2018/04/23

Monday, April 23, 2018 11:44 AM

Continue on storage classes:

PhyFieldVals.h

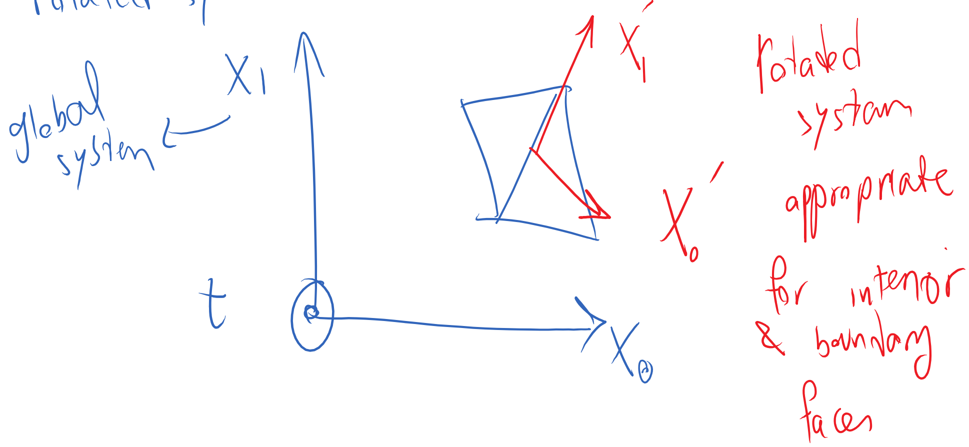
class PhyFieldVals

PhyElementFields cVal; // Cartesian values

PhyElementFields rVal; // rotated values

stores values in global Cartesian system

rotated system



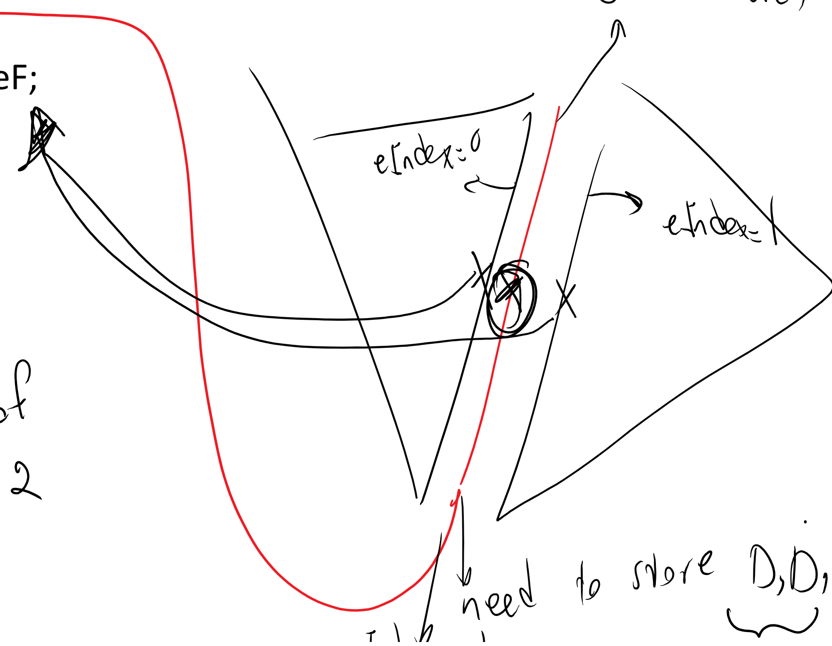
Storage for only one of the coordinate systems is:
class PhyElementFields

storage for element interior

PhyFieldElement eI;
vector<PhyFieldElement> eF;
bool bInterior;

$\int \hat{D}(D - D_{src})\,dv$

here eF is of size 2



need to store $D, \hat{D}, D_{src}$

eIndex = -1

need to store $\underbrace{D, \dot{D}}, D_{src}$

only depend on the interface element (-1)

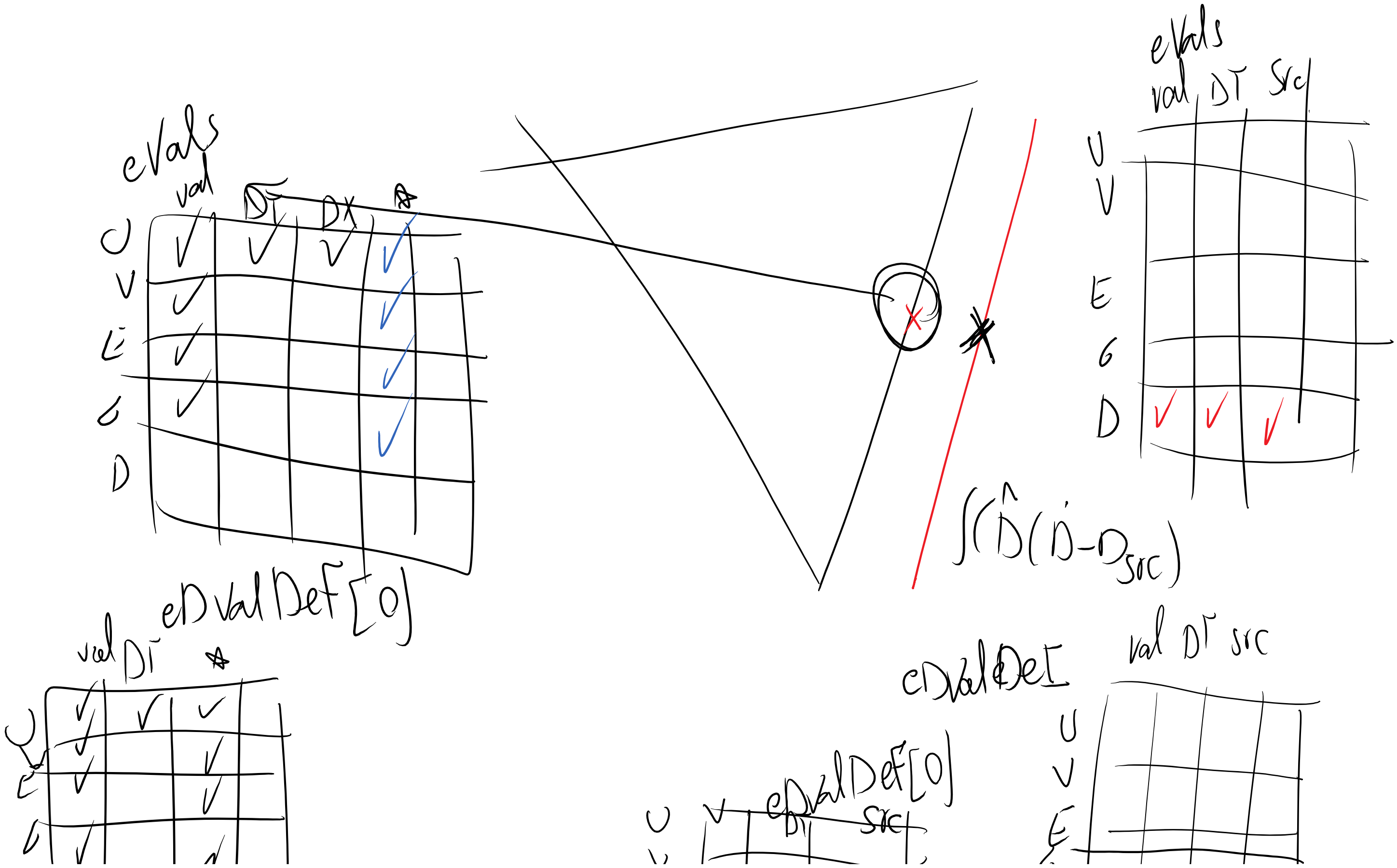
has dependency to eIndex =-1, 0, 1

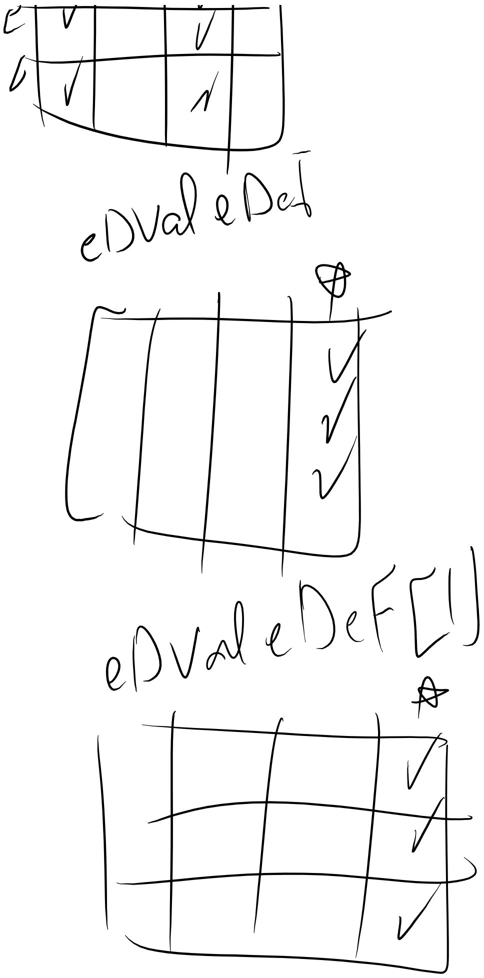
The last level of hierarchy are values stored for ONE of the elements in ONE coordinate system.

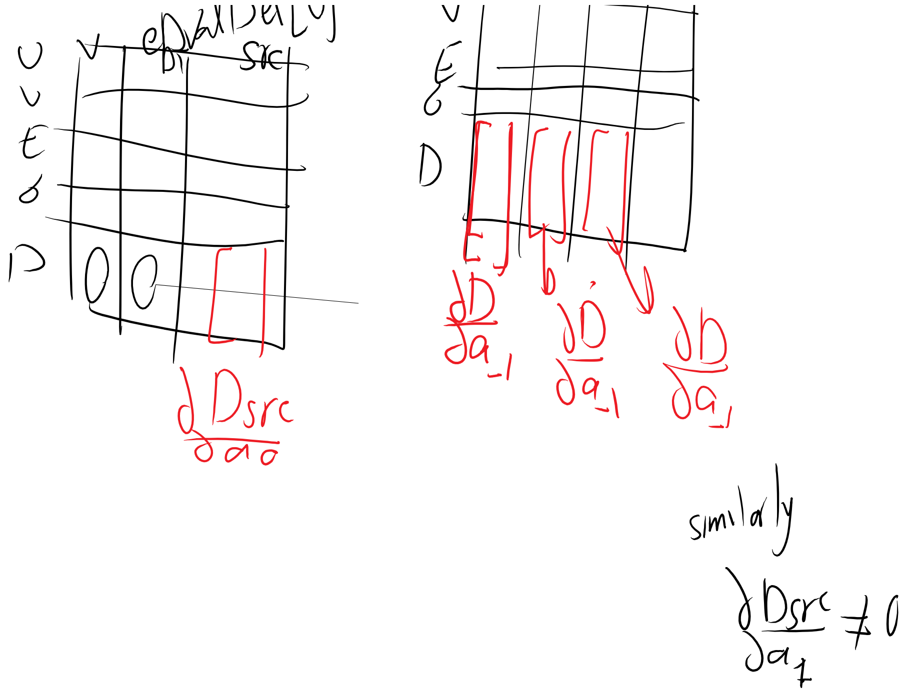
class PhyFieldElement

// values
mapPfc2Td eVals;

// derivatives (shapes)
mapPfc2DTd eDValDeI; // w.r.t. element interior (eIndex = -1)
vector< mapPfc2DTd > eDValDeF; // w.r.t. elements having facets at the PIC







The second storage we need at a quadrature point

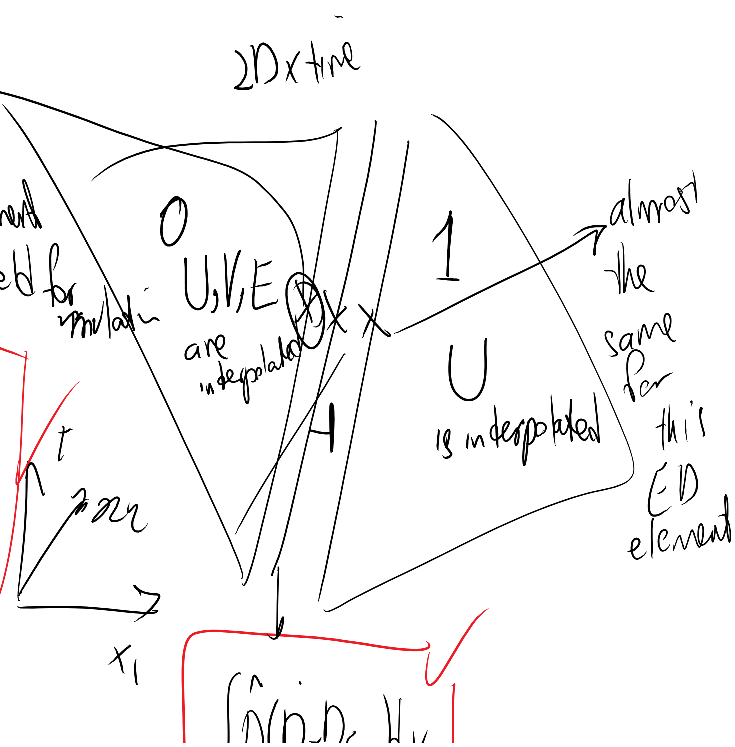
$$\int_Q (\hat{a}\, p + \hat{E}\, \sigma + \hat{u}(p b))\, dV + \ldots$$

for 0th element for 3 field formulation

$a:$

$$+ \int \hat{V} \sigma^* n_x ds - \hat{V} p^* n_t ds + [u]\hat{\sigma} n_t ds + [\dot{u}]\hat{\sigma} n_x ds + \hat{u}_b [u] n_t ds = 0$$

for this integration cell



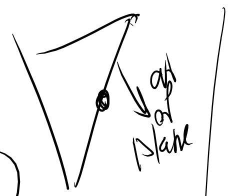
for this integration cell

x_1

$$\boxed{\int_{\Omega_I} \hat{D}(D - D_{src}) dV} + \int \hat{D}(D^* - D) n\, dS = 0$$

what is interpolated

eIndex 0



basis b_0

$U \quad (U_0, U_1) \quad p=3 \quad$ dof 2x20

$V \quad (V_0, V_1) \quad p=2 \quad$ 2x10

$b_1 \quad b_1$

$E \quad (E_{00}, E_{11}, E_{01}) \quad p=2$

$b_1 \quad b_1 \quad b_1$

3x10

90 dofs

eIndex 1

$U \quad (U_0, U_1) \quad p=3$

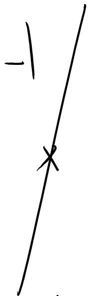
b_0

for eIndex 1

2x20

40 dofs

eIndex -1



$D \quad b_0 \quad p=3$

1x10

10 dofs

140 dofs for the patch

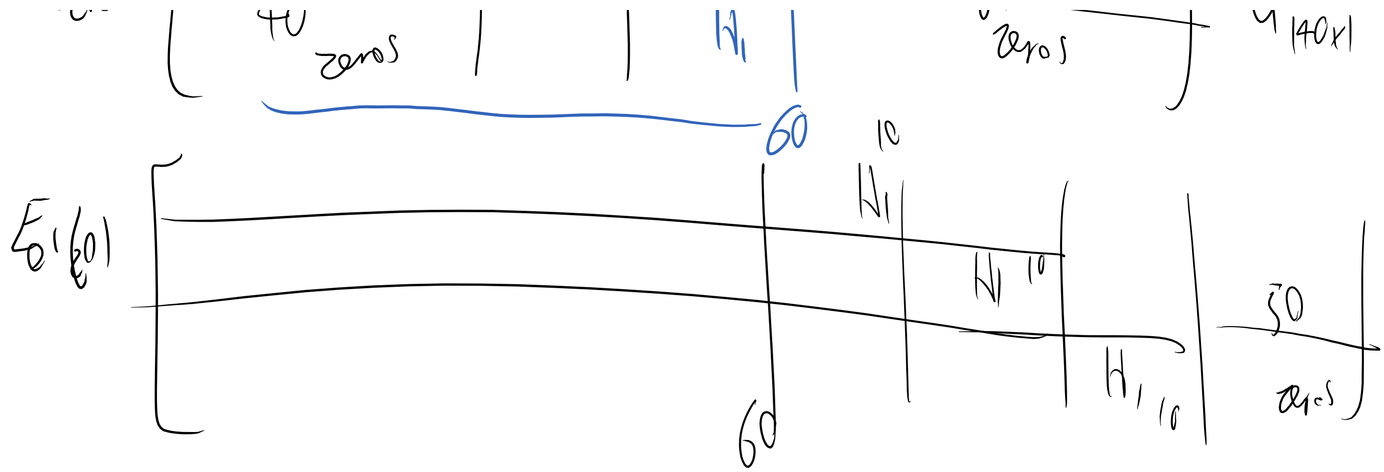
$$U_{el(0)} = \left[\begin{array}{cc|c} H_0 & & \\ & H_0 & 100 \text{ zeros} \end{array} \right] \begin{pmatrix} q_{patch} \\ 140\times1 \end{pmatrix}$$

20 20

b_0 is evaluated at the quad point

10 dofs

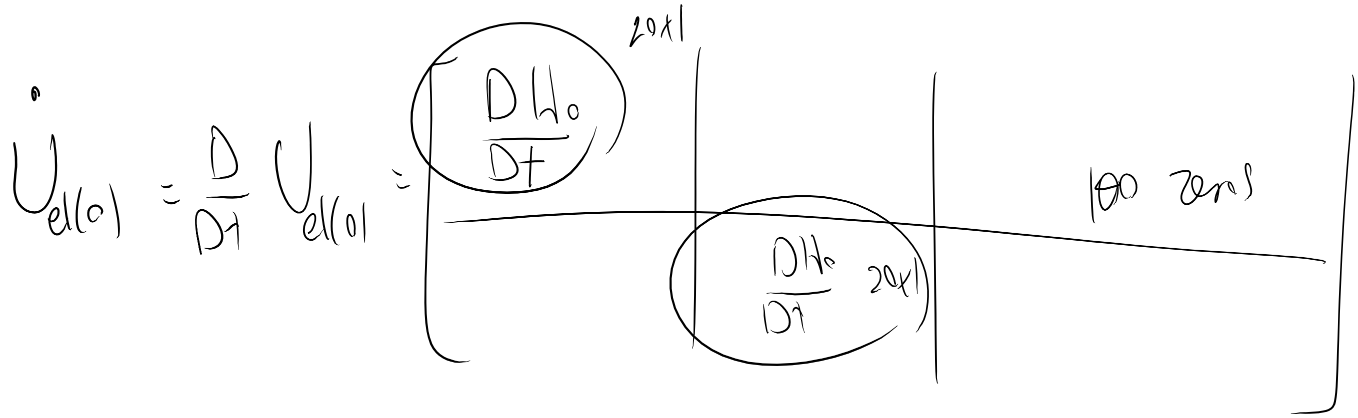
$$V_{el(0)} \left[\begin{array}{c|c|c|c} 40 \text{ zeros} & H_1 & H_1 & 80 \text{ zeros} \end{array} \right] q_{140\times1}$$



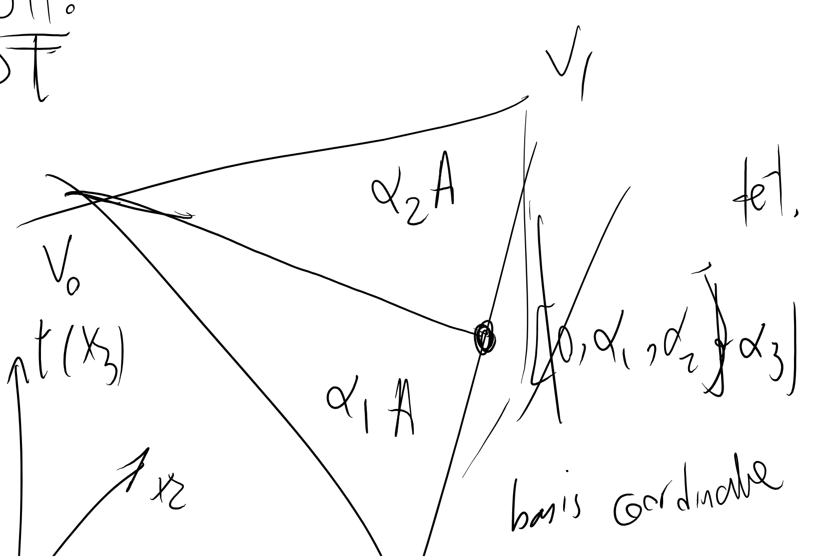
For element 0 we only need to pre-calculate two basis:

H_0, H_1

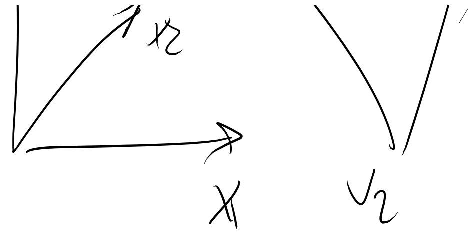
What if we needed to calculate uDot (time derivative of displacement)?



How do we calculate $\frac{DH_0}{DT}$



$H(x) = H(\alpha_1, \alpha_2, \alpha_3)$



basis coordinate

$x = [\alpha_1, \alpha_2, \alpha_3]$

$$\frac{dH}{dt} = \frac{\partial H}{\partial \alpha_i} \frac{\partial \alpha_i}{\partial T}$$

who provides this?

who computes this?

The basis of the pField can calculate dH/dxi:

class phyField
PhyBasisElement pBasis;

class PhyBasisElement
void getH_DH_D2H(eCoord& crd, VECTOR &H, V2TENSOR& DH, V3TENSOR& D2H, int derOrderMax) const;

basis coordinate

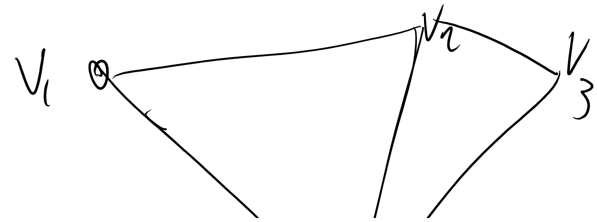
$$\frac{\partial x_i}{\partial X_I}$$

if $x = \alpha \rightarrow$ quad coordinate

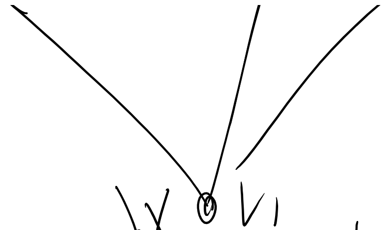
we need $\dfrac{\partial \alpha}{\partial X}$

global coordinate

e.g. $I=3$ $X_3 = T$



e.g. $I=3 \quad X_3=7$



Q1: How $\frac{\partial \alpha}{\partial X}$ is calculated?

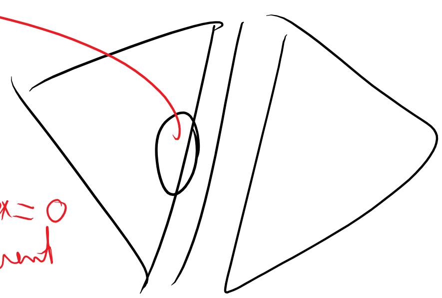
Q2: where $\frac{\partial \alpha}{\partial X}$ is stored?

$$\frac{\partial X}{\partial \alpha} = \left[V_1 - V_0 \,\middle|\, V_2 - V_0 \,\middle|\, V_3 - V_0 \right]^T$$

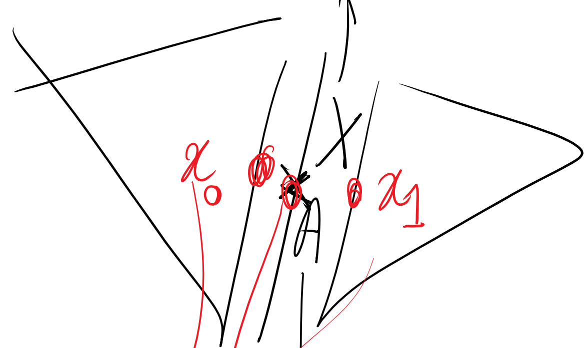
$$\frac{\partial \alpha}{\partial X} = \left(\frac{\partial X}{\partial \alpha} \right)^{-1}$$

general inverse

a) simple design
const Jacobian
geometry information
is stored at cIdx=0
element



in general
it's stored at basis
coordinates



class eCoord $\longleftrightarrow$ x

....

GeomPropBase e_gpb;

inside this we have

GCellGeomProp* geomPropPtr;

Purely geometry class
GMeshing\GCellGeomProp.h

```
class GCellGeomProp
{
    vector<double> v0; // coordinate of the first vertex
    MATRIX grad_dX_dAlpha;
```

for simplex

$$\frac{\partial X}{\partial \alpha} = \left[V_1 - V_0 \,\middle|\, V_2 - V_0 \,\middle|\, \cdots \,\middle|\, V_n - V_0 \right]$$

```
    MATRIX grad_dAlpha_dX ;
```

$$\frac{\partial \alpha}{\partial X} = \left(\frac{\partial X}{\partial \alpha} \right)^{-1}$$

general inverse

```
    double omega2Alpha;
```

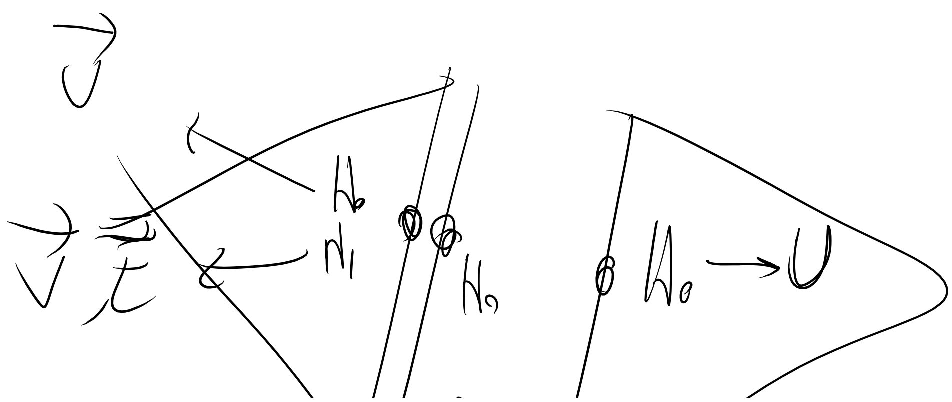
$$J = \underbrace{\det}_{\text{general}} \left(\frac{\partial X}{\partial \alpha} \right)$$

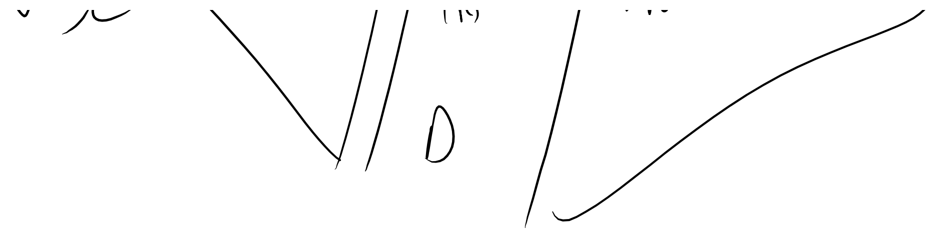
```
    // and other members related to normal vectors

    }
```

// look over this class for general geometry calculations

We need these shapes to

1. Calculate the values
2. Calculate the FEM shapes (der. w.r.t. element unknowns) of interpolated fields.





How about if we have a class that stores all these H's, $\frac{DH}{DX_I}$, $\frac{D^2H}{\partial X_I \partial X_J}$, ... at the quad point? $\Longrightarrow$

if interpolated

These are the building blocks of whatever we need to calculate FIELDS THAT ARE **INTERPOLATED**.

and their derivatives

U	V
∇U	∇V
$\vdots$	$\vdots$

physics\PhyStoreBasis.h
This file has storages for H, DH/DX, D2HDX2, … of distinct basis

---- look at this class
class IntHStorage (storage for all elements at a quad point)

class ElShapeStore (storage for ONE of the elements at the quad point)

Final note:
Where are these storages used? Pretty much everywhere.

Look at this function in PhyElementBase class

Coord

```
void PhyElementBase::ComputeFieldIntegrand(ptCoords& crds,
PhyFieldVals& fldVals, IntHStorage& basisShapes, int e_Index, compT cT,
phyFld fld, vsT cVH, rotT rT)
```

tensor fields & shapes

N, DN/DX, D^2N/Dx^2 ... stored here