1. **File Domain**
2. **Lớp Domain** (gồm 2 giá trị truyền vào là: dim và lim)

Định nghĩa miền của PDF và các giá trị biên của nó. Trong lớp này bao gồm:

1. Các giá trị khởi tạo:

* Dim: chiều của miền vd (1 or 2)
* Lim: giới hạn của mỗi chiều tại một cột. ([2xdim]) vd ([[-1],[1]])

1. Hàm scaleCoord(x): dùng để xử lý tọa độ để các input đầu vào của NN nằm trong đoạn [-1,1]

* Tham số truyền vào là x [n x dim] vd: [[0.5],[0.6]]
* Ouput: [n x dim] => kq: [[0.5],[0.6]]

1. Hàm isInside(x): dùng để xác định các tọa độ có nằm bên trong miền. Hàm này được định nghĩa lại (gọi lại) trong lớp con (vd như trong MeshDomain(Domain), Domain1D(Domain) )

* Tham số đầu vào là x tọa độ các điểm để kiểm tra
* Output là giá trị true or false của mỗi điểm trên tọa độ đầu vào

1. Hàm getMesh(): trả về một đối tượng Mesh tương ứng với lưới được tạo cho miền. Hàm này cũng cần định nghĩa lại trong lớp con.
2. **Lớp Mesh**

Lớp này dùng để lưu trữ và truy xuất các thuộc tính của một miền nào đó.

Lưu trữ các dữ liệu lưới cái mà được sử dụng cho việc trainning NN cũng như vẽ các trường (fields). Lớp này bao gồm các thuộc tính khởi tạo như:

* Dim: chiều của miền
* Dof: số điểm rời rạc
* Coordinates [dof x dim]: tọa độ của các điểm rời rạc.
* He: kích thước phần tử
* bIndNum: số lượng các chỉ số biên trong miền
* bdof [bIndNum x 1]: số các điểm rời rạc trên mỗi phân đoạn miền
* discNum [1 x dim]: số các điểm rời rạc tại mỗi miền (dành cho 2D)
* bDiscNum: mật độ các điểm rời rạc trên các biên (dành cho 2D)

1. **Lớp MeshDomain(Domain): kế thừa lớp Domain tham số đầu vào là một filemesh.**

Lớp này xử lý các miền phức tạp, bao gồm:

1. Hàm khởi tạo:

* Coordinates [dof x dim]: tọa độ của các điểm rời rạc bên trong miền.
* bIndNum: số lượng của các chỉ số trong miền
* bCoordinates [bIndNum x n x dim]: chứa danh sách các điểm rời rạc trên các biên.

1. Hàm isInside(x): dùng để xác định các tọa độ có nằm bên trong miền

* Tham số đầu vào là x tọa độ các điểm để kiểm tra
* Output là giá trị true or false của mỗi điểm trên tọa độ đầu vào

1. Hàm getMesh(x): trả về một đối tượng Mesh tương ứng với lưới được tạo cho miền.
2. **Lớp Domain1D(Domain): Lớp này cũng kế thừa Domain, tham số đầu vào là một khoảng thời gian (interval) cho trước ([-1,1])**
3. Hàm khởi tạo:

* Dim = 1: chiều của miền
* Lim = [[-1],[1]]: giới hạn của miền
* bIndNum = 2 : số lượng điểm biên
* measure = 1 – (-1) = 2: độ dài của biên

1. Hàm isInside(x, tol = 0): tham số truyền vào là x [n x 1] và tol là tham số mặc định => output là giá trị logic của các điểm trong x có nằm trong miền. vd (x = [[2],[1]] => [[false], [true]]) miền này là đoạn [-1,1]
2. Hàm getMesh(discNum = 100, bDiscNum=None, rfrac=0, sortflg=True, discTol=None): hàm này tạo ra một cấu trúc lưới và loại bỏ các điểm nằm ngoài miền.

* Dim = 1
* Dof = 100
* Lim = [[-1],[1]]
* bIndNum = 2
* bdof = [1,1]
* he = 2/101 = [0,019]
* coordinates =[[-0.98019802],…[]…] => size = 100
* bCoordinates = [[-1],[1]]

Gọi lại class Mesh(dim=dim, dof=dof,coordinates=coordinates,he=he,bIndNum=bIndNum, bdof=bdof, bCoordinates=bCoordinates,discNum=discNum) và truyền các tham số ở trên vào để tạo cấu trúc lưới 1 chiều.

1. Hàm meshTot(discNum=100): hàm này tạo ra cấu trúc lưới của miền đa giác và loại bỏ các điểm nằm ngoài biên của miền. Sự khác biệt với hàm getMesh () là hàm này tạo ra một lưới cho toàn bộ miền bao gồm các biên trong khi getMesh () xem xét các biên và miền bên trong riêng biệt.

* Dim = 1
* Dof =100
* Lim = [[-1],[1]]
* He = 2/99 = [0,02020202]
* Coordinates =[[-1],[-0.97979798],….[1]] => size = 100
* bIndNum = None
* bdof = None
* bCoordinates = None

Gọi lại class Mesh(dim=dim, dof=dof,coordinates=coordinates,he=he,bIndNum=bIndNum, bdof=bdof, bCoordinates=bCoordinates,discNum=discNum) và truyền các tham số ở trên vào để tạo cấu trúc lưới 1 chiều

1. **Lớp PolyDomain2D(Domain): Lớp này cũng kế thừa từ lớp Domain(), tham số đầu vào là một vertices = [[-1,-1],[1,-1],[1,1],[-1,1]], obsVertices =[], mặc định là một mảng rỗng**
2. Hàm khởi tạo bao gồm các thuộc tính:

Dim =2 , chiều của miền

Lim = [[-1, -1], [1,1]], giới hạn miền

Vertices = [[-1,-1],[1,-1],[1,1],[-1,1]]

VertexNum = 4, số vector của khung miền

obsNum = 0, số chướng ngại vật

obsVertices = []

bIndNum = 4, số biên

boundryGeom = [[-1,-1],[1,-1],[1,1],[-1,1]], giới hạn của mỗi biên

measure = 4.0, khu vực miền đa giác

1. Hàm boundaryLims(vertices): tham số truyền vào là vertices => output là mảng chứa tọa độ các góc của mỗi cạnh trong một hàng.

Vd: vertices = np.array([[-0.1, 0.5], [0.7, -0.7], [1.0, 1.0], [-1.0, 1.0]])

Kq: [[[-0.1,0.5],[ 0.7,-0.7]], [[ 0.7,-0.7],[1,1]],[[1, 1],[-1,1]],[[-1,1],[-0.1,0.5]]]

1. Hàm domPlot(addDescription=True, figNum=None,frameColor=None):

Hàm vẽ biên 2 chiều với tọa độ chạy từ -1 => 1

1. Hàm meshPlot(mesh):
2. Hàm isInside(x,tol=0):
3. Hàm getMesh(discNum=100,bDiscNum=50,rfrac=0,sortflg=True,discTol=None)
4. Hàm innerDisc(discNum,rfrac=0,sortflg=True,discTol=None)
5. Hàm boundaryDisc(vertices,bDiscNum,rfrac=0,sortflg=True)
6. Hàm meshTot(discNum=100,rfrac=0,sortflg=True)
7. **File UnityFunc.py**
8. **Class UF() dùng để định nghĩa các hàm tùy chỉnh, kiểm tra các giá trị đầu vào, class này bao gồm các hàm như:**
9. isnumber(x):hàm này dùng để kiểm tra các giá trị trong x có phải là số không, tham số đầu vào là x => output là true or false

vd: x=[1,2,3,4] => kết quả: true

x = [‘a’, 1,2] => kết quả: false

1. arrayIsnumber(x): hàm này dùng để kiểm tra mảng có chứa tất cả các số hay không, tham số đầu vào là x => output là true or false

vd: x = np.array([1,2,3,4]) => kết quả là true

x = np.array([‘a’,2,3,4]) => kết quả là false

1. isempty(x): hàm kiểm tra tham số đầu vào x có rỗng hay không, output là true or false

vd: x = [] => kết quả true

x = [1,2,3] => kết quả trả về false

1. isone(x): hàm này kiểm tra tham số đầu vào x có phải là None không, output là true or false

vd: x = None =>kết quả true

x = [1,2,3,4] => kết quả false

1. unpackList(x): tham số đầu vào là một danh sách các list phần tử => output là một list đơn

vd: x = [[1,2,4],[5,8,9],[1,2]] => kết quả: [1,2,4,5,8,9,1,2]

1. vstack(tup): input đầu vào là một list (tup) => output là mảng 2 chiều

vd: x= [1,2,4] => kết quả: [[1],[2],[4]]

x = [[1,2],[1,4]] => kết quả: [[1,2],[1,4]]

1. hstask(tup): input đầu vào là một list => output là mảng một chiều

vd: x = [[1,2],[3,4,5]] => kết quả: [1,2,3,4,5]

x = [1,3,4] => kết quả: [1,3,4]

1. nodeNum(x,val): hàm này tìm chỉ số trong tham số x có giá trị gần nhất với tham số val

vd: x = [[8],[1],[2],[4],[3],[0]]

val = [1,2,3,4,8,0,9]

* kết quả: [1,2,4,3,0,5,0]

1. nodeNumGrid(obj,x=None,y=None,t=None)
2. pairMats(mat1,mat2,reverse=False)
3. rejectionSampling(func, smfun, dof,dofT=None)
4. listSegment(vec,segdof,func=None)
5. reorderList(x,ind)
6. buildDict(keys,values)
7. l2Err(xTrue,xApp)
8. clearFolder(folderpath)
9. copyFile(filename,folderpath)
10. polyArea(x,y=None)
11. mergeDict(dictList)
12. csvRead(filename)
13. column(vec)
14. blkdiag(tup,empty=None)
15. rmEmptyList(val)
16. Type(x)
17. addNoise(sig,delta,distn=’Gaussian’,method=’additive’)
18. stem3(x,y)
19. stepFun(cellVal, cellLims,points)
20. mat2diffpack(filename,fieldname,mat)
21. meshNavigation(discNum,locind,step)
22. finiteDiff(dim,step,schema=’central’)
23. ANSYS(n,newCoord,filename,savepath=None)
24. inCircle(coord,cen,rad)
25. inEllipse(coord,cen,a,b,theta)
26. inTriangle(coord,corners)
27. inRectangle(coord,corners)
28. barPlot(data,ticks=None,lables=None,edgecolor=’white’,title=None)
29. checkBoundErr(interval, name,valType=float,valDefault=None,maxBound=None, warn=True,selfErr=True)
30. **File ADPDE.py:**
31. **Lớp ADPDE: tạo phương trình vi phân ADPE**

Lớp này gồm các tham số đầu vào:

Dim = 1

Domain: class Domain

tInterval = None

BCs = [[0.0, 1.0, <function ADPDE.\_\_in...D7F77FE18>], [0.0, 1.0, <function ADPDE.\_\_in...D7F77FD90>]]

Bctype = ['Dirichlet', 'Dirichlet']

IC = None

Cex = None

MORvar = None

Bao gồm các hàm như:

1. Hàm plotIC()
2. plotBC()
3. **File VarNet.py**
4. **File TFModel.py**
5. **File Operator\_1D.py**

Nhấn debug chỗ biến domain trong file Operator\_1D.py và cho chạy từng bước 1 em thấy:

Trong lớp Domain1D() chứa các giá trị: dim =1, interval = [-1,1] => lim = [[-1],[1]], bIndNum = 2, measure = 2

Sau khi xong lớp Domain1D() xong qua lớp ADPDE(domain, diff=D, vel=u, timeDependent=True, tInterval=[0,T], IC=IC,cEx=cExact) cũng cho chạy từng bước em thấy: nó gọi các hàm của lớp UF() trong file UnityFunc.py để check các tham số đầu vào như:

* Hàm isnone(tInterval) để check tham số tInterval và trả về kết quả của timeDependent là True or False
* Hàm isnumber(vel) để check vel có phải là số không
* Hàm isnone(BCs) check BCs có None không
* Hàm isnone(cEx) check cEx có None không

Và trong class ADPDE() có các giá trị khởi tạo là:

* Dim = 1
* Domain = gọi class Domain()
* timeDependent = True
* tInterval = [0, 2]
* BCs = [[0.0, 1.0, <function ADPDE.\_\_in...D7F77FE18>], [0.0, 1.0, <function ADPDE.\_\_in...D7F77FD90>]]
* Bctype = ['Dirichlet', 'Dirichlet']
* IC = gọi Function IC (<function IC at 0x000001D01E56B840>)
* Cex = gọi function cExact (<function cExact at 0x000001D02DF05158>)
* MORvar = None

Chạy tiếp qua dòng varnet = VarNet()

Cũng gọi các hàm trong lớp UF() để check các tham số

Gọi lớp varnet và khởi tạo các giá trị:

* Dim = 1
* timeDependent = True
* MORvar = None
* discNum = 20
* bDiscNum = None
* tDiscNum = 300
* MORdiscScheme = None
* modelId = ‘MLP’
* PDE = gọi class ADPDE() với các tham số giá trị như trên
* Vì modelId = ‘MLP’ => fixData = gọi class FIXData từ file VarNetUtility()
* tfData= gọi class TFNN() từ file TFModel

Trong class FIXData() có các giá trị khởi tạo:

* Dim = 1
* feDim = 2
* timeDependent = True
* integPnum = 2
* dof = 20
* bdof = [1,1]
* biDof0 = [300, 300, 20]
* nt0 = 6000
* hVec = [[0.0952381], [0.00666667]]
* biDimVal = 2.0
* detJvec = False
* lossVecflag = True
* uniform\_input = là mảng 2 chiều size = 600 (max = 2, min = -0.9047)
* uniform\_binput = là mảng 2 chiều size = 620(max = 2, min = -1)
* MORbacthNum = 1
* MORargInd = None
* MORdiscArg = None
* (Tất cả các giá trị khởi tạo còn lại đều = None)

Trong class TFNN() có các giá trị khởi tạo là:

* Dim =1
* inpDim = 2
* depth =1
* layerWidth =[20]
* modelId = ‘MLP”
* activationFun = [‘sigmoid’]
* timeDependent = True
* RNNdata =None
* ProcessorNum =1
* Processors = [‘/device:CPU:0’]
* Controller = ‘/device:CPU:0’
* lossOpt = {‘integWflag’: False, ‘isSource’: False}
* optimizer\_name = ‘adam’
* learning\_rate = 0.001