Applied Mathematics - MATH-0504
Documentation for getFD

Xavier ADRIAENS, Julien DULAR

November 23, 2018

1 Input

The inputs of the program consist in four files:

- a text file (`.txt`) that defines the geometry of the problem,
- a matlab file `geometryTags.m` that defines the type of each region of the geometry file,
- a matlab file `parameters.m` that defines
  - the name of the geometry file,
  - the equation to be solved in the domain (wave, diffusion, ...),
  - the specification of the boundary conditions (value of Dirichlet, ...),
  - the initial condition(s) (if needed),
  - the numerical parameters (time step, ...),
- a matlab file `nodeEquations.m` that contains the finite difference schemes.

1.1 Geometry file

Illustrations are proposed in Fig. 1.

- The two-dimensional domain where the equation must be solved is an open subset of $\mathbb{R}^2$ denoted by Ω. It can have an arbitrary shape and can be multiply connected. The boundary $\partial\Omega$ of this domain is denoted Γ and is partitioned into α_Γ sets with $\alpha_\Gamma \in \mathbb{N}_0$, denoted as Γ_a with $a \in \{1, \ldots, \alpha_\Gamma\}$.
- The corresponding geometry file is a `.txt` file. It contains an array of integers of size $(N_y,\ N_x)$ that defines the finite difference grid. Grid points inside the domain have the value 0 (by default), grid points on boundary Γ_a have the value a and grid points inside Ω^C (not in the problem domain) have the value -1. All lines of the file must have the same length. If you consider a non-rectangular domain Ω, complete a rectangle with -1 values.
- The geometry file is read by the `dlmread` Matlab function and put into a matrix $\mathbf{G}$.

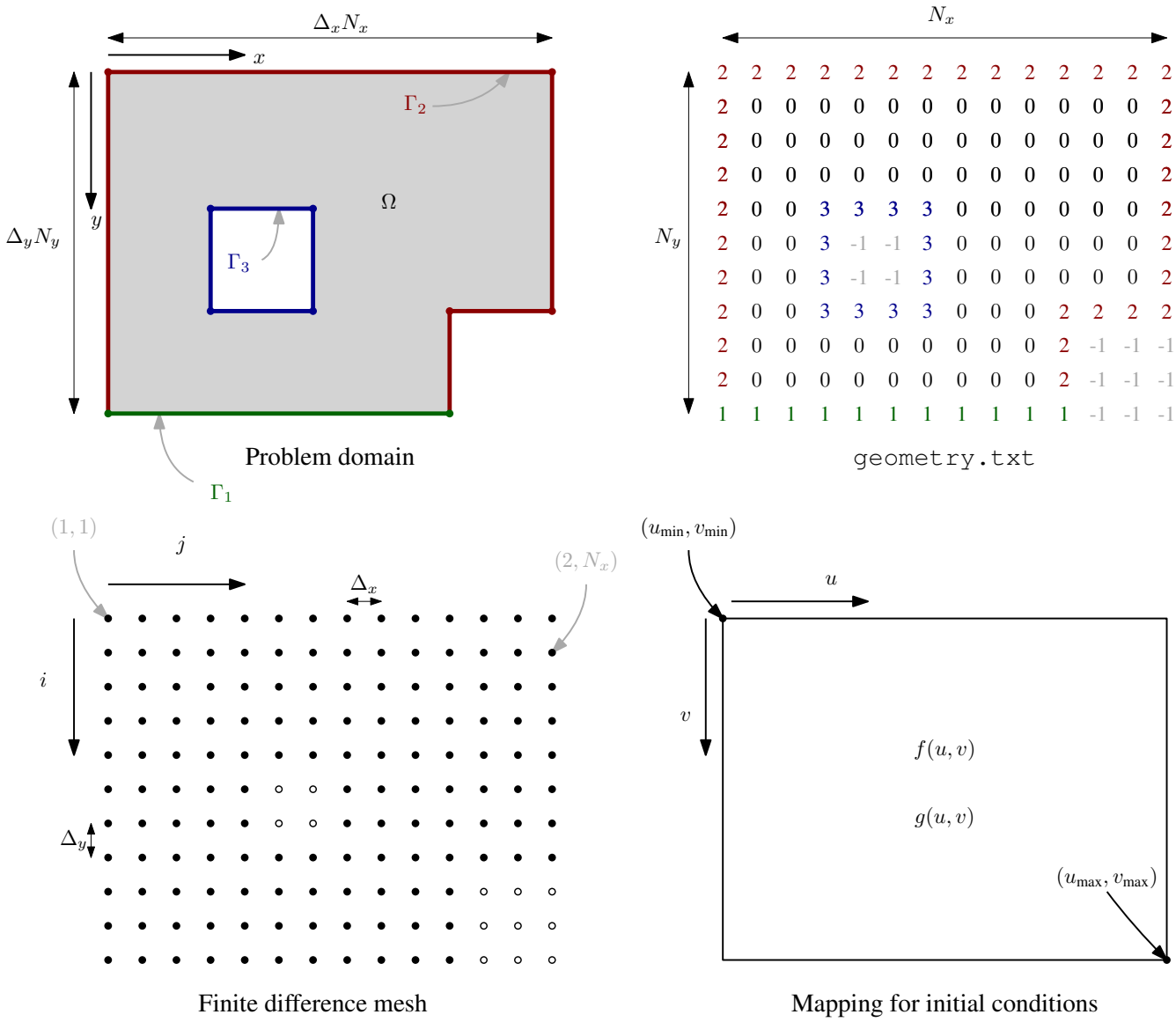

Figure 1: Summary schematics for conventions and notations.

1.2 Geometry tags file

The `geometryTags.m` file associates each integer value in the geometry file with a given type of point. By default, 0 corresponds to points inside the domain, defined as the `BULK`, and -1 are outside the domain. Each boundary part Γ_a is associated to a type of boundary condition among the following possibilities:

- `DIR`: First Dirichlet boundary condition $u = \phi$
- `DIR2`: Second Dirichlet boundary condition $u = \phi_2$
- `NEU`: Neumann boundary condition $\partial_n u = \text{Cst}_{\text{neu}}$
- `ABC`: Absorbing boundary condition $\partial_t u + a\partial_t u = \text{Cst}_{\text{abc}}$

So, from a given geometry, you are free to choose the boundary conditions on each Γ_a, separately. As an example, a possible `geometryTags.m` file associated to the problem introduced in Fig. 1 is

```
BULK = 0;
DIR = 1;
NEU = 2;
DIR2 = 3;
```

1.3 Parameters

The `parameters.m` file specify constants and function related to the problem to solve. A list of them is given below

- `filename`: name of the file containing the geometry
- `a`: diffusion coefficient, transport velocity or wave velocity
- `equation`: index to choose which equation to solve in the bulk
- `coord_upLeft_corner`: vector containing $(u_{\text{min}}, v_{\text{min}})$
- `coord_downRight_corner`: vector containing $(u_{\text{max}}, v_{\text{max}})$
- `neuCst`: Neumann boundary condition right hand side Cst_{neu}
- `abcCst`: Absorbing boundary condition right hand side Cst_{abc}
- `phi`: Dirichlet boundary condition right hand side ϕ (type 1)
- `phi2`: Dirichlet boundary condition right hand side ϕ_2 (type 2)
- `f`: Initial condition $u(t=0) = f$
- `g`: Initial condition $u_t(t=0) = g$
- `T_final`: Simulation time
- `Dt`: Time step
- `Dx`: Spatial step Δ_x
- `Dy`: Spatial step Δ_y

1.4 Node equations

- The `nodeEquations.m` file contains the finite difference schemes.
- A finite difference scheme of a *linear* partial difference equation with *constant coefficients* takes the following form, at a grid point (i, j) and for finding solution at time step $n + 1$,

$$\sum_{\beta=1}^{N_\beta} c_\beta u^{n+(\delta n)_\beta}_{i+(\delta i)_\beta\ ,\ j+(\delta j)_\beta} + c_0 = 0, \tag{1}$$

  with $N_\beta+1$ the total number of terms in the scheme, c_β the coefficient of these terms (c_0 corresponds to a possible independent term). The solution at grid point (i, j) and time step n is denoted $u^n_{i,j}$. The unknowns are $u^{n+1}_{i,j}$ and the solution is known at all preceding time steps.

- Such a scheme is encoded into a matrix E (`eqnBulk` in the code) of 4 columns and a number of lines corresponding to the number of terms in the scheme. The line of the matrix corresponding to term β writes

$$\begin{pmatrix}(\delta n)_\beta & (\delta i)_\beta & (\delta j)_\beta & c_\beta\end{pmatrix}. \tag{2}$$

  For c_0, the corresponding line writes

$$\begin{pmatrix}\text{NaN} & \text{NaN} & \text{NaN} & c_0\end{pmatrix}. \tag{3}$$

  If $c_0 = 0$, this line can be deleted.

- For example, a possible finite difference scheme for the diffusion equation $u_t - k(u_{xx} + u_{yy}) = 0$ writes (implicit, centred in space)

$$u_{i,j}^{n+1} - u_{i,j}^n - h_x(u_{i+1,j}^{n+1} - 2u_{i,j}^{n+1} + u_{i-1,j}^{n+1}) - h_y(u_{i,j+1}^{n+1} - 2u_{i,j}^{n+1} + u_{i,j-1}^{n+1}) = 0, \tag{4}$$

  with $h_x = k\Delta t/\Delta x^2$ and $h_y = k\Delta t/\Delta y^2$. The corresponding node equation matrix $\mathbf{E}$ writes

$$\mathbf{E} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2h_x \\ 1 & 0 & 0 & 2h_y \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & -h_x \\ 1 & 0 & -1 & -h_x \\ 1 & 1 & 0 & -h_y \\ 1 & -1 & 0 & h_y \\ \text{NaN} & \text{NaN} & \text{NaN} & 0 \end{pmatrix}, \quad \text{or, equivalently,} \quad \mathbf{E} = \begin{pmatrix} 1 & 0 & 0 & 1+2h_x+2h_y \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 1 & -h_x \\ 1 & 0 & -1 & -h_x \\ 1 & 1 & 0 & -h_y \\ 1 & -1 & 0 & h_y \end{pmatrix}.$$

  Note that it is not mandatory to gather terms corresponding to the same grid points into a single line (it is just more efficient). Also, the ordering of lines is arbitrary. Finally, if $c_0 = 0$, the corresponding line is useless (and might be deleted for efficiency).

- Be careful to the signs of the coefficients when you create a new node equation! Everything is assumed to be in the left-hand side in the scheme (like in equation (1)).

- The equations that are imposed on boundary nodes are specified the same way. However, boundary conditions typically depend on normal derivatives that depends on the boundary orientation. The $\mathbf{E}$ matrix related to a boundary equation is therfore slightly different if the boundary is above, below, to the right or to the left of the node. Instead of having only one matrix $\mathbf{E}$, boundary conditions have a matrix $\mathbf{E}$ for any possible case (corner, left boundary, right boundary, *etc*.).

- Boundary conditions using *ghost point* are implemented. Ghost points are an involved way to impose boundary conditions in finite differences.

2 Output

The output of the code is a three-dimensional matrix containing the solution u at each grid point and at each time step.