VMATDocumentation

Generated by Doxygen 1.8.6

Wed May 11 2016 16:21:08

Contents

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Namespace Documentation

4.1 greedyVMATsampling Namespace Reference

Classes

- class region
- class apertureList

  apertureList is a class definition of locs and angles that is always sorted.
- class vmat_class

  This defines the global VMAT class that contains most of the VMAT data to be used in the implementation Most of the values were defined as static attributes and only one instantiation at a time is possible.

Functions

- def readctvoxelinfo
- def readDmatrix
- def calcObjGrad
- def fvalidbeamlets
- def PPsubroutine
- def parallelizationPricingProblem
- def PricingProblem
- def solveRMC
- def printresults
- def colGen
- def plotAperture
- def updateOpenAperture

Variables

- string __author__ = 'wilmer'
- have_mkl = True
- list kappa = [6, 17, 28, 39, 50, 61, 72, 83, 94, 105, 116, 127, 138, 149, 160, 171, 11, 22, 33, 44, 55, 66, 77, 88, 99, 110, 121, 132, 143, 154, 165, 1, 175, 14, 25, 36, 47, 58, 69, 80, 91, 102, 113, 124, 135, 146, 157, 168, 3, 8, 19, 30, 41, 52, 63, 74, 85, 96, 107, 118, 129, 140, 151, 162, 172, 176, 0, 2, 4, 5, 7, 9, 10, 12, 13, 15, 16, 18, 20, 21, 23, 24, 26, 27, 29, 31, 32, 34, 35, 37, 38, 40, 42, 43, 45, 46, 48, 49, 51, 53, 54, 56, 57, 59, 60, 62, 64, 65, 67, 68, 70, 71, 73, 75, 76, 78, 79, 81, 82, 84, 86, 87, 89, 90, 92, 93, 95, 97, 98, 100, 101, 103, 104, 106, 108, 109, 111, 112, 114, 115, 117, 119, 120, 122, 123, 125, 126, 128, 130, 131, 133, 134, 136, 137, 139, 141, 142, 144, 145, 147, 148, 150, 152, 153, 155, 156, 158, 159, 161, 163, 164, 166, 167, 169, 170, 173, 174, 177]

- WholeCircle = False
- string rootFolder = '/media/wilmer/datadrive'
- string readfolder = rootFolder+'/Data/DataProject/HN/'
- string readfolderD = readfolder+'Dij/'
- string outputfolder = '/home/wilmer/Dropbox/Research/VMAT/output/'
- int degreesep = 60
- string objfile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/objectives/obj1.txt'
- string structurefile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/structureInputs.txt'
- string algfile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/algInputsWilmer.txt'
- string mm3voxels = rootFolder+'/Data/DataProject/HN/hn3mmvoxels.mat'
- list priority = [7, 24, 25, 23, 22, 21, 20, 16, 15, 14, 13, 12, 10, 11, 9, 4, 3, 1, 2, 17, 18, 19, 5, 6, 8]
- list mylines = [line.rstrip('\n') for line in open('/home/wilmer/Dropbox/Research/VMAT/VMATwPenCode/beamAngles.txt')]
- tuple fig = plt.figure(1)
- list catemp = [ ]
- list gatemp = [ ]
- tuple start = time.time()
- tuple data = vmat_class()
- tuple oldfolder = os.getcwd()

  ••••••••##########################
- tuple allFiles = glob.glob("∗VOILIST.mat")
- tuple allBeamInfos = glob.glob("∗Couch0_BEAMINFO.mat")
- tuple allNames = sorted(allFiles)
- tuple allBeamInfoNames = sorted(allBeamInfos)
- tuple numStructs = len(allFiles)
- tuple vdims = readctvoxelinfo()
- list numVoxels = vdims['x']
- list Vorg = [ ]
- tuple bigZ = np.zeros(numVoxels, dtype=int)
- tuple nVox = sum(bigZ)
- int counter = 0
- tuple originalVoxels = np.empty(numVoxels)
- list lines = [myline.split('\t') for myline in [line.rstrip('\n') for line in open(structurefile)]]

  Read in structures .
- list invec = [item for sublist in lines for item in sublist]

  Collapse the above expression to a flat list.
- tuple maskValueFull = np.zeros(nVox.astype(np.int64))

  Assignation of different values.
- tuple maskValueSingle = np.zeros(nVox.astype(np.int64))
- list s = priority[i]
- tuple norepeat = np.unique(originalVoxels[np.invert(np.isnan(originalVoxels))])
- tuple istarget = str(s)
- list tempindicesfull = originalVoxels[Vorg[s]]
- tuple tempindices = np.intersect1d(tempindicesfull, norepeat)
- int gastart = 0
- int gaend = 356
- int gastep = 2
- list ga = [ ]
- list ca = [ ]
- string fname = 'Gantry'

  Treatment of BEAMINFO data.
- string bletfname = readfolder+'Gantry'
- tuple nBPB = np.zeros(len(ga))
- tuple nDIJSPB = np.zeros(len(ga))

- tuple beamletCounter = np.zeros(data.numbeams + 1)

  Beginning of Troy's cpp code (interpreted, not copied) A comment This comes from first two lines in doseInputs txt file (troy's version)
- tuple binfoholder = sio.loadmat(bletfname)
- tuple N = len(data.yinter)

  After reading the beaminfo information.
- tuple M = len(data.xinter)
- int overallDijsCounter = 0

  Initial intensities are allocated a value of zero.
- list DlistT = [None]
- tuple pool = Pool(processes=8)
- tuple Allmats = pool.map(readDmatrix, range(0, data.numbeams))
- list ininter = [ ]
- functionData = data.functionData

  Collapse the above expression to a flat list.
- tuple quadHelperThresh = np.zeros(data.numvoxels)
- tuple quadHelperOver = np.zeros(data.numvoxels)
- tuple quadHelperUnder = np.zeros(data.numvoxels)
- tuple quadHelperAlphaBetas = np.zeros(data.numvoxels)
- tuple uDose = np.zeros(data.numvoxels)
- tuple oDose = np.zeros(data.numvoxels)
- tuple before = time.time()
- tuple pstar = colGen(0, WholeCircle, 16)
- tuple after = time.time()

4.1.1 Function Documentation

4.1.1.1 def greedyVMATsampling.calcObjGrad (*x*, *user_data* = None)

4.1.1.2 def greedyVMATsampling.colGen (*C*, *WholeCircle*, *initialApertures*)

4.1.1.3 def greedyVMATsampling.fvalidbeamlets (*i*, *index*)

4.1.1.4 def greedyVMATsampling.parallelizationPricingProblem (*i*, *C*, *C2*, *C3*, *b*, *vmax*, *speedlim*, *N*, *M*)

4.1.1.5 def greedyVMATsampling.plotAperture (*l*, *r*, *M*, *N*, *myfolder*, *iterationNumber*, *bestAperture*)

4.1.1.6 def greedyVMATsampling.PPsubroutine (*C*, *C2*, *C3*, *b*, *angdistancem*, *angdistancep*, *vmax*, *speedlim*, *predec*, *succ*, *N*, *M*, *thisApertureIndex*)

4.1.1.7 def greedyVMATsampling.PricingProblem (*C*, *C2*, *C3*, *b*, *vmax*, *speedlim*, *N*, *M*)

4.1.1.8 def greedyVMATsampling.printresults (*iterationNumber*, *myfolder*)

4.1.1.9 def greedyVMATsampling.readctvoxelinfo ( )

4.1.1.10 def greedyVMATsampling.readDmatrix (*i*)

4.1.1.11 def greedyVMATsampling.solveRMC ( )

4.1.1.12 def greedyVMATsampling.updateOpenAperture (*i*)

4.1.2 Variable Documentation

4.1.2.1 string greedyVMATsampling.__author__ = 'wilmer'

4.1.2.2 tuple greedyVMATsampling.after = time.time()

4.1.2.3 string greedyVMATsampling.algfile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/algInputsWilmer.txt'

4.1.2.4 tuple greedyVMATsampling.allBeamInfoNames = sorted(allBeamInfos)

4.1.2.5 tuple greedyVMATsampling.allBeamInfos = glob.glob("∗Couch0_BEAMINFO.mat")

4.1.2.6 tuple greedyVMATsampling.allFiles = glob.glob("∗VOILIST.mat")

4.1.2.7 tuple greedyVMATsampling.Allmats = pool.map(readDmatrix, range(0, data.numbeams))

4.1.2.8 tuple greedyVMATsampling.allNames = sorted(allFiles)

4.1.2.9 tuple greedyVMATsampling.beamletCounter = np.zeros(data.numbeams + 1)

Beginning of Troy's cpp code (interpreted, not copied) A comment This comes from first two lines in doseInputs txt file (troy's version)

Allocate memory

4.1.2.10 tuple greedyVMATsampling.before = time.time()

4.1.2.11 tuple greedyVMATsampling.bigZ = np.zeros(numVoxels, dtype=int)

4.1.2.12 tuple greedyVMATsampling.binfoholder = sio.loadmat(bletfname)

4.1.2.13 string greedyVMATsampling.bletfname = readfolder+'Gantry'

4.1.2.14 list greedyVMATsampling.ca = [ ]

4.1.2.15 list greedyVMATsampling.catemp = [ ]

4.1.2.16 int greedyVMATsampling.counter = 0

4.1.2.17 tuple greedyVMATsampling.data = vmat_class()

4.1.2.18 int greedyVMATsampling.degreesep = 60

4.1.2.19 list greedyVMATsampling.DlistT = [None]

4.1.2.20 tuple greedyVMATsampling.fig = plt.figure(1)

4.1.2.21 string greedyVMATsampling.fname = 'Gantry'

Treatment of BEAMINFO data.

4.1.2.22 greedyVMATsampling.functionData = data.functionData

Collapse the above expression to a flat list.

Read in the constraint file: NOTHING TO DO ############ FINISHED READING EVERYTHING #### Work with function data.

4.1.2.23 list greedyVMATsampling.ga = [ ]

4.1.2.24 int greedyVMATsampling.gaend = 356

4.1.2.25 int greedyVMATsampling.gastart = 0

4.1.2.26 int greedyVMATsampling.gastep = 2

4.1.2.27 list greedyVMATsampling.gatemp = [ ]

4.1.2.28 greedyVMATsampling.have_mkl = True

4.1.2.29 list greedyVMATsampling.ininter = [ ]

4.1.2.30 list greedyVMATsampling.invec = [item for sublist in lines for item in sublist]

Collapse the above expression to a flat list.

4.1.2.31 tuple greedyVMATsampling.istarget = str(s)

4.1.2.32 list greedyVMATsampling.kappa = [6, 17, 28, 39, 50, 61, 72, 83, 94, 105, 116, 127, 138, 149, 160, 171, 11, 22, 33, 44, 55, 66, 77, 88, 99, 110, 121, 132, 143, 154, 165, 1, 175, 14, 25, 36, 47, 58, 69, 80, 91, 102, 113, 124, 135, 146, 157, 168, 3, 8, 19, 30, 41, 52, 63, 74, 85, 96, 107, 118, 129, 140, 151, 162, 172, 176, 0, 2, 4, 5, 7, 9, 10, 12, 13, 15, 16, 18, 20, 21, 23, 24, 26, 27, 29, 31, 32, 34, 35, 37, 38, 40, 42, 43, 45, 46, 48, 49, 51, 53, 54, 56, 57, 59, 60, 62, 64, 65, 67, 68, 70, 71, 73, 75, 76, 78, 79, 81, 82, 84, 86, 87, 89, 90, 92, 93, 95, 97, 98, 100, 101, 103, 104, 106, 108, 109, 111, 112, 114, 115, 117, 119, 120, 122, 123, 125, 126, 128, 130, 131, 133, 134, 136, 137, 139, 141, 142, 144, 145, 147, 148, 150, 152, 153, 155, 156, 158, 159, 161, 163, 164, 166, 167, 169, 170, 173, 174, 177]

4.1.2.33 list greedyVMATsampling.lines = [myline.split('\t') for myline in [line.rstrip('\n') for line in open(structurefile)]]

Read in structures .

MATRIX CUT DONE Here all matrices are working with the same limits.

CHANGE THIS. Reading from txt file != good!!

Read in the objective file:

4.1.2.34 tuple greedyVMATsampling.M = len(data.xinter)

4.1.2.35 tuple greedyVMATsampling.maskValueFull = np.zeros(nVox.astype(np.int64))

Assignation of different values.

4.1.2.36 tuple greedyVMATsampling.maskValueSingle = np.zeros(nVox.astype(np.int64))

4.1.2.37 string greedyVMATsampling.mm3voxels = rootFolder+'/Data/DataProject/HN/hn3mmvoxels.mat'

4.1.2.38 list greedyVMATsampling.mylines = [line.rstrip('\n') for line in open('/home/wilmer/Dropbox/Research/VMAT/VMATw-PenCode/beamAngles.txt')]

4.1.2.39 tuple greedyVMATsampling.N = len(data.yinter)

After reading the beaminfo information.

Read CUT the data.

4.1.2.40 tuple greedyVMATsampling.nBPB = np.zeros(len(ga))

4.1.2.41 tuple greedyVMATsampling.nDIJSPB = np.zeros(len(ga))

4.1.2.42 tuple greedyVMATsampling.norepeat = np.unique(originalVoxels[np.invert(np.isnan(originalVoxels))])

4.1.2.43 tuple greedyVMATsampling.numStructs = len(allFiles)

4.1.2.44 list greedyVMATsampling.numVoxels = vdims['x']

4.1.2.45 tuple greedyVMATsampling.nVox = sum(bigZ)

4.1.2.46 string greedyVMATsampling.objfile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/objectives/obj1.txt'

4.1.2.47 tuple greedyVMATsampling.oDose = np.zeros(data.numvoxels)

4.1.2.48 tuple greedyVMATsampling.oldfolder = os.getcwd()

•••••••••###########################

4.1.2.49 tuple greedyVMATsampling.originalVoxels = np.empty(numVoxels)

4.1.2.50 string greedyVMATsampling.outputfolder = '/home/wilmer/Dropbox/Research/VMAT/output/'

4.1.2.51 int greedyVMATsampling.overallDijsCounter = 0

Initial intensities are allocated a value of zero.

4.1.2.52 tuple greedyVMATsampling.pool = Pool(processes=8)

4.1.2.53 tuple greedyVMATsampling.priority = [7, 24, 25, 23, 22, 21, 20, 16, 15, 14, 13, 12, 10, 11, 9, 4, 3, 1, 2, 17, 18, 19, 5, 6, 8]

4.1.2.54 tuple greedyVMATsampling.pstar = colGen(0, WholeCircle, 16)

4.1.2.55 tuple greedyVMATsampling.quadHelperAlphaBetas = np.zeros(data.numvoxels)

4.1.2.56 tuple greedyVMATsampling.quadHelperOver = np.zeros(data.numvoxels)

4.1.2.57 tuple greedyVMATsampling.quadHelperThresh = np.zeros(data.numvoxels)

4.1.2.58 tuple greedyVMATsampling.quadHelperUnder = np.zeros(data.numvoxels)

4.1.2.59 string greedyVMATsampling.readfolder = rootFolder+'/Data/DataProject/HN/'

4.1.2.60 string greedyVMATsampling.readfolderD = readfolder+'Dij/'

4.1.2.61 string greedyVMATsampling.rootFolder = '/media/wilmer/datadrive'

4.1.2.62 list greedyVMATsampling.s = priority[i]

4.1.2.63 tuple greedyVMATsampling.start = time.time()

4.1.2.64 string greedyVMATsampling.structurefile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/structureInputs.txt'

4.1.2.65 tuple greedyVMATsampling.tempindices = np.intersect1d(tempindicesfull, norepeat)

4.1.2.66 list greedyVMATsampling.tempindicesfull = originalVoxels[Vorg[s]]

4.1.2.67 tuple greedyVMATsampling.uDose = np.zeros(data.numvoxels)

4.1.2.68 tuple greedyVMATsampling.vdims = readctvoxelinfo()

4.1.2.69 list greedyVMATsampling.Vorg = [ ]

4.1.2.70 greedyVMATsampling.WholeCircle = False

Chapter 5

Class Documentation

5.1 greedyVMATsampling.apertureList Class Reference

apertureList is a class definition of locs and angles that is always sorted.

Public Member Functions

- def __init__

  constructor initializes empty lists
- def insertAngle

  Insert a new angle in the list of angles to analyse.
- def removeIndex

  Removes the index and its corresponding angle from the list.
- def removeAngle

  Looks for the angle and removes the index and the angle corresponding to it from the list.
- def __call__

  Overloads parenthesis operator in order to fetch the ANGLE given an index.
- def len

  Returns the length of this instantiation without the need to pass parameters.
- def isEmpty

  Returns True if the list is empty; otherwise returns False.

Public Attributes

- loc

  Location in index range(0,numbeams)
- angle

  Angles ranges from 0 to 360.

5.1.1 Detailed Description

apertureList is a class definition of locs and angles that is always sorted.

Its attributes are loc which is the numeric location; It has range 0 to 178 for the HN case; Angle is the numeric angle in degrees; It ranges from 0 to 358 degrees apertureList should be sorted in ascending order everytime you add a new element; User CAN make this safe assumption

5.1.2 Constructor & Destructor Documentation

5.1.2.1 def greedyVMATsampling.apertureList.__init__ (*self*)

constructor initializes empty lists

5.1.3 Member Function Documentation

5.1.3.1 def greedyVMATsampling.apertureList.__call__ (*self, index*)

Overloads parenthesis operator in order to fetch the ANGLE given an index.

Returns the angle at the ith location given by the index. First Find the location of that index in the series of loc Notice that this function overloads the parenthesis operator for elements of this class.

5.1.3.2 def greedyVMATsampling.apertureList.insertAngle (*self, i, aperangle*)

Insert a new angle in the list of angles to analyse.

Gets angle information and inserts location and angle In the end it sorts the list in increasing order

5.1.3.3 def greedyVMATsampling.apertureList.isEmpty (*self*)

Returns True if the list is empty; otherwise returns False.

5.1.3.4 def greedyVMATsampling.apertureList.len (*self*)

Returns the length of this instantiation without the need to pass parameters.

5.1.3.5 def greedyVMATsampling.apertureList.removeAngle (*self, tangl*)

Looks for the angle and removes the index and the angle corresponding to it from the list.

5.1.3.6 def greedyVMATsampling.apertureList.removeIndex (*self, index*)

Removes the index and its corresponding angle from the list.

Notice that it only removes the first occurrence; but if you have done everything correctly this should never be a problem

5.1.4 Member Data Documentation

5.1.4.1 greedyVMATsampling.apertureList.angle

Angles ranges from 0 to 360.

5.1.4.2 greedyVMATsampling.apertureList.loc

Location in index range(0,numbeams)

The documentation for this class was generated from the following file:

- greedyVMATsampling.py

5.2 greedyVMATsampling.region Class Reference

Public Member Functions

- def __init__

Public Attributes

- index
- sizeInVoxels
- indices
- fullIndices

Static Public Attributes

- tuple index = int()
- tuple sizeInVoxels = int()
- tuple indices = np.empty(1, dtype=int)
- tuple fullIndices = np.empty(1,dtype=int)
- target = False

5.2.1 Detailed Description

```
Contains all information relevant to a particular region
```

5.2.2 Constructor & Destructor Documentation

5.2.2.1 def greedyVMATsampling.region.__init__ (*self*, *iind*, *iindi*, *ifullindi*, *itarget*)

5.2.3 Member Data Documentation

5.2.3.1 tuple greedyVMATsampling.region.fullIndices = np.empty(1,dtype=int) [static]

5.2.3.2 greedyVMATsampling.region.fullIndices

5.2.3.3 tuple greedyVMATsampling.region.index = int() [static]

5.2.3.4 greedyVMATsampling.region.index

5.2.3.5 tuple greedyVMATsampling.region.indices = np.empty(1, dtype=int) [static]

5.2.3.6 greedyVMATsampling.region.indices

5.2.3.7 tuple greedyVMATsampling.region.sizeInVoxels = int() [static]

5.2.3.8 greedyVMATsampling.region.sizeInVoxels

5.2.3.9 greedyVMATsampling.region.target = False [static]

The documentation for this class was generated from the following file:

- greedyVMATsampling.py

5.3 greedyVMATsampling.vmat_class Class Reference

This defines the global VMAT class that contains most of the VMAT data to be used in the implementation Most of the values were defined as static attributes and only one instantiation at a time is possible.

Public Member Functions

- def calcDose
- def calcGradientandObjValue
- def __init__

Public Attributes

- currentDose
- dZdK
- objectiveValue
- voxelgradient
- aperturegradient
- numX

Static Public Attributes

- int numX = 0

  number of beamlets
- tuple numvoxels = int()

  number of voxels in the small voxel space
- int numstructs = 0

  number of structures/regions
- int numoars = 0

  number of organs at risk
- int numtargets = 0

  num of targets
- int numbeams = 0

  num of beams
- int totaldijs = 0

  num of nonzeros in Dij matrix
- tuple objectiveValue = float("inf")

  objectiveValue of the final function
- list beamletsPerBeam = [ ]

  number of beamlets per beam
- list dijsPerBeam = [ ]

  number of nonzeroes in Dij per beam
- list maskValue = [ ]

  non-overlapping mask value per voxel
- list fullMaskValue = [ ]

  complete mask value per voxel (A voxel may cover more than one structure = OAR's + T's)
- list regionIndices = [ ]
- list targets = [ ]
- list oars = [ ]
- list regions = [ ]
- list objectiveInputFiles = [ ]

- list constraintInputFiles = [ ]
- list algOptions = [ ]
- list functionData = [ ]
- list voxelAssignment = [ ]
- tuple notinC = apertureList()
- tuple caligraphicC = apertureList()
- string outputDirectory = ""
- string dataDirectory = ""
- tuple currentDose = np.empty(1)
- tuple currentIntensities = np.empty(1)
- list xinter = [ ]
- list yinter = [ ]
- list xdirection = [ ]
- list ydirection = [ ]
- list llist = [ ]
- list rlist = [ ]
- list voxelgradient = [ ]
- list scipygradient = [ ]
- list openApertureMaps = [ ]
- list diagmakers = [ ]
- float dZdK = 0.0
- list pointtoAngle = [ ]
- list Dlist = [ ]

5.3.1 Detailed Description

This defines the global VMAT class that contains most of the VMAT data to be used in the implementation Most of the values were defined as static attributes and only one instantiation at a time is possible.

But this should not be a problem. The file also contains functions to be used when you call the optimizer.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 def greedyVMATsampling.vmat_class.__init__ (*self*)

5.3.3 Member Function Documentation

5.3.3.1 def greedyVMATsampling.vmat_class.calcDose (*self*)

5.3.3.2 def greedyVMATsampling.vmat_class.calcGradientandObjValue (*self*)

5.3.4 Member Data Documentation

5.3.4.1 list greedyVMATsampling.vmat_class.algOptions = [ ] [static]

5.3.4.2 greedyVMATsampling.vmat_class.aperturegradient

5.3.4.3 list greedyVMATsampling.vmat_class.beamletsPerBeam = [ ] [static]

number of beamlets per beam

5.3.4.4 tuple greedyVMATsampling.vmat_class.caligraphicC = apertureList() [static]

5.3.4.5 list greedyVMATsampling.vmat_class.constraintInputFiles = [ ] [static]

5.3.4.6 tuple greedyVMATsampling.vmat_class.currentDose = np.empty(1) [static]

5.3.4.7 greedyVMATsampling.vmat_class.currentDose

5.3.4.8 tuple greedyVMATsampling.vmat_class.currentIntensities = np.empty(1) [static]

5.3.4.9 string greedyVMATsampling.vmat_class.dataDirectory = "" [static]

5.3.4.10 list greedyVMATsampling.vmat_class.diagmakers = [ ] [static]

5.3.4.11 list greedyVMATsampling.vmat_class.dijsPerBeam = [ ] [static]

number of nonzeroes in Dij per beam

5.3.4.12 list greedyVMATsampling.vmat_class.Dlist = [ ] [static]

5.3.4.13 float greedyVMATsampling.vmat_class.dZdK = 0.0 [static]

5.3.4.14 greedyVMATsampling.vmat_class.dZdK

5.3.4.15 list greedyVMATsampling.vmat_class.fullMaskValue = [ ] [static]

complete mask value per voxel (A voxel may cover more than one structure = OAR's + T's)

5.3.4.16 list greedyVMATsampling.vmat_class.functionData = [ ] [static]

5.3.4.17 list greedyVMATsampling.vmat_class.llist = [ ] [static]

5.3.4.18 list greedyVMATsampling.vmat_class.maskValue = [ ] [static]

non-overlapping mask value per voxel

5.3.4.19 tuple greedyVMATsampling.vmat_class.notinC = apertureList() [static]

5.3.4.20 int greedyVMATsampling.vmat_class.numbeams = 0 [static]

num of beams

5.3.4.21 int greedyVMATsampling.vmat_class.numoars = 0 [static]

number of organs at risk

5.3.4.22 int greedyVMATsampling.vmat_class.numstructs = 0 [static]

number of structures/regions

5.3.4.23 int greedyVMATsampling.vmat_class.numtargets = 0 [static]

num of targets

5.3.4.24 tuple greedyVMATsampling.vmat_class.numvoxels = int() [static]

number of voxels in the small voxel space

5.3.4.25 int greedyVMATsampling.vmat_class.numX = 0 [static]

number of beamlets

5.3.4.26 greedyVMATsampling.vmat_class.numX

5.3.4.27 list greedyVMATsampling.vmat_class.oars = [ ] [static]

5.3.4.28 list greedyVMATsampling.vmat_class.objectiveInputFiles = [ ] [static]

5.3.4.29 tuple greedyVMATsampling.vmat_class.objectiveValue = float("inf") [static]

objectiveValue of the final function

5.3.4.30 greedyVMATsampling.vmat_class.objectiveValue

5.3.4.31 list greedyVMATsampling.vmat_class.openApertureMaps = [ ] [static]

5.3.4.32 string greedyVMATsampling.vmat_class.outputDirectory = "" [static]

5.3.4.33 list greedyVMATsampling.vmat_class.pointtoAngle = [ ] [static]

5.3.4.34 list greedyVMATsampling.vmat_class.regionIndices = [ ] [static]

5.3.4.35 list greedyVMATsampling.vmat_class.regions = [ ] [static]

5.3.4.36 list greedyVMATsampling.vmat_class.rlist = [ ] [static]

5.3.4.37 list greedyVMATsampling.vmat_class.scipygradient = [ ] [static]

5.3.4.38 list greedyVMATsampling.vmat_class.targets = [ ] [static]

5.3.4.39 int greedyVMATsampling.vmat_class.totaldijs = 0 [static]

num of nonzeros in Dij matrix

5.3.4.40 list greedyVMATsampling.vmat_class.voxelAssignment = [ ] [static]

5.3.4.41 list greedyVMATsampling.vmat_class.voxelgradient = [ ] [static]

5.3.4.42 greedyVMATsampling.vmat_class.voxelgradient

5.3.4.43 list greedyVMATsampling.vmat_class.xdirection = [ ] [static]

5.3.4.44 list greedyVMATsampling.vmat_class.xinter = [ ] [static]

5.3.4.45 list greedyVMATsampling.vmat_class.ydirection = [ ] [static]

5.3.4.46 list greedyVMATsampling.vmat_class.yinter = [ ] [static]

The documentation for this class was generated from the following file:

- greedyVMATsampling.py

Chapter 6

File Documentation

6.1 greedyVMATsampling.py File Reference

Classes

- class greedyVMATsampling.region
- class greedyVMATsampling.apertureList

  apertureList is a class definition of locs and angles that is always sorted.
- class greedyVMATsampling.vmat_class

  This defines the global VMAT class that contains most of the VMAT data to be used in the implementation Most of the values were defined as static attributes and only one instantiation at a time is possible.

Namespaces

- greedyVMATsampling

Functions

- def greedyVMATsampling.readctvoxelinfo
- def greedyVMATsampling.readDmatrix
- def greedyVMATsampling.calcObjGrad
- def greedyVMATsampling.fvalidbeamlets
- def greedyVMATsampling.PPsubroutine
- def greedyVMATsampling.parallelizationPricingProblem
- def greedyVMATsampling.PricingProblem
- def greedyVMATsampling.solveRMC
- def greedyVMATsampling.printresults
- def greedyVMATsampling.colGen
- def greedyVMATsampling.plotAperture
- def greedyVMATsampling.updateOpenAperture

Variables

- string greedyVMATsampling.__author__ = 'wilmer'
- greedyVMATsampling.have_mkl = True

- list greedyVMATsampling.kappa = [6, 17, 28, 39, 50, 61, 72, 83, 94, 105, 116, 127, 138, 149, 160, 171, 11, 22, 33, 44, 55, 66, 77, 88, 99, 110, 121, 132, 143, 154, 165, 1, 175, 14, 25, 36, 47, 58, 69, 80, 91, 102, 113, 124, 135, 146, 157, 168, 3, 8, 19, 30, 41, 52, 63, 74, 85, 96, 107, 118, 129, 140, 151, 162, 172, 176, 0, 2, 4, 5, 7, 9, 10, 12, 13, 15, 16, 18, 20, 21, 23, 24, 26, 27, 29, 31, 32, 34, 35, 37, 38, 40, 42, 43, 45, 46, 48, 49, 51, 53, 54, 56, 57, 59, 60, 62, 64, 65, 67, 68, 70, 71, 73, 75, 76, 78, 79, 81, 82, 84, 86, 87, 89, 90, 92, 93, 95, 97, 98, 100, 101, 103, 104, 106, 108, 109, 111, 112, 114, 115, 117, 119, 120, 122, 123, 125, 126, 128, 130, 131, 133, 134, 136, 137, 139, 141, 142, 144, 145, 147, 148, 150, 152, 153, 155, 156, 158, 159, 161, 163, 164, 166, 167, 169, 170, 173, 174, 177]
- greedyVMATsampling.WholeCircle = False
- string greedyVMATsampling.rootFolder = '/media/wilmer/datadrive'
- string greedyVMATsampling.readfolder = rootFolder+'/Data/DataProject/HN/'
- string greedyVMATsampling.readfolderD = readfolder+'Dij/'
- string greedyVMATsampling.outputfolder = '/home/wilmer/Dropbox/Research/VMAT/output/'
- int greedyVMATsampling.degreesep = 60
- string greedyVMATsampling.objfile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/objectives/obj1.txt'
- string greedyVMATsampling.structurefile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/structureInputs.txt'
- string greedyVMATsampling.algfile = '/home/wilmer/Dropbox/IpOptSolver/TestData/HNdata/algInputsWilmer.txt'
- string greedyVMATsampling.mm3voxels = rootFolder+'/Data/DataProject/HN/hn3mmvoxels.mat'
- list greedyVMATsampling.priority = [7, 24, 25, 23, 22, 21, 20, 16, 15, 14, 13, 12, 10, 11, 9, 4, 3, 1, 2, 17, 18, 19, 5, 6, 8]
- list greedyVMATsampling.mylines = [line.rstrip('\n') for line in open('/home/wilmer/Dropbox/Research/VMAT/VMATwPenCode/beamAngles.txt')]
- tuple greedyVMATsampling.fig = plt.figure(1)
- list greedyVMATsampling.catemp = [ ]
- list greedyVMATsampling.gatemp = [ ]
- tuple greedyVMATsampling.start = time.time()
- tuple greedyVMATsampling.data = vmat_class()
- tuple greedyVMATsampling.oldfolder = os.getcwd()

  •••••••••###########################
- tuple greedyVMATsampling.allFiles = glob.glob("∗VOILIST.mat")
- tuple greedyVMATsampling.allBeamInfos = glob.glob("∗Couch0_BEAMINFO.mat")
- tuple greedyVMATsampling.allNames = sorted(allFiles)
- tuple greedyVMATsampling.allBeamInfoNames = sorted(allBeamInfos)
- tuple greedyVMATsampling.numStructs = len(allFiles)
- tuple greedyVMATsampling.vdims = readctvoxelinfo()
- list greedyVMATsampling.numVoxels = vdims['x']
- list greedyVMATsampling.Vorg = [ ]
- tuple greedyVMATsampling.bigZ = np.zeros(numVoxels, dtype=int)
- tuple greedyVMATsampling.nVox = sum(bigZ)
- int greedyVMATsampling.counter = 0
- tuple greedyVMATsampling.originalVoxels = np.empty(numVoxels)
- list greedyVMATsampling.lines = [myline.split('\t') for myline in [line.rstrip('\n') for line in open(structurefile)]]

  Read in structures .
- list greedyVMATsampling.invec = [item for sublist in lines for item in sublist]

  Collapse the above expression to a flat list.
- tuple greedyVMATsampling.maskValueFull = np.zeros(nVox.astype(np.int64))

  Assignation of different values.
- tuple greedyVMATsampling.maskValueSingle = np.zeros(nVox.astype(np.int64))
- list greedyVMATsampling.s = priority[i]
- tuple greedyVMATsampling.norepeat = np.unique(originalVoxels[np.invert(np.isnan(originalVoxels))])
- tuple greedyVMATsampling.istarget = str(s)
- list greedyVMATsampling.tempindicesfull = originalVoxels[Vorg[s]]

- tuple greedyVMATsampling.tempindices = np.intersect1d(tempindicesfull, norepeat)
- int greedyVMATsampling.gastart = 0
- int greedyVMATsampling.gaend = 356
- int greedyVMATsampling.gastep = 2
- list greedyVMATsampling.ga = [ ]
- list greedyVMATsampling.ca = [ ]
- string greedyVMATsampling.fname = 'Gantry'

  Treatment of BEAMINFO data.
- string greedyVMATsampling.bletfname = readfolder+'Gantry'
- tuple greedyVMATsampling.nBPB = np.zeros(len(ga))
- tuple greedyVMATsampling.nDIJSPB = np.zeros(len(ga))
- tuple greedyVMATsampling.beamletCounter = np.zeros(data.numbeams + 1)

  Beginning of Troy's cpp code (interpreted, not copied) A comment This comes from first two lines in doseInputs txt file (troy's version)
- tuple greedyVMATsampling.binfoholder = sio.loadmat(bletfname)
- tuple greedyVMATsampling.N = len(data.yinter)

  After reading the beaminfo information.
- tuple greedyVMATsampling.M = len(data.xinter)
- int greedyVMATsampling.overallDijsCounter = 0

  Initial intensities are allocated a value of zero.
- list greedyVMATsampling.DlistT = [None]
- tuple greedyVMATsampling.pool = Pool(processes=8)
- tuple greedyVMATsampling.Allmats = pool.map(readDmatrix, range(0, data.numbeams))
- list greedyVMATsampling.ininter = [ ]
- greedyVMATsampling.functionData = data.functionData

  Collapse the above expression to a flat list.
- tuple greedyVMATsampling.quadHelperThresh = np.zeros(data.numvoxels)
- tuple greedyVMATsampling.quadHelperOver = np.zeros(data.numvoxels)
- tuple greedyVMATsampling.quadHelperUnder = np.zeros(data.numvoxels)
- tuple greedyVMATsampling.quadHelperAlphaBetas = np.zeros(data.numvoxels)
- tuple greedyVMATsampling.uDose = np.zeros(data.numvoxels)
- tuple greedyVMATsampling.oDose = np.zeros(data.numvoxels)
- tuple greedyVMATsampling.before = time.time()
- tuple greedyVMATsampling.pstar = colGen(0, WholeCircle, 16)
- tuple greedyVMATsampling.after = time.time()

Index