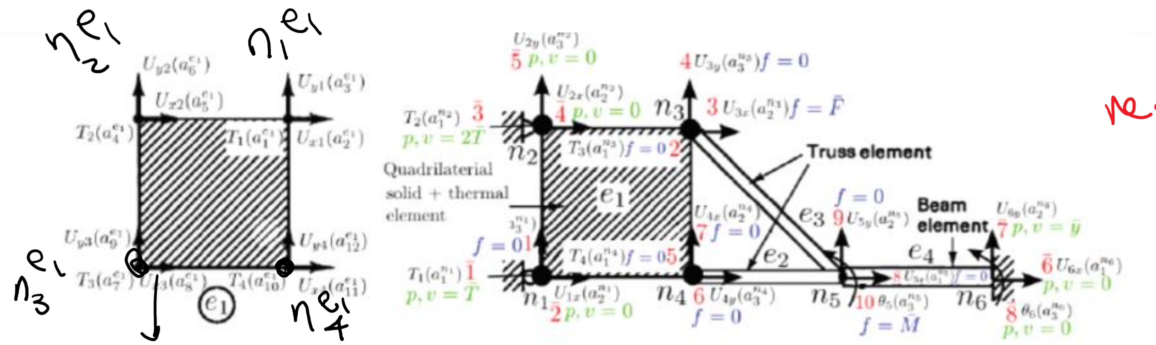
FEM2021/11/09-Coding

Tuesday, November 9, 2021 3:40 PM

Object oriented programming we deal with the interaction of classes

Here some important classes are element, node, Dof

--- Each class has a set of data members and functions



red global dof

solves solid U_x, U_y + thermal T

Data members of element class: examples from e_1

- id — 1
- neNodes — 4
- eNodes (LEM) — [3, 2, 1, 4] Map of element nodes
- edof (a → sln vector) — $a_{e_1} = [T^1\ U_x^1\ U_y^1 \mid T^2\ U_x^2\ U_y^2 \mid \ldots \mid T^4\ U_x^4\ U_y^4]$ ↓ $(\text{edof})_{e_1}$
- dofMap (M ↑ in nodes) — $[2\ 3\ 4 \mid \bar{3}\ \bar{4}\ \bar{5} \mid \bar{1}\ \bar{2}\ 1 \mid 5\ 6\ 7]$
- stiffness matrix — $K^{e_1}_{12\times12}$
- forces f^e_e, f^e_o, f^e_D — $f^e_e = (f^e_r + f^e_N + \ldots) - f^e_D$; f^e_o → other; f^e_D → Dirichlet
- eType — square
- {physics} → set, or list of

e_1

thermal, 2D elasticity

Examples of functions

virtual Calculate Stiffness (calculates k^e)

(A)

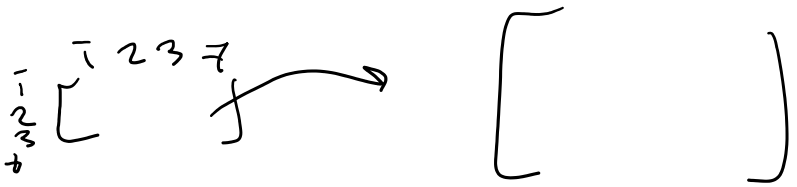
bar $k^e = \frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ beam = [ ] 4x4

Assemble Stiffness (Assembly k^e i to global structure)

same function for all elements

(B)

k^e, dofMap^e




```
class PhyElement
{
    friend ostream& operator<<(ostream& out, const PhyElement& dat);
public:
    virtual void setGeometry() = 0;
    virtual void setInternalMaterialProperties(PhyMaterial* pMat) = 0;
    void setNodeConnectivity_Sizes(int nNodeInElement, int ndofpnIn, vector<int>& eNodesIn, vector <PhyNode*>& eNodePtrsIn);

    void print(ostream& out);
    // Step 8: Element dof maps Me
    // Step 9: Set element dofs ae
    void setElementDofMap_ae(int ndofpn);

    // Step 10: Compute element sti
    ness/force (ke, foe (fre: source term; fNe: Neumann BC))
    virtual void Calculate_ElementStiffness_Force() = 0;   // example of type (A)

    // Step 11: Assembly from local to global system
    void AssembleStiffnessForce(MATRIX& globalK, VECTOR& globalF);   (B)

    // Step 14: Compute prescribed dof forces
    void UpdateElementForces_GlobalFp(VECTOR& Fp);

    // Step 15: Compute/output element specific data
    virtual void SpecificOutput(ostream& out) const {THROW("does not have implementation");};
```

Assembly has a general implementation for all element types:

```
void PhyElement::AssembleStiffnessForce(MATRIX& globalK, VECTOR& globalF)
{
    fee.resize(nedof);
    if (foe.size() == nedof)
        fee = foe;
    else
        fee = 0.0;

    int I, J;
    for (int i = 0; i < nedof; ++i)
    {
        I = dofMap[i];

....
}
```

=======

Group A are the functions that have different implementations
Like how the stiffness matrix is calculated

```
virtual void Calculate_ElementStiffness_Force() = 0;
```
// example of type (A)

Say I want to create a bar element

```
class PhyElementBar : public PhyElement
{
public:
    virtual void setGeometry();
    virtual void setInternalMaterialProperties(PhyMaterial* pMat);
    virtual void Calculate_ElementStiffness_Force();
    virtual void SpecificOutput(ostream& out) const;
    double L;
    double A;
    double E;
};
```

```
void PhyElementBar::Calculate_ElementStiffness_Force()
{
    // compute stiffness matrix:
    ke.resize(2, 2);
    double factor = A * E / L;
    ke(0, 0) = ke(1, 1) = factor;
    ke(1, 0) = ke(0, 1) = -factor;
}
```

$$k^{e}_{bar} = \frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

If you wanted to use a procedure based language

FORTRAN

Calculate Stiffness

```
{
    if (type == bar)
    {
        stiffness = AE/L [ 1  -1
                          -1   1 ]
    }
    else if ( beam )
    {
        ____________
    }
```