PDE Toolbox example

Gilbert François Duivesteijn

February 14, 2022

This is a simple example, showing how to use the Matlab PDE toolbox[1]. The recommended workflow is

- Create PDE[2]
- Define geometry[3]
- Apply boundary conditions[4]
- Specify coefficients[5]
- Set initial conditions[6]
- Generate mesh[7]
- Solve the PDE[8]

Let's analyse the L-shaped domain (figure 1) and solve the molecular diffusion equation

$$\frac{\partial u}{\partial t} = D\nabla^2 u \tag{1}$$

$$\tag{2}$$

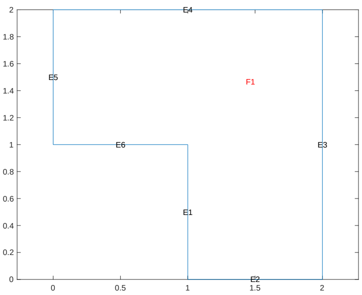
with boundary conditions

$$\frac{\partial u}{\partial \mathbf{n}} = 0 \qquad \text{on} \quad [E_2, E_3, E_4, E_5] \tag{3}$$

$$u = 0 \qquad \text{on} \quad [E_1, E_6] \tag{4}$$

$$\tag{5}$$

[1] PDE Toolbox documentation
[2] createpde
[3] decsg
[4] applyboundarycondition
[5] specifycoefficients
[6] setinitialconditions
[7] generatemesh
[8] solvepde



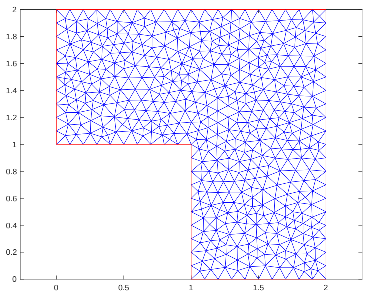


Figure 1: Computational domain with edge and face labels (left). Unstructured grid (right).

and initial condition

$$u(t=0) = e^{-16(x-\frac{1}{2})^2 - 16(y-\frac{1}{2})^2} \qquad \text{on} \quad F_1 \tag{6}$$

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

$$m\frac{\partial^2 u}{\partial t^2} + d\frac{\partial u}{\partial t} - \nabla \cdot (c\nabla u) + au = f \tag{7}$$

The coefficients to set are m, d, c, a and f. To define (2), we set $m = 0$, $d = 1$, $c = D$, $a = 0$ and $f = 0$.

```
%% Specify the PDE model

specifyCoefficients(    ...
    model,              ...
    'm', 0,             ...
    'd', 1,             ...
    'c', 0.05,          ...
    'a', 0,             ...
    'f', 0              ...
);
```

Note that the Dirichlet and Neumann boundary conditions are defined in a generic way, that can be tailored to the specific use case by setting the coefficients. The Dirichlet boundary condition implies that the solution u on a particular edge or face satisfies the equation

$$hu = r \tag{8}$$

where h and r are functions defined on $\partial\Omega$. Generalized Neumann boundary conditions imply that the solution u on the edge or face satisfies the equation

$$\mathbf{n} \cdot (c\nabla u) + qu = g \tag{9}$$

The coefficient c is the same as the coefficient of the second order differential operator in the generic partial differential equation (7). When specifying the boundary conditions, we give the type (e.g. 'neumann' or 'dirichlet'), the list of edges, as seen in figure 1 and the coefficients. The boundary conditions from (3) and (4) are:

```
%% Boundary conditions

applyBoundaryCondition(model,              ...
    'neumann',                             ...
    'edge', [2, 3, 4, 5],                  ...
    'g', 0,                                ...
    'q', 0                                 ...
);

applyBoundaryCondition(model,              ...
    'dirichlet',                           ...
    'edge', [1, 6],                        ...
    'u', 0);
```

Let's set the initial condition from equation (6) by creating a function at the end of the script and reference the function pointer in the function setInitialConditions.

```
%% Initial conditions.

setInitialConditions(model, @fn_u0);

%% Function declarations

function u0 = fn_u0(location)
    u0(1,:) = exp(-16*(location.x-1.5).^2 ...
                  -16*(location.y-1.5).^2);
end
```

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;
```

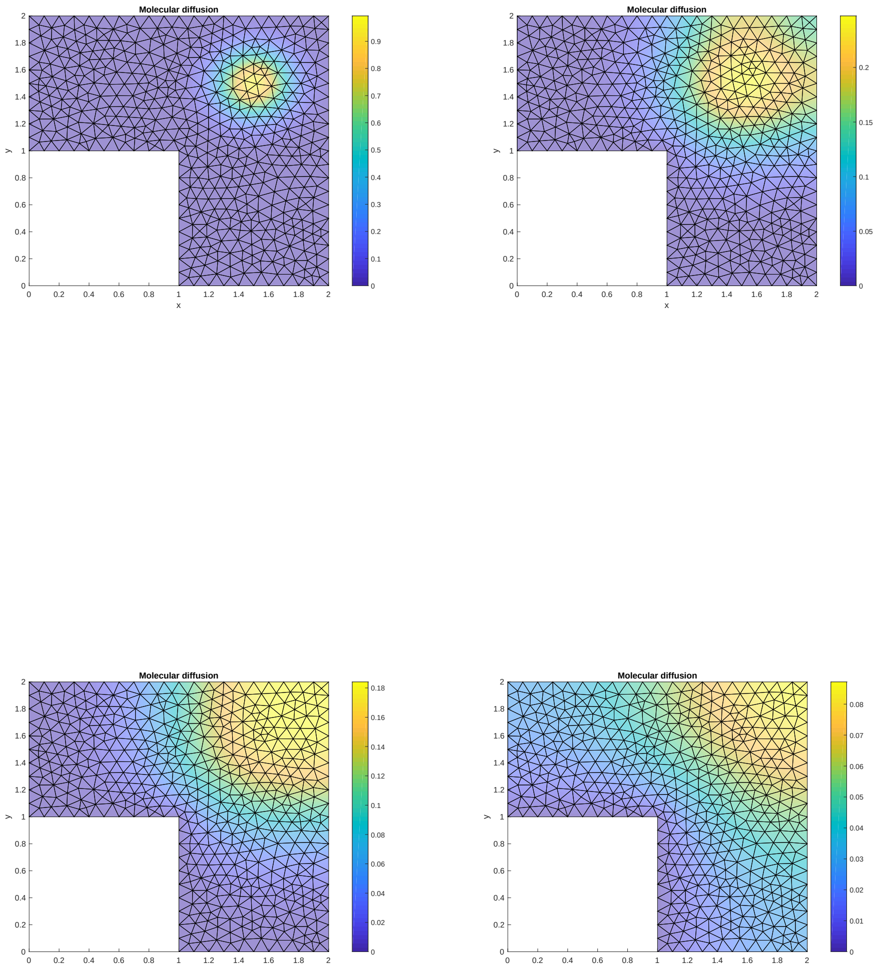


Figure 2: Solution u at $t = 0$, $t = 1$, $t = 2$ and $t = 10$.

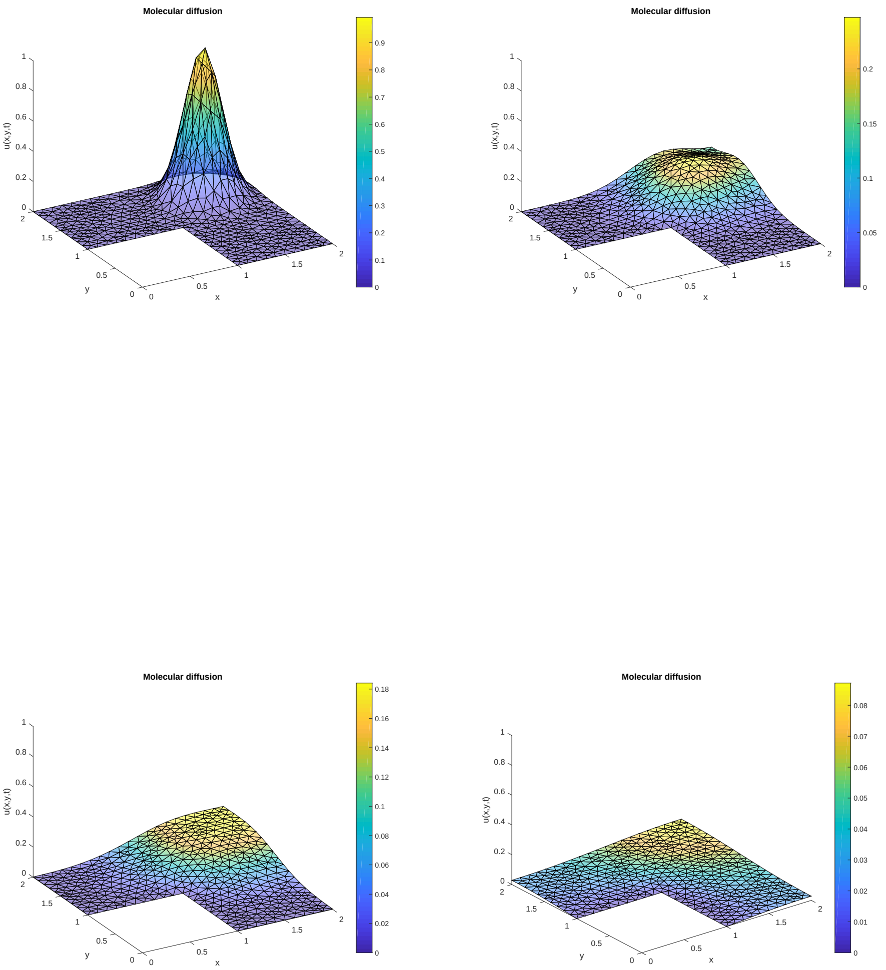


Figure 3: Solution u at $t = 0$, $t = 1$, $t = 2$ and $t = 10$.