Class hierarchy:

0. PhyPatch

1. PhyElement

```cpp
class PhyPatch: public PhyPatchData
{
    vector<PhyElementBase*> phyElementsBase;
}
```

```cpp
vector<GCH> allCells;
vector<GCH> activeCells;
```

```cpp
class PhyPatchData
{
}
```

PhyPatch receives these cells to form FEM objects.

These make the elements.

vector<PhyElementBase*> phyElementsBase;

here 3 elements

0. PhyPatch

1. PhyElement
2. PhyPhysics (each element has a list of physics) class PhyElementBase

Members of PhyPhysics and creation of PhyPhysics

For specific physics we need to derive them from a base PhyPhysics class.

There are many specific physics implementations. We use the notation of factory to create them.

PhyPhysics are created by a factory:

Physics/PhysicsFactory.h
PhyPhysics* createPhysics(subConfigRef subConRef);

PhyPhysics* createPhysics(subConfigRef subConRef)
{
    PhyPhysics* pp;
    int subConfigIndex = subConRef.subConfigIndex;
    int option;
    switch(subConRef.formulationT)
    {
        case CL:
            option = phyConf->subConf[subConfigIndex]->physics_options(0);
            pp = createCLInstance(option);
            break;
        case SL:
            pp = new SLPhysics();
            break;
    }

    // the use of the factory in PhyElement
    void PhyElementBase::setPhysics()
    {
        num_physics = descProp.subConfigRefs.size();
        physics.resize(num_physics);
        for(int i = 0; i < num_physics; i++)
        {
            physics[i] = createPhysics(descProp.subConfigRefs[i]);
            physics[i]->phyLocInElement = i;
            physics[i]->peParent = this;
            // physics[i]->patch = patch;
        }
    }

    By using this function we create the vector of PhyPhysics inside the element.

What is inside PhyPhysics:
2. PhyPhysics has a vector of interpolated tensor fields

```cpp
vector<PhyTensorField> pTFields;
```

4. class PhyTensorField

```cpp
vTensor<phyField> physicsFs;
```

For example, \( U \) has

\[
U = U_0 \text{ or } P_{2D}
\]

elastodynamics

\( E \) has \( E_{00}, E_{11}, E_{1}
\)

for 2D ED

5. phy field class:

```
class phyField

U = \sum \phi(x) \mathbf{a}_i
```
class phyField
PhyBasisElement pBasis;
VECTOR pCoef;

\[ U_0 = \sum_{i=0}^{N} \phi_i \cdot a_i \]

\[ \begin{bmatrix} a_0 \\ \vdots \\ a_{\text{dof}-1} \end{bmatrix} \]

\[ \begin{bmatrix} \phi_0 \\ \phi_1 \\ \vdots \\ \phi_{\text{dof}-1} \end{bmatrix} \]

start point is 160

store the location of dofs of different objects in patch dofs

dof patch

\[ \begin{bmatrix} 0 & 2 \times 20 & 0 \times 0 \\ 20 \times 20 & 2 \times 20 & 0 \times 0 \\ 0 \times 0 & 2 \times 20 & 2 \times 20 \end{bmatrix} \]
These starting point numbers play the role of dofMap for CFEMs and allow the code to assemble the matrices and vectors to the right place in global stiffness matrix.
Another important hierarchy of the Patch

\[
\begin{align*}
\langle \text{Physics} \rangle & \downarrow \quad U, V, E \\
\langle \text{Interpolated tensor field} \rangle & \\
\langle \text{Components} \rangle & \quad U_0, U_1
\end{align*}
\]
class PhyIntCellBase

PhyInt2EBasePtr interiorIntBase;
bool bInteriorInt;
vector<PhyInt2EBasePtr> facetIntBase;

If the cell is integrating any interior.

How many faces it integrates.
PhyElements and PICs are stored in PhyPatch

Class PhyPatch

....
vector<PhyElementBase*> phyElementsBase;
int num_elementsBase;

vector<PhyIntCellBase*> phylntsBase;
int num_phylntsBase;

Storage members:

Have to have a way to name tensors.

class PhyFldC

...  phyFld phyF;
compT  cT;  
  \begin{align*}
  & U, V, S, E, \ldots \\
  & \text{val} \quad D \tau, \n \end{align*}
\[ t_0^* = \left( t_0 \frac{Z^+ t_0 Z^-}{Z^+ + Z^+} + \frac{Z^+}{Z^+} (v^+ + v^-) \right) \]

(1-D)

We go from a fully bonded (\(D=0\)) to fully deflected solution.

\[ \text{val(\text{DR})...} \]

will need to store it at both places.
\[
\frac{ds_0}{da_I} = -\frac{Z}{Z^+ + Z^-} \left( \frac{Z}{Z^+ + Z^-} + \frac{Z}{Z^+} \frac{\partial D}{\partial a_I} \frac{r^+}{\partial a_I} \right) - \frac{Z}{Z^+ + Z^-} \frac{\partial D}{\partial a_I} \frac{r^+}{\partial a_I}
\]

domain of \( s_0 \)

are w.r.t. what?
So the challenge is that any component of a tensor can depend on all elements that are present at a PIC.

-----

The storage for all values and shapes for one quadrature points for one of the elements:

```cpp
class PhyFieldElement {
    mapPfc2Td eVals;
    mapPfc2DTd eDValDeI;
    vector< mapPfc2DTd > eDValDeF;
};
```