**DIFFERENCE BETWEEN LAR.PY AND LAR.JS**

|  |  |
| --- | --- |
| **LAR.py** | **LAR.js** |
| **GEOMETRY LAYER (USING PYPLASM)** | **UTILITY FUNCTION (oggetto *utils*)** |
| * ***View*** * ***Bezier*** (To create a Bezier curve of degree n from a list of n+1 d-points. Each point is given as a list of coordinates.) * ***CCOMB*** (To create the convex combination of a list of vectors. Each vector is given as a list of coordinates.) * ***EXPLODE*** (To explode a HPC scene, given three real scaling parameters- with EXPLODE0) * ***MKPOLS*** (To MaKe a list of HPC objects from a LAR model.) * ***LAR2PLASM*** (To transform a topological relation from LAR format (base-index = 0, like C or python)) * ***VERTS*** (To generate the vertices of a grid of points from a list of d lists (of equal length) of numbers.) * ***VERTEXTRUDE*** (Utility function to generate the output model vertices in a multiple extrusion of a LAR model.) * ***format*** (Transform from list of triples (row,column,vale) to scipy.sparse corresponding formats.) | * ***compareVertices*** * ***compareCells*** * ***Fill*** (Create an m x n k-filled matrix) * ***zeros*** (Create an m x n 0-filled matrix) * ***ones*** (Create an m x n 1-filled matrix) * ***flat*** (Return a flat version of the given array of arrays) * ***select*** (Return selected row in the given matrix as array of arrays) * ***chainPut1*** (Return an array of 0s but indexes set to 1) * ***chainPut0*** (Return an array of 1s but indexes set to 0) * ***binarize*** (Transformation from \*compressed matrix - cmat\* to \*binary matrix\*Return a (m x n) \*binary matrix\*) * ***unbinarize*** (Transformation from \*binary matrix\* to \*compressed matrix\*Return a (m x n) \*compressed matrix\*) |
| **BASIC LAR SOFTWARE LAYER** | **LAR BASIC OPERATIONS (oggetto ops)** |
| * ***cooCreateFromBrc*** * ***csrCreateFromCoo*** * ***csrCreate*** * ***csrGetNumberOfRows*** * ***csrGetNumberOfColumns*** * ***csrToMatrixRepresentation*** * ***csrToBrc***   (**coo** is the standard rep using non-ordered triples of numbers) | * ***extract*** (Return C\_{k-1,0} cells) * ***extrude*** (Return C\_{k+1,0} cells) * ***combine*** (Return C\_{i,j} = C\_{i,0} x C\_{j\_0}^T) * ***extract0*** (Return C\_{0,0} = C\_{1,0}^T x C\_{1\_0}) * ***boundarize*** (Makes C\_{i,i+1} matrix boundary operator) |
| **TOPOLOGY INTERFACE LAYER** | **TOPOLOGY** |
| * ***csrCreateTotalChain*** * ***csrCreateUnitChain*** * ***csrExtractAllGenerators*** * ***csrChainToCellList*** | * ***Topology*** (@constructor @param {Array} cells array of arrays of C\_d0 cells as vertices index) * ***get*** (Return C\_{i,j} matrix) * ***boundary*** (Return C\_{i,i+1} boundary operator matrix) * ***coboundary*** (Return C\_{i,i-1} boundary operator matrix) * ***extrude*** (Return extruded cells) |
| **TOPOLOGY QUERY LAYER** |
| * ***larCellAdjacencies*** * ***larCellIncidences*** * ***larBoundary*** * ***larBoundaryChain*** * ***larCoboundaryChain*** |
| **MODEL GEOMETRY LAYER** | **MODEL** |
| * ***larProduct*** (binary product of cell complexes) * ***cumsum*** (cumulative addition) * ***larExtrude*** (extrusion of simplicial complexes) * setup (extraction of facets of a cell complex) * ***larFacets*** (Estraction of (d-1)-cellFacets from model) * ***larSkeletons*** (extraction of skeletons of a (grid) cell complex) * ***larFacets*** (Estraction of (d-1)-cellFacets from model) * ***boundarGrid*** (Build the set of the outerCells of a cuboidal model) * ***outerVertexTest*** (Look whether v is on the boundary of a unconnected (multi-dim) interval [vmin\_0,vmax\_0, ... ,vmin\_n,vmax\_n]) | * ***Model*** (@constructor, @param {Array} vertices array of arrays of vertices coordinate,@param {Array} cells array of arrays of C\_d0 cells as vertices index) * ***isEmpty*** (Test if the model is empty) * ***getChain*** (Return list of ordered d-cells) * ***getCells*** (Return list of ordered d-cells) * ***submodel*** (Return a chain as a Model) * ***skeleton*** (Return d-skeleton of the model) * ***map*** (Return C\_{i,j} applied to j-chain) * ***boundary*** (Return p-boundary of the given p-chain) * ***coboundary*** (Return p-coboundary of the given p-chain) * ***extrude*** (Extrude model by a list of quotes) * ***orient*** (Orient model cells – ***not implemented?***) * ***filter*** (Filter out unindexed vertices) * ***clone*** (Return a deep cloned copy of this model) |
| **MATRIX UTILITY LAYER** |
| * ***csrIsA*** * ***csrGet*** * ***csrSet*** * ***csrAppendByRow*** * ***csrAppendByColumn*** * ***csrSplitByRow*** * ***csrSplitByColumn*** |
| **SPARSE MATRIX OPERATIONS LAYER** |
| * ***csrMaxFilter*** * ***csrBoundaryFilter*** * ***csrBinFilter*** * ***csrPredFilter*** |
| **IMAGING LAYER (ENUMERATIVE SCHEMES W INTEGER COORDS)** |
| crossGrow  xbar  ybar  delta  write2DBlock  read2DBlock  lar2DFromImageBlocks  write3DBlock  read3DBlock  lar3DFromImageBlocks |
| **APPLICATION LAYER (DEMO)** |
| **…** |

**Leggenda:**

esempio – dove viene eseguita una operazione di moltiplicazione tra matrici;