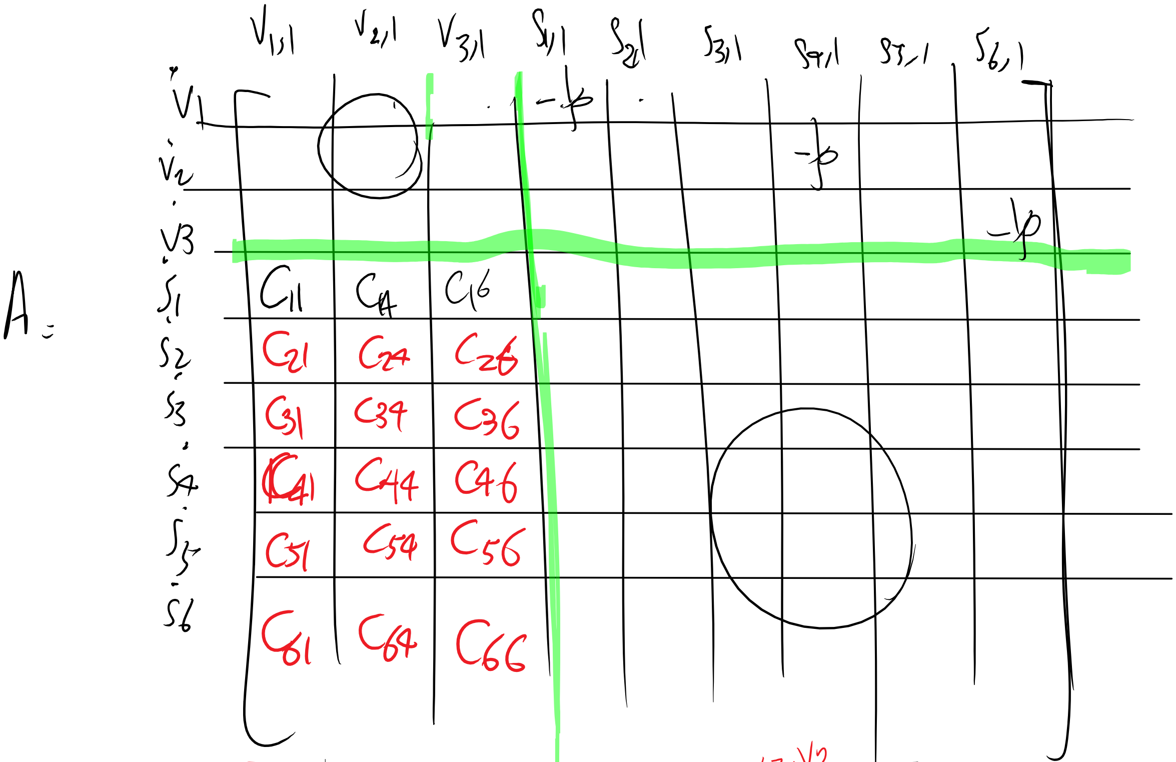
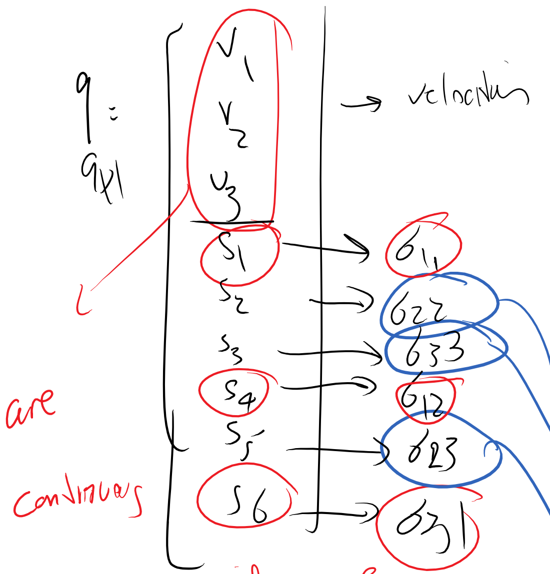
2018/04/13

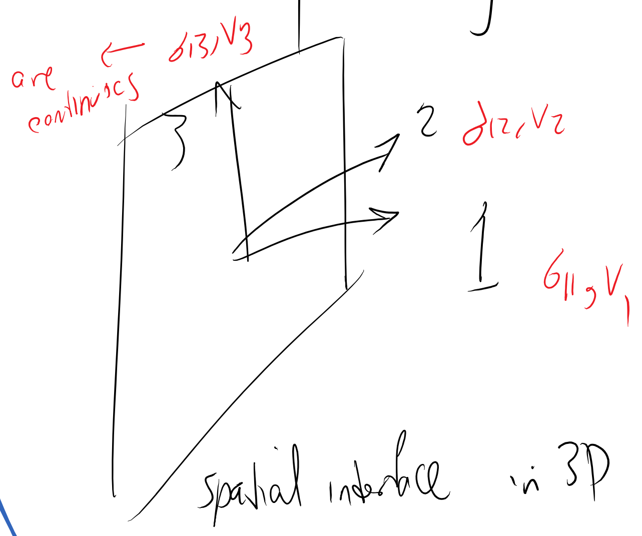
Friday, April 13, 2018 11:38 AM

$$\dot{q} + A_1 q_{,1} + A_{2,2} + A_3 q_{,3} = 0$$

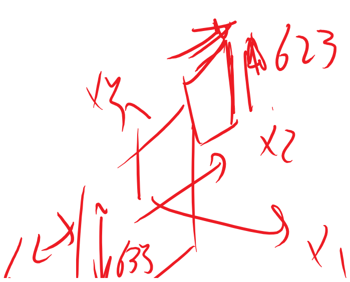


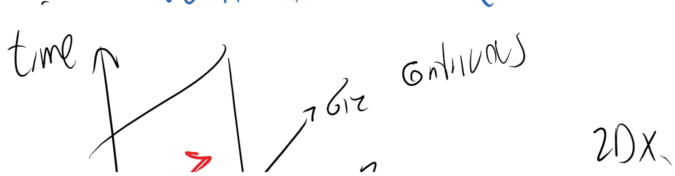


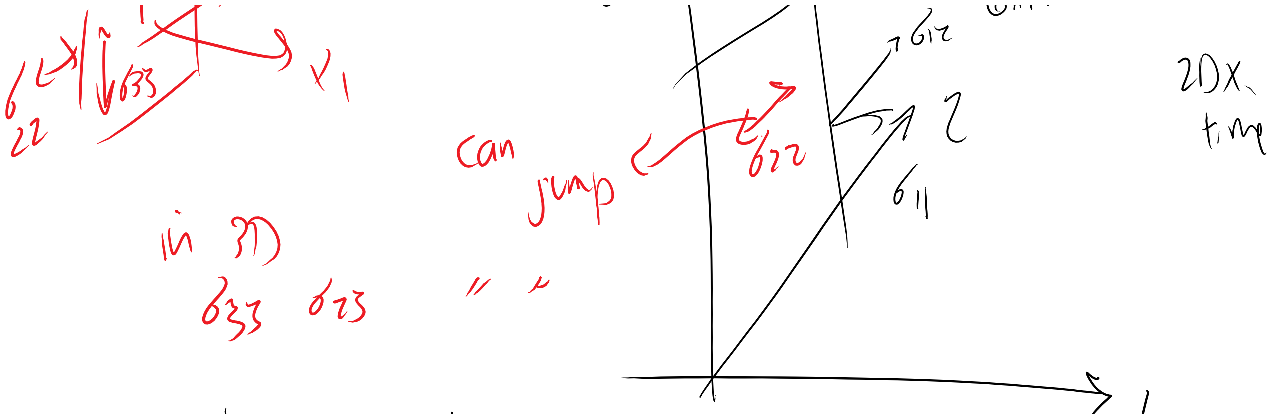
are continuous on a material interface



Can suffer jump on a vertical interface







since $\sigma_{22}, \sigma_{33}, \sigma_{23}$ can jump across time axes $(C=0)$ we have 3 zero eigenvalues

$9 - 3 = 6$ nonzero eigen values

→ 3 negative

3 positives

So, in
Zhan_2018_An exact Riemann solver for wave propagation in arbitrary anisotropic elastic media with fluid coupling.pdf
The process of solving eigenvalue problem for these anti-symmetric matrices is discussed ->

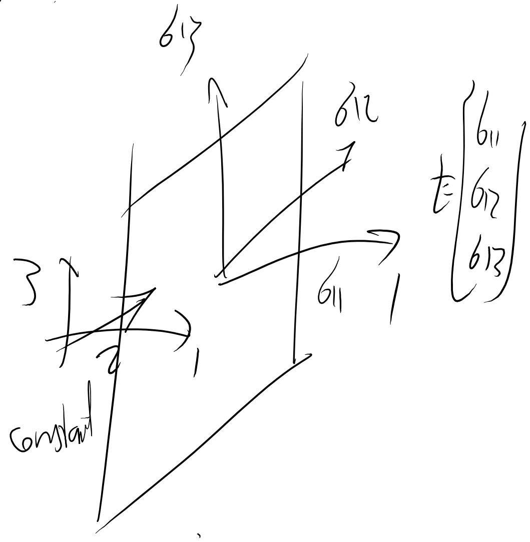
9x9 A matrix we end up solving a 3x3 matrix eigenvalue problem

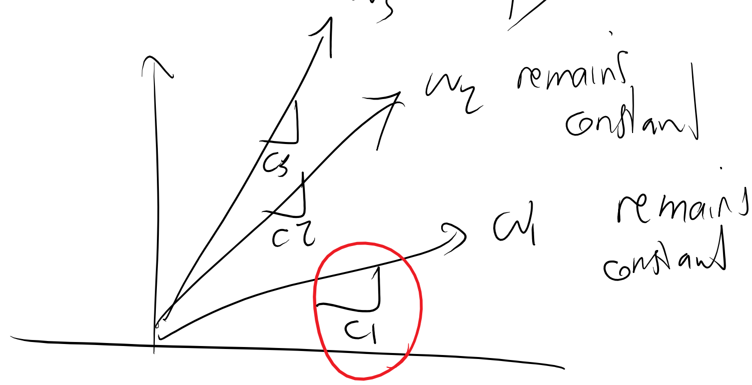
$$\mathbb{M}_{3\times3} = \mathbb{A}_{3\times6}\mathbb{B}_{6\times3} = \begin{pmatrix} \frac{\tilde{D}_{55}}{\rho} & \frac{\tilde{D}_{45}}{\rho} & \frac{\tilde{D}_{35}}{\rho} \\ \frac{\tilde{D}_{45}}{\rho} & \frac{\tilde{D}_{44}}{\rho} & \frac{\tilde{D}_{34}}{\rho} \\ \frac{\tilde{D}_{35}}{\rho} & \frac{\tilde{D}_{34}}{\rho} & \frac{\tilde{D}_{33}}{\rho} \end{pmatrix}$$

Interesting finding of the paper.

$$\omega_{3\times1} = t_{3\times1} - Z_{3\times3}\, \vec{v}_{3\times1}$$

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{12} \\ \sigma_{13} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_4 \\ S_6 \end{bmatrix}$$





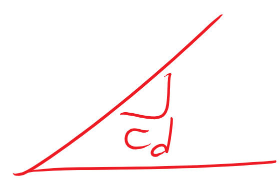
For isotropic material $\leftarrow c_1 = c_d = \sqrt{\dfrac{\lambda + 2\mu}{\rho}}$

dilatational

shear $\leftarrow c_2 = c_3 = c_s = \sqrt{\dfrac{\mu}{\rho}}$

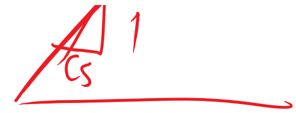
$$Z = \begin{bmatrix} \rho c_d & & 0 \\ 0 & \rho c_s & 0 \\ 0 & 0 & \rho c_s \end{bmatrix}$$

$$\Rightarrow \quad \omega_1 = t_1 - \rho c_d v_1$$

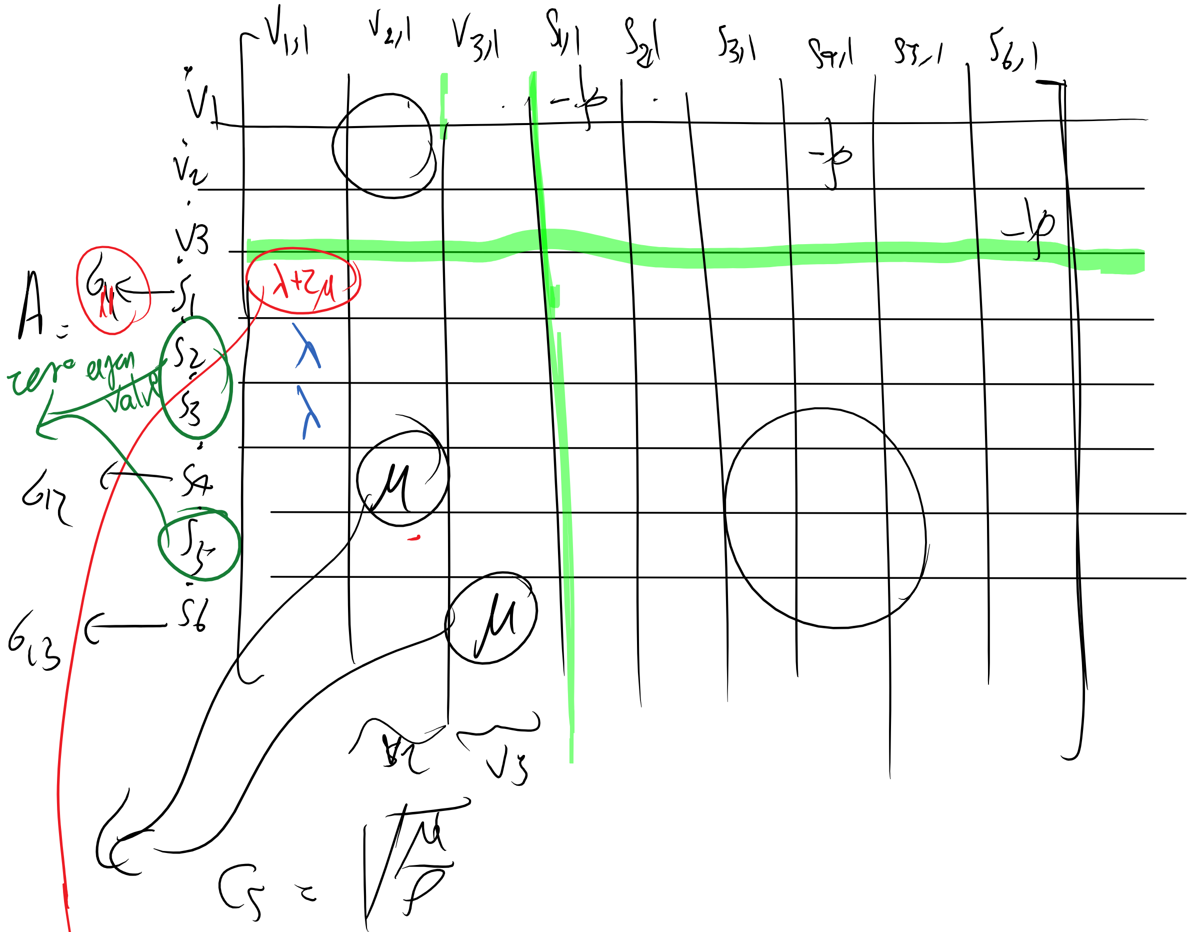


$$\omega_2 = t_2 - \rho c_s v_2$$

$$\omega_3 = t_3 - \rho c_s v_3$$



· For isotropic material A looks like



$$C_d = \sqrt{\frac{\lambda + 2\mu}{\rho}}$$

$$C_s = \sqrt{\frac{\mu}{\rho}}$$

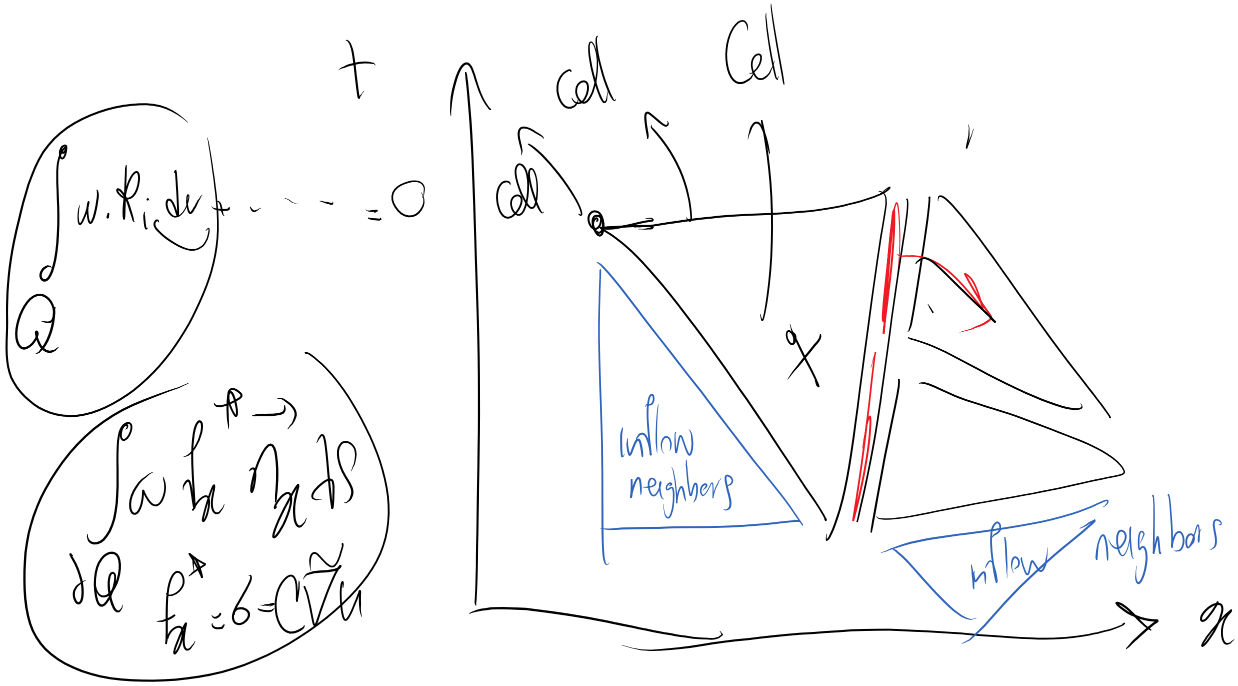
$\sigma_1 \longleftrightarrow v_1$

$\sigma_{12} \longleftrightarrow v_2$

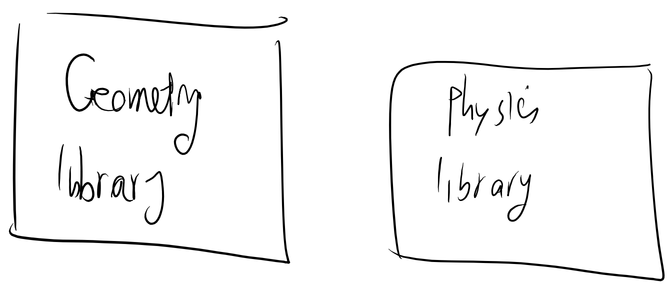
$\sigma_{13} \longleftrightarrow v_3$

Implementation:

Classes needed for a finite element implementation



Geometry objects



1. Cell
   Is a geometry object in 0D, 1D, 2D, 3D, … that provides certain functionalities (discussed later)

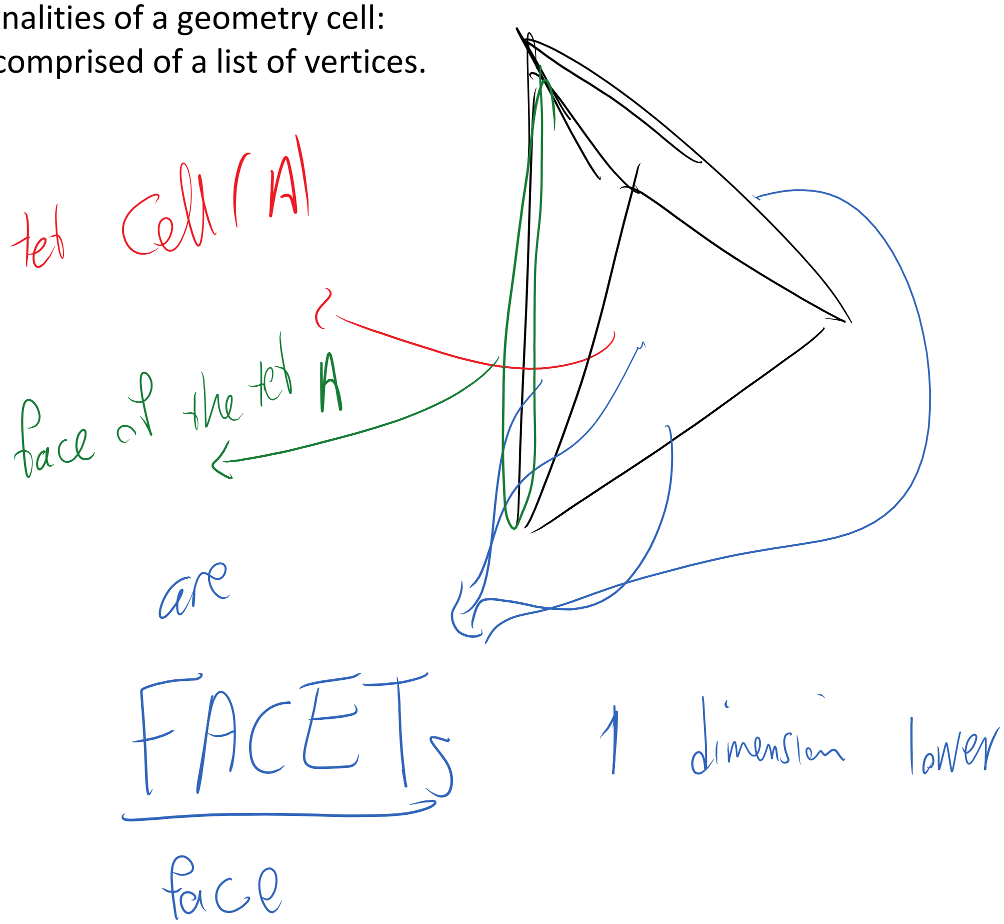
0D Cells are also vertices

Eventually element interiors, faces, etc. all will be built from geometry cells.

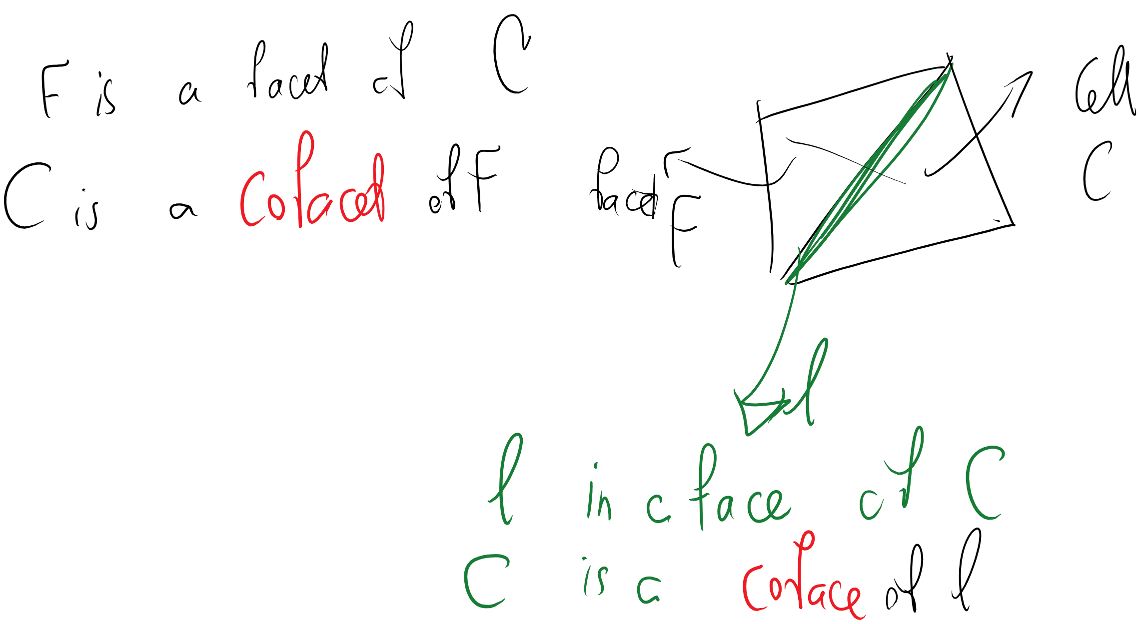
Functionalities of a geometry cell:

Functionalities of a geometry cell:

- It's comprised of a list of vertices.



The list of facets, facets of facets, etc. recursively is called the set of faces of a cell



- One of the functionalities we want is knowing facets, cofacets, faces, and cofaces of a cell

- We want to know the vertices of a cell

Geometry cell in the code:

GMeshing\GCell.h

class Gcell
List of vertices
void getVertices(vector<GVertexH>& verticesOut) const;

Facets, cofacets, and vertices of a cell

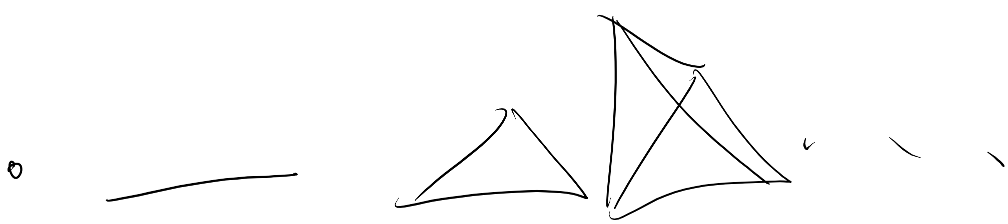
map<GCellID, GCellFacet> facets;
// vector<GCellID> extrusionFaces; // bottom, ..., top extrusion cells if any

vector< map<GRefinementLevel, GCellCofacet> > cofacets;
vector<GVertexH> vertices;

The Cell (Gcell) class is general. For any new element type, we need to derive a subclass for it

GMeshing\GCellSimplex.h

class GCellSimplex : public GCell → GCell Simplex is a sub lass of GCell



For each derived geometry cell type all we need to do is to implement a few virtual functions:

```
GCellSimplex(int geomCellOrderIn = 1, GCellID idIn = -1);
// other functions used in initialization
// the order of facets is important. It helps set the orver of vertices
virtual void setFacetsReadingMesh(vector<GCellID>& facetIDs, GCellH mesh);

virtual void TransferBaseFacetQuadCoord_2_BaseCofacetOrNbrQuadCoord(GCellH facetBase, const GQuadCoord& facetQuadCrd, GQuadCoord& cofacetOrNbrQuadCrd);
virtual void ComputeX_dXdAlpha_BaseCell(GCellH actualCell_no_b2t, GQuadCoord& quadCrd, vector<double>& X, GCellGeomProp& geomPropOut, bool compute_dX_dAlpha, bool computeX);
virtual void Compute_sdxFacet_from_sdxMatrix_in_Cofacet(GCellH facetBase, GCellGeomProp& geomPropOut);
```

Other things needed from a Gcell:

3. Neighborhood:

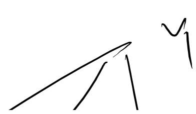
Can be shown by having (facets, cofacet, and containment (co-containment) information you can get any type of neightborhood information needed.

4. Geometry operations such as:
   a. Quadrature rule: giving the list of quadrature points, weights, for an integration order; Jacobians for face and interior integral.
   b. Normal vectors and volume.
   c. Bunch of coordinate transformation.

We have at least 3 different types of coordinate:

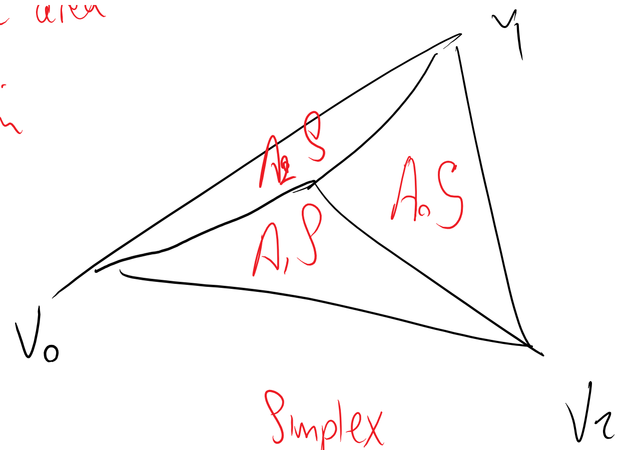
1 Alpha (A) Quadrature coordinate

S: surface area



S: surface area

used for integration



Simplex

for simplices

$\sum A_i = 1$

often we eliminate one

$(\cancel{A_0}, A_1, A_2, \ldots, A_n)$

in our code we drop the first one

2. X global coordinate

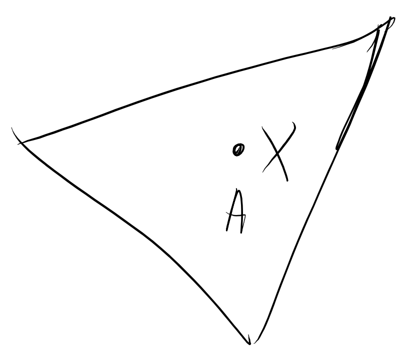
3. x basis coordinate

$T = T_0(\vec{x})\, a_0 +$

$T_1(\vec{x})\, a_1 + \ldots$

↓

basis



local coordinate

basis

basis are expressed in terms of a { local coordinate / basis coordinate

— Choices → Natural coordinate $x = A$

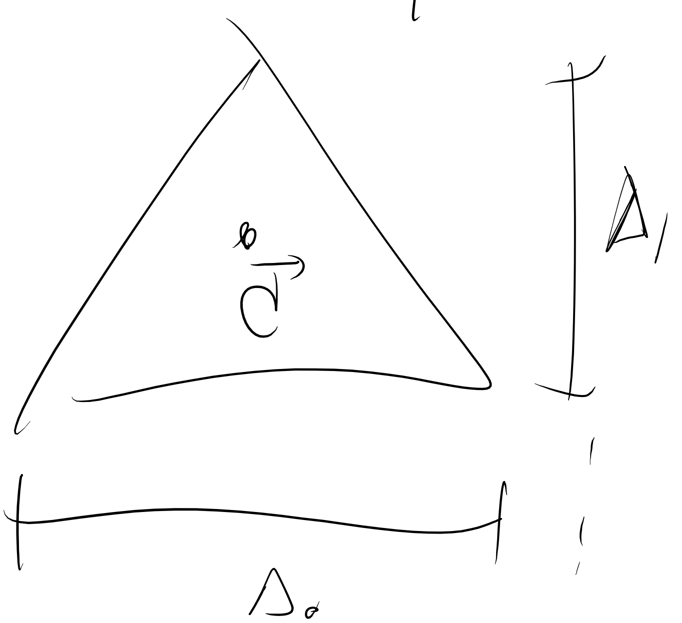
quadrature

$x = X$ ill-conditioning & loss of precision

$x = \frac{X - C}{\Delta}$

C → Centroid of element

Δ → span of element



Geometry needs to be able to do all coordinate transformations between A, X, x

This example shows what classes are needed

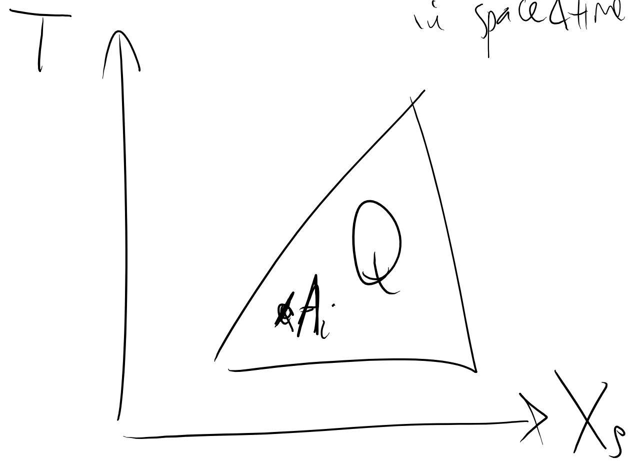
Inside integration

Elastodynamics $\dot{\rho}$ $\nabla \cdot \sigma$

Elastodynamics

$$\int_{Q} \hat{v} \, (\underbrace{\rho \ddot{u}}_{\dot{p}} - \underbrace{\nabla \cdot C \nabla u}_{\nabla \cdot \sigma} - \rho b) \, dV + \dots \int_{\partial Q}$$

u is 4th order in space&time



$X = (X_s, T)$

integration order $2 \times 4 - 1 - 2 = 5$

(1: weight, 2: sln)

Tri, int Order 5 $\longrightarrow$

$\langle$ Quad Pts $\rangle$;

$\langle W_i \rangle$ weights quad

$\langle A_i \rangle$ coordinates

stiffness — $K = 0$; $R = 0$ → residual n quad

for $i = 0$; $i < nQuad$; $++i$

A_i known

$A_i \longrightarrow x_i$ ↗ basis coordinate