# Manual on cell_geom.py

April 1, 2016

## 1 Table of Contents



## 2 Overview

In this file `class UnitCell` is defined, where possible inclusions can be added to the unit cell. The member methods of this class are constructor, `set_append_inclusion`, `add_mark_boundary`, `view_mesh`, and `view_domain`. The instance of this method is instantiated with a `Mesh` object in FEniCS. A `UnitCell` instance can be either two dimensional or three dimensional.

Classes for creation of inclusions are included in the current file, namely `InclusionCircle` and `InclusionRectangle`. Besides, `PeriodicBoundary_no_corner` is a class specifying the periodic map for periodic boundary condition in homogenization problem.

## 3 Inclusions

Setting a unit cell and its inclusions is introduced in this part. We first import modules

```
In [2]: from dolfin import *
        import sys
        sys.path.append('../')
        import cell_geom as geom
```

### 3.1 2D Case

**Import mesh and instantiation**

```
In [3]: mesh = Mesh(r"../m.xml")

        # Generate Inclusion
        inc1 = geom.InclusionCircle(2, (0.5, 0.5), 0.25)
        inc_group = {'circle_inc1': inc1}

        # Initiate UnitCell Instance with Inclusion
        cell = geom.UnitCell(mesh, inc_group)
        cell.view_domain()
```

**Multiple inclusions and append inclusion**

```
In [7]: mesh = UnitSquareMesh(40, 40, 'crossed')

        # Instantiation with inclusions
        inc1 = geom.InclusionCircle(2, (0.1, 0.1), 0.5)
        inc2 = geom.InclusionCircle(2, (0.9, 0.9), 0.5)
        inc_group_1 = {'circle_inc1': inc1, 'circle_inc2': inc2,}
        cell = geom.UnitCell(mesh, inc_group_1)
        cell.view_domain()

In [8]: # Another group of inlusions
        inc3 = geom.InclusionRectangle(2, 0.1, 0.3, 0.7, 0.9)
        inc4 = geom.InclusionRectangle(2, 0.7, 0.9, 0.1, 0.3)
        inc_group_2 = {'rect_inc3': inc3, 'rect_inc4': inc4}

        # Append inclusions and view
        cell.set_append_inclusion(inc_group_2)
        cell.view_domain()
```

### 3.2 3D Case

**Multiple inclusions and append inclusion**

```
In [11]: mesh = UnitCubeMesh(20, 20, 20)

         # 9 Inclusions with 8 corner inclusions and one sphere inclusion in the center
         inc = geom.InclusionCircle(3, 0.5)
         inc1 = geom.InclusionRectangle(3, 0., 0.3, 0., 0.3, 0., 0.3)
         inc2 = geom.InclusionRectangle(3, 0., 0.3, 0., 0.3, 0.7, 1.)
         inc3 = geom.InclusionRectangle(3, 0., 0.3, 0.7, 1., 0., 0.3)
         inc4 = geom.InclusionRectangle(3, 0., 0.3, 0.7, 1., 0.7, 1.)
         inc5 = geom.InclusionRectangle(3, 0.7, 1., 0., 0.3, 0., 0.3)
         inc6 = geom.InclusionRectangle(3, 0.7, 1., 0., 0.3, 0.7, 1.)
         inc7 = geom.InclusionRectangle(3, 0.7, 1., 0.7, 1., 0., 0.3)
         inc8 = geom.InclusionRectangle(3, 0.7, 1., 0.7, 1., 0.7, 1.)
         inc_group = {'circle': inc, 'corner1': inc1, 'corner2': inc2,
                      'corner3': inc3, 'corner4': inc4, 'corner5': inc5,
                      'corner6': inc6, 'corner7': inc7, 'corner8': inc8}
         cell = geom.UnitCell(mesh, inc_group)

         cell.view_domain()
```

## 4 Peirodic Boundary Condition

Periodic mapping for FunctionSpace initiallization. Both 2D case and 3D case are covered. This periodic mapping excludes corners of unit cell. In unit cell computation these corners are set fixed to prevent rigid body movement.

```
In [16]: # 2D
         a, b = 3, 6
         mesh_2d = UnitSquareMesh(a, b)
         FS_2d = geom.FunctionSpace(mesh_2d, 'CG', 1,
                                    constrained_domain=geom.PeriodicBoundary_no_corner(2))
         f = geom.Function(FS_2d)
```

```
        # DoF that are cancelled out
        print '2D periodic map'
        print 'original DoF =', (a + 1) * (b + 1), ';',
        print 'actual DoF =', f.vector().size(), ';',
        print 'the excluded DoF =', (a - 1 + b - 1)
```

```
2D periodic map
original DoF = 28 ; actual DoF = 21 ; the excluded DoF = 7
```

In [20]:
```
        # 3D
        a, b, c = 3, 6, 9
        mesh_3d = geom.UnitCubeMesh(a, b, c)
        FS_3d = geom.FunctionSpace(mesh_3d, 'CG', 1,
                                   constrained_domain=geom.PeriodicBoundary_no_corner(3))
        f = geom.Function(FS_3d)

        # DoF that are cancelled out
        print '3D periodic map'
        print 'original DoF =', (a + 1) * (b + 1) * (c + 1), ';',
        print 'actual DoF =', f.vector().size(), ';',
        print 'the excluded DoF =', (a - 1 + b - 1 + c - 1) * 3 + \
                (a - 1) * (b - 1) + (a - 1) * (c - 1) + (b - 1) * (c - 1)
```

```
3D periodic map
original DoF = 280 ; actual DoF = 169 ; the excluded DoF = 111
```