The 2DG Code

version 4.0

1 Data Structures

1.1 The `mesh` data structure

We will describe the `mesh` data structure with the help of the following (coarse) triangular mesh for the unit circle:

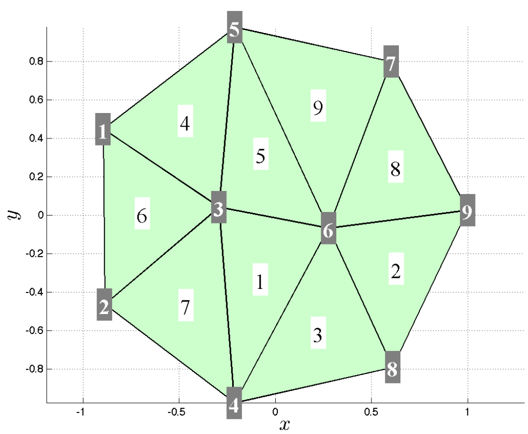

Figure 1: Example of a triangular mesh

The `mesh` data structure contains the following information:

- **The triangular linear mesh.** This mesh consists of `np` vertices and `nt` triangles. For the example in figure 1 we have `np = 9` and `nt = 9`.

`mesh.p[np,2]`: `x` and `y` coordinates of vertices in triangulation for simplicial mesh

```
>> mesh.p

ans =

   -0.8941    0.4479
   -0.8858   -0.4641
   -0.2922    0.0416
   -0.2113   -0.9774
   -0.2087    0.9780
    0.2769   -0.0665
    0.6029    0.7978
    0.6113   -0.7914
    0.9997    0.0243
```

`mesh.t[nt,3]`: Element vertices for simplicial mesh (numbered counterclockwise)

```
>> mesh.t

ans =

     4     6     3
     9     6     8
     8     6     4
     1     3     5
     5     3     6
     2     3     1
     4     3     2
     7     6     9
     7     5     6
```

- **Face and element connectivity information.** Here, `nf` is the number of faces (or edges in 2D) and can be calculated as `nf = (3*nt + nb)/2`, where `nb` is the number of boundary edges. For our example `nb = 7`, so `nf = 17`. We introduce two arrays: `mesh.f[nf,4]` and `mesh.t2f[nt,3]`.

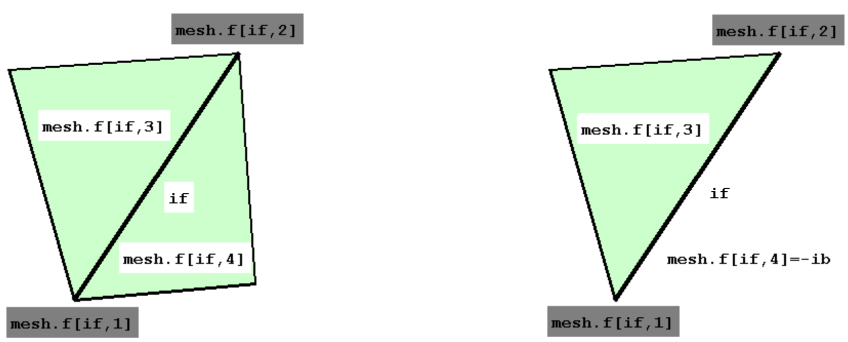

Figure 2: Definition of an interior face `if` (left), and a boundary face `if` on boundary `ib`.

A face is defined as the segment joining two vertices in the triangular mesh. `mesh.f(:,1:2)` are the indices of the two vertices in the mesh. `mesh.f(:,3)` is the index of the triangle to the left of the edge (when walking from `mesh.f(:,1)` to `mesh.f(:,2)`). For interior edges `mesh.f(:,4)` is the index of the triangle to the right of the edge. In addition, interior edges are always oriented so that `mesh.f(:,1) < mesh.f(:,2)`. For boundary edges `mesh.f(:,4)` is the *negative* of the boundary indicator. For the case of the circle we only have one boundary type and hence for these edges `mesh.f(:,4) = -1`. Note that this convention implies that all the boundary edges are oriented counterclockwise. Finally, for computational efficiency, the edges are ordered so that all interior edges are placed first and the boundary edges are last in the edge list.

`mesh.f[nf,4]`: `mesh.f(:,1:2)` are the indices of the two vertices in the mesh. `mesh.f(:,3)` is the index of the triangle to the left of the edge (when walking from `mesh.f(:,1)` to `mesh.f(:,2)`). Note that all boundary edges are last and that for all the interior edges `mesh.f(:,1) < mesh.f(:,2)`.

```
>> mesh.f

ans =

     3     6     5     1
     3     4     1     7
     4     6     1     3
     6     8     2     3
     6     9     8     2
     3     5     4     5
     1     3     4     6
     5     6     9     5
     2     3     6     7
     6     7     9     8
     8     9     2    -1
     4     8     3    -1
     5     1     4    -1
     1     2     6    -1
     2     4     7    -1
     9     7     8    -1
     7     5     9    -1
```

For this example there is only one geometric boundary `ib = -1`.

`mesh.t2f[nt,3]`: Triangle to face connectivity. `mesh.t2f[it,in]` contains the face number in element `it` which is opposite node `in` of element `it`. If the face orientation matches the element counterclockwise orientation then the face number is stored, otherwise the negative of the face number is stored.

```
>> mesh.t2f

ans =

    -1     2     3
     4    11    -5
    -3    12    -4
     6    13     7
     1    -8    -6
    -7    14     9
    -9    15    -2
     5    16   -10
     8    10    17
```

- **Geometry information**

  `mesh.fcurved[nf]`: Logical flag indicating which faces are curved. This flag should also be active for straight faces with non-constatn Jacobian.

```
>> mesh.fcurved

ans =

     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     1
     1
     1
     1
     1
     1
     1
```

  `mesh.tcurved[nt]`: Logical flag indicating which triangles have at least a curved face. This flag should also be active for non curved triangles with non-constant Jacobian.

```
>> mesh.tcurved

ans =

     0
     1
     1
     1
     0
     1
     1
     1
     1
```

- **Master Element information**

mesh.porder: Order of the complete polynomial used for approximation inside each element.

```
>> mesh.porder

ans =

     3
```

mesh.plocal[npl,3]: Parametric coordinates of the nodes in the master element. Note that mesh.plocal(:,1) = 1-mesh.plocal(:,2)-mesh.plocal(:,3). Also, npl = (mesh.porder+1)*(mesh.porder+2)/2. The order of the nodes is that shown in figure 3.

```
>> mesh.plocal

ans =

    1.0000         0         0
    0.6667    0.3333         0
    0.3333    0.6667         0
         0    1.0000         0
    0.6667         0    0.3333
    0.3333    0.3333    0.3333
         0    0.6667    0.3333
    0.3333         0    0.6667
         0    0.3333    0.6667
         0         0    1.0000
```

mesh.tlocal[ntl,3]: Element vertices for local auxiliary mesh. The element ordering is arbitrary. (Used for refinement and plotting).

```
>> mesh.tlocal

ans =

     1     2     5
     2     3     6
     3     4     7
     2     6     5
     3     7     6
     5     6     8
     6     7     9
     6     9     8
     8     9    10
```

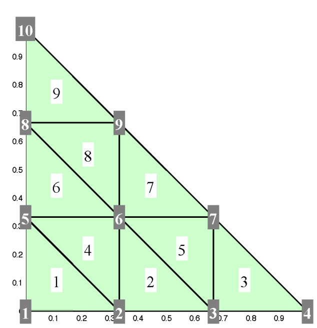

Figure 3: Node positions in master element and local auxiliary mesh connectivity.

- **FEM node locations**

  `mesh.dgnodes[npl,2,nt]`: `mesh.dgnodes[ipl,1:2,it]` are the `x` and `y` coordinates of the `ipl` local node in element `it`. Note that the nodes that lie on a curved boundary must be placed on the actual geometry as shown in figure 4.

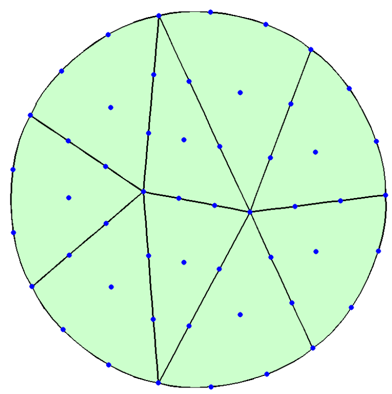

Figure 4: DG node placement for curved geometries.

1.2 The `master` data structure

The `master` data structure contains master element pre-computed information such as the parameters required for numerical integration, the value of the shape functions and their derivatives as well as connectivity information required for efficient assembly.

- **Master Element information**

  The variable `master.porder = mesh.porder` and the array `master.plocal = mesh.plocal` are duplicated for convenience.

`master.porder`:	Order of the complete polynomial used for approximation inside each element.
`master.plocal[npl,3]`:	Parametric coordinates of the nodes in the master element.
`master.ploc1d[npl1d]`:	Parametric coordinates for the 1D master element. Here, `npl1d = master.porder+1`

```
>> master.ploc1d

ans =

         0
    0.3333
    0.6667
    1.0000
```

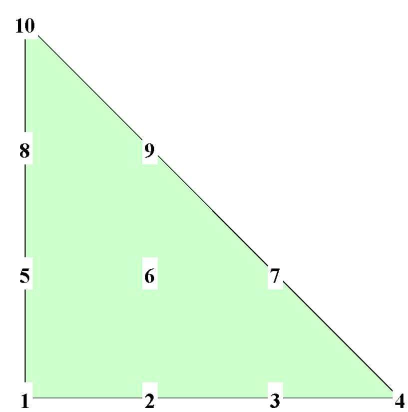

Figure 5: Local node numbering for a cubic triangular element.

- **Node Templates**

`master.corner[3]`: Node numbers of the three corner nodes (see figure 5).

```
>> master.corner

ans =

     1
     4
    10
```

`master.perm[npl1d,3,2]`: Nodes numbers of the nodes on each edge. Note that, `npl1d = master.porder+1`. `master.perm[:,ifl,1]` is the list of nodes on face `ifl` taken clockwise and `master.perm[:,ifl,2]` is the list of nodes on face `ifl` taken counterclockwise (see figure 5).

```
>> master.perm

ans(:,:,1) =

     4    10     1
     7     8     2
     9     5     3
    10     1     4


ans(:,:,2) =

    10     1     4
     9     5     3
     7     8     2
     4    10     1
```

- **Numerical Integration**

`master.gpts[ng,2]`: Parametric coordinates $\{(\xi,\eta) \mid \xi \geq 0,\ \eta \geq 0,\ 1-\xi-\eta \geq 0\}$ of the integration points in 2d. Here, `ng` is the number of 2D integration points.

`master.gwgh[ng]`: Integration weights in 2D.

`master.gp1d[ng1d]`: Parametric coordinates $\{\xi \mid \xi \geq 0,\ \xi \leq 1\}$ of the integration points in 1d. Here, `ng` is the number of 1D integration points.

`master.gw1d[ng1d]`: Integration weights in 1D.

- **Shape Functions**

`master.shap[npl,3,ng]`: Value of the 2D cardinal (or nodal) shape functions and their derivatives evaluated at the integration points. Here, `master.shap[npl,1,ng]` contains the value of the shape function and `master.shap[npl,2:3,ng]` contains the values of the derivatives of the shape functions with respect to ξ and η.

`master.sh1d[npl1d,2,ng1d]`: Value of the 1D cardinal (or nodal) shape functions and their derivatives evaluated at the integration points. Here, `master.sh1d[npl1d,1,ng1d]` contains the value of the shape function and `master.sh1d[npl1d,2,ng1d]` contains the values of the derivatives of the shape functions with respec to ξ.

- **Pre-computed Element Matrices**

`master.mass[npl,npl]`:	Mass matrix in 2D. The entry `master.mass[i,j]` corresponds to $\int_{K'} \phi_i \phi_j \, dK'$, where K' is the master element
`master.conv[npl,npl,2]`:	Convection matrices in 2D. The entry `master.conv[i,j,1]` corresponds to $\int_{K'} \phi_i \phi_{j,\xi} \, dK'$, whereas `master.conv[i,j,2]` corresponds to $\int_{K'} \phi_i \phi_{j,\eta} \, dK'$.
`master.ma1d[npl1d,npl1d]`:	Mass matrix in 1D. The entry `master.ma1d[i,j]` corresponds to $\int_{K'_{1d}} \phi_i \phi_j \, dK'_{1d}$, where K'_{1d} is the 1D master element.