Chair for Computational Analysis of Technical Systems | RWTH AACHEN UNIVERSITY

Isogeometric Analysis

Wintersemester 15/16

Homework 2
Heat Equation

As the easiest time-dependent modification of the Poisson problem in Exercise 7, we will consider in this homework the heat equation

$$\partial_t u - \lambda\,\Delta u = 0\,,$$

where λ is the thermal diffusivity. Since this is a key example for a parabolic PDE, and we have so far only dealt with elliptic problems, we need to take special care of the time derivative. One way to do this is the so-called Method of Rothe: The first step will consist of using Implicit Euler to discretize in time and then in the second step we will apply our IGA approach to discretize in space. More specifically, the time discretization leads to

$$\frac{u^{n+1}-u^n}{\Delta t} - \lambda\,\Delta u^{n+1} = 0\,,$$

which after shuffling around a few parameters can be written as

$$u^{n+1} - \Delta t\lambda\,\Delta u^{n+1} = u^n\,.$$

Insofar, the problem has been reduced to an elliptic PDE that can now be handled using the isogeometric method. By introducing the mass matrix $\boldsymbol{M}$ and the stiffness matrix $\boldsymbol{A}$ our fully discretized problem then becomes

$$(\boldsymbol{M} + \Delta t\lambda\,\boldsymbol{A})\,\boldsymbol{u}_h^{n+1} = \boldsymbol{M}\boldsymbol{u}_h^n\,. \tag{1}$$

A key point, which was already mentioned in the exercises, is that the matrices $\boldsymbol{M}, \boldsymbol{A}$ are assembled regardless of what type the boundary conditions are. Let Ω_D denote the Dirichlet part of the boundary, then we want to fix those entries in $\boldsymbol{u}_h^n$ that correspond to a basis function with support on the boundary Ω_D and as such also need to cancel out some of the equations in (1). But just canceling out the equations would make the left hand side of (1) a non-square matrix, where only for a subset of the entries of $\boldsymbol{u}_h^{n+1}$ is solved for. Hence one needs to carry over the Dirichlet values to the right hand side and by this homogenize the problem. For this purpose we will introduce a lifting function $\boldsymbol{u}_{h,D}^{n+1}$ that holds the Dirichlet values and is zero in the rest of the domain $\Omega\backslash\Omega_D$. Splitting $\boldsymbol{u}_h^{n+1}$ into $\boldsymbol{u}_{h,D}^{n+1}$ and a term $\boldsymbol{u}_{h,0}^{n+1}$ that is zero in all entries corresponding to Dirichlet nodes, we can write

$$\left[\boldsymbol{C}\boldsymbol{u}_{h,0}^{n+1}\right]_i = \left[\boldsymbol{M}\boldsymbol{u}_h^n - \boldsymbol{C}\boldsymbol{u}_{h,D}^{n+1}\right]_i \quad \forall i \in I\,, \tag{2}$$

where $\boldsymbol{C} = \boldsymbol{M} + \Delta t \lambda\, \boldsymbol{A}$ and I denotes the index set that corresponds to the degrees of freedom that are not subject to the Dirichlet boundary condition. Thus, we have now transformed our initial inhomogeneous problem to a problem with homogeneous Dirichlet boundary conditions. The letter also implies that we can use the methods of Exercise 7 to solve for $\boldsymbol{u}_{h,0}^{n+1}$.

Parameter		Value
Thermal diffusivity	λ	0.01
Time stepsize	Δt	1
Time period	T_{end}	15

Table 1: Parameters used for the simulation. For the sake of simplicity we assume that all quantities have been rescaled to dimensionless units.

To make things concrete we will once more consider Ω as the domain that is generated by `generate_testnurb`. Furthermore, we will choose the boundaries that correspond to $\eta = 0$ or $\eta = 1$ as Dirichlet boundary and the boundaries that correspond to $\xi = 0$ or $\xi = 1$ as Neumann boundaries with zero heat flux. More specifically, u shall be 100 at the boundary with $\eta = 1$ and shall be -10 at the boundary given by $\eta = 0$. The initial condition will be $u^0 = 30$. All other parameters are listed in Table 1.

From the considerations above it should be clear that the final algorithm has the following design:

```
function heat_eq_example()

% we want to simulate on 25 elements
refinement_level=5;
ref_nurb = nurb_knot_refinement(generate_testnurb(), refinement_level -1);

% set up mass and stiffness matrix
YOUR_CODE

% construct the index set I that corresponds to the inner DOFs, cf. Exercise 7
YOUR_CODE

% set up initial conditions
solution = YOUR_CODE;

% store matrix C in the variable mat
mat = YOUR_CODE;

% time loop
for YOUR_CODE
        % store old solution
        old_solution = solution;

        % set up Dirichlet boundary conditions
        YOUR_CODE
        % copy these terms to the corresponding part of the solution vector
        YOUR_CODE

        % calculate the right hand side
        rhs = YOUR_CODE;

        % solve the linear system only for the inner degrees and store
        % the result in the solution vector
        YOUR_CODE

        % reshape the solution vector into a coefficients array, multiply
        % with the weights, and finally plot the result
        YOUR_CODE
end
end
```

The final result of the computation can be seen in Figure 1.

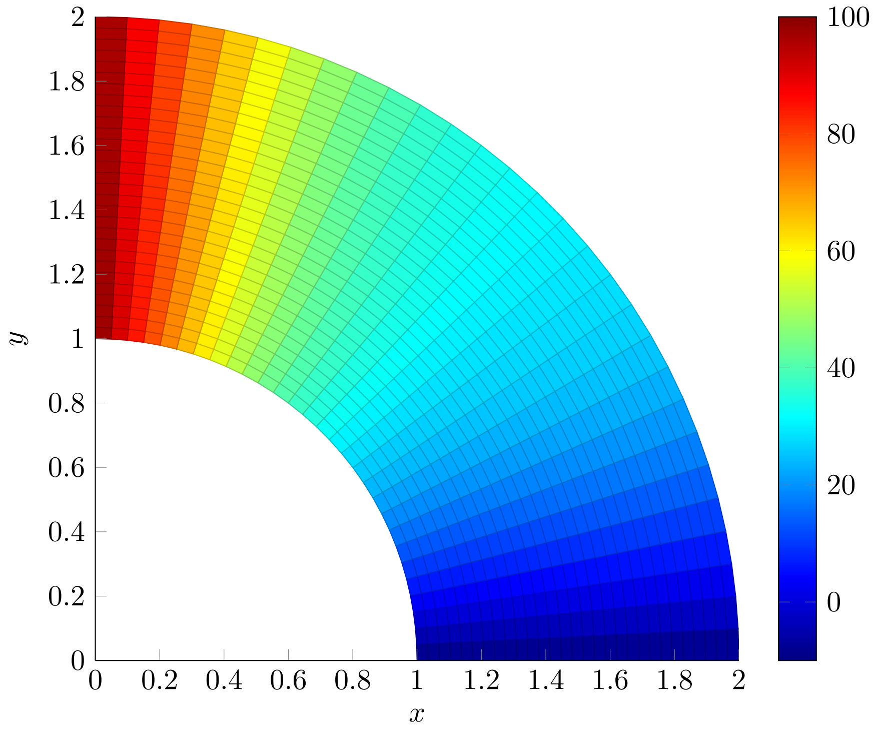

Figure 1: Surface plot of the solution.

Your task is now to fill out the missing steps in the provided source code and perform the calculation using the given parameters. Hand in a final report describing your steps as well as the result. This final report is required to be

- narrative in its nature, and collected in a pdf-file,

as well as include

- a picture like Figure 1,
- the current code.

In addition, please submit

- your code for the homework, as well as the Exercises 1 - 7, in a separate zip/tar-file.

Furthermore, this homework assignment is to be solved **individually** (not as a team) and handed in **online** by using the L^2P system. Solutions are accepted until **January 24th 2016, 8 PM**.

Contact:
Florian Zwicke, M.Sc. · zwicke@cats.rwth-aachen.de