2D Poisson equation

Gilbert François Duivesteijn

February 14, 2022

This Matlab implementation is for verification and validation for the python code "2D Poisson equation, finite difference". The Matlab PDE Toolbox v2017a is used. The Jupyter Notebook is located at [1]. Let's analyse the domain on the unit square (figure 1) and solve the 2D Poisson equation

$$\kappa \nabla^2 u + f = 0 \tag{1}$$

with

$$f(x, y) = -\sin(x)\cos(y), \tag{2}$$

boundary conditions

$$u = 0 \qquad \text{on} \quad [E_1, E_2, E_3, E_4] \tag{3}$$

$$\tag{4}$$

and initial condition

$$u(t = 0) = 0 \qquad \text{on} \quad F_1 \tag{5}$$

Matlab has a generic PDE formula that you can tune to your problem by setting the coefficients. The solvepde function models the equation:

$$m\frac{\partial^2 u}{\partial t^2} + d\frac{\partial u}{\partial t} - \nabla \cdot (c\nabla u) + au = f \tag{6}$$

The coefficients to set are m, d, c, a and f. To define (1), we set $m = 0$, $d = 1$, $c = \kappa$, $a = 0$ and $f = @fn\_fxy$.

```
%% Specify the PDE model
specifyCoefficients(    ...
    model,              ...
    'm', 0,             ...
    'd', 1,             ...
    'c', 1,             ...
    'a', 0,             ...
    'f', @fn_fxy        ...
);
```

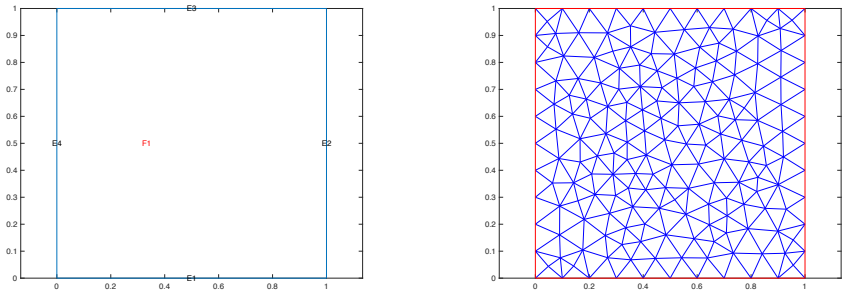

Figure 1: Computational domain with edge and face labels (left). Unstructured grid (right).

The function `fn_fxy` is implemented in a separate file

```
%% file: fn_fxy.m
function f = fn_fxy( location, state )
    f(1,:) = -sin(location.x) .* cos(location.y);
end
```

The Dirichlet boundary condition implies that the solution u on a particular edge or face satisfies the equation

$$hu = r \tag{7}$$

where h and r are functions defined on $\partial\Omega$. The boundary conditions from (3) can be constructed by specifying $u = 0$:

```
%% Boundary conditions
applyBoundaryCondition(model,            ...
    'dirichlet',                         ...
    'edge', [1, 2, 3, 4],                        ...
    'u', 0);
```

Let's set the initial condition from equation (5) by creating a function and reference the function pointer in the function setInitialConditions.

```
%% Initial conditions.
setInitialConditions(model, @fn_u0);
```

Generate the mesh, solve the PDE and get the results from the result object:

```
%% Generate mesh.
generateMesh(model, 'Hmax', 0.1);

%% Time domain
t = 0:0.01:10;

%% Solve the PDE
result = solvepde(model, t);
u = result.NodalSolution;
```

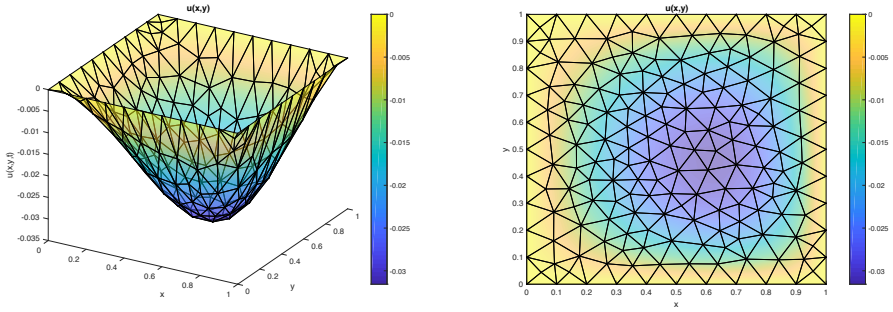

Figure 2: Solution u at $t = \infty$

References

(1) Poisson equation, finite difference.ipynb

(2) Matlab PDE Toolbox documentation