OOPDELight in a Nutshell

Part 1

Preface

OOPDELight is a small excerpt from the OOPDE class library. It contains all files to solve PDEs of second order on intervals in 1D and rectangle geometries in 2D.

The classes derived from PDE superclass are not included, but all necessary second level classes and some user classes for basic OOPDE operations.

Simple 1D Problems

The only possible domain in 1D is the interval, $\Omega_I = [a,b] \subset \mathbb{R}$. The basic differential equation is here an ODE (since we have only one dimension). It reads

$$-(cu_x)_x + b \cdot u_x + au = f\,,$$

plus a boundary condition. We have in OOPDE Neumann, Robin, periodic and Dirichlet boundary condition implemented. Let be Γ the boundary of Ω_I.

Of cause, we can use the FEM to solve a **bounday value problem** for this **ODE**. Let $I = (-1,1)$ and $\Gamma_I = \{-1,1\}$.

FIrst, we discretize the domain. Plotting an 1D grid makes less sense (but it may possible) so we only call the display method.

```
interval = Interval([-1,1],0.1);
interval.display;
```

```
interval =
    Grid object of class Interval with
    33 mesh points
    2 edges/lateral faces
    32 elements.
```

Define Lagrange elements.

```
FEM = Lagrange11D;
```

Let $c = 1$, $b = -1$ $a = 1$ and

$$f = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ otherwise} \end{cases}.$$

We assume homogenious Nemann boundary conditions, natural in FEM, i.e. no entries from that appear in the linear system.

We need the full set of matrices for stiffness K, mass M and convection terms C as well as a source term vector F.

```
[K,M,F] = FEM.assema(interval,1,1,'10*(x>0)');
C = FEM.convection(interval,0);
```

Solve the linear system and plot the solution over the grid.

```
u = (K+M+C)\F;
interval.plot(u,'Color',[0 0.25 0]);
```

Now we change the boundary condition and consider homogenious Dirichlet boundary conditions.

First we have to define the boundary segments, where the conditions should be fulfilled. In 1D, we have only the two boundary points. Here, we set $u = 0$ on Γ_I, i.e everwere on the boundary.

For that, we create a boundary matrix. In this matrix, OOPDE stores the relation boundary segments and boundary conditions. (Every boundary segment can have a different condition.) The method dirichletBC has here on argument that stores the value as char. After creating the boundary condition matrix we can use assemb to compute the matrices and vectors related with the boundary conditions. The fist output arguments are devoted to Robin/Neumann boudayry conditions, while the second pair, here H and R store the contribution of Dirichlet boundary conditions to the linear system.

```
interval.makeBoundaryMatrix(interval.dirichletBC('0'));
[~,~,H,R] = FEM.assemb(interval);
```

Again, solve the linear system. Note, that we have to write `H'*H` here. The factor `1e2` is a penalty factor to ensure that the conditions holds.

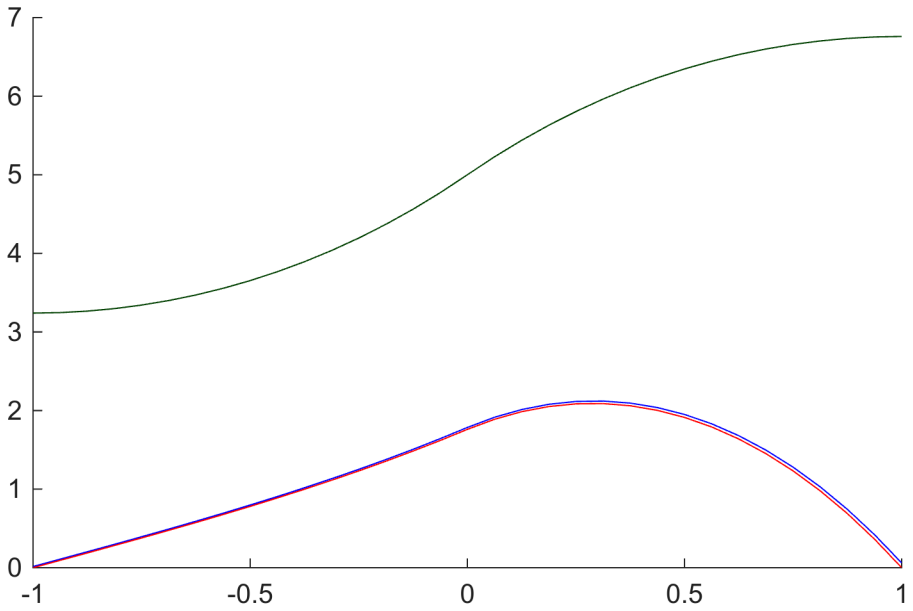
```
u = (K+M+C+1e2*(H'*H))\F;
interval.plot(u,'Color','blue');
```

To compare the penalty approach with a Lagrange approach, the Dirichlet boundary conditions is seen here as a (equality) constraint. we also implement is here. The linear system reads now

$$\begin{bmatrix} K+M+C & H' \\ H & 0 \end{bmatrix} \begin{bmatrix} u \\ \lambda \end{bmatrix} = \begin{bmatrix} F \\ R \end{bmatrix}.$$

The variable λ is a Langrange multiplier. It is here a vector of length two.

```
y = [K+M+C H'
     H    [0 0
           0 0]]\[F
                  R];
u = y(1:end-2);
interval.plot(u,'Color','red')
```



In the figure, you see both solution in red and blue, respectively.

Time Dependent Problems

Since the PDE classes are not shipped with OOPDELight, for the solution of time dependent problems we need to implement our owen solvers. We consider a simple problem in 1D.

$du_t - (cu_x)_x + b \cdot u_x + au = f$ in $\Omega \times (t_0, t_1)$ together with periodic boundary conditions $u(a) = u(b)$ and $u_t(a) + u_t(b) = 0$ and an initial value function $u(t_0) = u_0$.

Let $d = 1$, $c = 0.02$, $b = 3$, $a = 0$ and $f = 0$. This models a system with less diffusion, some convection that srives the solution from left to right, and without sources. We expect solution riding left to right, slowly disappearing.

For a better experince, we refine the interval once. Of couse, we have to reassemble the linear system. SInce $a = 0$,

```
interval.refineMesh;
[K,D,~] = FEM.assema(interval,0.02,1,0);
C = FEM.convection(interval,3);
```

For the periodic boundary conditions we call `FEM.periodic` method.

```
P = FEM.periodic(interval);
```

Let $u_0 = sin(\pi x)$. We implement an implicite Euler schema.

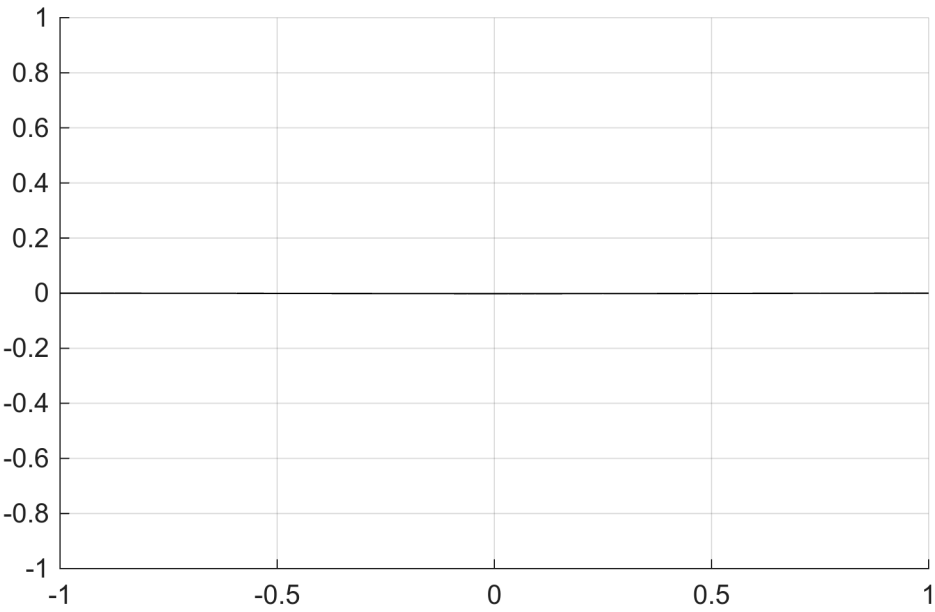
$(M + \delta t(K + C + P))u_{k+1} = Mu_k,$

Note that we use here a never ending loop, so we break it if the system is near the stable state $\|u\| \approx 0$.

```
u_old = sin(interval.x*pi)';
```

```
dt = 0.01;
while true
    figure(2)
    cla
    interval.plot(u_old,'Color','black');
    axis([-1 1 -1 1]); grid on
    drawnow
    u_new = (D+dt*(K+C+1e2*P))\D*u_old;
    u_old = u_new;

    if norm(u_old) < 0.01
        fprintf('Loop breaks with ||u|| < 0.01\n');
        break
    end
end
```



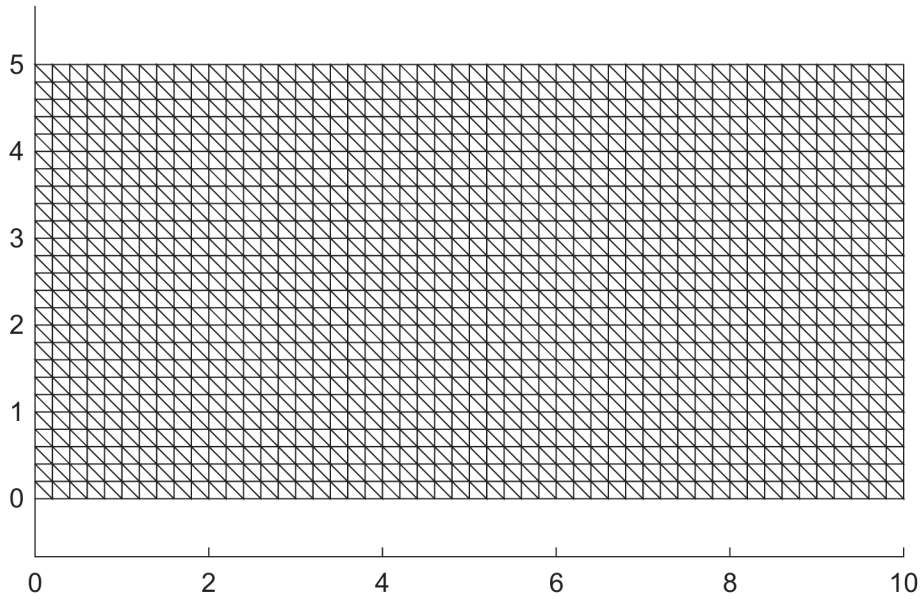

```
Loop breaks with ||u|| < 0.01
```

Simple 2D Problems

THe steps here are the same as in the 1D case, only some names of constructor methods change. Instead of Interval we have to handle Rectangle objects.

Disctetize the rectangle $\Omega_R = [0, 10] \times [0, 5]$ with meshwidth $h = 0.2$ by a triangles mesh. Plot the grid.

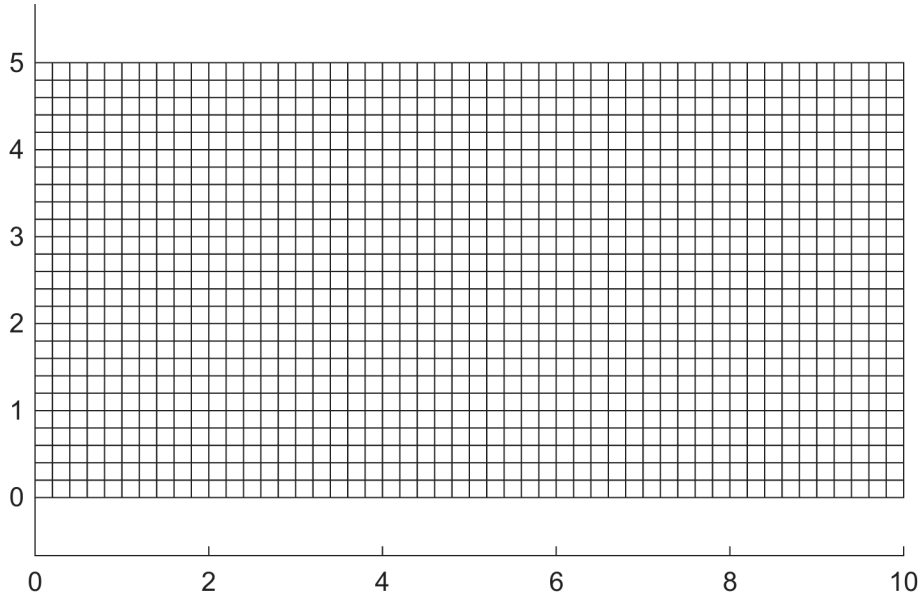
```
domain = Rectangle(0,10,0,5,0.2);
domain.plot;
```



As you can see, it is a regular cross mesh.

If you with to use a rectangualr mesh, use the class RectangleR.

```
domain2 = RectangleR(0,10,0,5,0.2);
domain2.plot;
```



You see: Same meshsize, same number of nodes but different number of elements.

To get information on the grids, you can call the display methods:

```
domain.display;
```

```
domain =
    Grid object of class Rectangle with
    1326 mesh points
```

```
    150 edges/lateral faces
    2500 elements.
```

```
domain2.display;
```

```
domain2 =
    Grid object of class RectangleR with
    1326 mesh points
    150 edges/lateral faces
    1250 elements.
```

To chose a FEM, we have to call a constructor method. We have in OOPDELight only (bi) linear Lagrange elements available. We need Lagrange12D for the triangle geometries and Bilinear2D for the rectangular based geometries. We assign them to two variables, FEM and FEMR.

```
FEM = Lagrange12D();
FEMR = Bilinear2D();
```

Now we define a PDE problem. OOPDE can solve linear PDEs of 2nd order pit of the box.

Consider

$$-\nabla \cdot (C \nabla u) + b \cdot \nabla u + au = f$$

together with homogenious Neumann boundary conditions on the rectangle Ω_R. OOPDELIght uses the finite elements method and discretizes a weak form of the PDE. By using Green formula, one can include Neumann boundayry conditions directly in the weak formulation, and in particular, homogenious Neumann boundary conditions disappear in the weak formulation. They are natural in FEM.

Using this we have to assemble only the matrices related with the domain. Assume $C = 3$ and let $b = [0 \quad 0]^{\mathsf{T}}$, $a = 1$ and

$$f = \begin{cases} 1 \text{ if } x > 5 \\ 0 \text{ otherwise} \end{cases}.$$

We need stiffness matrix K, mass matrix M, and source vector F. Convection matrix C is all zero.

We first use the triangle meshes and then the rectangle mesh.

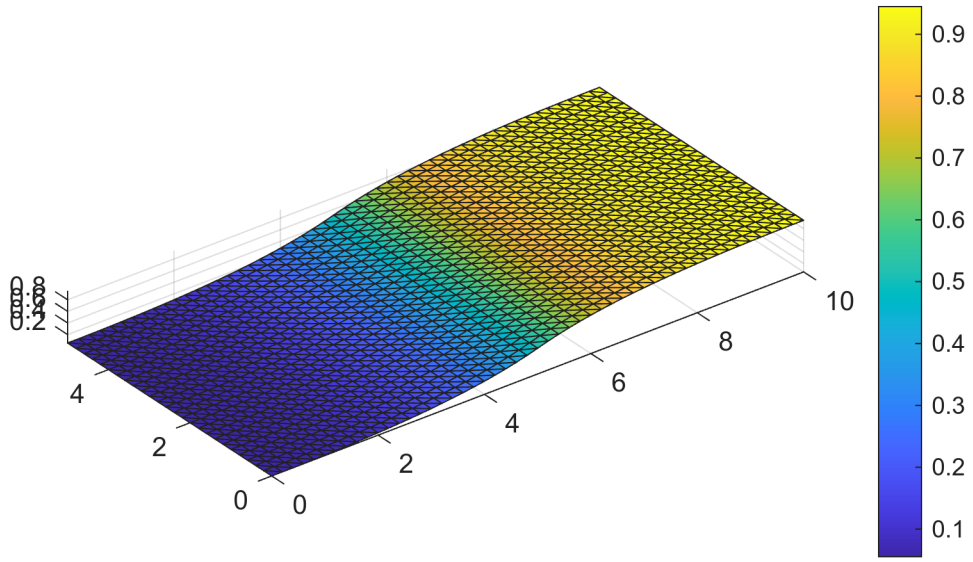
```
[K,M,F] = FEM.assema(domain,3,1,'x>5');
```

Now we need to solve the linear system $(K + M)u = F$.

```
u = (K+M)\F;
```

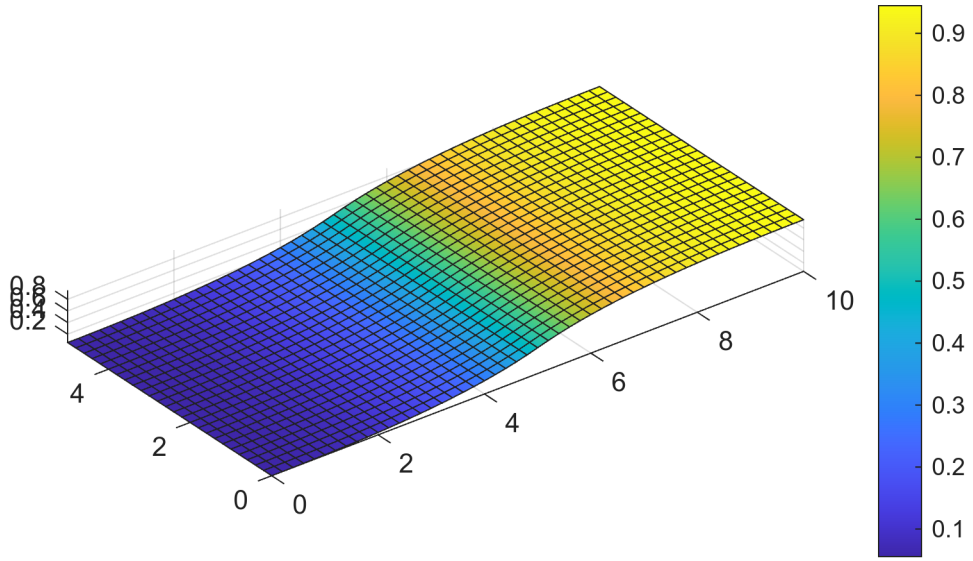
In OOPDELight, all plotting is in the sense of "plot over the domain", so it is a method of the geometry classes.

```
domain.plot(u);
```



The same now using the rectangular grid. Replace FEM by FEMR and domain by domain2.

```
[K,M,F] = FEMR.assema(domain2,3,1,'x>5');
u = (K+M)\F;
domain2.plot(u);
```



In Part 2

Handle boudary conditions in 2D. Time dependent problems in two space dimensions.