ND-Tile

API Documentation

November 17, 2016

Contents

1 Module Tiling

Tile an N-Dimensional Domain containing Point objects depending on a Tile decision function.

1.1 Class OutputWriter

object
    Tiling.OutputWriter

1.1.1 Methods

__init__(*self*, *otype*=`None`, *null*=`False`)

If otype==None, then OutputWriter will print to stdout. Otherwise, open a file named otype for writing. If null, write immediately returns and does nothing.

Overrides: object.__init__

close(*self*)

If OutputWriter has a file open, then close it.

write(*self*, *content*)

Print content to stdout if no file object is available in ofile. If there is a file object in ofile, then write content to the file.

When writing content to a file, append a newline.

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.1.2 Properties

Name	Description
Inherited from object	
__class__	

1.2 Class BCTypes

object
Tiling.BCTypes

1.2.1 Methods

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.2.2 Properties

Name	Description
Inherited from object	
__class__	

1.2.3 Class Variables

Name	Description
up	**Value:** `+ 1`
none	**Value:** `None`
down	**Value:** `-1`
tile	**Value:** `+ 2`
point	**Value:** `+ 3`
all_types	**Value:** `+ 4`

1.3 Class BCDim

object
Tiling.BCDim

1.3.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.3.2 Properties

Name	Description
Inherited from object __class__	

1.4 Class TETypes

object
Tiling.TETypes

Tiling Error Types

1.4.1 Methods

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __init__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.4.2 Properties

Name	Description
Inherited from object __class__	

1.4.3 Class Variables

Name	Description
cannot_start_point	**Value:** `0`
cannot_enclose_enough_points	**Value:** `1`
few_points_remain	**Value:** `2`

1.5 Class DMCycle

object
Tiling.DMCycle

1.5.1 Methods

__**init**__(*self*, *dm*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

cycle(*self*)

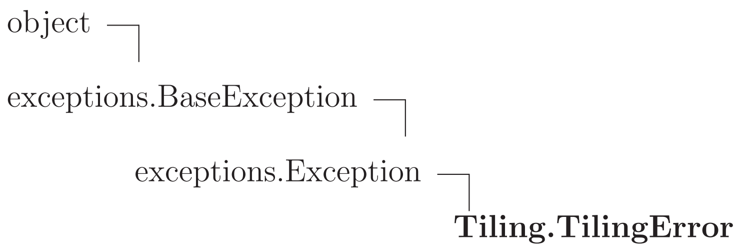
Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.5.2 Properties

Name	Description
Inherited from object	
__class__	

1.6 Class TilingError



Error class for various kinds of tiling errors that can arise.

1.6.1 Methods

__init__(*self*, *err_type*, *err_tile*=`None`, *scratch_points*=`None`, *message*=`''`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from exceptions.Exception

__new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

__format__(), __hash__(), __reduce_ex__(), __sizeof__(), __subclasshook__()

1.6.2 Properties

Name	Description
Inherited from exceptions.BaseException	
args, message	

continued on next page

Name	Description
Inherited from object __class__	

1.7 Class Point

object ─┐
Tiling.Point

1.7.1 Methods

__init__(*self*, *r*=[], *v*=None)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

norm_dist_to_pt(*self*, *b*)

order_nn(*self*, *plist*=[])

get_average_dist_nn(*self*, *plist*=[], *num_neighbors*=1)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.7.2 Properties

Name	Description
Inherited from object __class__	

1.8 Class Plane

object ─┐
Tiling.Plane

1.8.1 Methods

__init__(*self*, *points*=`None`, *fit_guess*=`[]`, *dm*=`None`, *lo*=`[]`, *hi*=`[]`, *writer*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

close(*self*)

Manually cleanup. Used for closing open file handles.

print_fit_report(*self*, *writer*=`None`)

Prints report of the fit to the writer. If writer==None, use self.writer.

compute_pars(*self*, *points*, *fit_guess*=`[]`)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.8.2 Properties

Name	Description
Inherited from object __class__	

1.9 Class Tile

object

Tiling.Tile

1.9.1 Methods

__init__(*self*, *points*=`[]`, *lo*=`[]`, *hi*=`[]`, *fit_guess*=`[]`, *dm*=`None`, *smask*=`None`, *virtual*=`False`, *writer*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

close(*self*)

Manually cleanup. Used for closing open file handles.

extend_dimension(*self*, *di*, *dx*, *surface*, *direction*)

get_dim_thickness(*self*, *di*)

Given the dimension di, return the thickness of this tile along di.

get_thinnest_dimension(*self*)

Find the dimension di in which this Tile is thinnest.

Also find the thickness dx along dimension di.

Return (di, dx)

order_thinnest_dimensions(*self*)

Return the dimensions of this Tile in a list ordered by the Tile thickness in each dimension from smallest to largest.

colocated_with(*self*, *btile*, *di*=`-1`)

Determine whether self and btile are colocated.

If the optional argument di is provided, determines whether self and btile are colocated only considering dimension di.

gen_vertices(*self*)

Return a generator for the vertices of this Tile.

create_surface(*self*, *di*, *surface*)

Make and return the surface of this Tile defined by the constant dimension (di) where the surface normal of Tile along di on this surface lies in the direction given by (surface)

get_surfaces(*self*, *dj*=`-1`)

Return the surfaces for this Tile as Tile objects.

The distinguishing feature of the surface relative to this Tile is that, although the surface and Tile are of the same dimensionality, there is at least one dimension di in which the surface tile has lo[di] == hi[di] == constant.

In case this tile is already a surface, then only set the constraint lo[di] == hi[di] == constant if lo[di] != hi[di].

If dj is provided, only return surfaces for which lo[dj] == hi[dj] == constant.

Otherwise return all such surfaces if dj is not provided (-1). I'm using -1 here because 0 is a valid dimension but tests as a boolean False.

If dj is provided and it is not a nonconstant dimension, then return an empty list.

get_constant_dimensions(*self*)

Find all dimensions di for which lo[di] == hi[di] == constant

Return a list of tuples [(di, constant), ...] satisfying that condition.

get_nonconstant_dimensions(*self*)

Find all dimensions di for which lo[di] != hi[di]

Return a list of such dimensions di.

print_tile_report(*self*, *tile_number*=`None`, *writer*=`None`)

Prints report of this Tile to the writer. If writer==None, use self.writer.

print_fit_report(*self*, *writer*=`None`)

Prints report of the fit on this Tile to the writer. If writer==None, use self.writer.

get_volume(*self*, *dom_lo*=`[]`, *dom_hi*=`[]`)

Computes volume of the Tile. If [lo, hi] is undefined, return None. If dom_lo and dom_hi are supplied, normalize the tile dimensions by the dimensions of domain lo and hi first before computing a normalized volume.

boundary_minimize(*self*)

Given the points in the Tile, set the boundary defined by [lo, hi] to the minimum surface enclosing the points.

extend_points(*self*, *plist*=`[]`)

Given the list of points (plist), extends the Tile if necessary to enclose them.

Set the Tile boundaries to the minimum volume enclosing the provided points.

Do nothing if no points are provided.

overlaps_point_dimension(*self*, *refpoint*, *di*)

Checks to see if self overlaps refpoint in the dimension di:

refpoint must be a Point object

di must be an integer in range(self.dm)

get_point_occlusions(*self*, *points*, *di*)

Given a list of points (points), find the points which will occlude self along dimension di and thus can set bounds.

Return a list of such points.

get_point_constraints(*self*, *points*, *di*, *bcdi*=`None`)

Get point based [lo, hi] constraints along dimension di for this Tile.

Return boundary conditions along dimension di in bcdi.

Calculates point occlusions from the list of points (points)

overlaps_tile_dimension(*self*, *reftile*, *di*)

Checks to see if self overlaps reftile in the dimension di:

reftile must be a Tile object

di must be an integer in range(self.dm).

overlaps_tiles(*self*, *tlist*=`[]`)

Checks to see if self overlaps any of the tiles in tlist

Returns list of tiles in tlist which overlap self

Returns the empty list if no tiles in tlist overlap self

get_tile_occlusions(*self*, *tiles*, *di*)

Given a list of tile objects (tiles), find the tiles in tiles which are not self which will occlude self along dimension di and thus can set bounds.

Return a list of such tiles.

whether_occludes_tile(*self*, *atile*, *di*)

Determine whether self and atile occlude along dimension di.

Return True if they occlude, False otherwise.

By design, occlusion is false if the tiles overlap along dimension di or if they are the same tile.

whether_osculates_tile(*self*, *atile*, *di*, *direction*=`BCTypes.none`)

Determine whether self and atile osculate on any surface in dimension di.

If direction is supplied, then the osculation surface relative to self should have the surface normal mask equal to direction.

Find the surface of self which is osculated (sface)

Also find the ctile which is the intersection of the osculating surfaces of self and atile.

Return (sface, ctile) in case atile osculates self.

Otherwise return (None, None)

get_tile_intersection(*self*, *atile*)

Return the tile which is the intersection of self and atile.

get_tile_constraints(*self*, *tiles*, *di*, *bcdi*=`None`)

Get tile based [lo, hi] constraints along dimension di for this Tile.

Return boundary conditions along dimension di in bcdi.

Calculates tile occlusions from the list tiles.

get_hypothetical_extend(*self*, *points*=`[]`, *avoid_tiles*=`None`, *greedy_absorb_points*=`[]`)

extend_min_volume(*self*, *plist*=`[]`, *avoid_tiles*=`None`, *decision_fun*=`None`, *dom_lo*=`[]`, *dom_hi*=`[]`)

Given the list of points (plist), extends the Tile by adding one point from plist to Tile where the point is selected from plist such that it minimizes the normalized volume of Tile. The normalized volume is the product of tile dimensions, each normalized by the extent of the domain in each dimension, passed via dom_lo and dom_hi arguments. If either of those arguments are not supplied, then do not normalize the volume.

Returns plist where the selected point is popped from the list.

If avoid_tiles is passed, it should be a list of Tile objects. The current Tile will then only be extended such that it does not intersect the tiles in avoid_tiles.

If a function is passed as decision_fun, this Tile will be passed to the 'decision function' to determine whether to extend the tile. decision_fun should take a single Tile argument and return True or False

which_points_within(*self*, *pointlist*=`[]`, *lo*=`[]`, *hi*=`[]`)

get_subtile(*self*, *lo*, *hi*)

do_plane_fit(*self*)

get_coeff_det(*self*)

get_L2_norm_resd(*self*)

get_tilde_resd(*self*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.9.2 Properties

Name	Description
Inherited from object __class__	

1.10 Class Domain

object
    Tiling.Domain

1.10.1 Methods

__init__(*self*, *points*=`[]`, *lo*=`[]`, *hi*=`[]`, *dm*=`None`, *plot_lo*=`[]`, *plot_hi*=`[]`, *point_normalize*=`True`, *plot_dimfrac*=`0.9`, *last_domain_slice*=`(None,None)`, *dlabels*=`[]`, *ilabel*=`None`, *logfile*=`None`, *summaryfile*=`None`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

close(*self*)

Manually cleanup. Used for closing open file handles.

photogenic_plot_limits(*self*, *dimfrac*=`0.9`)

Compute the appropriate plot limits [self.plot_lo, self.plot_hi] for which 2-D domain slices will plot the domain [self.lo, self.hi] such that the domain extents in each dimension occupy the fraction dimfrac of their corresponding axis.

plot_domain_slice(*self*, *dimx*=`0`, *dimy*=`1`, *save_num*=`None`, *show_tile_id*=`True`, *save_last_figure*=`False`, *underlay_figure_axis*=`None`, *show_plot*=`False`)

plot_domain_slice_scratch(*self*, *stile*, *dimx*=`0`, *dimy*=`1`, *save_num*=`None`, *show_tile_id*=`True`, *save_last_figure*=`False`, *underlay_figure_axis*=`None`, *show_plot*=`False`)

Given a scratch tile, add it to the tiles in this domain, plot the domain slice, and then pop the tile from the domain tiles so the domain is unaffected.

print_domain_report(*self*, *writer*=`None`)

bc_init_mask_points(*self*, *plist*)

propagate_tile_perturbation(*self*, *from_tile*, *di*, *dx*, *surface*, *direction*, *ignore_tiles*=`[]`, *dry_run*=`False`)

Extend the boundaries of from_tile along dimension di by length dx.

Find the other tiles in the domain which extending this boundary would either interfere with or pull away from if it already osculates them along dimension di. Correct those tile dimensions and propagate the changes throughout the domain recursively. Never shrink a tile by more than dx if it's of width dx or smaller. In that case, return False to indicate the propagation could not succeed.

The surface mask smask of this Tile will be checked and if it is not equal to BCTypes.none to indicate this Tile is a surface along dimension di, then expand the surface outwards in the direction of smask.

Surface and Direction should be either BCTypes.up or BCTypes.down to indicate whether lo or hi is to be shifted and in what direction.

If ignore_tiles, then ignore propagating the perturbation to the tiles in ignore_tiles. This is an aid for recursively propagating throughout the domain.

If dry_run == True, then do not actually perform any propagation but do check the domain tiles recursively to see if the propagation is allowed.

Returns False if the perturbation could not be applied to from_tile, returns True otherwise.

multi_propagate_tile_perturbation(*self*, *tosc*, *di*, *dx*, *surface*, *direction*, *ignore_tiles*=`[]`, *dry_run*=`False`)

Wrapper for propagate_tile_perturbation that takes a list tosc of (btile, sface, ctile) where btile is to be perturbed and propagated in direction if sface's surface mask matches surface.

tosc contents are as returned by self.get_osculating_tiles.

Returns True if the propagation succeeded, False otherwise.

Only actually applies the propagation if dry_run == False.

get_tile_boundaries(*self*, *atile*, *di*, *allow_bc_types*=`[BCTypes.all_types]`)

Given atile and a dimension di, return the [lo, hi] boundaries in a BCDim object.

Account for the types of boundary conditions listed in allow_bc_types.

set_tile_boundaries(*self*, *atile*, *allow_bc_types*=`[BCTypes.all_types]`)

Given atile, sets its [lo, hi] boundaries in each dimension.

Also updates the boundary masks for adjacent points in the tiling list scratch_points.

tiling_decision_function(*self*, *L2r_thresh*=`None`, *coeff_det_thresh*=`None`, *tilde_resd_thresh*=`None`, *tilde_resd_factor*=`None`)

form_tile(*self*, *decision_function*=`None`, *plot_intermediate*=`False`)

extend_existing_tiles(*self*, *decision_function*=`None`)

Extends all existing tiles, gobbling up scratch_points as possible.

Do this by figuring out which tile can best include each of the remaining scratch_points, given the decision_function constraint.

bound_existing_tiles(*self*)

Given the tiles in self.tiles, update all their tile-based boundaries until no further updates can be made.

get_osculating_tiles(*self*, *atile*, *di*, *direction*=`BCTypes.none`, *get_other_sface*=`False`, *return_other_tile*=`False`)

Get the tiles in Domain which osculate atile along the dimension di. Return them as a list tosc = [(sface, ctile), ...] where sface and ctile are as in Tile.whether_osculates_tile

If direction == BCTypes.up or direction == BCTypes.down then only return the tiles which osculate atile such that the osculating surface of atile has a surface normal oriented along direction. Otherwise, if direction == BCTypes.none, return tiles which osculate atile in any direction along di. Note that this direction should be relative to atile regardless the value of get_other_sface.

If get_other_sface, then sface will correspond to the surface of the Tile which osculates self.

If return_other_tile, will return the tuples [(stile, sface, ctile), ...] where stile is the Tile of which sface is the surface.

do_empty_tiling(*self*, *plot_tile_surfaces*=`False`)

Creates empty virtual subtiles to cover the domain dom.

Adds the created virtual subtiles to the domain dom.

Returns True if virtual Tiles were created.

Returns False if no virtual Tiles could be created.

Because the Domain Tile loop doesn't update itself as Tiles are added to the Domain, you should loop over this function until it returns None to indicate the entire Domain has been Tiled.

shrink_virtual_tiles(*self*)

Shrink a virtual tile V in the domain to zero volume by rearranging neighboring tiles.

The algorithm is outlined below: Pop a virtual tile V off the Domain's list of virtual_tiles. Shrink virtual tile V by identifying its thinnest dimension di (of width W) and finding the tiles B of maximum volume with surfaces S which V osculates along di (B may be up or down relative to di, but not both). Take the surfaces S and form a virtual tile (SW) of thickness W extending from S in the direction of V. Find all tiles T, T != V, which SW overlaps. Shrink all tiles T away from B in the dimension di by length W. Expand tiles B into the volume of SW. Remove virtual tile V from domain. Return and Repeat until no virtual tiles remain.

Real or virtual tiles will have a problem if they osculate the virtual tile but are thinner than the virtual tile in the osculating dimension. To get around that, shrink_virtual_tiles should check to see if its smallest dimension is not thicker than its osculating tiles along that dimension. If that's not true, then it will not be possible to reduce such a virtual tile.

Reallocate points to real tiles and repeat fitting to update stats. This has to be done in whatever code calls this function.

create_virtual_tiles(*self*, *make_plots*=`False`, *plot_tile_surfaces*=`False`)

Tile all untiled space in the domain into virtual tiles and add them to self.virtual_tiles.

Returns True if a new virtual tile was created. Returns False otherwise.

static_tile_assign_points(*self*)

Statically assign points to the tiles in the Domain and update the fits on those tiles. Existing points in the tiles are reset to only those points assigned here.

do_domain_tiling(*self*, *L2r_thresh*=`None`, *coeff_det_thresh*=`None`, *tilde_resd_thresh*=`None`, *tilde_resd_factor*=`None`, *attempt_virtual_shrink*=`False`, *plot_tile_surfaces*=`False`, *plot_intermediate*=`False`, *plot_tiling*=`False`, *plot_final*=`True`)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.10.2 Properties

Name	Description
Inherited from object __class__	

Index