Quick Introduction to `NGSolve`

What is `NGSolve`?

- `NGSolve` is an **all-in-one** high performance multiphysics finite element software.
- Official Website: https://ngsolve.org/
- There are many similar softwares: Ansys, Comsol, Abaqus, deal.ii, MFEM, FEniCS, etc.

Installation

```
pip install --upgrade ngsolve
```

```
pip install --upgrade webgui_jupyter_widgets
```

Key Components

- Mesh
- Finite Element Space (built on Mesh)
- Weak Form/Linear Form (built on FES)
- Linear System Solver

Mesh

`NGSolve` uses `NetGen` to create mesh from geometry objects.

geometry objects can be constructed with `CSG2d` or `CSG` (Constructed Solid Geometry), where the geometry objects can easily compute

- intersection `*`
- difference `-`
- union `+`

FES

In `NGSolve`, there are multiple choices of FES: `H1`, `L2`, `HDiv`, `HCurl`, and one can mix them. Typically, the FES of H1 is constructed by

```
fes = H1(mesh, order=order, dirichlet=[])
```

The Dirichlet boundary constraints should be supplied using the labels.

Trial & Test Functions

Then we need trial & test functions on the FES to prepare the weak form and linear form.

```
u = fes.TrialFunction()
v = fes.TestFunction()
```

Weak Form

The bilinear form is constructed in a way like

```
a = BilinearForm(fes, symmetric=True)
a += grad(u)*grad(v)*dx + u * v * dx
```

or equivalently

```
a = grad(u)*grad(v)*dx + u * v * dx
```

Linear Form

The linear form is similar.

```
L = LinearForm(fes)
L += f * v * dx
```

or equivalently

```
L = f*v*dx
```

Assemble System

The linear system is formed by calling

```
a.Assemble()

L.Assemble()
```

Solve

- The matrix with the bilinear form is `a.mat`
- The vector with the linear form is `L.vec`

The solving process is usually like (if there are constraints, then slightly different)

```
sol = GridFunction(fes)

sol.vec.data = a.mat.Inverse(freedofs=fes.FreeDofs()) * L.vec
```

Example: Poisson Equation in Unit Square

$$-\Delta u = f(x) \quad \text{in } D, \quad u|_{\partial D} = g(x)$$

Example: Steady Heat Equation

$$-\nabla \cdot (k(x)\nabla u) = f(x) \quad \text{in } D, \quad u|_{\Gamma_1} = g(x), \quad \partial_n u|_{\Gamma_2} = 0$$

Q: What if $k(x)$ has high constrast?

The stiffness matrix computed from $\int_D k(x)u(x)v(x)dx$ lose accuracy due to inaccurate integration.

Example: Wave Equation with A Scatterer (soft)

$$-\Delta u - k^2 u = f, \quad \partial_n u - iku|_{\Gamma_D} = 0, \quad u|_{\Gamma_S} = 0$$

where f is a pulse source.