Heat Transfer Analysis on a Glass Cup using FEniCS

Levent Aydinbakar

Computer Programming Course
Department of Mechanical Engineering
Bursa Technical University

Abstract

In this study, a heat transfer example is studied using the open-source tools Gmsh, ParaView and FEniCS. The heat diffusion in a glass cup is investigated solving the transient heat conduction equation in two space dimensions. First, the geometry is made using Gmsh. Later, a mesh is generated in the same tool. This mesh is imported into a Python script using FEniCS. Following this, the strong form of the heat diffusion equation is written and it is converted into a variational formulation. This variational formulation is written in the Python script using the Unified Form Language of FEniCS. Finally, the computations are made and the results are visualized in ParaView. As a result of this study, the temperature distribution in the glass cup is found.

December 13, 2022

Contents

List of Tables

List of Figures

1 Problem Definition

In two dimensional heat transfer problem shown in Figure 1 there is a glass cup with the heat diffusion coefficient of $\alpha = 0.34$ mm^2/s filled with water at 89°C in a room at 22°C This study assumes that

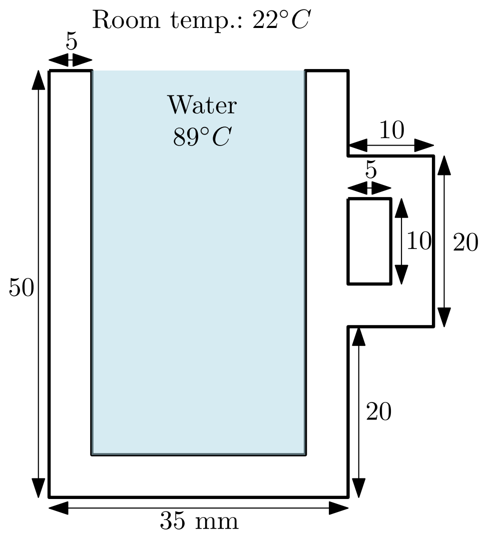

Figure 1: Schematic of the problem

- the water and room temperatures are constant,
- the convection and radiation heat transfers are neglected, and
- the heat diffusion coefficient of the glass cup is constant in space and time.

2 Mesh generation

The open-source tool Gmsh [1] is used for generating the Computer Aided Design (CAD) of the computational domain. Then, using the same tool, the mesh is generated.

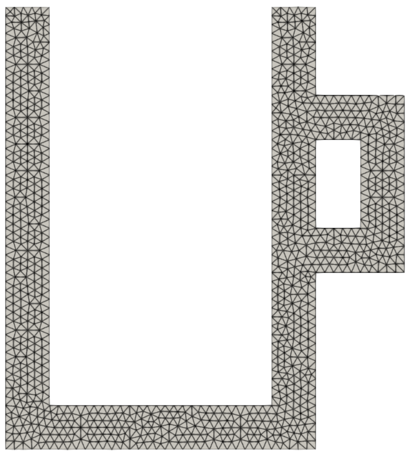

Figure 2: Mesh on the computational domain

Figure 2 shows the triangular mesh elements on the computational domain.

Table 1: The number of points (np) and the number of elements (ne) in the mesh

np	ne
$1,138$	$1,966$

The number of nodes and the number of elements are given in Table 1.

3 Boundary Conditions

The boundary conditions are set as constant and 22°C on the red, and 89°C on the blue boundaries shown in Figure 3.

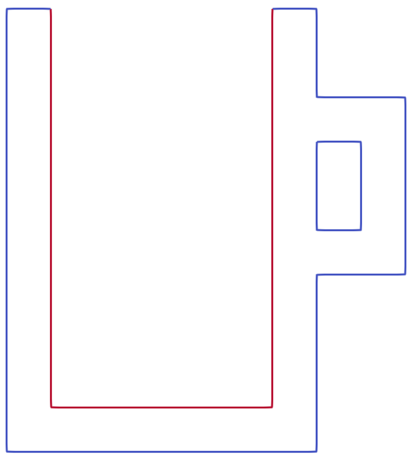

Figure 3: Boundary conditions. The *red* and *blue* colors represent the boundaries where the temperatures are set to 89°C and 22°C, respectively

4 Formulation

The heat diffusion (or transient heat conduction) equation is written as follows:

$$\frac{\partial u}{\partial t} = \nabla \cdot (\alpha \nabla u), \tag{1}$$

where u, t and α are temperature, time and diffusion coefficient. $\frac{\partial u}{\partial t}$ is temperature change in time and can be written as follows

$$\frac{\partial u}{\partial t} = \frac{u - u^{t_c - 1}}{\Delta t}, \tag{2}$$

using a first-order bacward Euler scheme in finite difference method. Here, t_c represents the current time step, $t_c - 1$ is the previous time step. The heat diffusion equation then can be written as,

$$\frac{u - u^{t_c - 1}}{\Delta t} = \nabla \cdot (\alpha \nabla u). \tag{3}$$

We can write the weak form of this equation by multiplying each term with a test function w and integrating over computational domain Ω as follows:

$$\int_\Omega \frac{u - u^{t_c - 1}}{\Delta t} w \mathrm{d}\Omega = \int_\Omega \nabla \cdot (\alpha \nabla u) w \mathrm{d}\Omega. \tag{4}$$

The right-hand-side of this equation is arranged by integration by parts as,

$$\int_\Omega \nabla \cdot (\alpha \nabla u) w \mathrm{d}\Omega = \int_\Gamma \alpha w \frac{\partial u}{\partial n} \mathrm{d}\Gamma - \int_\Omega \nabla w \cdot (\alpha \nabla u) \mathrm{d}\Omega. \tag{5}$$

Here, Γ represents boundary of the Ω and $\frac{\partial u}{\partial n}$ is the derivative of temperature in the outward normal direction on the boundary. We need to set the first term on the right-hand-side to zero because $w = 0$ on Γ where the solution of u is known. Hence, the heat diffusion equation can be written as follows:

$$\int_\Omega \frac{u - u^{t_c - 1}}{\Delta t} w \mathrm{d}\Omega = -\int_\Omega \nabla w \cdot (\alpha \nabla u) \mathrm{d}\Omega. \tag{6}$$

By reordering this, we obtain the weak form as:

$$\int_\Omega u w \mathrm{d}\Omega + \int_\Omega \Delta t \alpha \nabla u \cdot \nabla w \mathrm{d}\Omega = \int_\Omega u^{t_c - 1} w \mathrm{d}\Omega. \tag{7}$$

5 Computations

The variational formulation of the transient heat conduction equation can be written in a Python script using the Unified Form Language (UFL) [2] as follows:

```
F = u*w*dx + dt*alpha*dot(grad(u), grad(w))*dx - ut*w*dx
```

Here, `w`, `dx`, `dt`, `alpha` and `ut` represent the $\mathbf{w}$, $\mathrm{d}\Omega$, Δt, α and $\mathbf{u}^{t_c-1}$ in Equation (7). The `grad()` and `dot()` are functions from FEniCS [3, 4, 5] to take the gradient of a vector and to take the dot product of two vectors.

The complete script for this computations are given in Appendix A.

6 Results

The script given in Appendix A generates an output file in XDMF format. This output can be opened and visualization can be made using ParaView [6].

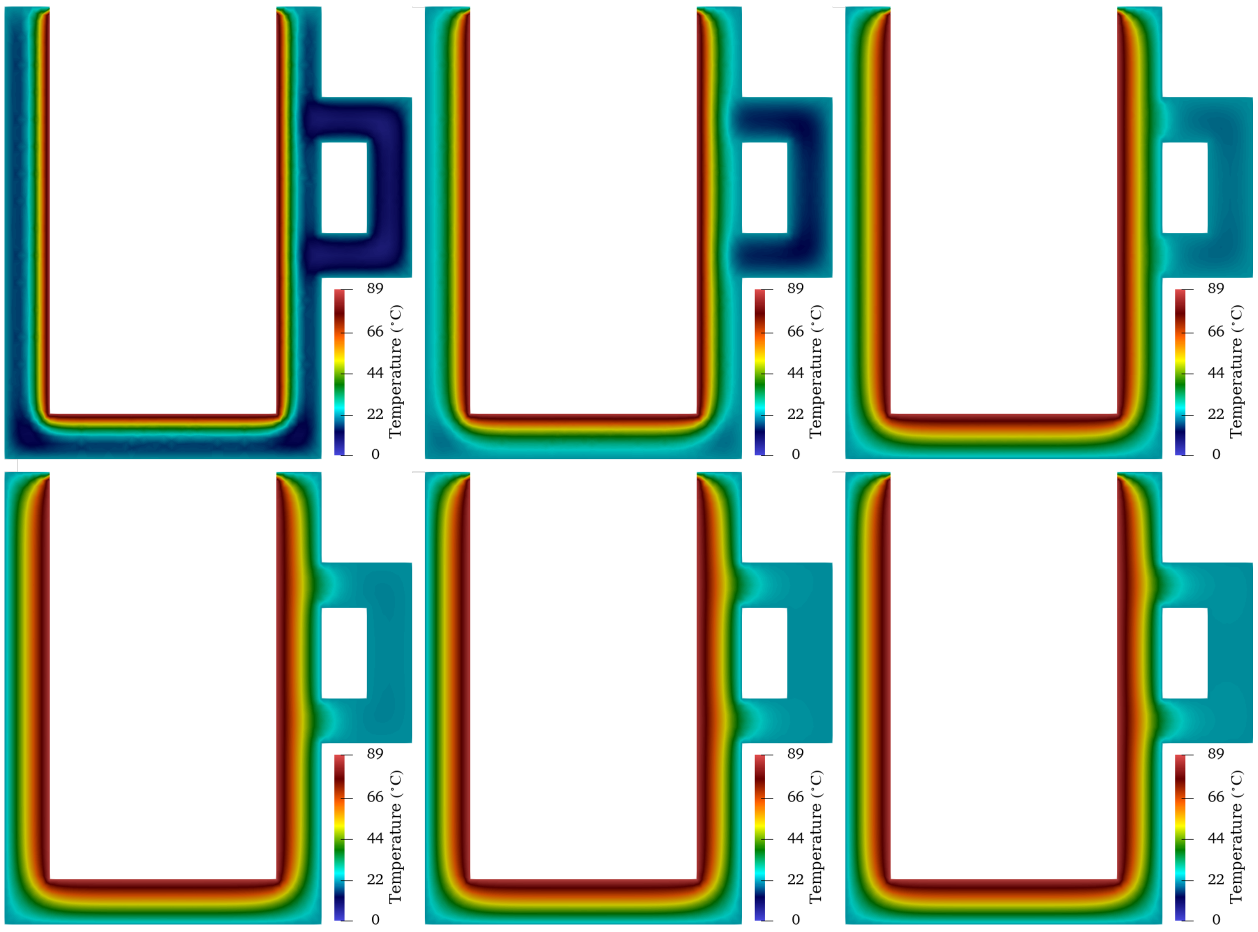

Figure 4: Temperature distribution at different instances. $t = 5$ sec, $t = 10$ sec, $t = 20$ sec, $t = 30$ sec, $t = 50$ sec, and $t = 100$ sec from left to right, from top to bottom

Appendices

A The complete script

```
# Import necessary modules
from dolfin import *

# Read mesh in XDMF
mesh=Mesh()
with XDMFFile("MESH/cup2d_mesh.xdmf") as f:
  f.read(mesh)

# Read boundary numbers from XDMF
dim=mesh.topology().dim()
mvc = MeshValueCollection("size_t", mesh, dim)
with XDMFFile("MESH/cup2d_bcs.xdmf") as f:
  f.read(mvc, "boundary_numbers")
  boundaries=cpp.mesh.MeshFunctionSizet(mesh, mvc)

# Define function space
V = FunctionSpace(mesh, 'CG', 1)

# Set the boundary condition
inside  = DirichletBC(V, Constant(89), boundaries, 1)
outside = DirichletBC(V, Constant(22), boundaries, 2)
bcs=[inside, outside]

# The trial and test functions
u = TrialFunction(V)
w = TestFunction(V)

# Diffusion coefficient
alpha = 0.34

# Time step size
dt = 1
T=200
nstep=int(T/dt)
ut = Function(V)

# Formulation
F = u*w*dx + dt*alpha*dot(grad(u), grad(w))*dx - ut*w*dx
a, L = lhs(F), rhs(F)

# Time-stepping loop
u = Function(V)
t=0
vtkfile = File('OUTPUT/u.pvd')
for i in range(nstep):
  solve(a == L, u, bcs)
  ut.assign(u)
  u.rename("Temperature","T")
  vtkfile << (u)
  t+=dt
```

References

[1] Gmsh. `https://gmsh.info/`. Accessed: 2022-12-16.

[2] M.S. Alnaes, Olgaard K.B. Logg, A., M.E. Rognes, and G.N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*, 40, 2014.

[3] FEniCS. `https://fenicsproject.org/`. Accessed: 2022-12-16.

[4] Martin Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E Rognes, and Garth N Wells. The fenics project version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[5] A. Logg, K.-A. Mardal, and G.N. Wells. Automated solution of differential equations by the finite element method. 2012.

[6] ParaView. `https://www.paraview.org/`. Accessed: 2022-12-16.