Finite Element Method for Heat Transfer

Rezgar Shakeri Leila Ghaffari

University of Colorado Boulder

Rezgar.Shakeri@colorado.edu *Leila.Ghaffari@colorado.edu*

December 10, 2020

Link to the Repository

Outline

Heat Equation

- Predicts the temperature of the body (domain) subjected to the heat source
- Derived from Fourier's law and the conservation of energy
- Of fundamental importance in diverse scientific fields

Fourier's Law: $$\boldsymbol{q} = -k\nabla u$$

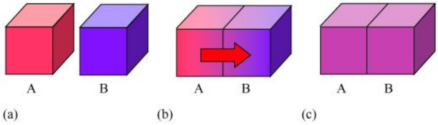


Figure: Heat conduction

Heat Equation

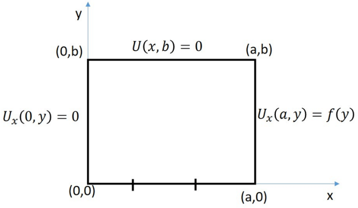
$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u = \alpha \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right), \tag{1}$$

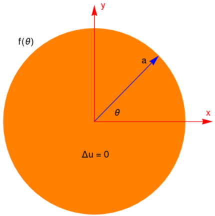
$u(x, y, z, t) =$ Temperature

$\alpha = \frac{k}{c_p \rho} =$ Thermal Diffusivity $c_p =$ Specific Heat Capacity

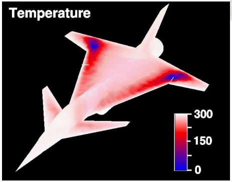
$k =$ Thermal conductivity $\rho =$ Mass density

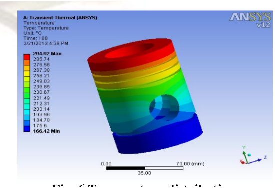
We can solve (1) analytically in **simple geometries** like rectangular (cubic in 3D) or circular by the **Separation of Variables** method.





But how about the complex region? Like aircraft, blade of the turbine?!





Finite Element Method (FEM)

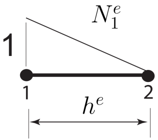
- Divide the domain to small pieces called **Element**
- Implement the weak form of **Heat Equation** for each element
- Expand the unknown variable (here temperature) using shape functions at each element
- Assemble the general form of the equation and find the solution

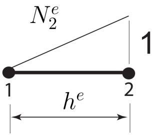
$$
\begin{aligned}
&\frac{\partial^2 u}{\partial x^2} = f(x) \Longrightarrow \int_0^L w(x)\left[\frac{\partial^2 u}{\partial x^2} - f(x)\right] dx = 0 \\
&\left(w\frac{\partial u}{\partial x}\right)_0^L - \int_0^L \frac{\partial w}{\partial x}\frac{\partial u}{\partial x} dx - \int_0^L w(x) f(x) dx = 0
\end{aligned}
\tag{2}
$$

Above equation is correct for each piece of the domain (element)

Finite Element Method (FEM)

$$
\begin{aligned}
u^{h^e}(x) = \sum_{a=1}^{n_{\text{en}}} N_a^e(x) d_a^e &= N_1^e(x) d_1^e + N_2^e(x) d_2^e \\
&= \begin{bmatrix} N_1^e & N_2^e \end{bmatrix} \begin{bmatrix} d_1^e \\ d_2^e \end{bmatrix} \\
&= \boldsymbol{N}^e(x) \cdot \boldsymbol{d}^e
\end{aligned}
$$





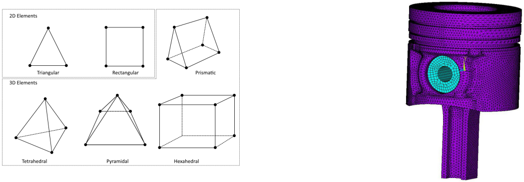
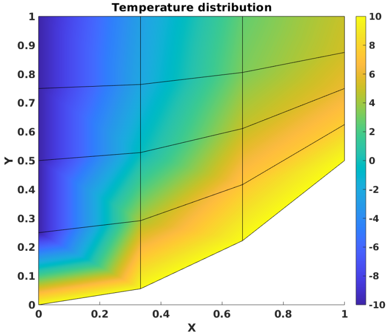


Figure: Different types of element and example of the Piston mesh

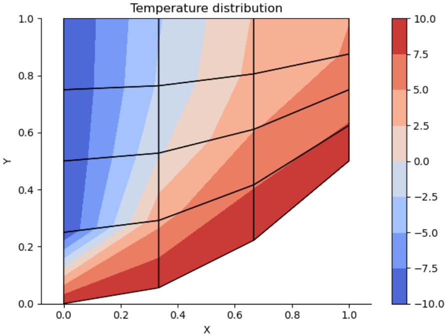
New Features

- MATLAB to Python
- Command line options
- Modularity
- Travis CI, Unit and Functional tests
- Documentation, Comments, User Manual
- First release - v1.0

MATLAB to Python



(a) MATLAB



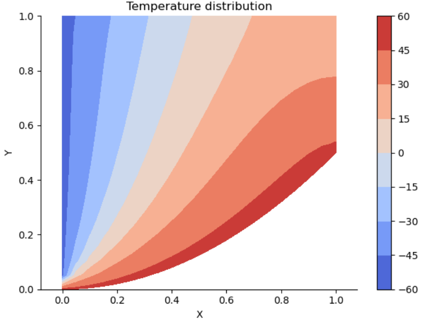
(b) Python

Figure: Temperature distribution, bottom and left edges are subjected to the 10 and -10 boundary conditions. Right and top edges are insulated.

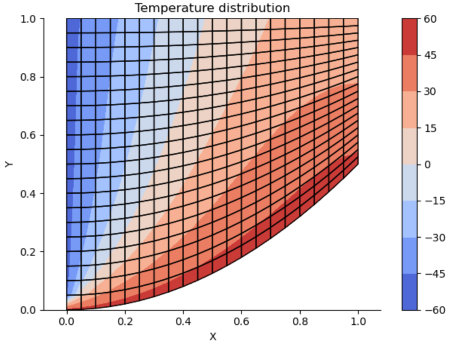
we found temperature by solving $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$

Command line options

```
python src/heat2d.py --num_elm_x 20 --num_elm_y 20
--T0_bottom 50 --T0_left -50 --heat_source 400 --flux_top
100 --grid
```



(a) Without grid



(b) With grid

Modularity

src_flux()

setup_ID_LM()

Dirichlet_BCs()

NeumannBCs()

connectivity()

assembly()

heat2delem()

phys_coord()

setup()

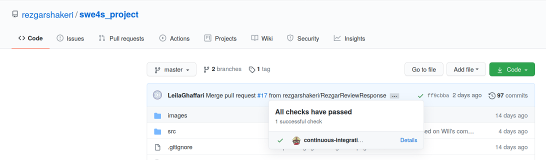
gauss()

Travis CI, Unit and Functional tests

```yaml
language: python

before_install:
    - wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
    - bash Miniconda3-latest-Linux-x86_64.sh -b
    - . /home/travis/miniconda3/etc/profile.d/conda.sh
    - conda update --yes conda
    - conda config --add channels r
    - conda create --yes -n test
    - conda activate test
    - conda install --yes python=3.8
    - conda install -y pycodestyle
    - conda install -y numpy
    - conda install -y matplotlib

script:
    - python src/test_FE_subroutines.py
    - bash src/test_heat2d.sh
    - pycodestyle src/test_FE_subroutines.py
    - pycodestyle src/heat2d.py
    - pycodestyle src/FE_subroutines.py
    - pycodestyle src/plot.py
```



Documentation, Comments, User Manual

```python
def phys_coord(nelx, nely):
    """ This function returns the physical coordinates of the nodes.

    Input:
    ------
    nelx:   integer
            number of elements in the x direction.
    nely:   integer
            number of elements in the y direction.

    Output:
    -------
    x:      float (1d array)
            the coordinate of the node in the x direction
    y:      float (1d array)
            the coordinate of the node in the y direction

    The geometry we are working on is like the following.
    (for nelx = 2, nely = 2)
    6---------7----------8
    |         |   (3)    |
    |   (2)   |      ----5
    |      ---4-----/    |
    3-----/   |   (1)    |
    |         |      ----2
    |   (0)   |     /
    |     ----1----/
    0----/
    There are 4 elements (numbering in parenthesis), and 9 nodes.
    Bottom edge (0 to 2) is y=0.5x^2. (see src/test_subroutines.py)
    This function returns x,y as 9x2 array for the above mesh.
    """
```

(a) Documentation

```python
# Get the setup properties
nel, lpx, lpy, nnp, ndof, nen, neq = setup(nelx, nely)

# Divide [0,1] by lpx (mesh in the x direction)
x0 = np.linspace(0, 1, lpx)
y0 = 0.5 * x0**2              # the bottom geometry line

y = np.zeros((nnp, 1))
for i in range(0, lpx):
    # Divide [0,1] by lpy (mesh in the y direction)
    y1 = np.linspace(y0[i], 1, lpy)
    for j in range(0, lpy):
        y[i + j*lpx] = y1[j]   # collection of y

x = np.zeros((nnp, 1))
for i in range(0, lpy):
    for j in range(0, lpx):
        x[j + i*lpx] = x0[j]   # collection of x

return x, y
```

(b) Comments

User Manual/README

First release - v1.0

Releases 1

v1.0 Latest
31 minutes ago

Packages

No packages published
Publish your first package

Contributors 2

LeilaGhaffari Leila Ghaffari

rezgarshakeri

Languages

- **Python** 98.2%
- **Shell** 1.8%

Question