SIMPLE EXAMPLE: HEAT CONDUCTION

Sakina Rehman

May 3, 2020

This example focuses on solving a heat conduction problem, which is written in the weak formulation of partial differential equations (PDEs). Find the temperature $u \epsilon H^1(\Omega)$ such that for all $v \epsilon H_0^1(\Omega)$ holds

$$\int_\Omega v(\frac{\delta u}{\delta t}) + \int_\Omega c \nabla v . \nabla u = 0, \forall v, u(x,0) = g(x), u(x,t) = \begin{cases} -2x \epsilon \Gamma_{left} \\ -2x \epsilon \Gamma_{right} \end{cases}$$

where c is the thermal diffusivity. The following steps are required to solve this problem using *SfePy*:

- The domain Ω must be discretized to create a finite element mesh. The mesh can be loaded from the *meshes* folder or alternatively generated by the code (simple shapes)

```
filename_mesh = 'meshes/3d/cylinder.mesh'
```

- Regions are domains of integration and allow the user to define the initial and boundary conditions. The code below defines the domain Ω and the boundaries Γ_{left} and Γ_{right}

```
regions = {
        'Omega' : 'all'
        'Left' : ('vertices in (x < 0.00001)', 'facet'),
        'Right' : ('vertices in (x > 0.099999)', 'facet'),
}
```

- The field is defined as the discrete function spaces which can be defined using the number of components, region name, data type etc. The field can either be defined on a whole cell subdomain or on a surface region.

```
fields = {
        'temperature' : ('real', 1, 'Omega', 1),
}
```

- These discrete function spaces (FE spaces) can now be used to define variables. Variables can be in three forms: unknown field (for state variables), test (virtual) field and the parameter field, which is for variables with a known degree of freedom (DOF). The '1' in the code below shows a history size of 1, as the previous time step state is required for the numerical derivative. The value 'u' below is the name of the unknown variable.

```
variables = {
        'u' : ('unknown field', 'temperature', 0, 1),
        'v' : ('test field',  'temperature', 'u'),
}
```

- Materials can be given as the constant parameter c, as part of the material 'm'.

```
materials = {
        'm' : ({'c' : 1.0e-5},),
}
```

- The essential boundary conditions will also be set as constants. In this case, the value of u will be -2 and 2 on Γ_{right} and Γ_{left} respectively

```
ebcs = {
        'u1' : ('Left', {'u.0' : 2.0}),
        'u2' : ('Right', {'u.0' : -2.0}),
}
```

- To define the initial conditions, *NumPy* must be imported. The initial conditions apply to the entire domain Ω and ic max is a constant defined outside the function

```
import numpy as np

def get_ic(coors, ic):
        x, y, z = coors.T
        return 2 - 40.0 * x + ic_max * np.sin(4 * np.pi * x / 0.1)
functions = {
        'get_ic' : (get_ic,),
}
ics = {
        'ic' : ('Omega', {'u.0' : 'get_ic'}),
}
```

- The PDEs are now built as a combination of linear predefined terms. Each term has its own quadrature order and a region of integration. The integral specifies a numerical quadrature order.

```
  integrals = {
      'i' : 2,
  }
  equations = {
        'Temperature' : """dw_volume_dot.i.Omega(v, du/dt)
                           + dw_laplace.i.Omega(m.c, v, u) = 0"""
}
```

This simulation (full code in Appendix X) is then run in the Jupyter Notebook terminal using:

```
sfepy-run simple poisson_short_syntax.py
```

References