Isogeometric Analysis

Wintersemster 15/16

Report on HomeWork 2

Author:
Kumar Saurabh

1 Problem Description

The objective is to solve the time dependent Poisson problem, given by Eqn (1)

$$\partial_t u - \lambda \Delta u = 0 \tag{1}$$

where λ is the thermal diffusivity.

2 Initial and Boundary Condition

The domain Ω is generated by using the function `generate_testnurb`. The boundary conditions are listed below:

Boundary	Boundary Type	Value
$\eta = 0$	Dirichlet	-10
$\eta = 1$	Dirichlet	100
$\xi = 0$	Neuman	0
$\xi = 1$	Neuman	0

The initial condition is given as $u(0) = 30$.

3 Parameters

In order to simulate the problem, the following parameters are used:

Parameter		Value
Thermal diffusivity	λ	0.01
Time step	$\triangle t$	1
Final time	T_{end}	15

4 Steps Involved

To solve the time dependent Poison problem (Eqn (1)), Method of Rothe is used. The first step will consist of Implicit Euler to discretize in time

and then IGA approach to discretize in space. The final discretized equation takes the form:

$$(\mathbf{M} + \triangle t\lambda\mathbf{A})u_h^{n+1} = \mathbf{M}u_h^n \tag{2}$$

where $\mathbf{M}$ and $\mathbf{A}$ represent the mass and stiffness matrix respectively.

- In the first step, the nurb is refined with the help of code `nurb_knot_refinement.m`. The `refinement_level` is set to 5 in order to set up the simulation with 25 elements.
- The mass matrix and stiffness matrix is set up using the file `assemble_matrix_2d.m`. The file named `blk_uv` and `blk_dudv` from previous exercises is used to set up the mass matrix and the stiffness matrix respectively.
- To incorporate the Dirichlet condition, the unknown u_h^{n+1} is split into $u_{h,0}^{n+1}$ and $u_{h,D}^{n+1}$, where $u_{h,0}^{n+1}$ is zero in all the entries corresponding to Dirichlet node and $u_{h,D}^{n+1}$ holds the Dirichlet values and is zero in the rest of the domain. Finally, the equation (2) can be written as:

  $$[\mathbf{C}u_{h,0}^{n+1}] = [\mathbf{M}u_h^n - \mathbf{C}u_{h,D}^{n+1}] \quad \forall i \in I \tag{3}$$

  where $\mathbf{C} = (\mathbf{M}+\triangle t\lambda\mathbf{A})$ and I denotes the index set that corresponds to the degrees of freedom that are not subject to the Dirichlet boundary condition. The linear system is solved only for the inner degrees of freedom.

5 Code

The Matlab code `heat_eq_example.m` is used to compute the steps mentioned in Section (4). It makes use of the code `assemble_matrix_2d.m` from Exercise 4-6.

Code 1: heat_eq_example.m

```
1 function heat_eq_example()
2
3 addpath('Exercise1/');
4 addpath('Exercise2/');
5 addpath('Exercise3/');
```

```
addpath('Exercise4-6/');
addpath('Exercise7/');

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

delta_t = 1;
lambda = 0.01;
%Mass and Stiffness Matrix
[mat, ~] = assemble_matrix_2d(false, ndof, ref_nurb, ...
    @blk_dudv, @rhs_testfun);
[mat1, ~] = assemble_matrix_2d(false, ndof, ref_nurb, ...
    @blk_uv, @rhs_testfun);


mat = mat*(delta_t*lambda);
mat = mat + mat1;


n = ref_nurb.number(1);
m = ref_nurb.number(2);

all_dofs = [1 : n*m];

% equation numbers that correspond to weighting functions ...
    with support on
% the boundary with eta=0
boundary1 = 1:n;
% boundary with eta=1
boundary2 = (m-1)*n+1:1:n*m;
boundary = union(boundary1,boundary2);
% boundary = union(boundary,1:n:m*(n-1));
% % boundary with eta=1
% boundary = union(boundary,n:n:n*m);


inner_dofs = setdiff(all_dofs,boundary);

% initialize solution vector to zero, including the ...
    boundary terms
solution = ones(n*m,1)*30;

```

```
46
47 solution(boundary) = 0;
48
49 uhD = zeros(n*m,1);
50 uh0 = solution;
51 uhD(boundary1) = -10;
52 uhD(boundary2) = 100;
53 time = 0;
54 while (time < 15)
55
56     old_solution = solution;
57
58     old_solution(boundary) = uhD(boundary);
59
60     rhs = mat1*old_solution - mat*uhD;
61
62     solution(inner_dofs) = mat(inner_dofs,inner_dofs) \ ...
           rhs(inner_dofs);
63
64
65     time = time + delta_t;
66
67 end
68 solution(boundary) = uhD(boundary);
69
70 coeffs = reshape(solution,ndof,n,m);
71 % remember that the nurb_eval routine needs the ...
      coefficients premultiplied
72 % with the weights
73 for i=1:ndof
74     coeffs(i,:,:) = coeffs(i,:,:).*ref_nurb.coeffs(4,:,:);
75 end
76 % show a surface plot of the solution
77 draw_nurb_surf(ref_nurb, [30 30], ...
78     @(xi,eta) nurb_eval(ref_nurb, coeffs, ndof, [xi; eta]));
79
80
81 end
```

6 Visualization

Figure (1) shows the results of simulation of equation (1) along with the parameters and initial and boundary conditions described in the Section (3)

and (2) respectively.

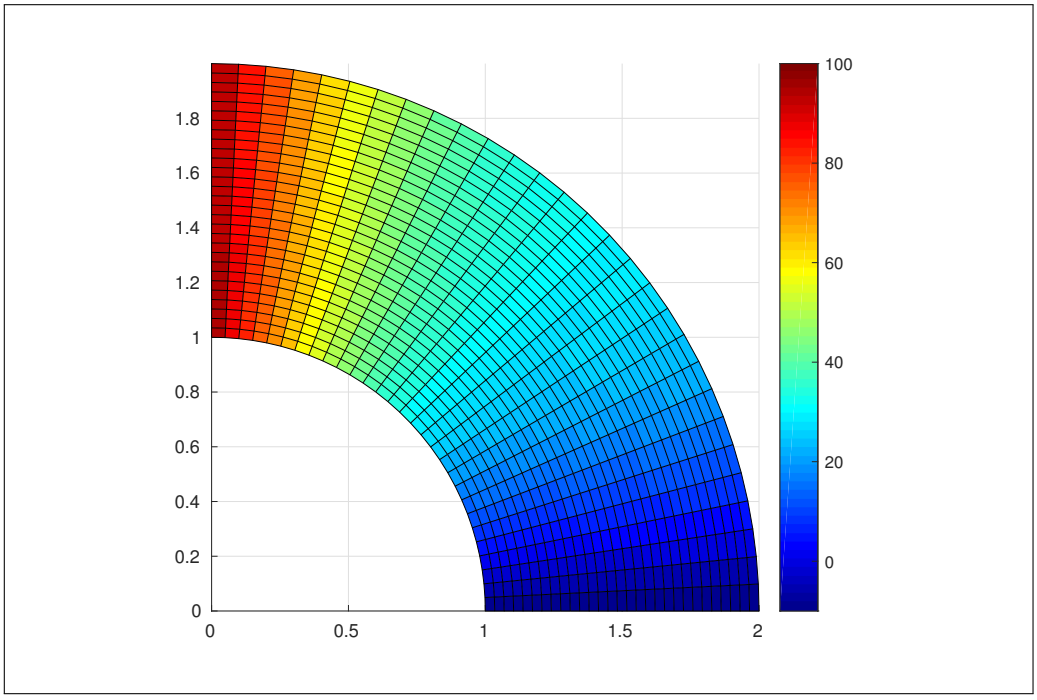

Figure 1: Surface plot of the solution

7 Result

The Equation (1) is basically a time-dependent diffusion equation. From the Figure (1) it can be seen that the temperature at the both the end of Dirichlet Condition remains fixed whereas the diffusion of the heat from higher temperature to a lower temperature takes place in interior of the domain. The region close to the boundary with higher temperature is higher as compared with those closer to the other boundary. The value of zero Neuman boundary condition ensures that there is no heat flux across the boundary.