Solid primitives by domain mapping

The `mapper.jl` file contains the implementation of several parametric primitives, including curves, surfaces and solids embedded in either 2D or 3D.

The constructive approach is common to all methods. It consists in generating a simplicial or cuboidal decomposition of a simple geometrical domain in u,v or u,v,w parametric space. Then a change of coordinates, e.g. from Cartesian to polar or cylindrical coordinates, is applied to the vertices of the cellular complex decomposing the domain.

So, the mapped domain produces a curved manifold in 2D or 3D space. To obtain a closed curved surface, i.e. a manifold-without-boundary, as in the case of a 2-sphere in 3D, or of the toroidal surface in 3D, a suitable identification of oincident mapped points is performed.

List of currently available primitives

The mapper module aims to provide the tools needed to apply both dimension-independent affine transformations and general simplicial maps to geometric objects and assemblies developed within the LAR scheme. A large number of surfaces and primitives solids are definable using the map function and the local parametrizations.

Curves

1. Primitive one-dimensional objects:

- `circle` - Circle centered in the origin
- `helix` - Helix curve about the z axis

Surfaces

2. Primitive two-dimensional objects:

- `disk` - Disk centered in the origin
- `helicoid` - Helicoid about the z axis
- `ring` - Ring centered in the origin
- `cylinder` - Cylinder surface with z axis
- `sphere` - Spherical surface of given radius
- `toroidal` - Toroidal surface of given radiuses
- `crown` - Half-toroidal surface of given radiuses

Solids

3. Primitive three-dimensional objects:

- `box` - Solid box of given extreme vectors
- `ball` - Solid Sphere of given radius
- `rod` - Solid cylinder of given radius and height
- `hollowCyl` - Hollow cylinder of given radiuses and height
- `hollowShere` - Hollow sphere of given radiuses
- `torus` - Solid torus of given radiuses
- `pizza` - Solid pizza of given radiuses

Implementation

The coding of the generating functions for the various geometric primitives follows the below guidelines:

- **Higher level function interface.** Every generating function is of type

$$fun : parms_1 \rightarrow (parms_2 \rightarrow results),$$

  with $parms_1 = p_1 \times p_2 \times cdots \times p_m$ and $parms_1 = q_1 \times q_2 \times cdots \times q_n$. The p_i parameters concern the specification of the coordinate functions of the mapping. The q_j parameters ($1 \leq j \leq n \in \{1, 2, 3\}$) affect the discretization of mapping domain.

- *Simplicial or cuboidal decomposition.* Different discretization primitives `simplexGrid()` or `larCuboids()` are used for the two cases. Both primitives are dimension-independent, i.e. may decompose 1D, 2D, 3D,..., nD domains, depending only on the array `shape` of the generated cellular complex. The complex is generated in LAR format `(vertices,cells)`, where vertices have integer coordinates.

- *Coordinate functions.* Are applied to the integer `vertices`, so producing their mapped instances and store them in a `Array{Array{Int64,1},1}`

- *Complex simplification.* Finally, the geometrically coincident vertices are identified, the generated cells are translated to the new vertex indices, and cells are simplified from multiple identical indices. This may induce the *sewing* of domain boundaries according to expected topology of the curved manifold and/or the reduction of independent vertices in cells of the complex.