Master's Degree Numerical Methods in Engineering

Coupled Problems

Computer Homework

Author:
Mariano Tomás Fernandez

Professor:
Javier Principe
Joan Baiges

June 17^{th}, 2020
Academic Year 2019-2020

Contents

Computer homework

1. *Solve a single heat transfer problem. The domain is [0,1]. Fix u = 0 in both boundaries.*
   (a) Study the effect of changing the value to the thermal diffusion coefficient kappa.
   (b) Study the effect of changing the source term value.
   (c) Study the effect of changing the number of elements, evaluate the convergence rate of the error in the maximum heat value in the domain.
2. *Solve two independent heat transfer problems with kappa = 1, source = 1. The first problem subdomain is [0, 0.25]. The second problem subdomain is [0.25,1]. Fix u in x=0 and x=1, leave it free in the interface between subdomains. Comment on the results.*
3. *Solve the previous problem in a Monolithic way.*
   (a) *Study* `HP_SolveMonolithic.m` *and relate it to what was explained in theory. Comment on the results.*
   (b) *Modify the kappa parameter of one of the subdomains. Comment on the results.*
4. *Solve the previous problem (kappa = 1 in both subdomains) in an iterative manner (Dirichlet Neumann). Apply Neumann boundary conditions at the interface in the first (left) subdomain, and Dirichlet boundary conditions at the interface in the second subdomain.*
   (a) *Evaluate the convergence of the iterative scheme (in terms of u at the interface).*
   (b) *Increase the value for kappa at subdomain 1 (x100). Comment on the convergence rate.*
   (c) *Diminish the value for kappa at subdomain 1(/100). Comment on the convergence rate.*
   (d) *Motivate the previous results in terms of the stability of the coupling scheme.*
5. *Implement a relaxation scheme*
   (a) *Relaxation scheme in terms of a fixed relaxation parameter w.*
   (b) *Aitken relaxation scheme.*

1 First exercise

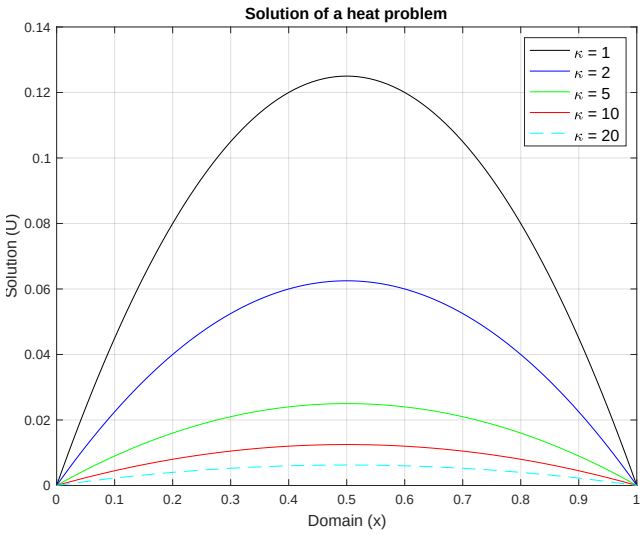
A single heat transfer problem needs to be solved for a domain $\Omega = [0, 1]$ with Dirichlet boundary conditions $u(0) = u(1) = 0$.

1.1 First task

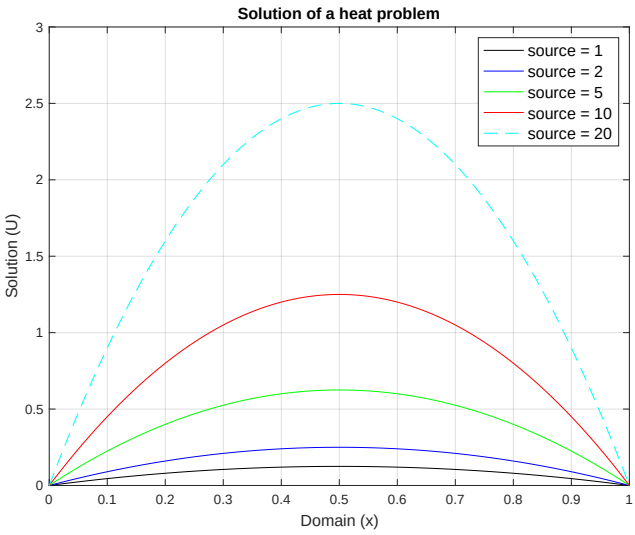
The effect of changing the thermal diffusion coefficient is studied by solving the problem for different κ values from 1 to 20, and their results are shown in Figure 1a. As seen in the image, for $\kappa = 1$ the solution has a maximum $U = 0.125$ whereas for the $\kappa = 20$ the solution has a maximum of $U = 0.0063$. Therefore, for bigger diffusion coefficient and constant sources the solution is smaller.

1.2 Second task

The effect of changing the source of heat in the problem is studied by solving the problem for different s values from 1 to 20, and their results are shown in Figure 1b. As seen in the image, for $s = 1$ the solution has a maximum $U = 0.125$ whereas for the $\kappa = 20$ the solution has a maximum of $U = 2.5$. Logically, for bigger source of heat and the heat equation solution has bigger values.



(a) Varying κ

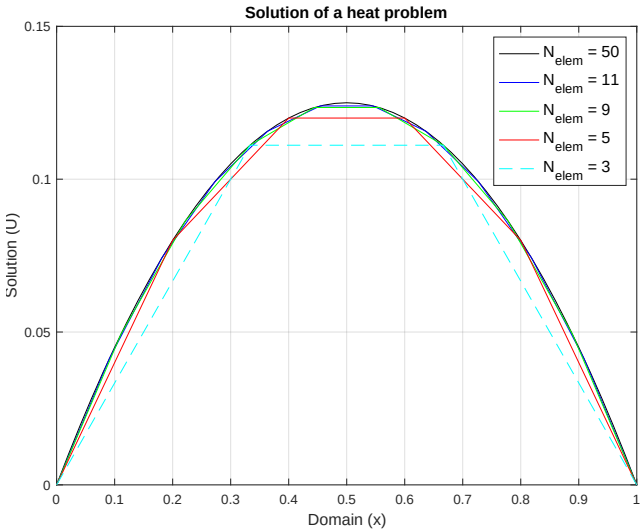


(b) Varying source term

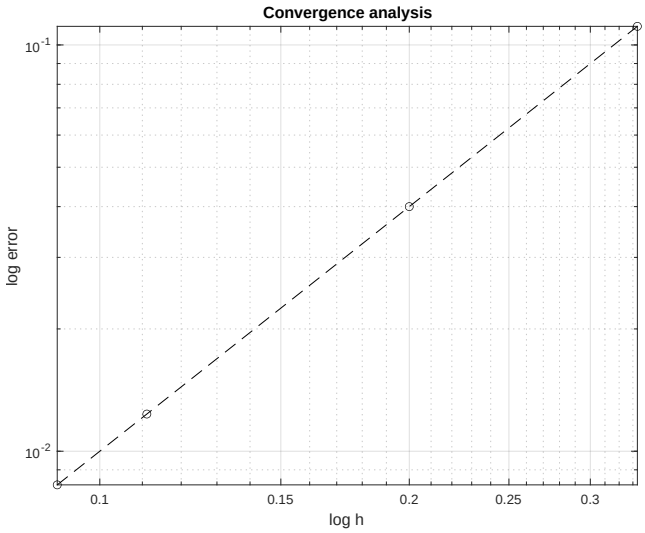
Figure 1: Solution of a heat transfer problem for varying κ and s from 1 to 20.

1.3 Third task

The idea is now to obtain the relationship between the maximum value of the solution in the domain of interest with the size of the mesh. To do so, first a dense mesh will be run and its maximum will be taken as the '*exact*' value to compare with the coarser meshes studied. As can be seen in Figure 2b the convergence is linear for increasing amount of elements in the domain. The values of the solution start getting really close in shape from $N_{elem} \geq 9$, even though the error is still $c \cdot 10^{-2}$.



(a) Solution for various meshes



(b) Mesh convergence (log-log error)

Figure 2: Analysis of mesh refinement for heat problem solution.

2 Second exercise

The domain is split into two subdomains right and left from $x = 0.25$. As seen in Figure 3, when no transmission conditions are imposed, the problem is not continuous as there are two different values of the solution in the same mesh point ($x = 0.25$).

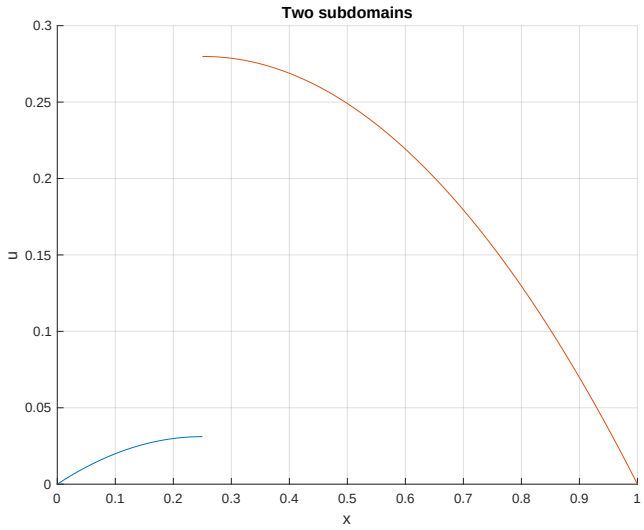


Figure 3: Solution of a heat transfer problem for varying κ from 1 to 20.

3 Third exercise

3.1 First task

The `HP_SolveMonolithic.m` code generates a matrix of *Num_Nodes1+Num_Nodes2-1* to perform the couple of the two problems and stores the matrix the two subdomains. First the matrix of subdomain one is stored in *1:Num_Nodes1*, then the matrix of subdomain 2 is stored from *Num_Nodes1:(Num_Nodes1+Num_Nodes2-1)*, same scheme is performed with independent vectors F. The boundary of the two subdomains is located at the

end node of *Num_Nodes1*, that is why one node is substracted form the size of the matrix, and in that node the components of matrix from subdomain 1 and subdomain 2 are summed.

Now, the results presented in Figure 4 reflects the correct application of the monolithic scheme as expected, same flux and solution for the boundary between subdomains.

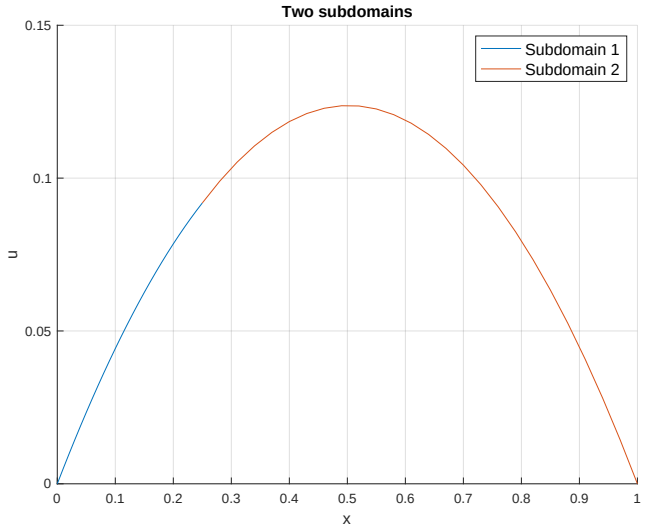


Figure 4: Monolithic solution of heat transfer equation for two subdomains.

3.2 Second task

When modifying one of the κ parameters in a subdomain, the monolithic scheme cannot handle the continuity of fluxes but does it for the solution, as virtually the solution is calculated only once and then translated to each of the subdomains. Saying this, when one subdomain has bigger κ than the other, its results are lower (as seen in Figure 1a) and the enforcement of equal solution without imposition of fluxes shows results like those seen in Figure 5. In this case $\kappa_2 = 20$ and $\kappa_1 = 1$.

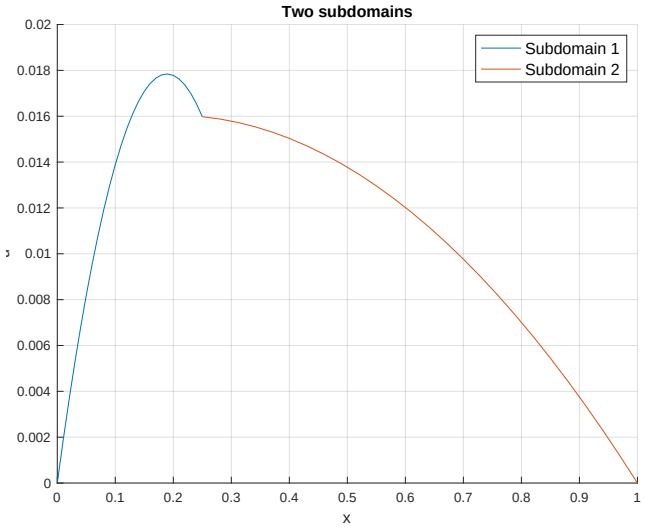


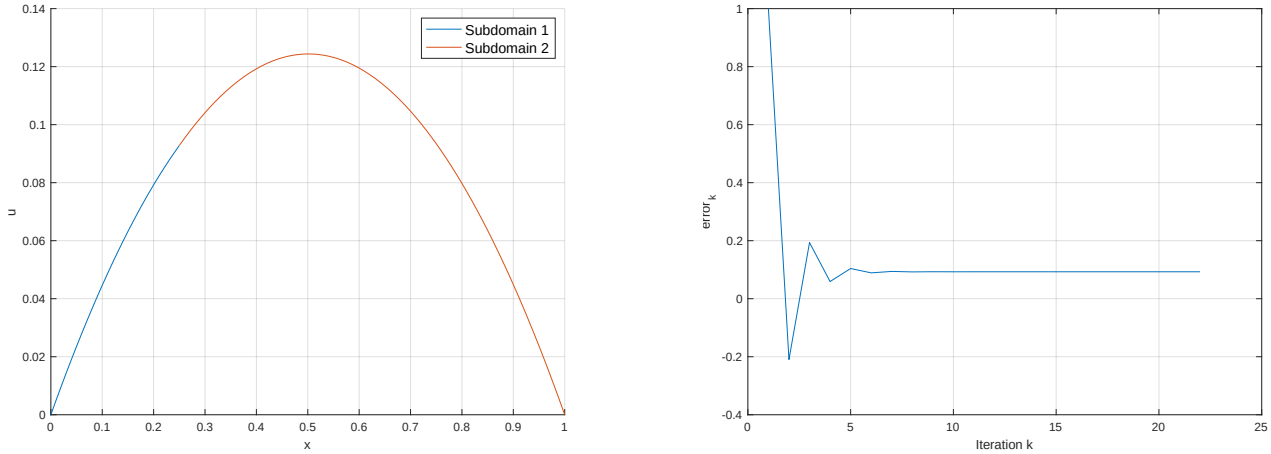
Figure 5: Monolithic solution of heat transfer equation for two subdomains with different κ parameters.

4 Fourth exercise

The idea is now to implement an iterative scheme for with Neumann-Dirichlet conditions in the two subdomains. To do this, first on Ω_1 Neumann conditions are applied using solutions of u_2 from the previous step and, then Dirichlet conditions are imposed in Ω_2 using results from u_1. As seen, this scheme uses both solutions and relates them iteratively to merge into a smooth solution where both transmission conditions are comply. In Figure 6a the results are shown and this description is checked.

4.1 First task

To check the convergence of the code a series of different values of u in the interface are registered while increasing the iteration step. The results show a linear convergence after the first iteration (See Figure 6b) but the results do not seem to be completely right.



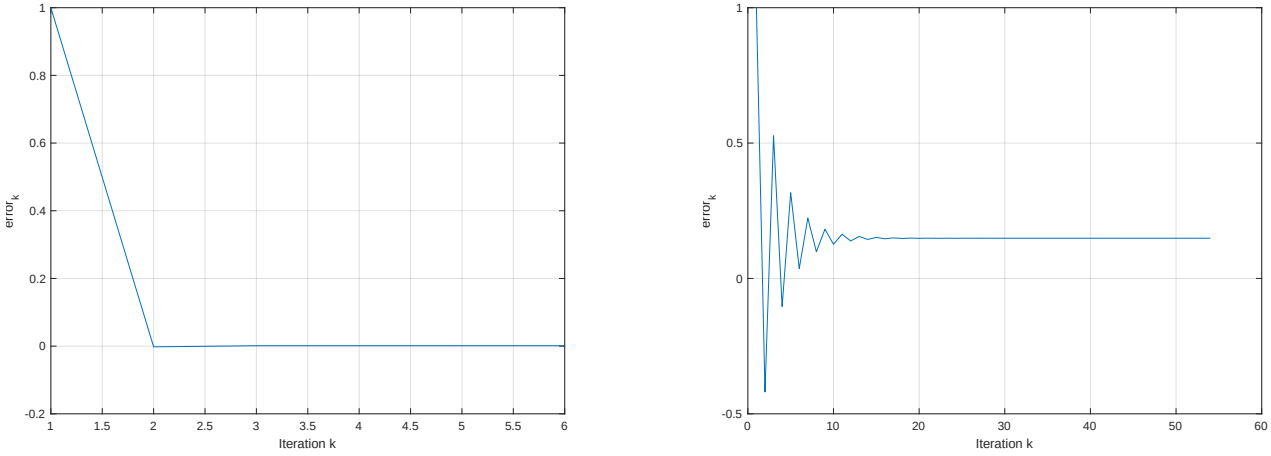
(a) Obtained solution

(b) Iteration convergence (log k -log error)

Figure 6: Dirichlet-Neumannn algorithm.

4.2 Second and third tasks

The idea is now to analyse the convergence for a difference κ value between each of the subdomains. As mentioned before the convergence of this implementation is not completely right, and with this task it is confirmed how the convergence is not granted.



(a) Increasing $\kappa_1 = 100\ \kappa_2$.

(b) Decreasing $\kappa_1 = 1/100\ \kappa_2$.

Figure 7: Dirichlet-Neumannn algorithm.

Comment

I had a problem during the implementation of the *Dirichlet-Neumann* scheme (related to the way I was plotting the resuts). This is why I could not finish the assignment on time, please accept my apologies.

A Implementation codes

A.1 General code exercise 1 to 3

```
clear all;
clc;
% %Domain
% Data.inix = 0;
% Data.endx = 1.0;
% Data.nelem = 100;
% %Physical
% Data.kappa = 1;
% Data.source = 1;
% Boundary conditions
% Dirichlet
% Data.FixLeft = 1; %0, do not fix it, 1: fix it
% Data.LeftValue = 0;
% Data.FixRight =1;
% Data.RightValue = 0;
% Neumann
% Data.FixFluxesLeft = 0;
% Data.LeftFluxes = 0;
% Data.FixFluxesRight = 0;
% Data.RightFluxes = 0;

%First task
%To solve the first part of the assignment Domain = [0,1] u(0)=u(1)=0
% Data.nelem = 50;
% HeatProblem = HP_Initialize(Data);
% HeatProblem = HP_Build(HeatProblem);
% Solution1 = HP_Solve(HeatProblem);
% Data.nelem = 11;
% HeatProblem = HP_Initialize(Data);
% HeatProblem = HP_Build(HeatProblem);
% Solution2 = HP_Solve(HeatProblem);
% Data.nelem = 9;
% HeatProblem = HP_Initialize(Data);
% HeatProblem = HP_Build(HeatProblem);
% Solution3 = HP_Solve(HeatProblem);
% Data.nelem = 5;
% HeatProblem = HP_Initialize(Data);
% HeatProblem = HP_Build(HeatProblem);
% Solution4 = HP_Solve(HeatProblem);
% Data.nelem = 3;
% HeatProblem = HP_Initialize(Data);
% HeatProblem = HP_Build(HeatProblem);
% Solution5 = HP_Solve(HeatProblem);
%
% figure(1)
% plot(Solution1.Solution.coord,Solution1.Solution.U,'k')
% hold on
% ylim([0,0.15])
% plot(Solution2.Solution.coord,Solution2.Solution.U,'b')
% hold on
% plot(Solution3.Solution.coord,Solution3.Solution.U,'g')
% hold on
% plot(Solution4.Solution.coord,Solution4.Solution.U,'r')
% hold on
% plot(Solution5.Solution.coord,Solution5.Solution.U,'c--')
% hold on
% grid on
% legend('N_{elem} = 50','N_{elem} = 11','N_{elem} = 9','N_{elem} = 5','N_{elem} = 3','FontSize
    ',12)
```

```
% hold on
% title('Solution of a heat problem','FontSize',12)
% hold on
% xlabel('Domain (x)','FontSize',12)
% ylabel('Solution (U)','FontSize',12)
% %axes('FontSize',12)
% saveas(figure(1),'task3.pdf')
%
% %computing the error
% v_ref = max(Solution1.Solution.U);
% error = [abs(max(Solution2.Solution.U)-v_ref)/v_ref; abs(max(Solution3.Solution.U)-v_ref)/v_ref
    ; abs(max(Solution4.Solution.U)-v_ref)/v_ref; abs(max(Solution5.Solution.U)-v_ref)/v_ref];
% h = [1/11; 1/9; 1/5; 1/3];
%
% figure(2)
% loglog(h,error,'k--o')
% hold on
% grid on
% title('Convergence analysis','FontSize',12)
% xlabel('log h','FontSize',12)
% ylabel('log error','FontSize',12)
% saveas(figure(2),'task3_2.pdf')

%
% %Second task
% %To solve the second part of the assignment Domain1 = [0,0.25] u(0)=0
% %Domain2 = [0.25,1.0] u(1)=0 ; in the interface leave it "free"
% %First re-write the boundaries of Domain1
% %%Domain
% Data.inix = 0;
% Data.endx = 0.25;
% Data.FixRight =0;
% Data.FixLeft = 1; % fixed\
% Data.LeftValue = 0;
% Data.FixRight = 0; %not Fixed
% %Neumann
% Data.FixFluxesRight = 0;
% Data.RightFluxes = 0;
%
%
%
% HeatProblem = HP_Initialize(Data);
% HeatProblem = HP_Build(HeatProblem);
% Solution2 = HP_Solve(HeatProblem);
% HP_Plot(Solution2,2);
%
% Data2 = Data;
%
% Data2.inix = 0.25;
% Data2.endx = 1;
% Data2.FixLeft = 0; %Not fixed
% Data2.FixRight = 1; %Fixed
% Data2.RightValue = 0;
% %Neumann
% Data2.FixFluxesLeft = 0;
% Data2.LeftFluxes = 0;
%
% HeatProblem2 = HP_Initialize(Data2);
% HeatProblem2 = HP_Build(HeatProblem2);
% Solution2 = HP_Solve(HeatProblem2);
% HP_Plot(Solution2,2);

%Third task
%To solve monolithically the previous problem the only need is to execute
%the solve monolitically function
% %Physical
%Domain
Data.inix = 0;
Data.endx = 0.25;
Data.nelem = 25;
%Physical
```

```
Data.kappa = 1;
Data.source = 1;
%Boundary conditions
%Dirichlet
Data.FixLeft = 1; %0, do not fix it, 1: fix it
Data.LeftValue = 0;
Data.FixRight = 0;
Data.RightValue = 0;
%Neumann
Data.FixFluxesLeft = 0;
Data.LeftFluxes = 0;
Data.FixFluxesRight = 0;
Data.RightFluxes = 0;

Data3 = Data;

Data3.inix = 0.25;
Data3.endx = 1;
Data3.FixLeft = 0;
Data3.FixRight = 1;

Data3.kappa = 20;

%Solving a monolithic problem
%Data subdomain 1
HeatProblem = HP_Initialize(Data);
HeatProblem = HP_Build(HeatProblem);
%Data subdomain 2
HeatProblem3 = HP_Initialize(Data3);
HeatProblem3 = HP_Build(HeatProblem3);

%Solve Monolithic problem
[HeatProblem,HeatProblem3] = HP_SolveMonolithic(HeatProblem,HeatProblem3);
HP_Plot(HeatProblem,3)
hold on
HP_Plot(HeatProblem3,3)
legend('Subdomain 1','Subdomain 2','FontSize',12);
```

A.2 Dirichelt-Neumann conditions

```
% Application of Dirichlet-Neumann conditions
clear all;
clc;
% close all
clear variables

%Subdomain 1
Data.inix = 0;
Data.endx = 0.25;
Data.nelem = 25;
%Physical
Data.kappa = 0.5;
Data.source = 1;
%Boundary conditions
%Dirichlet
Data.FixLeft = 1; %0, do not fix it, 1: fix it
Data.LeftValue = 0;
Data.FixRight =0;       % No fix
Data.RightValue = 0.5;    % useless
%Neumann
Data.FixFluxesLeft = 0;
Data.LeftFluxes = 0;
Data.FixFluxesRight = 1; %Fix right fluxes
Data.RightFluxes = 0;

%Subdomain 2
Data2.inix = 0.25;
Data2.endx = 1;
Data2.nelem = 75;
%Physical
Data2.kappa = 1;
```

```
Data2.source = 1;
%Boundary conditions
%Dirichlet
Data2.FixLeft = 1; %0, do not fix it, 1: fix it
Data2.LeftValue = 1;
Data2.FixRight = 1;
Data2.RightValue = 0;
%Neumann
Data2.FixFluxesLeft = 0;
Data2.LeftFluxes = 0;
Data2.FixFluxesRight = 0;
Data2.RightFluxes = 0;

% Initial calculation
HeatProblem1 = HP_Initialize(Data);
HeatProblem2 = HP_Initialize(Data2);
HeatProblem1.Solution.uRight = 1; %initial cond setting
HeatProblem2.Solution.FluxesLeft = 0;
u = 1;

Data2.LeftValue = HeatProblem1.Solution.uRight;

k = 1; % initial iteration counter k
eps = 1e-6; % allowed error
Guess = 1;
niter = 100; % max iteration number
error = 10;
error5 = zeros(niter,1);
Sol_n1 = 10; % initial guess value

while abs(HeatProblem1.Solution.uRight-Sol_n1)>1e-9
    %Subdomain 2
    %Impose Dir BC
    Data2.LeftValue = HeatProblem1.Solution.uRight;
    Sol_n1  = HeatProblem1.Solution.uRight;
    %Solve
    HeatProblem2 = HP_Initialize(Data2);
    HeatProblem2 = HP_Build(HeatProblem2);
    HeatProblem2 = HP_Solve(HeatProblem2); %solve in subdomain 2

    %Subdomain 1
    %Impose Neu BC
    Data.RightFluxes = -HeatProblem2.Solution.FluxesLeft;
    %Solve
    HeatProblem1 = HP_Initialize(Data);
    HeatProblem1 = HP_Build(HeatProblem1);
    HeatProblem1 = HP_Solve(HeatProblem1); %solve in subdomain 1

    Guess = [Guess, HeatProblem1.Solution.uRight];
end

HP_Plot(HeatProblem1,1)
grid on
% ylim([0,0.1])
hold on
HP_Plot(HeatProblem2,1)
legend('Subdomain 1','Subdomain 2','FontSize',12);

k = [1:1:size(Guess,2)];
figure(2)
plot(k,Guess);
grid on
xlabel('Iteration k')
ylabel('error_k')
saveas(figure(2),'convergence_k2.pdf')

%version 2

%     [boundval1,Solution1,Solution2]=Iterative_DN(boundval0,Data,Data2);
%     error = abs(boundval1-boundval0)
%     if(error<eps)
```

```
%          break
%      else
%          k=k+1;
%          boundval0 = boundval0+error;
%      end
%
% function [boundval1,Solution1,Solution2]=Iterative_DN(boundval,Data,Data2)
%      Data2.LeftValue = boundval; %Impose u_2(1) equal u_1(end)
%
%      HeatProblem2=HP_Initialize(Data2);
%      HeatProblem2=HP_Build(HeatProblem2);
%      Solution2=HP_Solve(HeatProblem2);
%
%      Data.FluxesLeft = -Solution2.Solution.FluxesRight; %Impose Neumann BC
%
%      HeatProblem1=HP_Initialize(Data);
%      HeatProblem1=HP_Build(HeatProblem1);
%      Solution1=HP_Solve(HeatProblem1);
%
%      boundval1 = Solution1.Solution.uRight; %Recover the n+1 result
%                                             %of interest
% end

%Version1

%      Data2.LeftValue = boundval %imposing Dirichlet on subd 2
%
%      % Solve subdomain 2
%      HeatProblem2 = HP_Initialize(Data2);
%      HeatProblem2 = HP_Build(HeatProblem2);
%      Solution2 = HP_Solve(HeatProblem2);
%
%      % Enforce onto problem 1
%      Data.RightFluxes = - Solution2.Solution.FluxesLeft %imposing Neumann on subd 1
%
%      % Solve subdomain 1
%      HeatProblem1 = HP_Initialize(Data);
%      HeatProblem1 = HP_Build(HeatProblem1);
%      Solution1 = HP_Solve(HeatProblem1);
%
%      boundval1 = Solution1.Solution.uRight; %update Dirichlet result n+1
%
%      if (abs(boundval1-boundval)>eps)
%          if abs(Solution1.Solution.FluxesRight+Solution2.Solution.FluxesLeft)>eps
%              k = k+1;
%              boundval = boundval1
%              fluxval = Solution1.Solution.FluxesRight
%          else
%              k = k+1;
%              boundval = boundval1
%              fluxval = Solution1.Solution.FluxesRight
%          end
%      elseif abs(Solution1.Solution.FluxesRight+Solution2.Solution.FluxesLeft)>eps
%          k = k+1;
%          boundval = boundval1
%          fluxval = Solution1.Solution.FluxesRight
%      else
%          break
%      end
```