001
epoch 130999 loss 0.0012725169071927667 lr 0.0001
epoch 131599 loss 0.0012671281583607197 lr 0.0001
epoch 132199 loss 0.0012616491876542568 lr 0.0001
epoch 132799 loss 0.0016163669060915709 lr 0.0001
epoch 133399 loss 0.0012522912584245205 lr 0.0001
epoch 133999 loss 0.0012459547724574804 lr 0.0001
epoch 134599 loss 0.0012405159650370479 lr 0.0001
epoch 135199 loss 0.001235472853295505 lr 0.0001
epoch 135799 loss 0.0013481288915500045 lr 0.0001
epoch 136399 loss 0.001225199201144278 lr 0.0001
epoch 136999 loss 0.0038420718628913164 lr 0.0001
epoch 137599 loss 0.0012151392875239253 lr 0.0001
epoch 138199 loss 0.0012101340107619762 lr 0.0001
epoch 138799 loss 0.001285574515350163 lr 0.0001
epoch 139399 loss 0.0012015477987006307 lr 0.0001
epoch 139999 loss 0.0011955127120018005 lr 0.0001
epoch 140599 loss 0.001190667157061398 lr 0.0001
epoch 141199 loss 0.001490274560637772 lr 0.0001
epoch 141799 loss 0.001181192696094513 lr 0.0001
epoch 142399 loss 0.0011764556402340531 lr 0.0001
epoch 142999 loss 0.0012248312123119831 lr 0.0001
epoch 143599 loss 0.0011671691900119185 lr 0.0001
epoch 144199 loss 0.001278382376767695 lr 0.0001
epoch 144799 loss 0.0011581832077354193 lr 0.0001
epoch 145399 loss 0.0011534699006006122 lr 0.0001
epoch 145999 loss 0.0011516594095155597 lr 0.0001
epoch 146599 loss 0.0011570571223273873 lr 0.0001
epoch 147199 loss 0.0011458457447588444 lr 0.0001
epoch 147799 loss 0.0011357878101989627 lr 0.0001
epoch 148399 loss 0.0011314069852232933 lr 0.0001
epoch 148999 loss 0.0011429133592173457 lr 0.0001
epoch 149599 loss 0.0011228211224079132 lr 0.0001
epoch 150199 loss 0.0011185153853148222 lr 0.0001
epoch 150799 loss 0.0011463185073807836 lr 0.0001
epoch 151399 loss 0.0011101074051111937 lr 0.0001
epoch 151999 loss 0.00110632402356714 lr 0.0001
epoch 152599 loss 0.0011017845245078206 lr 0.0001
epoch 153199 loss 0.001097709173336625 lr 0.0001
epoch 153799 loss 0.0010942226508632302 lr 0.0001
epoch 154399 loss 0.0010894794249907136 lr 0.0001
epoch 154999 loss 0.0011140677379444242 lr 0.0001
epoch 155599 loss 0.0010819155722856522 lr 0.0001
epoch 156199 loss 0.001077463966794312 lr 0.0001
epoch 156799 loss 0.0010771432425826788 lr 0.0001
epoch 157399 loss 0.0010696412064135075 lr 0.0001
epoch 157999 loss 0.0010732844239100814 lr 0.0001
epoch 158599 loss 0.001061866176314652 lr 0.0001
epoch 159199 loss 0.0012138312449678779 lr 0.0001
epoch 159799 loss 0.0010541722876951098 lr 0.0001
epoch 160399 loss 0.0010508238337934017 lr 0.0001
epoch 160999 loss 0.0010466146050021052 lr 0.0001
epoch 161599 loss 0.001063227653503418 lr 0.0001
epoch 162199 loss 0.0010391969699412584 lr 0.0001
epoch 162799 loss 0.0010355054400861263 lr 0.0001
epoch 163399 loss 0.0010326997144147754 lr 0.0001
epoch 163999 loss 0.0010320238070562482 lr 0.0001
epoch 164599 loss 0.001027878257445991 lr 0.0001
epoch 165199 loss 0.0010210676118731499 lr 0.0001
epoch 165799 loss 0.0012441660510376096 lr 0.0001
epoch 166399 loss 0.0010139635996893048 lr 0.0001
epoch 166999 loss 0.0013603824190795422 lr 0.0001
epoch 167599 loss 0.001006987993605435 lr 0.0001
epoch 168199 loss 0.001003469224087894 lr 0.0001
epoch 168799 loss 0.0010008731624111533 lr 0.0001
```

```
epoch 169399 loss 0.0009965953649953008 lr 0.0001
epoch 169999 loss 0.0012297882931306958 lr 0.0001
epoch 170599 loss 0.000989942462183535 lr 0.0001
epoch 171199 loss 0.0009865202009677887 lr 0.0001
epoch 171799 loss 0.0009831966599449515 lr 0.0001
epoch 172399 loss 0.0009805442532524467 lr 0.0001
epoch 172999 loss 0.0009766032453626394 lr 0.0001
epoch 173599 loss 0.0009758672676980495 lr 0.0001
epoch 174199 loss 0.0009701382950879633 lr 0.0001
epoch 174799 loss 0.0011366343824192882 lr 0.0001
epoch 175399 loss 0.0009636631002649665 lr 0.0001
epoch 175999 loss 0.0009614520822651684 lr 0.0001
epoch 176599 loss 0.0009573090355843306 lr 0.0001
epoch 177199 loss 0.0009553582640364766 lr 0.0001
epoch 177799 loss 0.0009510974632576108 lr 0.0001
epoch 178399 loss 0.0009479818399995565 lr 0.0001
epoch 178999 loss 0.0009463885799050331 lr 0.0001
epoch 179599 loss 0.0009418114786967635 lr 0.0001
epoch 180199 loss 0.0009423771407455206 lr 0.0001
epoch 180799 loss 0.0009357422241009772 lr 0.0001
epoch 181399 loss 0.0009737479267641902 lr 0.0001
epoch 181999 loss 0.0009297723299823701 lr 0.0001
epoch 182599 loss 0.0009657627670094371 lr 0.0001
epoch 183199 loss 0.000923803832847625 lr 0.0001
epoch 183799 loss 0.0026553887873888016 lr 0.0001
epoch 184399 loss 0.000917960365768522 lr 0.0001
epoch 184999 loss 0.0009150364785455167 lr 0.0001
epoch 185599 loss 0.0009231722797267139 lr 0.0001
epoch 186199 loss 0.0009093124535866082 lr 0.0001
epoch 186799 loss 0.0010348663199692965 lr 0.0001
epoch 187399 loss 0.0009036569390445948 lr 0.0001
epoch 187999 loss 0.0009132901905104518 lr 0.0001
epoch 188599 loss 0.0008980510174296796 lr 0.0001
epoch 189199 loss 0.0008952510543167591 lr 0.0001
epoch 189799 loss 0.0008935104706324637 lr 0.0001
epoch 190399 loss 0.0008899305248633027 lr 0.0001
epoch 190999 loss 0.0008871405152603984 lr 0.0001
epoch 191599 loss 0.0008971292991191149 lr 0.0001
epoch 192199 loss 0.000881611427757889 lr 0.0001
epoch 192799 loss 0.0009558864985592663 lr 0.0001
epoch 193399 loss 0.0008762393845245242 lr 0.0001
epoch 193999 loss 0.0009000951540656388 lr 0.0001
epoch 194599 loss 0.0008709533722139895 lr 0.0001
epoch 195199 loss 0.0008689198875799775 lr 0.0001
epoch 195799 loss 0.0008657217840664089 lr 0.0001
epoch 196399 loss 0.0010799571173265576 lr 0.0001
epoch 196999 loss 0.0008644809713587165 lr 0.0001
epoch 197599 loss 0.0018203334184363484 lr 0.0001
epoch 198199 loss 0.0008554608793929219 lr 0.0001
epoch 198799 loss 0.0008685323991812766 lr 0.0001
epoch 199399 loss 0.000850404379889369 lr 0.0001
epoch 199999 loss 0.0008479232201352715 lr 0.0001
epoch 200599 loss 0.002981209196150303 lr 0.0001
epoch 201199 loss 0.0008431891910731792 lr 0.0001
epoch 201799 loss 0.000845744158141315 lr 0.0001
epoch 202399 loss 0.0008380587096326053 lr 0.0001
epoch 202999 loss 0.0008452783804386854 lr 0.0001
epoch 203599 loss 0.0008332821889780462 lr 0.0001
epoch 204199 loss 0.0008308352553285658 lr 0.0001
epoch 204799 loss 0.0008286162046715617 lr 0.0001
epoch 205399 loss 0.0008266647928394377 lr 0.0001
epoch 205999 loss 0.0022595690097659826 lr 0.0001
epoch 206599 loss 0.0008213974069803953 lr 0.0001
epoch 207199 loss 0.0008190243388526142 lr 0.0001
epoch 207799 loss 0.0008167411433532834 lr 0.0001
epoch 208399 loss 0.0008145829197019339 lr 0.0001
epoch 208999 loss 0.000812091922853142 lr 0.0001
epoch 209599 loss 0.000821189780253917 lr 0.0001
epoch 210199 loss 0.0008075269288383424 lr 0.0001
epoch 210799 loss 0.0008053557830862701 lr 0.0001
epoch 211399 loss 0.0008459774544462562 lr 0.0001
epoch 211999 loss 0.0008008199511095881 lr 0.0001
epoch 212599 loss 0.0007986972341313958 lr 0.0001
epoch 213199 loss 0.0008551576174795628 lr 0.0001
epoch 213799 loss 0.0007941818330436945 lr 0.0001
epoch 214399 loss 0.0007950672297738492 lr 0.0001
epoch 214999 loss 0.0008563206065446138 lr 0.0001
epoch 215599 loss 0.0007876862655393779 lr 0.0001
epoch 216199 loss 0.0007855308358557522 lr 0.0001
epoch 216799 loss 0.0007834986317902803 lr 0.0001
epoch 217399 loss 0.0007819927996024489 lr 0.0001
epoch 217999 loss 0.0009484554175287485 lr 0.0001
epoch 218599 loss 0.0007771301898173988 lr 0.0001
epoch 219199 loss 0.0007749435608275235 lr 0.0001
epoch 219799 loss 0.0026204281020909548 lr 0.0001
epoch 220399 loss 0.0007709958590567112 lr 0.0001
epoch 220999 loss 0.0007688080659136176 lr 0.0001
epoch 221599 loss 0.0007670892518945038 lr 0.0001
epoch 222199 loss 0.0007660521660000086 lr 0.0001
epoch 222799 loss 0.0007626798469573259 lr 0.0001
epoch 223399 loss 0.0007655178778804839 lr 0.0001
epoch 223999 loss 0.0007586258579976857 lr 0.0001
epoch 224599 loss 0.0007567163556814194 lr 0.0001
epoch 225199 loss 0.0007547594723291695 lr 0.0001
```

```
epoch 225799 loss 0.0007639449904672801 lr 0.0001
epoch 226399 loss 0.0007515932666137815 lr 0.0001
epoch 226999 loss 0.0007496136822737753 lr 0.0001
epoch 227599 loss 0.0007604198763146996 lr 0.0001
epoch 228199 loss 0.0007450474658980966 lr 0.0001
epoch 228799 loss 0.0007478661718778312 lr 0.0001
epoch 229399 loss 0.0007412493578158319 lr 0.0001
epoch 229999 loss 0.0012809529434889555 lr 0.0001
epoch 230599 loss 0.0007374132983386517 lr 0.0001
epoch 231199 loss 0.0007827709196135402 lr 0.0001
epoch 231799 loss 0.0007339036674238741 lr 0.0001
epoch 232399 loss 0.0007336960989050567 lr 0.0001
epoch 232999 loss 0.0007299352437257767 lr 0.0001
epoch 233599 loss 0.0007366508943960071 lr 0.0001
epoch 234199 loss 0.0007262870785780251 lr 0.0001
epoch 234799 loss 0.0008968001930043101 lr 0.0001
epoch 235399 loss 0.0007237930549308658 lr 0.0001
epoch 235999 loss 0.0007220976403914392 lr 0.0001
epoch 236599 loss 0.000719347211997956 lr 0.0001
epoch 237199 loss 0.0007173082558438182 lr 0.0001
epoch 237799 loss 0.0007188196759670973 lr 0.0001
epoch 238399 loss 0.0007137920474633574 lr 0.0001
epoch 238999 loss 0.0007120738737285137 lr 0.0001
epoch 239599 loss 0.0007102854433469474 lr 0.0001
epoch 240199 loss 0.0007086432306095958 lr 0.0001
epoch 240799 loss 0.000714843743480742 lr 0.0001
epoch 241399 loss 0.0032962544355541468 lr 0.0001
epoch 241999 loss 0.0007057130569592118 lr 0.0001
epoch 242599 loss 0.0007017557509243488 lr 0.0001
epoch 243199 loss 0.0007241873536258936 lr 0.0001
epoch 243799 loss 0.0017580848652869463 lr 0.0001
epoch 244399 loss 0.0006967608351260424 lr 0.0001
epoch 244999 loss 0.00163069658447057 lr 0.0001
epoch 245599 loss 0.0006935118581168354 lr 0.0001
epoch 246199 loss 0.0006917886785231531 lr 0.0001
epoch 246799 loss 0.0006904805777594447 lr 0.0001
epoch 247399 loss 0.0006889217766001821 lr 0.0001
epoch 247999 loss 0.0006869558710604906 lr 0.0001
epoch 248599 loss 0.000685345206875354 lr 0.0001
epoch 249199 loss 0.0011788546107709408 lr 0.0001
epoch 249799 loss 0.0006821558927185833 lr 0.0001
epoch 250399 loss 0.000680559198372066 lr 0.0001
epoch 250999 loss 0.0006790586048737168 lr 0.0001
epoch 251599 loss 0.0006774788489565253 lr 0.0001
epoch 252199 loss 0.0006759076495654881 lr 0.0001
epoch 252799 loss 0.000674365262966603 lr 0.0001
epoch 253399 loss 0.0006728183943778276 lr 0.0001
epoch 253999 loss 0.0006714300252497196 lr 0.0001
epoch 254599 loss 0.0006698215729556978 lr 0.0001
epoch 255199 loss 0.001226587686687708 lr 0.0001
epoch 255799 loss 0.0006667575798928738 lr 0.0001
epoch 256399 loss 0.0006652608863078058 lr 0.0001
epoch 256999 loss 0.0006637947517447174 lr 0.0001
epoch 257599 loss 0.0006623085937462747 lr 0.0001
epoch 258199 loss 0.0006608347175642848 lr 0.0001
epoch 258799 loss 0.000659329118207097 lr 0.0001
epoch 259399 loss 0.0014005739940330386 lr 0.0001
epoch 259999 loss 0.0006566359661519527 lr 0.0001
epoch 260599 loss 0.0006549846148118377 lr 0.0001
epoch 261199 loss 0.0006535059656016529 lr 0.0001
epoch 261799 loss 0.0006523372721858323 lr 0.0001
epoch 262399 loss 0.0006523207412101328 lr 0.0001
epoch 262999 loss 0.0006492239772342145 lr 0.0001
epoch 263599 loss 0.0006480090087279677 lr 0.0001
epoch 264199 loss 0.0006463773897849023 lr 0.0001
epoch 264799 loss 0.00064626190578565 lr 0.0001
epoch 265399 loss 0.0006435881368815899 lr 0.0001
epoch 265999 loss 0.0006421908619813621 lr 0.0001
epoch 266599 loss 0.0008365173707716167 lr 0.0001
epoch 267199 loss 0.0006585791707038879 lr 0.0001
epoch 267799 loss 0.0006380189443007112 lr 0.0001
epoch 268399 loss 0.000636720797047019 lr 0.0001
epoch 268999 loss 0.0008950792253017426 lr 0.0001
epoch 269599 loss 0.0006338914972729981 lr 0.0001
epoch 270199 loss 0.000632556970231235 lr 0.0001
epoch 270799 loss 0.0006312167388387024 lr 0.0001
epoch 271399 loss 0.0006298767984844744 lr 0.0001
epoch 271999 loss 0.000628718757070601 lr 0.0001
epoch 272599 loss 0.004621272440999746 lr 0.0001
epoch 273199 loss 0.0006269602454267442 lr 0.0001
epoch 273799 loss 0.000624622218310833 lr 0.0001
epoch 274399 loss 0.0006232867599464953 lr 0.0001
epoch 274999 loss 0.0006226326222531497 lr 0.0001
epoch 275599 loss 0.0006206603138707578 lr 0.0001
epoch 276199 loss 0.000619396276306361 lr 0.0001
epoch 276799 loss 0.0006181286880746484 lr 0.0001
epoch 277399 loss 0.0006423341692425311 lr 0.0001
epoch 277999 loss 0.000615592289250344 lr 0.0001
epoch 278599 loss 0.0006143011851236224 lr 0.0001
epoch 279199 loss 0.0009945151396095753 lr 0.0001
epoch 279799 loss 0.0006178194307722151 lr 0.0001
epoch 280399 loss 0.0006105161155574024 lr 0.0001
epoch 280999 loss 0.0006364659639075398 lr 0.0001
epoch 281599 loss 0.0006080017192289233 lr 0.0001
```

```
epoch 282199 loss 0.0006800142000429332 lr 0.0001
epoch 282799 loss 0.0006056584534235299 lr 0.0001
epoch 283399 loss 0.0006043227040208876 lr 0.0001
epoch 283999 loss 0.0006031218799762428 lr 0.0001
epoch 284599 loss 0.0006051408126950264 lr 0.0001
epoch 285199 loss 0.0006007038173265755 lr 0.0001
epoch 285799 loss 0.0005994775565341115 lr 0.0001
epoch 286399 loss 0.000598271784838289 lr 0.0001
epoch 286999 loss 0.000599535705987364 lr 0.0001
epoch 287599 loss 0.0005958889378234744 lr 0.0001
epoch 288199 loss 0.0007471393910236657 lr 0.0001
epoch 288799 loss 0.0005940305418334901 lr 0.0001
epoch 289399 loss 0.0005924143479205668 lr 0.0001
epoch 289999 loss 0.0005912024644203484 lr 0.0001
epoch 290599 loss 0.000611347088124603 lr 0.0001
epoch 291199 loss 0.0005888922023586929 lr 0.0001
epoch 291799 loss 0.0005877499934285879 lr 0.0001
epoch 292399 loss 0.000586682406719774 lr 0.0001
epoch 292999 loss 0.0007185132126323879 lr 0.0001
epoch 293599 loss 0.0005843777908012271 lr 0.0001
epoch 294199 loss 0.0005831716698594391 lr 0.0001
epoch 294799 loss 0.0009258983773179352 lr 0.0001
epoch 295399 loss 0.0005810022121295333 lr 0.0001
epoch 295999 loss 0.0005797826452180743 lr 0.0001
epoch 296599 loss 0.000711894768755883 lr 0.0001
epoch 297199 loss 0.0005811086157336831 lr 0.0001
epoch 297799 loss 0.0005780308856628835 lr 0.0001
epoch 298399 loss 0.0005753650330007076 lr 0.0001
epoch 298999 loss 0.0005744104273617268 lr 0.0001
epoch 299599 loss 0.0005732096615247428 lr 0.0001
epoch 300199 loss 0.0005720920744352043 lr 0.0001
epoch 300799 loss 0.0006125787040218711 lr 0.0001
epoch 301399 loss 0.0005699898465536535 lr 0.0001
epoch 301999 loss 0.0005688653909601271 lr 0.0001
epoch 302599 loss 0.000579701503738761 lr 0.0001
epoch 303199 loss 0.0005667409859597683 lr 0.0001
epoch 303799 loss 0.0005656574503518641 lr 0.0001
epoch 304399 loss 0.0005646076169796288 lr 0.0001
epoch 304999 loss 0.0005639560404233634 lr 0.0001
epoch 305599 loss 0.0005625433987006545 lr 0.0001
epoch 306199 loss 0.0005614871624857187 lr 0.0001
epoch 306799 loss 0.0009186008246615529 lr 0.0001
epoch 307399 loss 0.000559771026019007 lr 0.0001
epoch 307999 loss 0.0005583860329352319 lr 0.0001
epoch 308599 loss 0.000557350751478225 lr 0.0001
epoch 309199 loss 0.0005744832451455295 lr 0.0001
epoch 309799 loss 0.0005552941001951694 lr 0.0001
epoch 310399 loss 0.0005551499198190868 lr 0.0001
epoch 310999 loss 0.0005532866343855858 lr 0.0001
epoch 311599 loss 0.000552243844140321 lr 0.0001
epoch 312199 loss 0.0005512614734470844 lr 0.0001
epoch 312799 loss 0.0006876098341308534 lr 0.0001
epoch 313399 loss 0.0005492622731253505 lr 0.0001
epoch 313999 loss 0.0005482568521983922 lr 0.0001
epoch 314599 loss 0.0005477251834236085 lr 0.0001
epoch 315199 loss 0.0005463139968924224 lr 0.0001
epoch 315799 loss 0.0005452994373627007 lr 0.0001
epoch 316399 loss 0.0005461893742904067 lr 0.0001
epoch 316999 loss 0.0005433986079879105 lr 0.0001
epoch 317599 loss 0.0005423938855528831 lr 0.0001
epoch 318199 loss 0.0005711683188565075 lr 0.0001
epoch 318799 loss 0.0005404769908636808 lr 0.0001
epoch 319399 loss 0.0005395396728999913 lr 0.0001
epoch 319999 loss 0.0005385514814406633 lr 0.0001
epoch 320599 loss 0.000537631509359926 lr 0.0001
epoch 321199 loss 0.0005366636905819178 lr 0.0001
epoch 321799 loss 0.0005357186892069876 lr 0.0001
epoch 322399 loss 0.0005566164036281407 lr 0.0001
epoch 322999 loss 0.0005338393966667354 lr 0.0001
epoch 323599 loss 0.0005329025443643332 lr 0.0001
epoch 324199 loss 0.0010964645771309733 lr 0.0001
epoch 324799 loss 0.0008405018597841263 lr 0.0001
epoch 325399 loss 0.0005301451892592013 lr 0.0001
epoch 325999 loss 0.0006483595934696496 lr 0.0001
epoch 326599 loss 0.0005285093211568892 lr 0.0001
epoch 327199 loss 0.0005274048307910562 lr 0.0001
epoch 327799 loss 0.0005264789797365665 lr 0.0001
epoch 328399 loss 0.0005256101721897721 lr 0.0001
epoch 328999 loss 0.0005246628425084054 lr 0.0001
epoch 329599 loss 0.0006443960010074079 lr 0.0001
epoch 330199 loss 0.0005228887312114239 lr 0.0001
epoch 330799 loss 0.0005219693412072957 lr 0.0001
epoch 331399 loss 0.0005215138080529869 lr 0.0001
epoch 331999 loss 0.00052018923452124 lr 0.0001
epoch 332599 loss 0.00052335683722049 lr 0.0001
epoch 333199 loss 0.0005184580804780126 lr 0.0001
epoch 333799 loss 0.0005175585392862558 lr 0.0001
epoch 334399 loss 0.0005760063068009913 lr 0.0001
epoch 334999 loss 0.0005163319874554873 lr 0.0001
epoch 335599 loss 0.0005200225277803838 lr 0.0001
epoch 336199 loss 0.0005141678848303854 lr 0.0001
epoch 336799 loss 0.0005132173537276685 lr 0.0001
epoch 337399 loss 0.000678871525451541 lr 0.0001
epoch 337999 loss 0.0006305938586592674 lr 0.0001
```

```
epoch 338599 loss 0.0005106645403429866 lr 0.0001
epoch 339199 loss 0.000533122627530247 lr 0.0001
epoch 339799 loss 0.0005089818732813001 lr 0.0001
epoch 340399 loss 0.0005085525335744023 lr 0.0001
epoch 340999 loss 0.00050753029063344 lr 0.0001
epoch 341599 loss 0.0009375774534419179 lr 0.0001
epoch 342199 loss 0.0005056179943494499 lr 0.0001
epoch 342799 loss 0.0011454191990196705 lr 0.0001
epoch 343399 loss 0.0005041163531132042 lr 0.0001
epoch 343999 loss 0.0005085391458123922 lr 0.0001
epoch 344599 loss 0.0005022955592721701 lr 0.0001
epoch 345199 loss 0.0005014960188418627 lr 0.0001
epoch 345799 loss 0.0005006666760891676 lr 0.0001
epoch 346399 loss 0.0004998953081667423 lr 0.0001
epoch 346999 loss 0.0004990436136722565 lr 0.0001
epoch 347599 loss 0.0004983709659427404 lr 0.0001
epoch 348199 loss 0.0004989359877072275 lr 0.0001
epoch 348799 loss 0.0004966185078956187 lr 0.0001
epoch 349399 loss 0.0004958812496624887 lr 0.0001
epoch 349999 loss 0.0005213727708905935 lr 0.0001
epoch 350599 loss 0.004030040930956602 lr 0.0001
epoch 351199 loss 0.000493968662340194 lr 0.0001
epoch 351799 loss 0.0004926391411572695 lr 0.0001
epoch 352399 loss 0.0004931920557282865 lr 0.0001
epoch 352999 loss 0.0004924433887936175 lr 0.0001
epoch 353599 loss 0.000490356411319226 lr 0.0001
epoch 354199 loss 0.0004907840047962964 lr 0.0001
epoch 354799 loss 0.0004887485411018133 lr 0.0001
epoch 355399 loss 0.0004890724085271358 lr 0.0001
epoch 355999 loss 0.0004872161371167749 lr 0.0001
epoch 356599 loss 0.0005424373666755855 lr 0.0001
epoch 357199 loss 0.00048570020589977503 lr 0.0001
epoch 357799 loss 0.00048492790665477514 lr 0.0001
epoch 358399 loss 0.0005057439557276666 lr 0.0001
epoch 358999 loss 0.00048340222565457225 lr 0.0001
epoch 359599 loss 0.0004829097306355834 lr 0.0001
epoch 360199 loss 0.0004819456662517041 lr 0.0001
epoch 360799 loss 0.0004811459220945835 lr 0.0001
epoch 361399 loss 0.0004805508360732347 lr 0.0001
epoch 361999 loss 0.0004796457360498607 lr 0.0001
epoch 362599 loss 0.0005991465295664966 lr 0.0001
epoch 363199 loss 0.0004782312025781721 lr 0.0001
epoch 363799 loss 0.0008372975280508399 lr 0.0001
epoch 364399 loss 0.00047673442168161273 lr 0.0001
epoch 364999 loss 0.0004759494331665337 lr 0.0001
epoch 365599 loss 0.0004804643976967782 lr 0.0001
epoch 366199 loss 0.0004745075129903853 lr 0.0001
epoch 366799 loss 0.0007348109502345324 lr 0.0001
epoch 367399 loss 0.0004731023800559342 lr 0.0001
epoch 367999 loss 0.00047234000521712005 lr 0.0001
epoch 368599 loss 0.0004889925476163626 lr 0.0001
epoch 369199 loss 0.0004709439817816019 lr 0.0001
epoch 369799 loss 0.00047237329999916255 lr 0.0001
epoch 370399 loss 0.0004694905655924231 lr 0.0001
epoch 370999 loss 0.0004687857290264219 lr 0.0001
epoch 371599 loss 0.0004682254802901298 lr 0.0001
epoch 372199 loss 0.00046739360550418496 lr 0.0001
epoch 372799 loss 0.0004673583316616714 lr 0.0001
epoch 373399 loss 0.00046595753519795835 lr 0.0001
epoch 373999 loss 0.0012191111454740167 lr 0.0001
epoch 374599 loss 0.0004668016918003559 lr 0.0001
epoch 375199 loss 0.0004639076651073992 lr 0.0001
epoch 375799 loss 0.00046323498827405274 lr 0.0001
epoch 376399 loss 0.0004630116163752973 lr 0.0001
epoch 376999 loss 0.0004618375969585031 lr 0.0001
epoch 377599 loss 0.0008686744840815663 lr 0.0001
epoch 378199 loss 0.00046245657722465694 lr 0.0001
epoch 378799 loss 0.0004597723891492933 lr 0.0001
epoch 379399 loss 0.00046397067490033805 lr 0.0001
epoch 379999 loss 0.00045843416592106223 lr 0.0001
epoch 380599 loss 0.0008067372255027294 lr 0.0001
epoch 381199 loss 0.00045708558172918856 lr 0.0001
epoch 381799 loss 0.00045646430226042867 lr 0.0001
epoch 382399 loss 0.000455744651844725 lr 0.0001
epoch 382999 loss 0.00045745191164314747 lr 0.0001
epoch 383599 loss 0.00045445398427546024 lr 0.0001
epoch 384199 loss 0.0004537491768132895 lr 0.0001
epoch 384799 loss 0.003872239962220192 lr 0.0001
epoch 385399 loss 0.0004537741478998214 lr 0.0001
epoch 385999 loss 0.0004518096393439919 lr 0.0001
epoch 386599 loss 0.0004511465085670352 lr 0.0001
epoch 387199 loss 0.0005748207913711667 lr 0.0001
epoch 387799 loss 0.0004498204798437655 lr 0.0001
epoch 388399 loss 0.0006799204857088625 lr 0.0001
epoch 388999 loss 0.0004680720157921314 lr 0.0001
epoch 389599 loss 0.0004478993359953165 lr 0.0001
epoch 390199 loss 0.00044727016938850284 lr 0.0001
epoch 390799 loss 0.0004466217942535877 lr 0.0001
epoch 391399 loss 0.00044715034891851246 lr 0.0001
epoch 391999 loss 0.0004453553119674325 lr 0.0001
epoch 392599 loss 0.002855796366930008 lr 0.0001
epoch 393199 loss 0.00044761228491552174 lr 0.0001
epoch 393799 loss 0.0004435033770278096 lr 0.0001
epoch 394399 loss 0.00044284568866714835 lr 0.0001
```

```
epoch 394999 loss 0.0004959466750733554 lr 0.0001
epoch 395599 loss 0.00044160251854918897 lr 0.0001
epoch 396199 loss 0.0007789171067997813 lr 0.0001
epoch 396799 loss 0.00044092664029449224 lr 0.0001
epoch 397399 loss 0.0004397366719786078 lr 0.0001
epoch 397999 loss 0.00048429236630909145 lr 0.0001
epoch 398599 loss 0.0005684943171218038 lr 0.0001
epoch 399199 loss 0.00043797047692351043 lr 0.0001
epoch 399799 loss 0.000437374459579587 lr 0.0001
epoch 400399 loss 0.0004367106012068689 lr 0.0001
epoch 400999 loss 0.00043655469198711216 lr 0.0001
epoch 401599 loss 0.00043550075497478247 lr 0.0001
epoch 402199 loss 0.0036772347521036863 lr 0.0001
epoch 402799 loss 0.002184787765145302 lr 0.0001
epoch 403399 loss 0.0004350380040705204 lr 0.0001
epoch 403999 loss 0.0004909222479909658 lr 0.0001
epoch 404599 loss 0.0004331967211328447 lr 0.0001
epoch 405199 loss 0.00043215026380494237 lr 0.0001
epoch 405799 loss 0.0004313381214160472 lr 0.0001
epoch 406399 loss 0.0004307585768401623 lr 0.0001
epoch 406999 loss 0.00043018217547796667 lr 0.0001
epoch 407599 loss 0.000481863331515342 lr 0.0001
epoch 408199 loss 0.0004291379009373486 lr 0.0001
epoch 408799 loss 0.0004284441820345819 lr 0.0001
epoch 409399 loss 0.00042787299025803804 lr 0.0001
epoch 409999 loss 0.0004272742662578821 lr 0.0001
epoch 410599 loss 0.0004267488548066467 lr 0.0001
epoch 411199 loss 0.00042616971768438816 lr 0.0001
epoch 411799 loss 0.0005317393224686384 lr 0.0001
epoch 412399 loss 0.00042567247874103487 lr 0.0001
epoch 412999 loss 0.0004350515955593437 lr 0.0001
epoch 413599 loss 0.0004238785186316818 lr 0.0001
epoch 414199 loss 0.0004233178333379328 lr 0.0001
epoch 414799 loss 0.00042275237501598895 lr 0.0001
epoch 415399 loss 0.000627092900685966 lr 0.0001
epoch 415999 loss 0.0005054818466305733 lr 0.0001
epoch 416599 loss 0.0004210698534734547 lr 0.0001
epoch 417199 loss 0.0004205159784760326 lr 0.0001
epoch 417799 loss 0.0004199525574222207 lr 0.0001
epoch 418399 loss 0.0004212664207443595 lr 0.0001
epoch 418999 loss 0.0004231779312249273 lr 0.0001
epoch 419599 loss 0.00041830874397419393 lr 0.0001
epoch 420199 loss 0.000419247051468119 lr 0.0001
epoch 420799 loss 0.000422801764216274 lr 0.0001
epoch 421399 loss 0.00041668282938189805 lr 0.0001
epoch 421999 loss 0.00041612060158513486 lr 0.0001
epoch 422599 loss 0.00041558980592526495 lr 0.0001
epoch 423199 loss 0.0004658166435547173 lr 0.0001
epoch 423799 loss 0.0004145448619965464 lr 0.0001
epoch 424399 loss 0.00041430353303439915 lr 0.0001
epoch 424999 loss 0.00041917309863492846 lr 0.0001
epoch 425599 loss 0.00041448380216024816 lr 0.0001
epoch 426199 loss 0.00042237466550432146 lr 0.0001
epoch 426799 loss 0.0004123349499423057 lr 0.0001
epoch 427399 loss 0.0004115592164453119 lr 0.0001
epoch 427999 loss 0.00041078400681726635 lr 0.0001
epoch 428599 loss 0.001014204230159521 lr 0.0001
epoch 429199 loss 0.0010933773592114449 lr 0.0001
epoch 429799 loss 0.0004147759173065424 lr 0.0001
epoch 430399 loss 0.0004090434522368014 lr 0.0001
epoch 430999 loss 0.00040816448745317757 lr 0.0001
epoch 431599 loss 0.0004076570039615035 lr 0.0001
epoch 432199 loss 0.0004127082065679133 lr 0.0001
epoch 432799 loss 0.00040662960964255035 lr 0.0001
epoch 433399 loss 0.0004060920618940145 lr 0.0001
epoch 433999 loss 0.0004168407467659563 lr 0.0001
epoch 434599 loss 0.00041182435234077275 lr 0.0001
epoch 435199 loss 0.0004131869354750961 lr 0.0001
epoch 435799 loss 0.00040428046486340463 lr 0.0001
epoch 436399 loss 0.0004035088641103357 lr 0.0001
epoch 436999 loss 0.0004030052514281124 lr 0.0001
epoch 437599 loss 0.0004270925419405103 lr 0.0001
epoch 438199 loss 0.0004059994244016707 lr 0.0001
epoch 438799 loss 0.0004014859441667795 lr 0.0001
epoch 439399 loss 0.0004010163538623601 lr 0.0001
epoch 439999 loss 0.00040051116957329214 lr 0.0001
epoch 440599 loss 0.0014635538682341576 lr 0.0001
epoch 441199 loss 0.0003995297884102911 lr 0.0001
epoch 441799 loss 0.0003990325203631073 lr 0.0001
epoch 442399 loss 0.0014298231108114123 lr 0.0001
epoch 442999 loss 0.00040063707274384797 lr 0.0001
epoch 443599 loss 0.0003976031730417162 lr 0.0001
epoch 444199 loss 0.0003970542747993022 lr 0.0001
epoch 444799 loss 0.0003965873329434544 lr 0.0001
epoch 445399 loss 0.0003960751637350768 lr 0.0001
epoch 445999 loss 0.00039562489837408066 lr 0.0001
epoch 446599 loss 0.0017447342397645116 lr 0.0001
epoch 447199 loss 0.00039482314605265856 lr 0.0001
epoch 447799 loss 0.0008484896970912814 lr 0.0001
epoch 448399 loss 0.00039645933429710567 lr 0.0001
epoch 448999 loss 0.0003932363470084965 lr 0.0001
epoch 449599 loss 0.0003927050274796784 lr 0.0001
epoch 450199 loss 0.0003930026141460985 lr 0.0001
epoch 450799 loss 0.00039174433914013207 lr 0.0001
```

```
epoch 451399 loss 0.00039132602978497744 lr 0.0001
epoch 451999 loss 0.0008857931825332344 lr 0.0001
epoch 452599 loss 0.00039078042027540505 lr 0.0001
epoch 453199 loss 0.0003915097040589899 lr 0.0001
epoch 453799 loss 0.00040018983418121934 lr 0.0001
epoch 454399 loss 0.0003889090148732066 lr 0.0001
epoch 454999 loss 0.00038847618270665407 lr 0.0001
epoch 455599 loss 0.00038805865915492177 lr 0.0001
epoch 456199 loss 0.00038754669367335737 lr 0.0001
epoch 456799 loss 0.00038709197542630136 lr 0.0001
epoch 457399 loss 0.0037755174562335014 lr 0.0001
epoch 457999 loss 0.00038616242818534374 lr 0.0001
epoch 458599 loss 0.00043257736251689494 lr 0.0001
epoch 459199 loss 0.000402679055696353 lr 0.0001
epoch 459799 loss 0.00038481064257211983 lr 0.0001
epoch 460399 loss 0.00038436095928773284 lr 0.0001
epoch 460999 loss 0.00038419532938860357 lr 0.0001
epoch 461599 loss 0.0003834779199678451 lr 0.0001
epoch 462199 loss 0.00041074634646065533 lr 0.0001
epoch 462799 loss 0.00038282075547613204 lr 0.0001
epoch 463399 loss 0.00038206944009289145 lr 0.0001
epoch 463999 loss 0.00038165124715305865 lr 0.0001
epoch 464599 loss 0.00038123628473840654 lr 0.0001
epoch 465199 loss 0.0003879536525346339 lr 0.0001
epoch 465799 loss 0.00038028915878385305 lr 0.0001
epoch 466399 loss 0.0003798605757765472 lr 0.0001
epoch 466999 loss 0.00039400163223035634 lr 0.0001
epoch 467599 loss 0.00037896280991844833 lr 0.0001
epoch 468199 loss 0.00037850733497180045 lr 0.0001
epoch 468799 loss 0.0003780836414080113 lr 0.0001
epoch 469399 loss 0.0003776949888560921 lr 0.0001
epoch 469999 loss 0.0003771970805246383 lr 0.0001
epoch 470599 loss 0.00037679701927118003 lr 0.0001
epoch 471199 loss 0.00037635944318026304 lr 0.0001
epoch 471799 loss 0.00037592955050058663 lr 0.0001
epoch 472399 loss 0.0003754817880690098 lr 0.0001
epoch 472999 loss 0.0003789659822359681 lr 0.0001
epoch 473599 loss 0.0003746206348296255 lr 0.0001
epoch 474199 loss 0.0003741947002708912 lr 0.0001
epoch 474799 loss 0.0003737795923370868 lr 0.0001
epoch 475399 loss 0.00040575143066234887 lr 0.0001
epoch 475999 loss 0.0003729123855009675 lr 0.0001
epoch 476599 loss 0.0003725052811205387 lr 0.0001
epoch 477199 loss 0.0003720730892382562 lr 0.0001
epoch 477799 loss 0.00040165442624129355 lr 0.0001
epoch 478399 loss 0.00037124776281416416 lr 0.0001
epoch 478999 loss 0.0003708541044034064 lr 0.0001
epoch 479599 loss 0.0004295867111068219 lr 0.0001
epoch 480199 loss 0.00036997924325987697 lr 0.0001
epoch 480799 loss 0.00036960048601031303 lr 0.0001
epoch 481399 loss 0.00037589360726997256 lr 0.0001
epoch 481999 loss 0.000377755262888968 lr 0.0001
epoch 482599 loss 0.00039587021456100047 lr 0.0001
epoch 483199 loss 0.00036831331090070307 lr 0.0001
epoch 483799 loss 0.001055592903867364 lr 0.0001
epoch 484399 loss 0.00038367771776393056 lr 0.0001
epoch 484999 loss 0.0016684618312865496 lr 0.0001
epoch 485599 loss 0.00036626963992603123 lr 0.0001
epoch 486199 loss 0.0003658573841676116 lr 0.0001
epoch 486799 loss 0.00036906119203194976 lr 0.0001
epoch 487399 loss 0.0005374337779358029 lr 0.0001
epoch 487999 loss 0.00036463019205257297 lr 0.0001
epoch 488599 loss 0.000856159022077918 lr 0.0001
epoch 489199 loss 0.0003638640628196299 lr 0.0001
epoch 489799 loss 0.00036347529385238886 lr 0.0001
epoch 490399 loss 0.00036997318966314197 lr 0.0001
epoch 490999 loss 0.0003626379475463182 lr 0.0001
epoch 491599 loss 0.0003622284275479615 lr 0.0001
epoch 492199 loss 0.0005978976842015982 lr 0.0001
epoch 492799 loss 0.00036145211197435856 lr 0.0001
epoch 493399 loss 0.0004438463947735727 lr 0.0001
epoch 493999 loss 0.0003610891290009022 lr 0.0001
epoch 494599 loss 0.0003635162429418415 lr 0.0001
epoch 495199 loss 0.0003598766925279051 lr 0.0001
epoch 495799 loss 0.0003594899899326265 lr 0.0001
epoch 496399 loss 0.00035907907295040786 lr 0.0001
epoch 496999 loss 0.00036486584576778114 lr 0.0001
epoch 497599 loss 0.00035831218701787293 lr 0.0001
epoch 498199 loss 0.0003579370677471161 lr 0.0001
epoch 498799 loss 0.001111219055019319 lr 0.0001
epoch 499399 loss 0.00035717658465728164 lr 0.0001
epoch 499999 loss 0.0005764150409959257 lr 0.0001
```

Perfomr all the epochs, even small problem can be very difficult to solve.

Let us compute a specific instance of the problem! Namely we compute $\boldsymbol{\pi}(\boldsymbol{\mu}_{test})$.

What is the output?

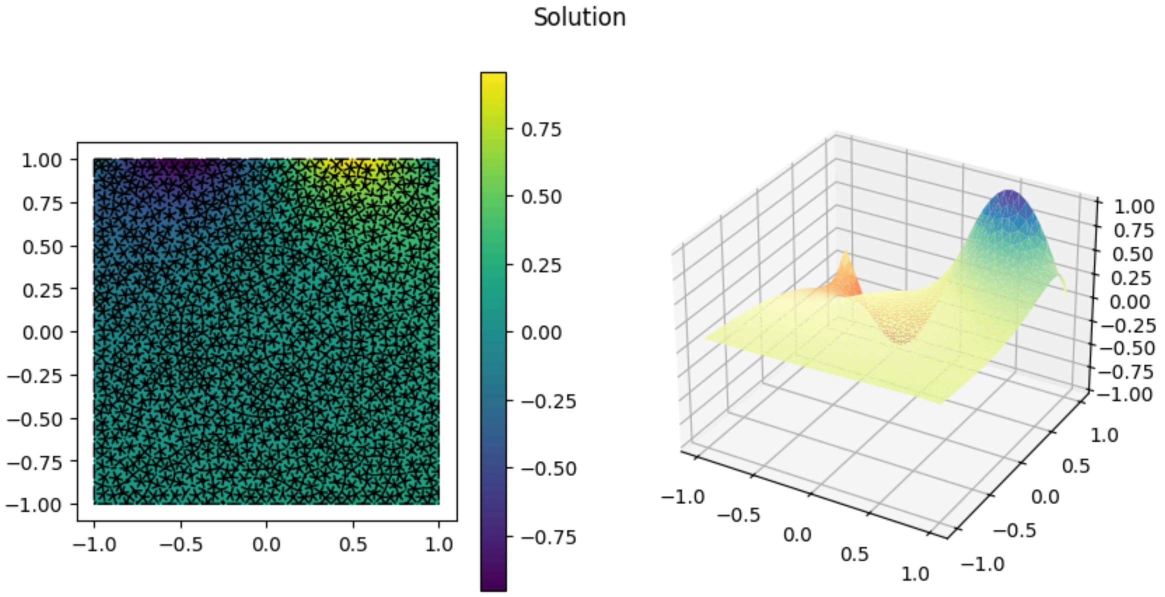
Let us compare it with the full solution.

What do I have to do?

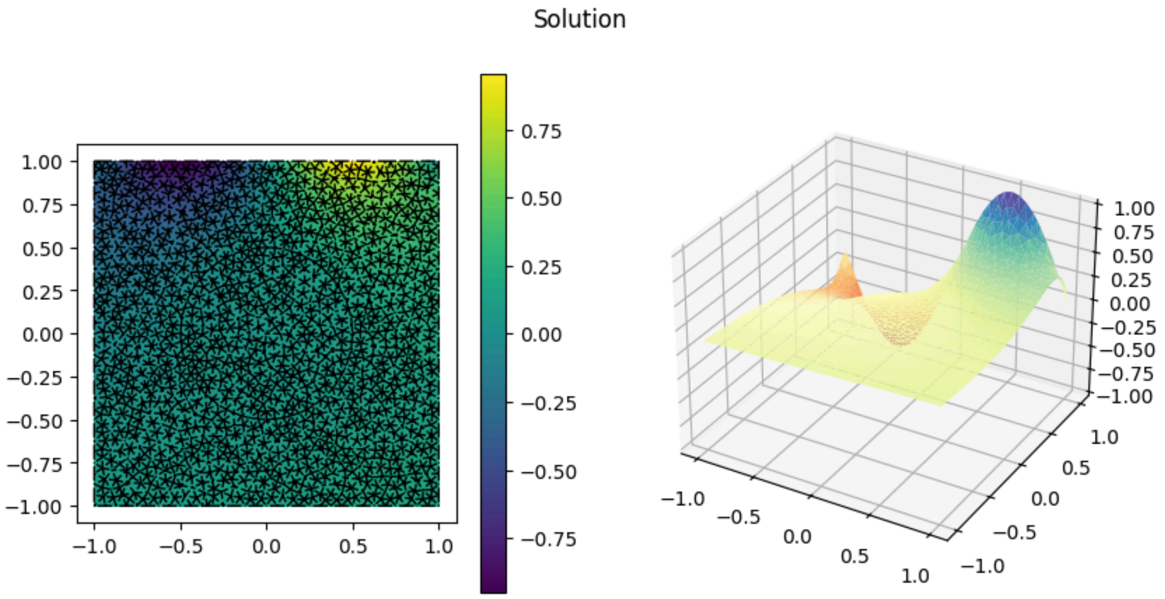
In [29]:
```python
x_test = [[6., .1, 1.]] # Test a new parameter
x_test = np.float32(x_test)
x_test = torch.tensor(x_test)

reduced_solution = np.asarray(net(x_test).detach().numpy())[0] # To make comparison wrt the other

print(reduced_solution) # In dimension 9
```

```
[ 0.7663319 -3.6407268  7.041379  -4.36864   17.088823   7.298813
 -0.7655187 -2.172597  -1.567853 ]
```

In [30]:
```python
# Project the solution to see it

nn_proj_reduced_solution = basis_functions @ reduced_solution
mu = x_test[0]
thetaA2 = mu[0].item()
thetaf1 = mu[1].item()
mu_3 = mu[2].item()
thetaA1 = 1
Dirichlet_top = gedim.AssembleStrongSolution(Dirichlet_Term, 3, problemData, lib)


gedim.PlotSolution(mesh, dofs, strongs, nn_proj_reduced_solution, Dirichlet_top)
# The problem is quite constant to zero and is also tricky for the NN
# Variability can help better to understand in which direction goes
```



In [31]:
```python
##### full #####

stiffness = thetaA1*stiffness1 + thetaA2*stiffness2
weakTerm_down = thetaf1*weakTerm_down1
f = weakTerm_down - (thetaA1*stiffnessStrong1 + thetaA2*stiffnessStrong2) @ Dirichlet_top
full_solution = gedim.LUSolver(stiffness, f, lib)

full_solution = gedim.LUSolver(stiffness, f, lib)
gedim.PlotSolution(mesh, dofs, strongs, full_solution, Dirichlet_top) # Plotting the FOM solution (the ground trouth)
```



Let us perform an error analysis and comment on the speed up!

In [32]:
```python
### compute error
import time
```

```
abs_err = []
rel_err = []
testing_set = np.random.uniform(low=P[:, 0], high=P[:, 1], size=(100, P.shape[0]))
speed_up = []

print("Computing error and speedup analysis") # Compute the erorr

for mu in testing_set:

  thetaA2 = mu[0]
  thetaf1 = mu[1]
  mu_3 = mu[2]

  #### I DO NOT NEED THE SOLVER #####
    # No need to assemble --> no solve any kind of system, just call the net
  Dirichlet_top = gedim.AssembleStrongSolution(Dirichlet_Term, 3, problemData, lib) ## label

  ##### full #####
  stiffness = thetaA1*stiffness1 + thetaA2*stiffness2
  weakTerm_down = thetaf1*weakTerm_down1
  f = weakTerm_down - (thetaA1*stiffnessStrong1 + thetaA2*stiffnessStrong2) @ Dirichlet_top


  start_fom = time.time()
  full_solution = gedim.LUSolver(stiffness, f, lib)
  time_fom = time.time() - start_fom
  # gedim.PlotSolution(mesh, dofs, strongs, full_solution, Dirichlet_top)

  #### reduced #####
  x_test = [[mu[0], mu[1], mu[2]]]
  x_test = np.float32(x_test)
  x_test = torch.tensor(x_test)

  start_rom = time.time()
  reduced_solution = np.asarray(net(x_test).detach().numpy())[0]
  time_rom = time.time() - start_rom

  speed_up.append(time_fom/(time_rom))

  proj_reduced_solution = basis_functions@reduced_solution
  # gedim.PlotSolution(mesh, dofs, strongs, proj_reduced_solution, Dirichlet_top)

  ### computing error
  error_function = full_solution - proj_reduced_solution
  error_norm_squared_component = np.transpose(error_function) @ inner_product @ error_function
  absolute_error = np.sqrt(abs(error_norm_squared_component))
  print(absolute_error)
  abs_err.append(absolute_error)

  full_solution_norm_squared_component = np.transpose(full_solution) @  inner_product @ full_solution
  relative_error = absolute_error/np.sqrt(abs(full_solution_norm_squared_component))
  rel_err.append(relative_error)
  print(relative_error)

# There are a lot of variablity in the errors
```

```
Computing error and speedup analysis
0.02821184718236291
0.014908633525250271
0.07616442397872149
0.015541296347706929
0.03766335037575535
0.009276231385921324
0.05719365989801668
0.014810570624140118
0.032620691231664285
0.013750165623966614
0.0405729203923825
0.008127863478883427
0.05900610769209333
0.012249218116076082
0.037401050686081634
0.00839834230956689
0.09512669743667924
0.019592785374797927
0.07050750194533502
0.01470807803004408
0.06127180256855271
0.012129656663648057
0.02045637606995963
0.0043147546530735055
0.03503938564104698
0.009261329038292005
0.027333812545155343
0.017505357024757827
0.03951699594194714
0.016443723627136062
0.08257323132842302
0.01638081395057979
0.06555599461016136
0.013988320972191347
0.07212403030351155
0.02072005363901202
0.03876076559983442
0.008767042409436874
1.617604829332954
0.32895825887977853
0.6837192040850134
0.13847357856053077
0.04227339745004451
0.008397214339237256
0.07107951639756235
0.014676881620522131
0.054006771452411784
0.01275873094622701
0.013584195980631736
0.002795474810406504
0.026187963547102074
0.0055709659663471895
0.014725626437153218
0.003224641106580426
0.04400968793924805
0.009126878694495221
0.11382070016326835
0.06607058559205269
0.025826928527340687
0.027011695659077774
0.12416705473559358
0.02476798895299906
0.12071354604773803
0.026734027571896756
0.848377814631573
0.23700054189685804
0.0301390785421146
0.005938991748945168
0.03994837708127216
0.01029693468220226
0.05864930779207655
0.09757275142895484
0.05222779560472152
0.01376082765536707
0.04962777831498411
0.01207310970875782
0.028332547102752438
0.006070245816001522
0.09597048445667407
0.018731500462683046
0.0806101128565935
0.01631531949678877
0.0508171262637631
0.010785368019487871
0.07579283997743448
0.014916081319311943
0.03032970539556055
0.008434558880418979
0.013676386742586626
0.015857943963793236
0.057536609127093984
0.01224344250687304
0.024063544788619797
```

0.0059847118000395565
0.06714673296915058
0.014161498823338203
0.0158390565678505
0.00351061268557254
0.10958906682472694
0.02512548460668675
0.07364061497463294
0.016042582923767874
0.03712234313207918
0.026629184662925128
0.18778004412376584
0.04096727218872616
0.073777323859217
0.015066172058200638
0.16487615440226092
0.035400075104476424
0.07141392493678779
0.015539143216420522
0.0806946941639414
0.18077789236094427
0.05335103303966353
0.017673243401288254
0.11511707916205155
0.024982797651049587
0.06083735599622492
0.013544730967062614
0.04618391024426444
0.01391309194879253
0.08708278588102569
0.03686957888036641
0.016344155356257144
0.0033485565402340254
0.016097016850897573
0.003814105035193591
0.021730814466563324
0.021399425189198167
0.0432727196424626
0.0200445618367447
0.049014847497908236
0.02242070571935231
0.05319665180134205
0.01103159474043813
0.03933850711836065
0.024427266351966796
0.059283806168710575
0.017123443343034655
0.04029981816534516
0.008459305005729228
0.111916504028514
0.06739874384843333
0.06802217628258674
0.01403614745650105
0.017945667588458933
0.006001709797772595
0.14830013814688423
0.029585892273652332
0.07350810023087762
0.015122213887077888
0.13902998554973475
0.03011458408323742
0.02869912253462392
0.007682396753743027
0.022177875693441942
0.0045974341376816856
0.030578279996783905
0.024810113788121604
0.07977351679861787
0.016355267540372146
0.01772061690014818
0.01748602118857569
0.010792307023187654
0.011486059075650102
0.026888157636357567
0.005537986544494978
0.028685813633302318
0.01325211116497352
0.05411497881739731
0.017036741554642836
0.032044132049254874
0.012080230070039399
0.04542204537910534
0.013394853151140886
0.051836454350734675
0.01089871263494314
0.04122167953060133
0.009400513101232138
0.02468839452260874
0.008687949834085553
0.18721314540977405
0.041956038855593944
0.03853943298655953
0.008858270355406662
0.055640885491766004

```
0.012030640481916658
0.03670355895898002
0.01020055555347715
0.03499766784063283
0.045967811210061285
0.0160053355085197
0.003198011996363442
0.01980685656861594
0.004062878535178802
0.18200220166313089
0.036942629016226886
0.07384122795203243
0.014784985514387804
```

In [33]:

```python
# The error increase, because we are using ML and so loose accuracy --> the better part is that we go faster
print("avarege relative error = ", np.mean(rel_err) )
print("avarege absolute error = ", np.mean(abs_err) )
print("avarege speed_up = ", np.mean(speed_up) ) # BUT we are faster!

# See the graph in notes 3 for more details on the error
```

```
avarege relative error =  0.02530663317527609
avarege absolute error =  0.08746066292620291
avarege speed_up =  36.87536589607264
```