UNIVERSITÀ
POLITECNICA
DELLE MARCHE

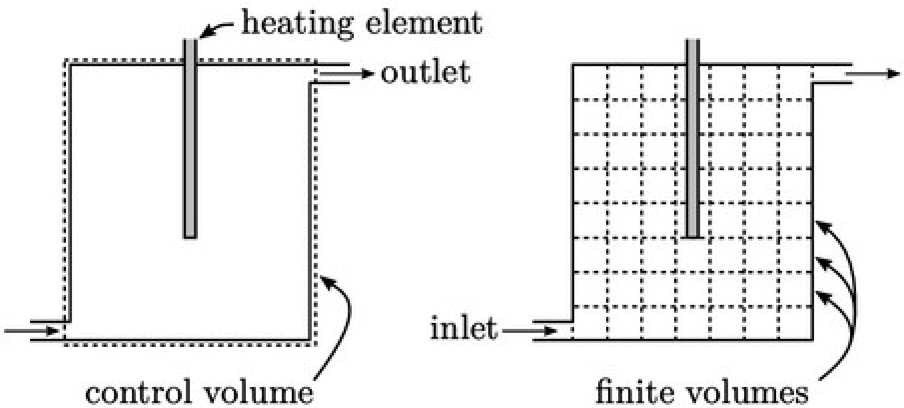
Numerical Heat Transfer
for Applications

Introduction to Finite Volume Method

Dr Valerio D’Alessandro

Finite volume concept

The finite volume method (FVM) adopts the **idea** of **control volumes** used to create models of physical systems. A control volume represents a region of space, which is generally fixed, enclosed by a surface through which fluid flows in and out.



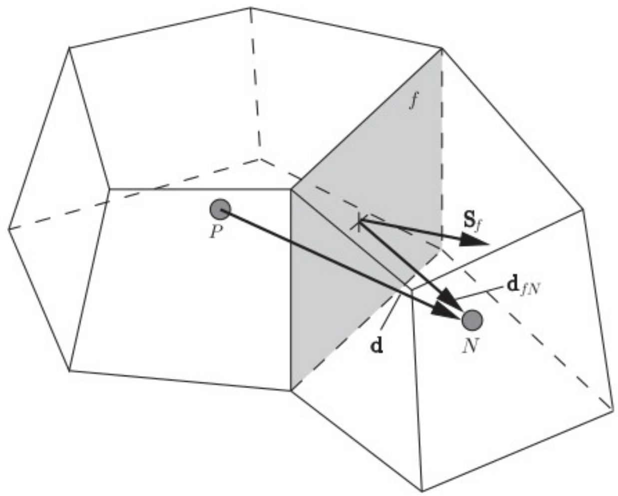
FVM applies conservation equations, by balancing fluxes at the bounding surface.
In FVM a single control volume is not used to describe the entire physical system. Differently, the domain is discritized into multiple connected finite volumes.

Conservation equations are applied to each volume, ensuring that the fluxes of mass, momentum and energy across finite volumes' surfaces are physically consistent.

Finite volume concept

$$\int_{V_P} (\mathbf{x} - \mathbf{x}_P)\, dV = \mathbf{0}$$

In FVM, fluxes at the faces and sources over the element are evaluated following the **mean value approach**, i.e., using the value at the centroid of the surface (midpoint rule) and cell.



$$\int_f \boldsymbol{\Psi} \cdot d\mathbf{S} = \boldsymbol{\Psi}_f \cdot \mathbf{S}_f \qquad \int_{V_P} \phi dV = \phi_P V_P$$

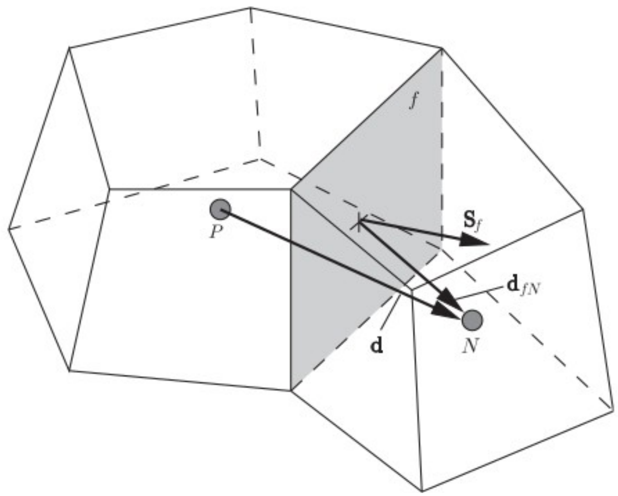
The order of accuracy of the mean value approach can be obtained using a Taylor expansion of the generic variable within the reference element:

$$\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P + O\left(|\mathbf{x} - \mathbf{x}_P|^2\right)$$

$$\int_{V_P} \phi(\mathbf{x})\, dV = \int_{V_p} \left[\phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P + O\left(|\mathbf{x} - \mathbf{x}_P|^2\right)\right] dV$$

Finite volume concept

$$\int_{V_P} (\mathbf{x} - \mathbf{x}_P)\, dV = \mathbf{0}$$



$$\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla \phi)_P + O\left(|\mathbf{x} - \mathbf{x}_P|^2\right)$$

$$\int_{V_P} \phi(\mathbf{x})\, dV = \int_{V_p} \left[\phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla \phi)_P\right] dV + \\ + \int_{V_p} \left[O\left(|\mathbf{x} - \mathbf{x}_P|^2\right)\right] dV$$

$$\int_{V_P} \phi(\mathbf{x})\, dV = \int_{V_p} \phi_P dV + \underbrace{\left(\int_{V_P} (\mathbf{x} - \mathbf{x}_P)\, dV\right)}_{=0} \cdot (\nabla \phi)_P + \int_{V_p} \left[O\left(|\mathbf{x} - \mathbf{x}_P|^2\right)\right] dV$$

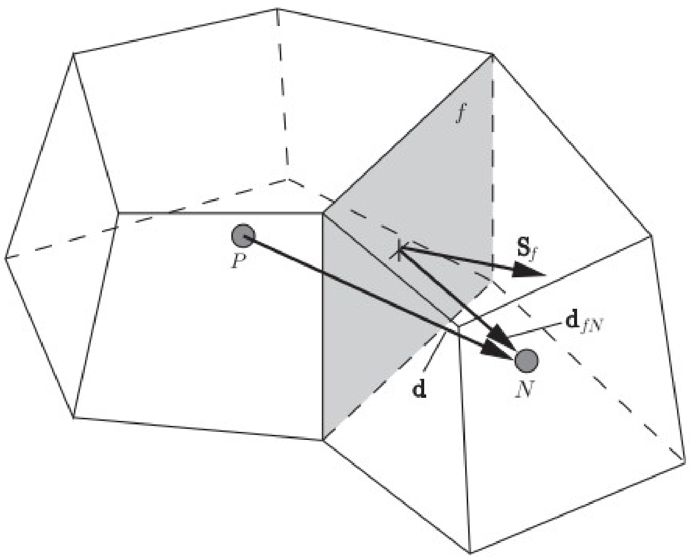
$$\int_{V_P} \phi(\mathbf{x})\, dV = \phi_P V_P + O\left(|\mathbf{x} - \mathbf{x}_P|^2\right) V_P$$

$$\boxed{\underbrace{\frac{1}{V_P} \int_{V_P} \phi(\mathbf{x})\, dV}_{\overline{\phi_P}} = \phi_P + \mathbf{O}\left(|\mathbf{x} - \mathbf{x_P}|^{\mathbf{2}}\right)}$$

The mean value approximation is second order accurate.

Finite volume discretization of Laplace equation

The method is based on discretising the integral form of governing equations over each control volume.



$$\int_{V_P} (\mathbf{x} - \mathbf{x}_P)\, dV = \mathbf{0}$$

$$\boxed{\nabla^2 T = 0 \Longrightarrow \int_{V_P} \nabla^2 T dV = 0}$$

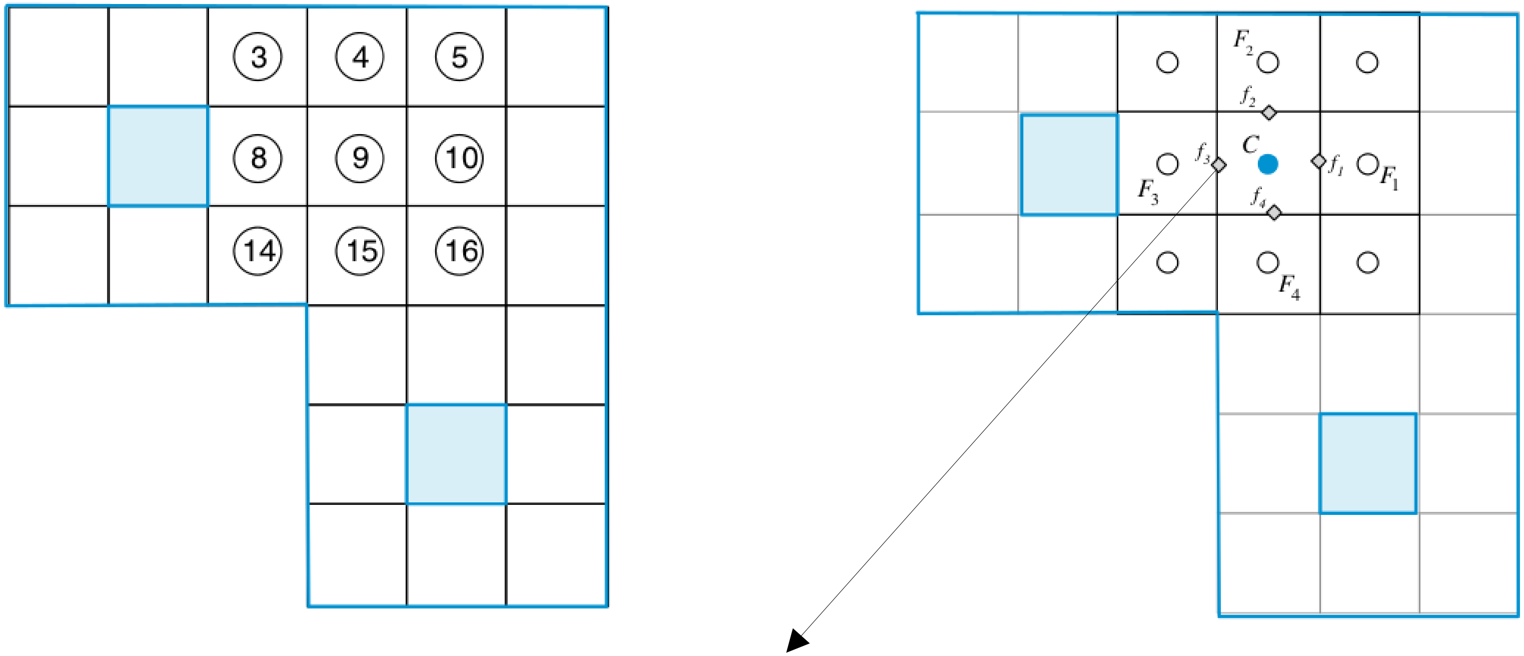
$$\int_{V_P} \nabla^2 T dV = \int_{V_P} \nabla \cdot (\nabla T)\, dV$$

$$\int_{V_P} \nabla \cdot (\nabla T)\, dV = \int_{\partial V_P} \nabla T \cdot d\mathbf{S} \simeq \sum_f (\nabla T)_f \cdot \mathbf{S}_f$$

$$\boxed{\sum_f (\nabla T)_f \cdot \mathbf{S}_f = 0}$$

Finite volume discretization of Laplace equation

$$\boxed{\sum_f (\nabla T)_f \cdot \mathbf{S}_f = 0}$$

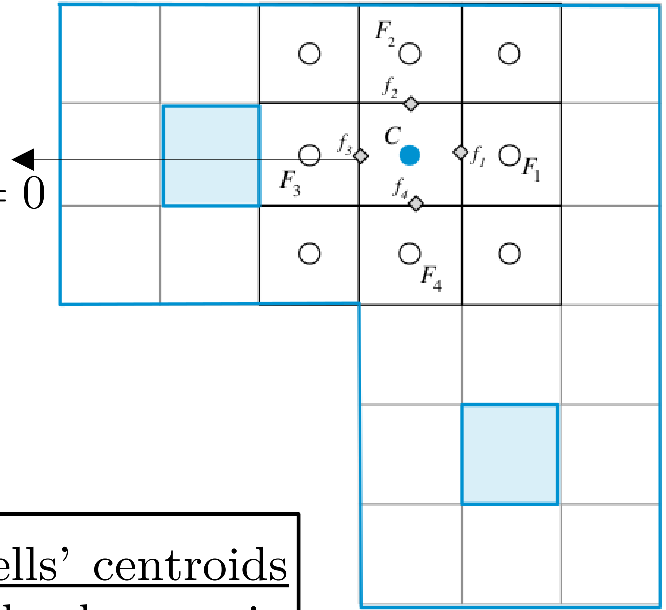


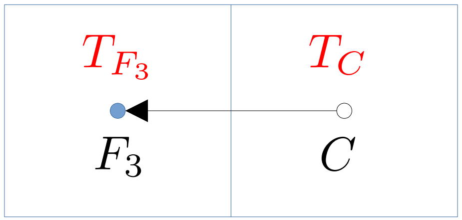
$$(\nabla T)_{f_1} \cdot \mathbf{S}_{f_1} + (\nabla T)_{f_2} \cdot \mathbf{S}_{f_2} + (\nabla T)_{f_3} \cdot \mathbf{S}_{f_3} + (\nabla T)_{f_4} \cdot \mathbf{S}_{f_4} = 0$$

Finite volume discretization of Laplace equation

$$\sum_f (\nabla T)_f \cdot \mathbf{S}_f = 0$$

$$(\nabla T)_{f_1} \cdot \mathbf{S}_{f_1} + (\nabla T)_{f_2} \cdot \mathbf{S}_{f_2} + \underbrace{(\nabla T)_{f_3} \cdot \mathbf{S}_{f_3}}_{= \frac{T_{F_3} - T_C}{|\mathbf{F_3} - \mathbf{C}|} |\mathbf{S}_{f_3}|} + (\nabla T)_{f_4} \cdot \mathbf{S}_{f_4} = 0$$

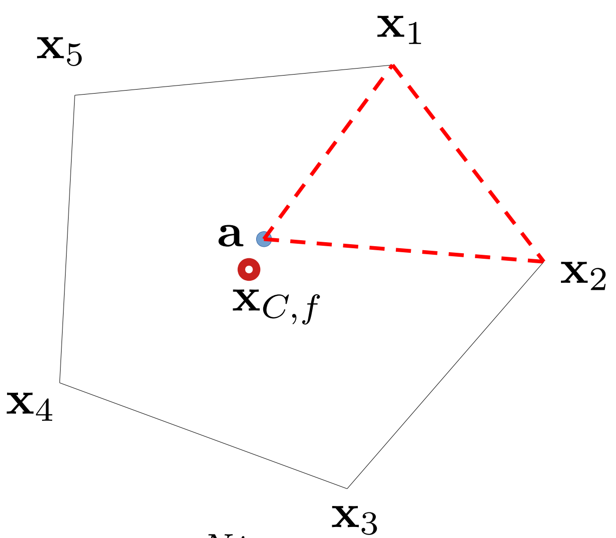




Faces' area and cells' centroids distance have to be known in FVM context.

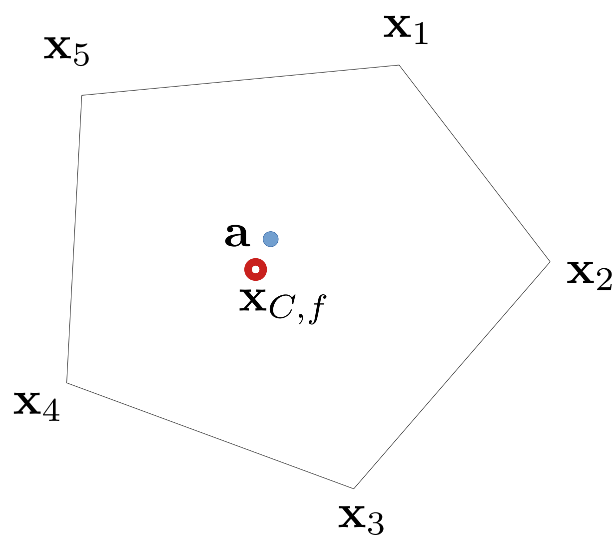
Mesh geometric quantities

FVM discretization requires information about mesh geometric entities such as: *the volume of elements, the area of faces, the centroids of elements and faces, the alignment of faces with the vectors joining the cells' centroids.*
These data are not typically available in mesh file/files but are computed.



$$S_f = \sum_{i=1}^{Nt} S_{t,i}$$

Polygon area can be evaluated as the sum of the sub-triangles area.

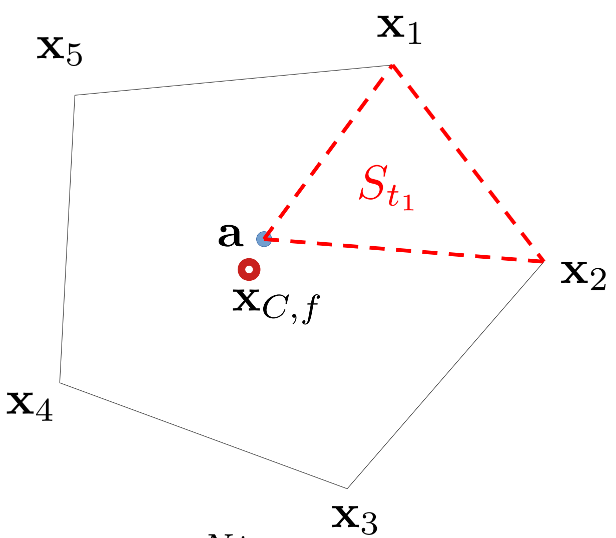


$$\mathbf{x}_{C,f} = \underbrace{\frac{1}{N_v} \sum_{i=1}^{N_v} \mathbf{x}_i}_{\mathbf{a}}$$

Approximate evalutation of face's center

Mesh geometric quantities

FVM discretization requires information about mesh geometric entities such as: *the volume of elements, the area of faces, the centroids of elements and faces, the alignment of faces with the vectors joining the cells' centroids.*
These data are not typically available in mesh file/files but are computed.



$$S_{t_1} = \frac{1}{2} \left| (\mathbf{x}_1 - \mathbf{a}) \wedge (\mathbf{x}_2 - \mathbf{a}) \right|$$

$$S_{t_2} = \ldots$$

$$\boxed{\begin{aligned} \mathbf{x}_{C,f} &= \frac{1}{S_t} \sum_{i=1}^{N_t} S_{t,i} \mathbf{C}_{t,i} \\ \mathbf{C}_{t,i} &= \frac{1}{3} \left(\mathbf{x}_i + \mathbf{x}_{i+1} + \mathbf{a} \right) \end{aligned}}$$

$$S_f = \sum_{i=1}^{Nt} S_{t,i}$$

Polygon area can be evaluated as the sum of the sub-triangles area.

$$\mathbf{a} = \frac{1}{N_v} \sum_{i=1}^{N_v} \mathbf{x}_i$$

Mesh geometric quantities

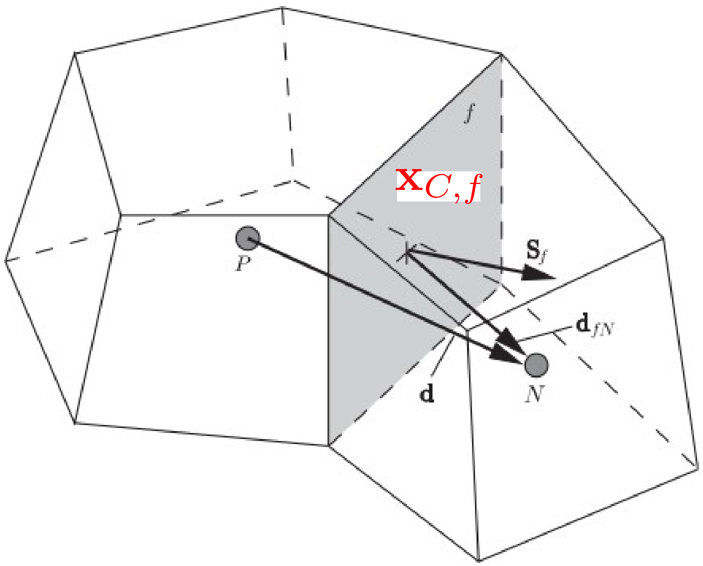
The cell's volume can be calculated by Gauss's theorem.

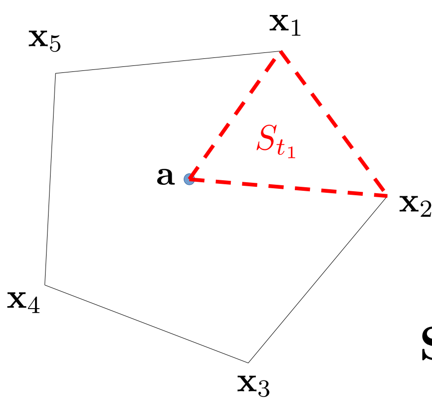
$$V_P = \int_{V_P} dV$$

$\nabla \cdot \mathbf{x} = 3$ where $\mathbf{x}$ is used to describe the position.

$$V_P = \int_{V_P} dV = \frac{1}{3}\int_{V_P} (\nabla \cdot \mathbf{x})\, dV \quad \Rightarrow \quad V_P = \frac{1}{3}\int_{\partial V_P} \mathbf{x} \cdot d\mathbf{S}$$

$$\boxed{V_P = \frac{1}{3}\sum_f \mathbf{x}_{C,f} \cdot \mathbf{S}_f}$$



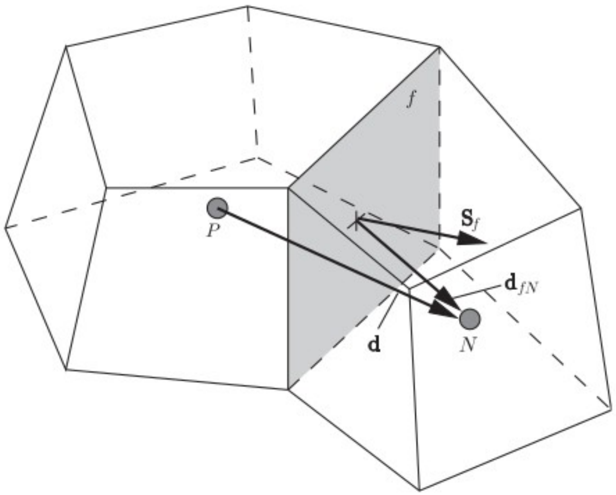


$$\mathbf{S}_f = \sum_{i=1}^{Nt} \mathbf{S}_{t,i}$$

$$\mathbf{S}_{\mathbf{t_1}} = \frac{1}{2}(\mathbf{x}_1 - \mathbf{a}) \wedge (\mathbf{x}_2 - \mathbf{a})$$

Mesh geometric quantities

The cells' centroid can evaluated as follows



$$\mathbf{x}_P = \frac{1}{N_f}\sum_{i=1}^{N_f}\mathbf{x}_{C,f}$$

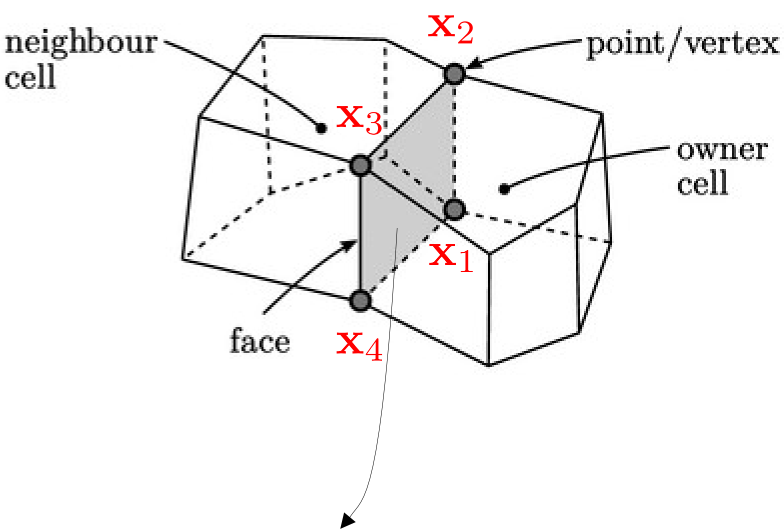
$$\boxed{\mathbf{x}_P = \frac{1}{V_P}\int_{V_p}\mathbf{x}dV}$$

Approximate evalutation of cells' center

$$\nabla|\mathbf{x}|^2 = 2\mathbf{x} \quad\Rightarrow\quad \mathbf{x}_P = \frac{1}{2V_P}\int_{V_p}\nabla|\mathbf{x}|^2 dV = \frac{1}{2V_P}\int_{\partial V_p}|\mathbf{x}|^2 d\mathbf{S}$$

$$\boxed{\mathbf{x}_P \simeq \frac{1}{2V_P}\sum_f |\mathbf{x}|_f^2\,\mathbf{S}_f}$$

Mesh connectivities



$$\mathbf{x}_C = \frac{1}{N_v}\sum_{i=1}^{N_v}\mathbf{x}_i$$

- **Element connectivities**
  a) Neighbour cells
  b) Bounding faces
  c) Defining vertices
- **Faces connectivities**
  a) Elements sharing the cells
  b) Defining vertices
- **Vertex connectivities**
  a) List of sharing elements
  b) List of sharing faces

OpenFOAM

OpenFOAM® (Open source Field Operation And Manipulation) is an object-oriented C++ framework that can be used to build a variety of computational solvers for problems in continuum mechanics with a focus on finite volume discretization. OpenFOAM® also includes several ready solvers, utilities, and applications that can be directly used.

OpenFOAM®

https://www.openfoam.com/

OF case structure

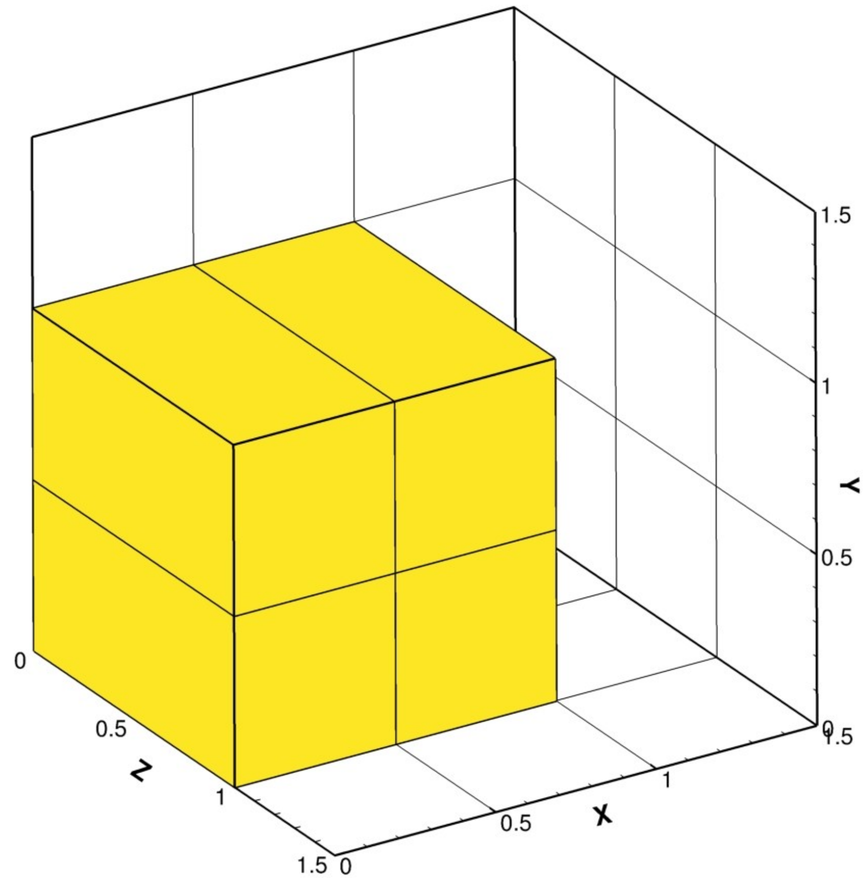
```
0/
...U
...p
...T
constant/
...polyMesh/
...transportProperties
...turbulenceProperties
system/
...controlDict
...fvSchemes
...fvSolution
```

OF mesh

```
...polyMesh/
   .../boundary
   .../points
   .../faces
   .../owner
   .../neighbour
```

OpenFOAM mesh

```
...polyMesh/
    .../boundary
    .../points
    .../faces
    .../owner
    .../neighbour
```



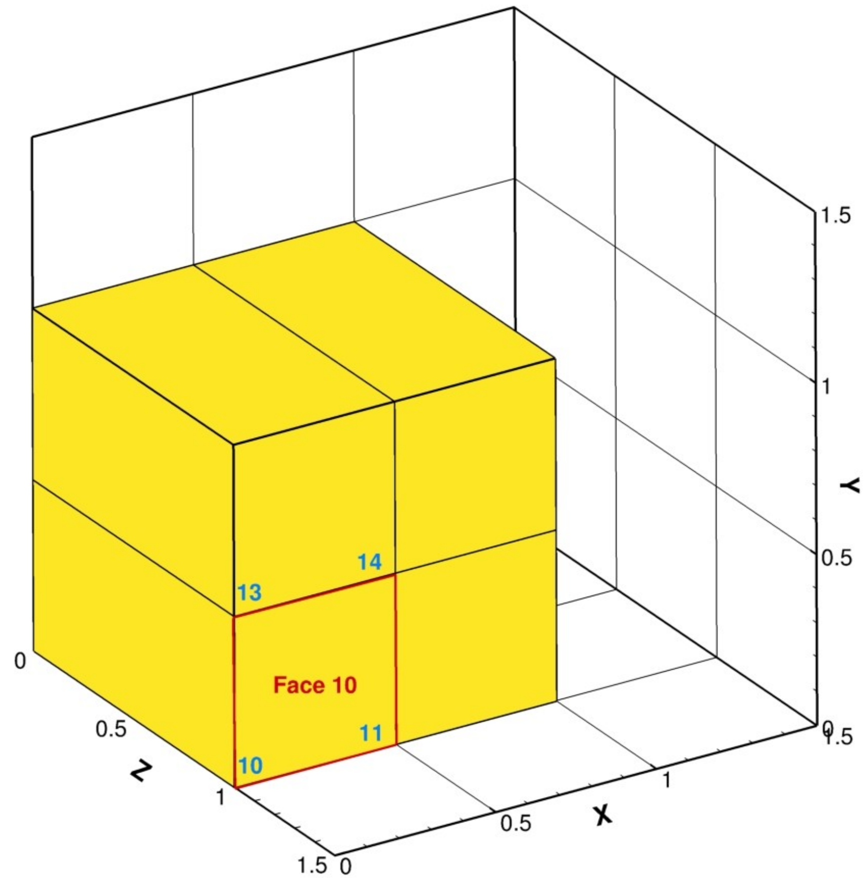
points

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       vectorField;
    location    "constant/polyMesh";
    object      points;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


18              // N. of vertices
(
(0 0 0)         // x , y , z – vertex 1
(0.5 0 0)       // x , y , z – vertex 2
(1 0 0)         // x , y , z – vertex 3
(0 0.5 0)
(0.5 0.5 0)
(1 0.5 0)
…
)
```

OpenFOAM mesh

```
...polyMesh/
   .../boundary
   .../points
   .../faces
   .../owner
   .../neighbour
```



faces

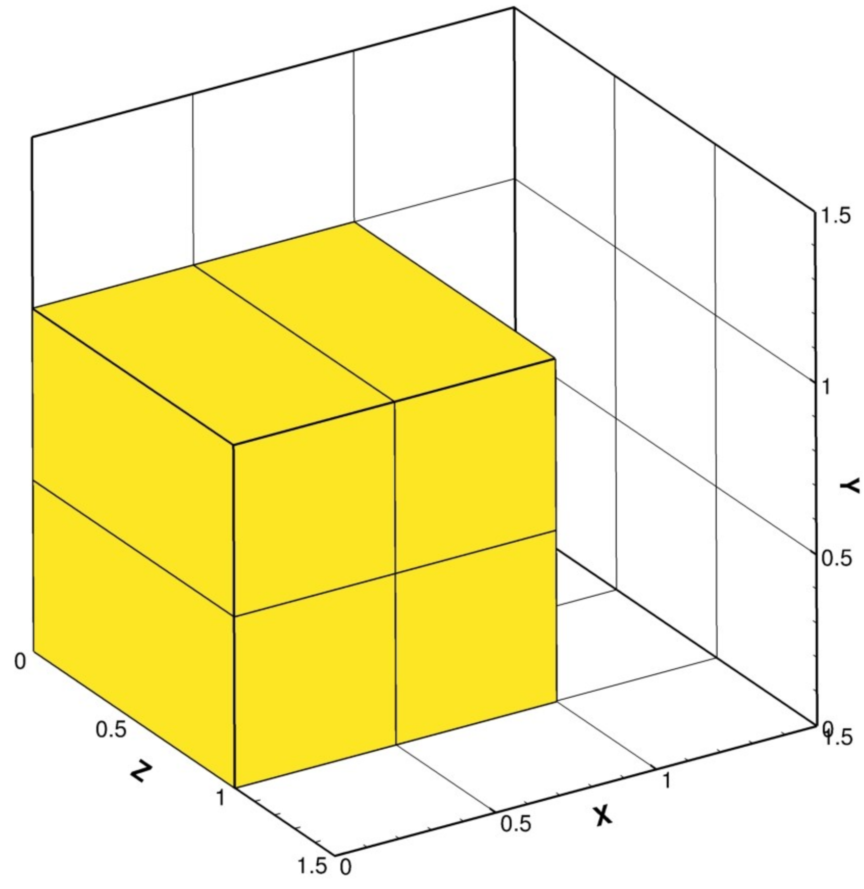
```
FoamFile
{
    version     2.0;
    format      ascii;
    class       faceList;
    location    "constant/polyMesh";
    object      faces;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

20  // N. of faces
(
4(4 13 10 1)
4(4 3 12 13)
4(4 13 14 5)
4(7 16 13 4)
4(1 10 9 0)
4(1 2 11 10)
…
)
```

4 vertices
(
Vertex 4
Vertex 13
Vertex 10
Vertex 1
)

OpenFOAM mesh

```
...polyMesh/
   .../boundary
   .../points
   .../faces
   .../owner
   .../neighbour
```

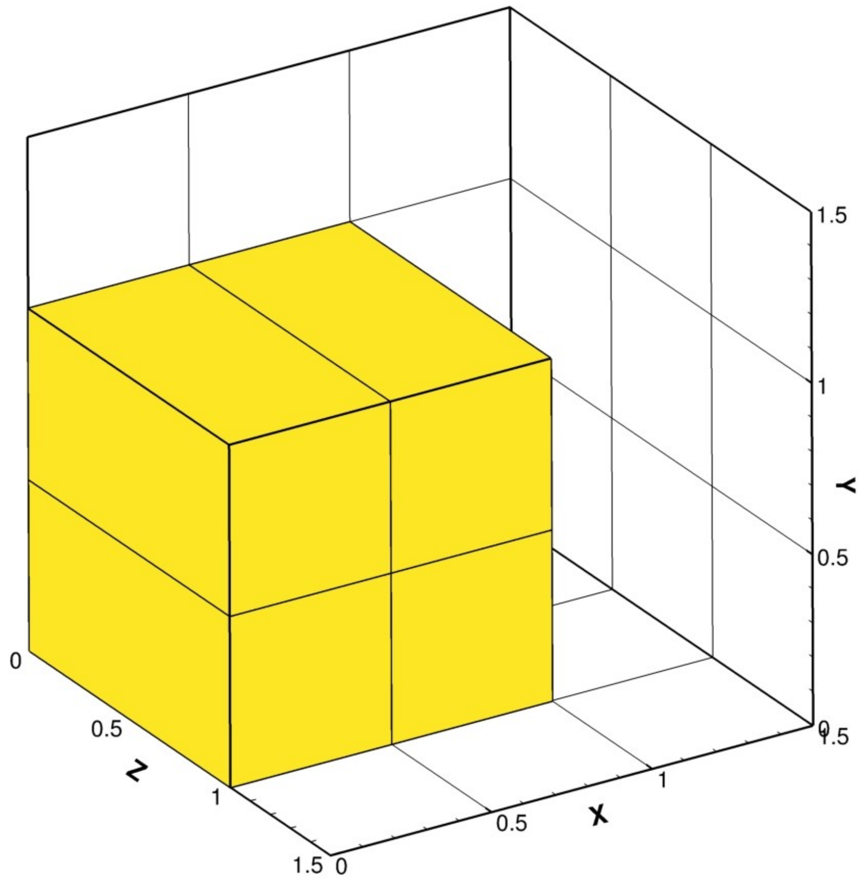


owner

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       labelList;
    note        "nPoints:18  nCells:4
nFaces:20  nInternalFaces:4";
    location    "constant/polyMesh";
    object      owner;
}
// * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * //
20  // N. of faces
(
0               // owner of face1
0               // owner of face2
1
2
0
1
…
)
```

OpenFOAM mesh

```
...polyMesh/
   .../boundary
   .../points
   .../faces
   .../owner
   .../neighbour
```



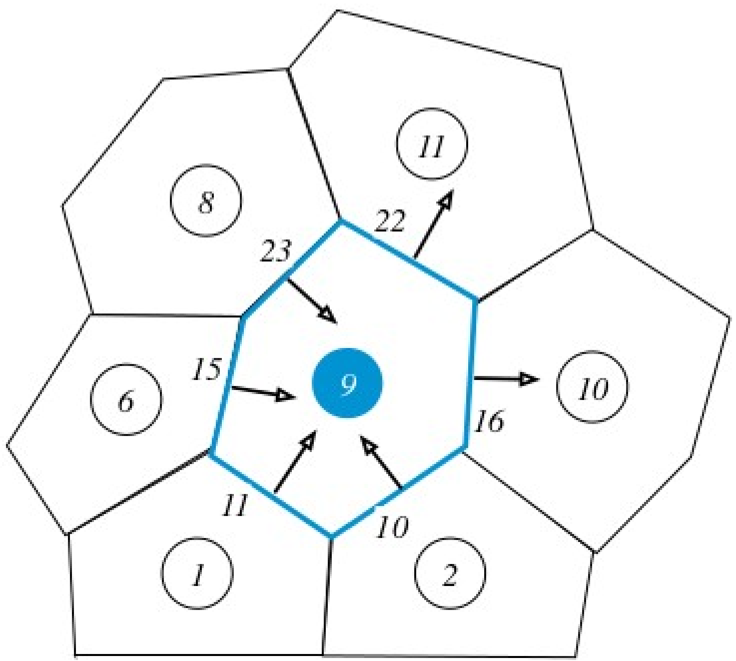
neighbour

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       labelList;
    note        "nPoints:18  nCells:4
nFaces:20  nInternalFaces:4";
    location    "constant/polyMesh";
    object      neighbour;
}
// * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * //

4 // N. of  INTERNAL faces
(
1              //neighbour of face1
2              //neighbour of face2
3              ...
3              ...
)
```

OpenFOAM mesh

```
...polyMesh/
   .../boundary
   .../points
   .../faces
   .../owner
   .../neighbour
```



Face 22 - Owner 9 ; Neigh 11

Face 23 - Owner 8 ; Neigh 9

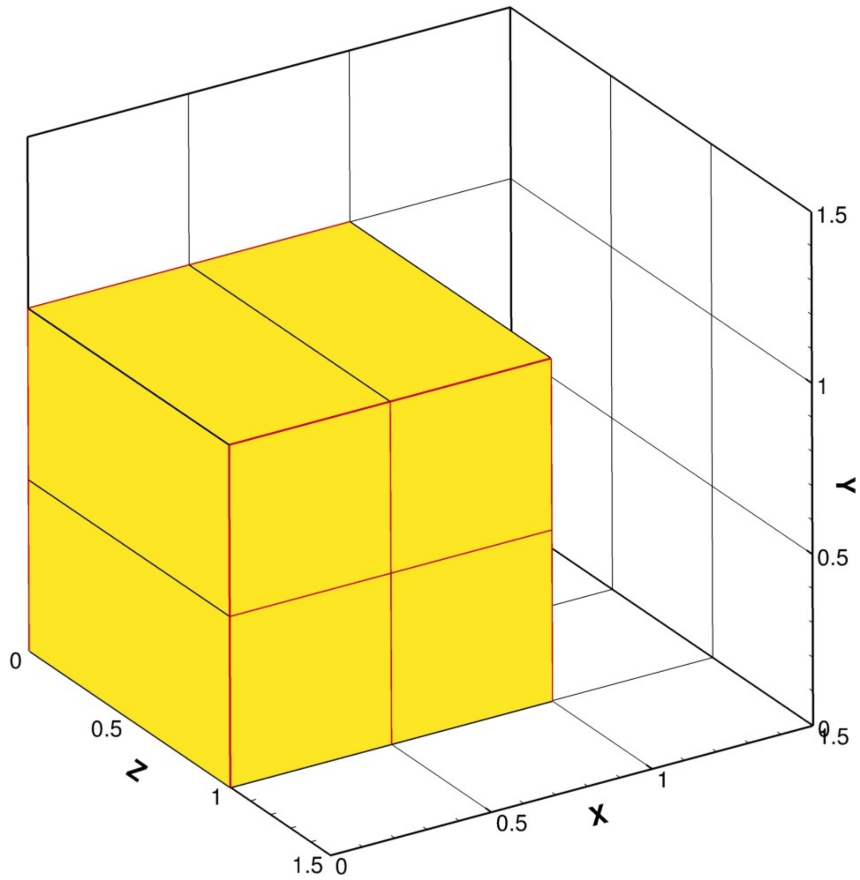
…

Face 16 - Owner 9 ; Neigh 10

Boundary faces have only owner.

OpenFOAM mesh

```
...polyMesh/
   .../boundary
   .../points
   .../faces
   .../owner
   .../neighbour
```



boundary

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * *
//
5        // N. of boundaries
(
    bottom
    {
        type            wall;
        nFaces          2;
        startFace       4;
    }
    frontAndBack
    {
        type            empty;
        nFaces          8;
        startFace       6;
    }
```

Ex1a_FVM – Computing mesh geometric quantities

Matlab code

```
clear all
clc
format long e

m = readOpenFoamMesh('cube');
%
Nf = m.numberOfFaces;
Nc = m.numberOfElements;
%
%---------------------------------------
% Computing faces' centre
%
p  = zeros(3,1);
for i=1:Nf
   pm     = zeros(3,1);
   iN     = m.faces(i).iNodes;
   for j=1:length(iN)
    p      = m.nodes(iN(j)).centroid;
    pm     = pm + p;
   end
   m.faces(i).Cf = pm/length(iN);
end
```

readOpenFoamMesh

The function allows the possibility to read in Matlab environment the polyMesh files of an OpenFOAM case (the function is available on the course gitHub repository).

m = readOpenFoamMesh (‘cube’) ;

structure containing mesh informations (points, faces, connectivities, ...)

case directory name (this dir contains 0, constant and system)

Ex1a_FVM – Computing mesh geometric quantities

$$\mathbf{m} = \mathbf{readOpenFoamMesh}\ (\text{‘}\mathbf{cube}\text{’})\ ;$$

```
m =

  struct with fields:

  nodes: [1×18 struct]
  numberOfNodes: 18
  caseDirectory: 'cube'
  numberOfFaces: 20
  numberOfElements: 4
  faces: [1×20 struct]
  numberOfInteriorFaces: 4
  boundaries: [1×5 struct]
  numberOfBoundaries: 5
  numberOfPatches: 5
  elements: [1×4 struct]
  numberOfBElements: 16
  numberOfBFaces: 16
```

m.nodes

Fields	centroid	index	iFaces	iElements
1	[0;0;0]	1	[5,7,15]	1
2	[0.5000;0;0]	2	[1,5,6,7,9]	[1,2]
3	[1;0;0]	3	[6,9,17]	2
4	[0;0.5000;0]	4	[2,7,11,15,16]	[1,3]
5	[0.5000;0....	5	[1,2,3,4,7,9,1...	[1,2,3,4]
6	[1;0.5000;0]	6	[3,9,13,17,18]	[2,4]
7	[0;1;0]	7	[11,16,19]	3
8	[0.5000;1;0]	8	[4,11,13,19,20]	[3,4]
9	[1;1;0]	9	[13,18,20]	4
10	[0;0;1]	10	[5,8,15]	1
11	[0.5000;0;1]	11	[1,5,6,8,10]	[1,2]
12	[1;0;1]	12	[6,10,17]	2
13	[0;0.5000;1]	13	[2,8,12,15,16]	[1,3]
14	[0.5000;0....	14	[1,2,3,4,8,10,...	[1,2,3,4]
15	[1;0.5000;1]	15	[3,10,14,17,18]	[2,4]
16	[0;1;1]	16	[12,16,19]	3
17	[0.5000;1;1]	17	[4,12,14,19,20]	[3,4]
18	[1;1;1]	18	[14,18,20]	4

Ex1a_FVM – Computing mesh geometric quantities

$$\mathbf{m = readOpenFoamMesh\ (`cube')\ ;}$$

```
m =

  struct with fields:

  nodes: [1×18 struct]
  numberOfNodes: 18
  caseDirectory: 'cube'
  numberOfFaces: 20
  numberOfElements: 4
  faces: [1×20 struct]
  numberOfInteriorFaces: 4
  boundaries: [1×5 struct]
  numberOfBoundaries: 5
  numberOfPatches: 5
  elements: [1×4 struct]
  numberOfBElements: 16
  numberOfBFaces: 16
```

m.faces

Fields	iNodes	index	iOwner	iNeighbour
1	[5,14,11,2]	1	1	2
2	[5,4,13,14]	2	1	3
3	[5,14,15,6]	3	2	4
4	[8,17,14,5]	4	3	4
5	[2,11,10,1]	5	1	-1
6	[2,3,12,11]	6	2	-1
7	[5,2,1,4]	7	1	-1
8	[14,13,1...	8	1	-1
9	[5,6,3,2]	9	2	-1
10	[14,11,1...	10	2	-1
11	[8,5,4,7]	11	3	-1
12	[17,16,1...	12	3	-1
13	[6,5,8,9]	13	4	-1
14	[15,18,1...	14	4	-1
15	[4,1,10,13]	15	1	-1
16	[7,4,13,16]	16	3	-1
17	[6,15,12,3]	17	2	-1
18	[6,9,18,15]	18	4	-1
19	[8,7,16,17]	19	3	-1
20	[9,8,17,18]	20	4	-1

Ex1a_FVM – Computing mesh geometric quantities

Matlab code

```
clear all
clc
format long e

m = readOpenFoamMesh('cube');
%
Nf = m.numberOfFaces;
Nc = m.numberOfElements;
%
%-------------------------------------
% Computing faces' centre
%
p  = zeros(3,1);
for i=1:Nf
   pm     = zeros(3,1);
   iN     = m.faces(i).iNodes;
   for j=1:length(iN)
    p      = m.nodes(iN(j)).centroid;
    pm     = pm + p;
   end
   m.faces(i).Cf = pm/length(iN);
end
```

$$\mathbf{x}_{C,f} = \frac{1}{N_v} \sum_{i=1}^{N_v} \mathbf{x}_i$$

```
iN     = m.faces(i).iNodes;
is the list of the nodes belonging to i-th face

p      = m.nodes(iN(j)).centroid;
Are the x,y,z coordinates of the iN(j) node.
Basically, the index-j scrolls inside the iN list.
```

Ex1a_FVM – Computing mesh geometric quantities

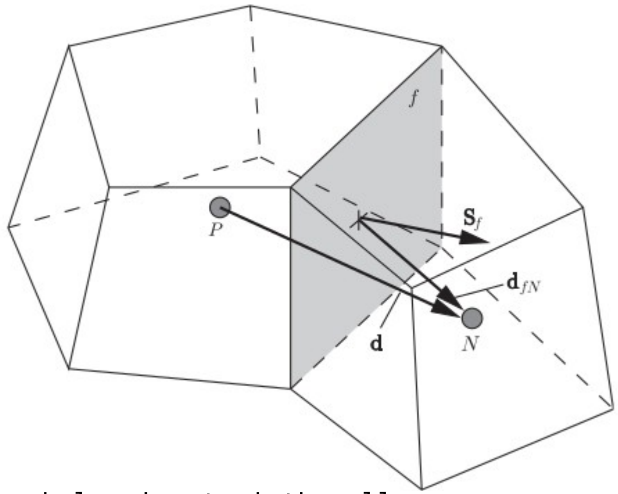
Matlab code

```
clear all
clc
format long e

m = readOpenFoamMesh('cube');
%
Nf = m.numberOfFaces;
Nc = m.numberOfElements;
%
%----------------------------------------
% Computing cells' centroid
%

for i=1:Nc
      Cc = zeros(3,1);
      for j=1:length(m.elements(i).iFaces)
        ifaces = m.elements(i).iFaces(j);
        Cc     = Cc + m.faces(ifaces).Cf;
      end
        Cc=Cc/length(m.elements(i).iFaces);
        m.elements(i).Cc  = Cc;
end
```

$$\mathbf{x}_P = \frac{1}{N_f}\sum_{i=1}^{N_f}\mathbf{x}_{C,f}$$



`m.elements(i).iFaces`
is the list of the faces belonging to i-th cell

`ifaces = m.elements(i).iFaces(j);`
ifaces is the label of `iFaces(j)`. Basically, the ifaces scrolls inside the element’s faces list

`m.faces(ifaces).Cf;`
Are the x,y,z coordinates of the aforementioned ifaces label

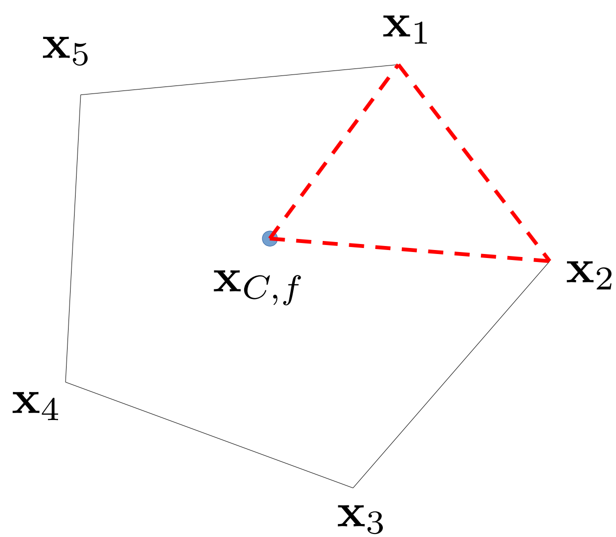
Ex1a_FVM – Computing mesh geometric quantities

Matlab code

```
clear all
clc
format long e

m = readOpenFoamMesh('cube');
%
Nf = m.numberOfFaces;
Nc = m.numberOfElements;
%
%-------------------------------------
% Computing faces' normal
%

for i=1:Nf
   iN            = m.faces(i).iNodes;
   x1            = m.nodes(iN(1)).centroid;
   x2            = m.nodes(iN(2)).centroid;
   c             = m.faces(i).Cf;
   p1            = x1 - c;
   p2            = x2 - c;
   Sf            = cross(p1,p2);
   m.faces(i).nf = Sf/(sqrt(sum(Sf.*Sf)) + eps) ;
end
```



$$\mathbf{p}_1 = (\mathbf{x}_1 - \mathbf{x}_{C,f}) \qquad \mathbf{p}_2 = (\mathbf{x}_2 - \mathbf{x}_{C,f})$$

$$\mathbf{n} = \frac{\mathbf{p}_1 \wedge \mathbf{p}_2}{|\mathbf{p}_1 \wedge \mathbf{p}_2|}$$

we approximate : $\mathbf{a} = \mathbf{x}_{C,f}$

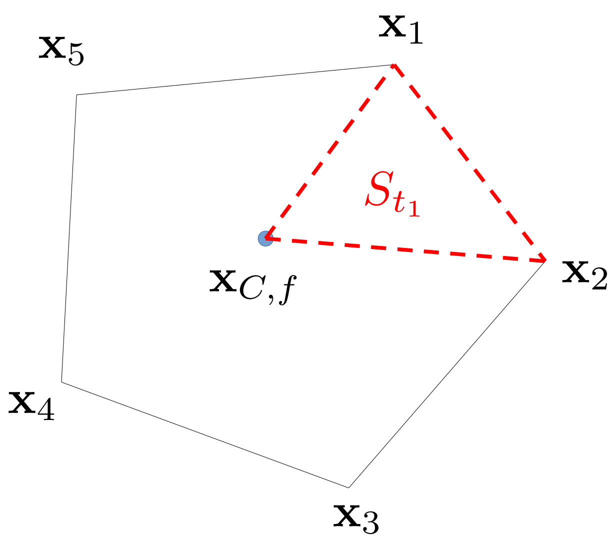
Ex1a_FVM – Computing mesh geometric quantities

Matlab code

```
clear all
clc
format long e

m = readOpenFoamMesh('cube');
%
Nf = m.numberOfFaces;
Nc = m.numberOfElements;
%
%---------------------------------------
% Computing faces' area
%
for i=1:Nf
    iN             = m.faces(i).iNodes;
    c              = m.faces(i).Cf;
    magSf          = 0.0;
    for j=1:length(iN)-1
        a1         = m.nodes(iN(j  )).centroid - c;
        a2         = m.nodes(iN(j+1)).centroid - c;
        vec        = cross(a1,a2);
        magSf      = magSf + 0.5*sqrt(sum(vec.*vec));
    end
        a1         = m.nodes(iN(end  )).centroid - c;
        a2         = m.nodes(iN(1    )).centroid - c;
        vec        = cross(a1,a2);
        magSf      = magSf + 0.5*sqrt(sum(vec.*vec));

    m.faces(i).Af = magSf;
end
```



$$S_{t_1} = \frac{1}{2}\left|(\mathbf{x}_1 - \mathbf{x}_{C,f}) \wedge (\mathbf{x}_2 - \mathbf{x}_{C,f})\right|$$

$$S_f = \sum_{i=1}^{Nt} S_{t,i}$$

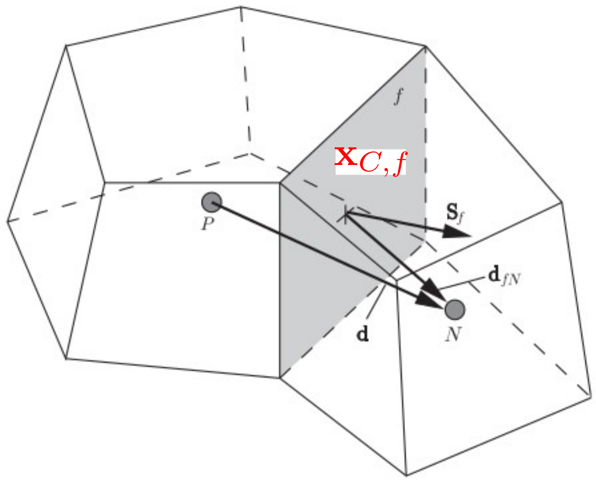
Ex1a_FVM – Computing mesh geometric quantities

Matlab code

```
clear all
clc
format long e

m = readOpenFoamMesh('cube');
%
Nf = m.numberOfFaces;
Nc = m.numberOfElements;
%
%---------------------------------------
% Computing cells' volume

for i=1:Nc
    vol = 0.0;
    for j=1:length(m.elements(i).iFaces)
        ifaces = m.elements(i).iFaces(j);
%
        n      = m.faces(ifaces).nf;
        Af     = m.faces(ifaces).Af;
        cf     = m.faces(ifaces).Cf;
        Sv     = Af*n*(m.elements(i).faceSign(j));
%
        vol    = vol + (1./3.)*sum(cf.*Sv);
    end
    m.elements(i).volC  = vol;
end
```



$$V_P = \frac{1}{3}\sum_f \mathbf{x}_{C,f} \cdot \mathbf{S}_f$$

Ex1b_FVM – Plotting scalar fields

Matlab code

```
clear all
clc
format long e

caseName = 'cube_nonOrtho';
m = readOpenFoamMesh(caseName);
%
Nf = m.numberOfFaces;
Nc = m.numberOfElements;
%
%-------------------------------------
…

Vc = zeros(Nc,1);
for i=1:Nc
    Vc(i) = m.elements(i).volC ;
end
wrtfld(1, m , Vc, 'Vol', caseName)
```

wrtfld (time, m, field, name, caseName)

The function write a generic field allowing its visualization through Tecplot, Paraview, … (the function is available on the course gitHub repository).

'Vol' is the name of field once it is visualized in post-processing stage.