Master's Degree Numerical Methods in Engineering

Coupled Problems

Computer Assignment: Heat Transfer

Authors:
Aren Khaloian

Professor:
Javier Principe
Joan Baiges

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

1 Problem 1

Solve a single heat transfer problem. The domain is [0,1]. Fix u = 0 in both boundaries.

- Study the effect of changing the value to the thermal diffusion coefficient kappa.
- Study the effect of changing the source term value.
- Study the effect of changing the number of elements, evaluate the convergence rate of the error in the maximum heat value in the domain.

In order to solve the problem the given codes are initiated. In order to plot the difference of the needed parameter an extra loop is added on the general code and the needed parameter initiated as a matrix of values. The obtained values are saved in an additional matrix and plotted after the loop is done.

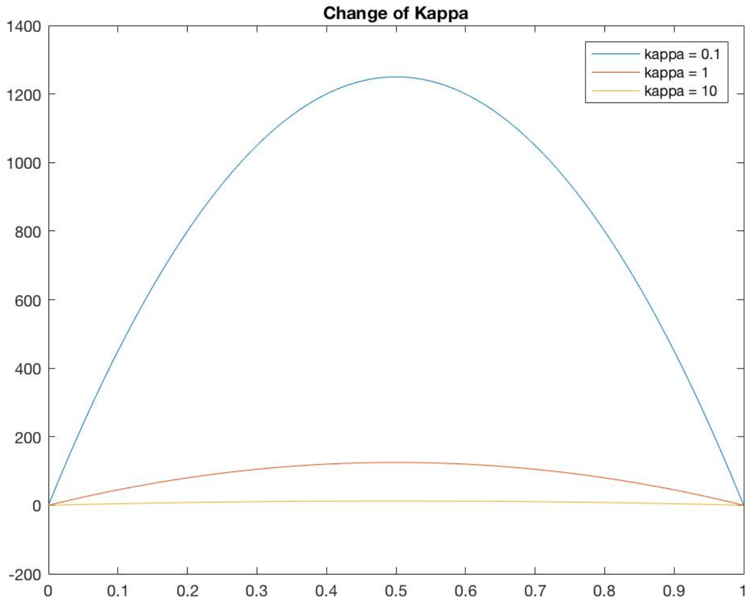

Figure 1: Change of kappa from 0.1 to 10

This case is with a source of 1000 as seen if the kappa is low the effect of the source is much higher and much higher values are obtained for U in comparison with higher values for kappa.

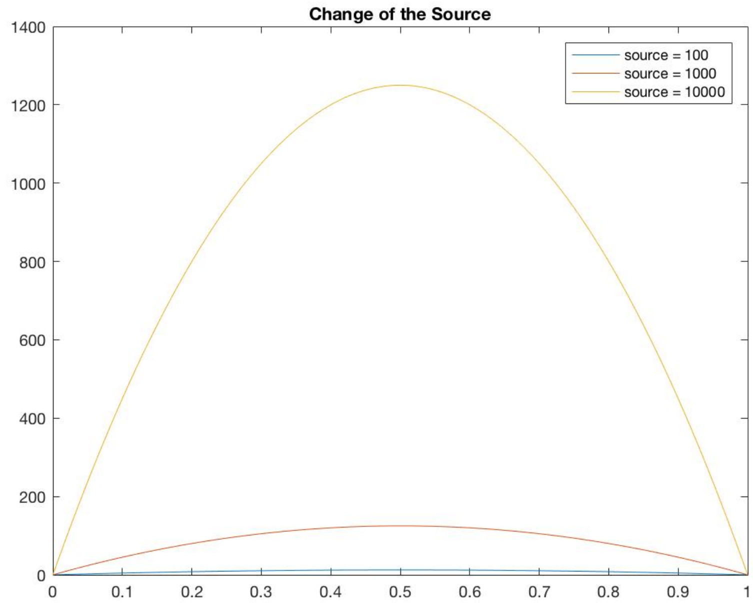

Figure 2: Change of source from 100 to 10000

As was possible to guess for higher values of the source term higher values of U are reached in the middle point. On both ends Dirichlet boundary conditions of value zero is prescribed that is why the effect of the source term is seen in the midpoint of the domain.

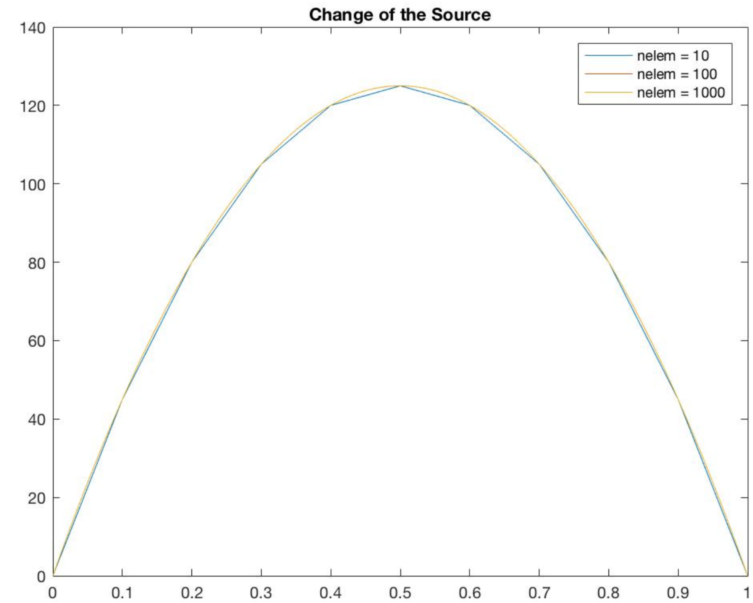

Figure 3: Change of the number of elements from 10 to 100

For the case of the number of elements it was needed to change the code again because the size of the obtained matrices was not equal in the stages. The solutions of each step and the coordinates of the nodes for each mesh was saved in a different matrix and plotted afterwards. As seen in figure.3 above there is not much difference between the values obtained because the problem was very simple, except for the case of 10 elements that because of the low number of elements the linear change is seen which is not correct. So in case the number of elements is high enough so the model converges then there is no large difference between the solutions.

2 Problem 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.

For this problem the code from problem 1 was modified with a difference that more parameters should have been initialized in matrix form in this case. After initializing the domain and the boundary conditions the problem is solved.

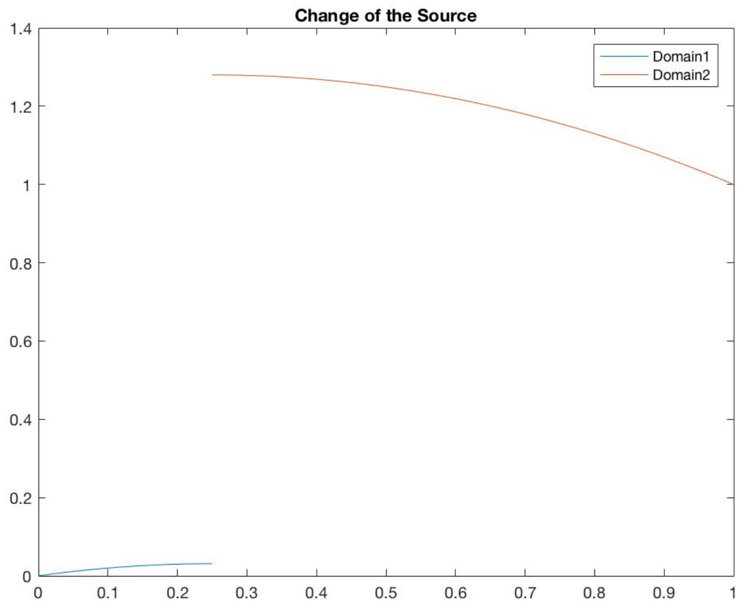

Figure 4: 2 subdomains solved with the initial code

As seen above because of no connection between the two subdomains and no prescribed constraints the plots are discontinuous in the point of connection. In order to get a real solution from the two subdomains it is needed for them to be connected and values checked on the interface.

3 Problem 3

Solve the previous problem in a Monolithic way.

- Study $HP_SolveMonolithic.m$ and relate it to what was explained in theory. Comment on the results.
- Modify the kappa parameter of one of the subdomains. Comment on the results.

For this case the two subdomains are initialized in different data trees with the prescribed values. Afterwards the $HP_SolveMonolithic.m$ code was used to calculate and using the $HP_Plot.m$ code the solutions are plotted.

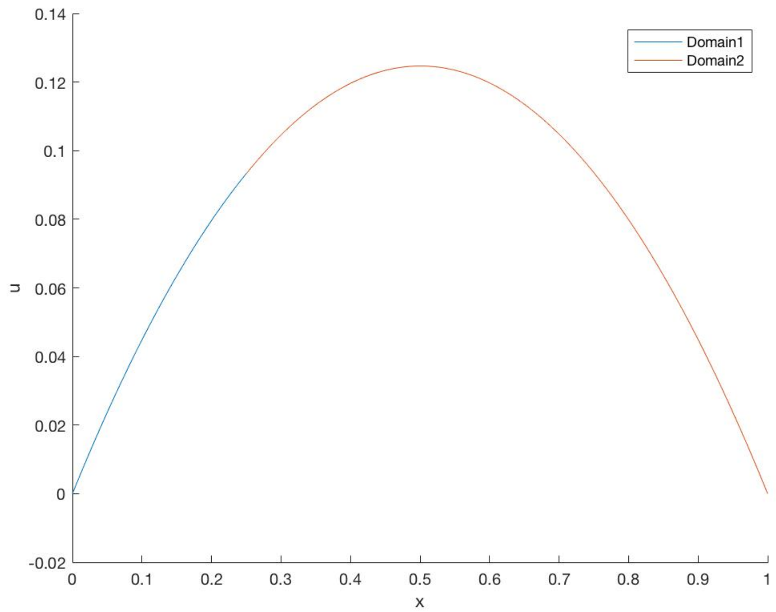

Figure 5: 2 subdomains solved in the monolithic way

Comparing the figure obtained by the monolithic algorithm with the normal, it is seen that while in the normal algorithm on the interface there was a huge gap between the U values of the two subdomains the values obtained for the monolithic algorithm are equal and the plot is continuous as the first part.

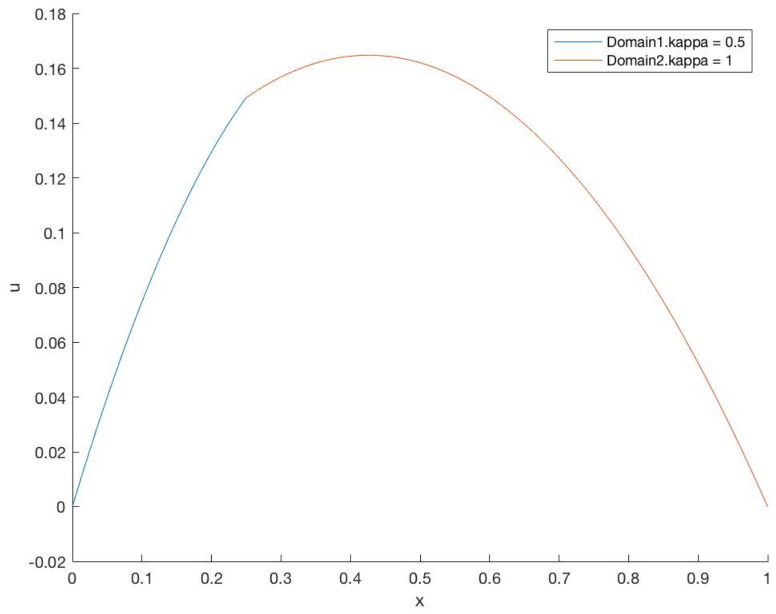

Figure 6: 2 subdomains with different kappa values solved in the monolithic way

In the case of different kappa values for the two subdomains it is seen that using the monolithic algorithm the U value at the interface is equal for the two subdomains but in comparison with the previous point that the kappas were equal the point of connection is no longer smooth and there is an edge on the interface so the derivative of U is not continuous on the interface anymore.

4 Problem 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.

- Evaluate the convergence of the iterative scheme (in terms of u at the interface).
- Increase the value for kappa at subdomain 1 (x100). Comment on the convergence rate.
- Diminish the value for kappa at subdomain 1(/100). Comment on the convergence rate.

For the case of the iterative Dirichlet-Neumann algorithm the two subdomains were initialized as before and after prescribing the input parameters a loop was created for the iterations. In the loop after solving subdomain 1 with the Dirichlet boundary condition the value of the left side of the second subdomain is changed to the value on the interface obtained by the first subdomain. In the second step subdomain 2 is solved and the fluxes calculated in the interface is taken to subdomain 1 and saved. By the end of each iteration the difference between the U value at the interface for subdomain 2 is checked with the value from the previous iteration and in case a tolerance is reached the loop breaks.

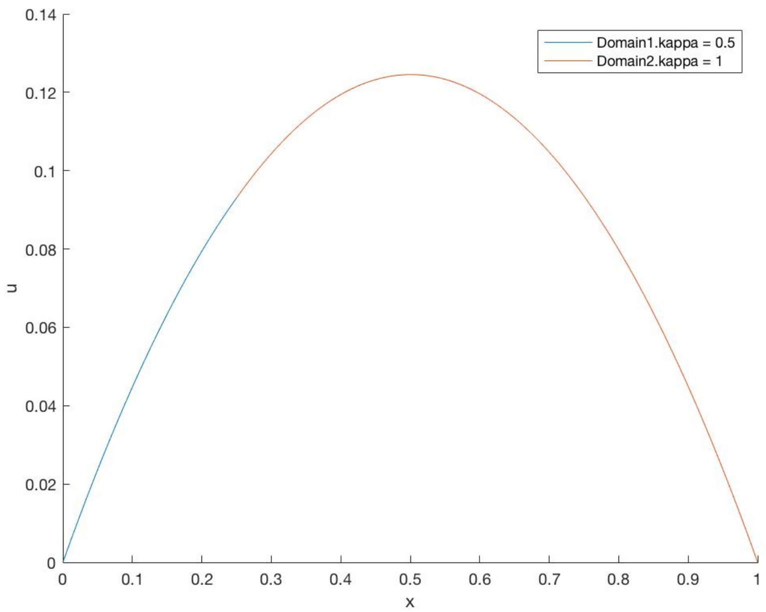

Figure 7: Iterative Dirichlet-Neumann method

As seen the solution is exactly the same as the solution obtained using the monolithic algorithm. For a tolerance of 10^{-8} 18 iterations were done until convergence. Afterwards the value of kappa for subdomain 1 from 1 to 100 the results below was obtained.

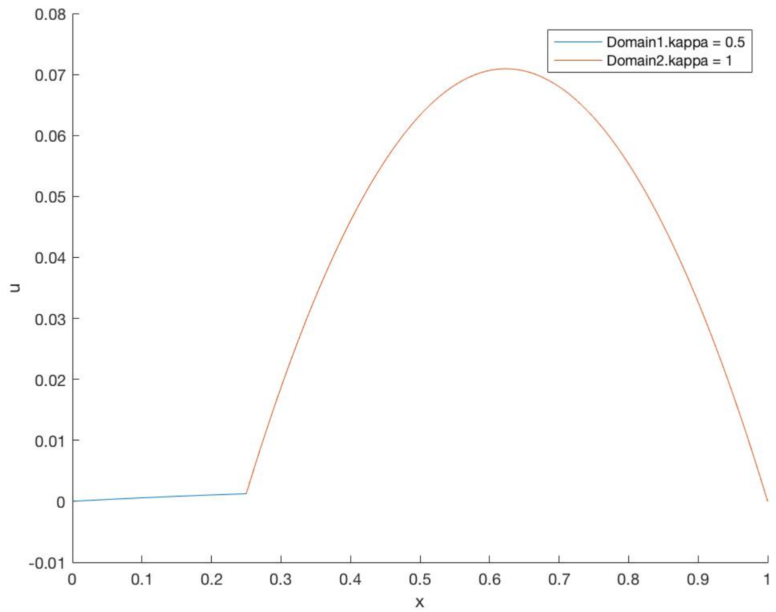

Figure 8: Iterative Dirichlet-Neumann method with kappa1 = 100

As expected the solutions obtained are continuous on the U values but it can be seen that there is an edge on the interface so the derivative of U is not continuous. For a tolerance of 10^{-8} 6 iterations were done until convergence. So in comparison with the previous section it is seen that for higher values of kappa the model converges faster.
For part 'c' it is seen that for a kappa value of 0.01 for subdomain 1 the model does not converge and the solution blows up. Different values smaller than 1 were tested for the model and for a minimum of kappa = 0.3 solutions were obtained. In order to control the iterations done the code was modified and a maximum number of iterations was defined. For a maximum number of iterations of 40 and kappa1 = 0.01 the solution below was obtained.

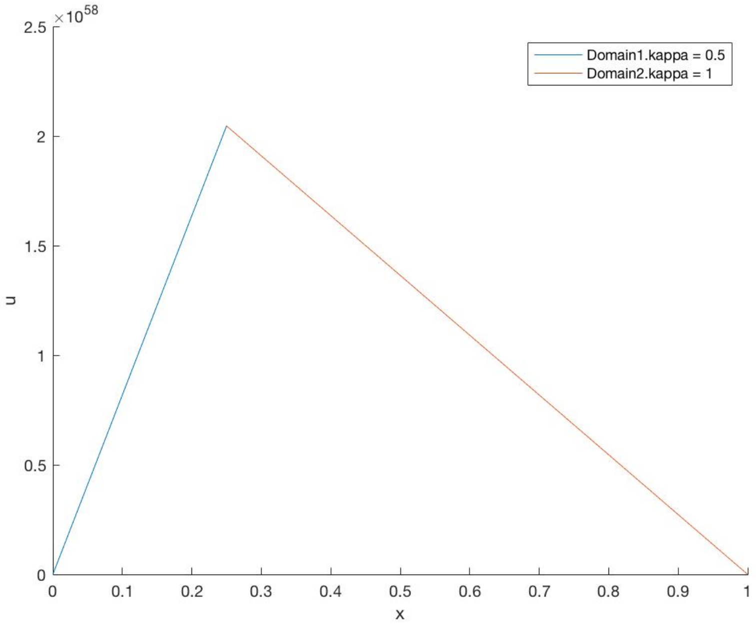

Figure 9: Iterative Dirichlet-Neumann method with kappa1 = 0.01

5 Problem 5

Implement a relaxation scheme

- Relaxation scheme in terms of a fixed relaxation parameter w.
- Aitken relaxation scheme.

For Problem 5 both the relaxation with fixed w value and the Aitken method a code was developed which in the beginning it is chosen between the methods. In the relaxation scheme with fixed value for w, in case w is equal to 1 the same plot as the previous part is obtained. The stability of this method is very much related to the value assigned to w. In case a small value like 0.01 is taken for w the method is not stable and does not converge to the solution.

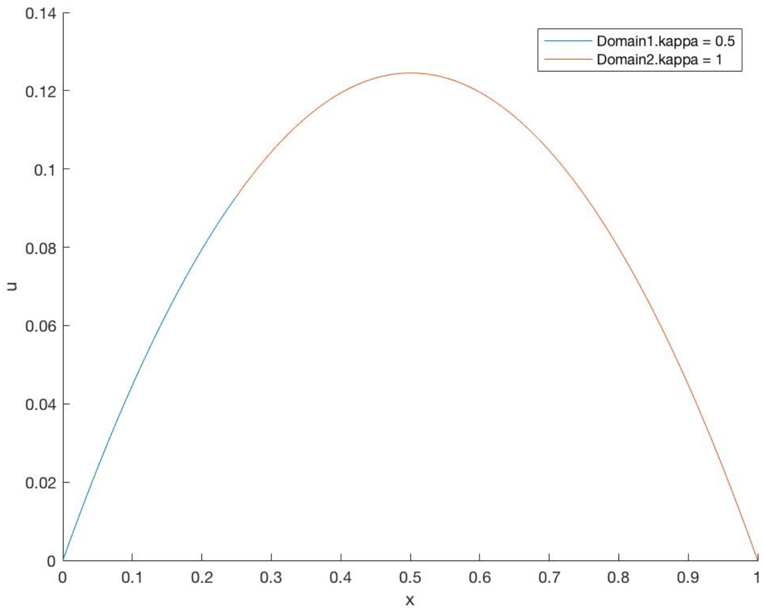

Figure 10: Relaxation scheme with w = 1

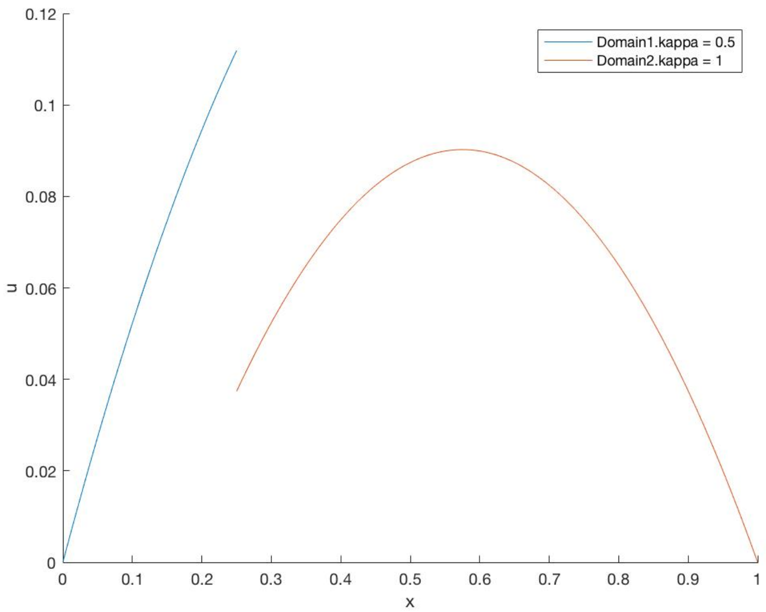

Figure 11: Relaxation scheme with w = 0.01

In case the w is a stable value if we use large values for kappa it is seen that the method converges and plots similar to the ones obtained in the previous parts is obtained. The main advantage of the relaxation scheme and the previous problem is that in case we take low values for kappa 1 like 0.01 where previously there was no convergence in this case using the correct value of the w we can reach convergence. As seen below kappa 1 was taken as 0.01 and using a w of 0.01 we can see that convergence is reached and a logical solution is obtained.

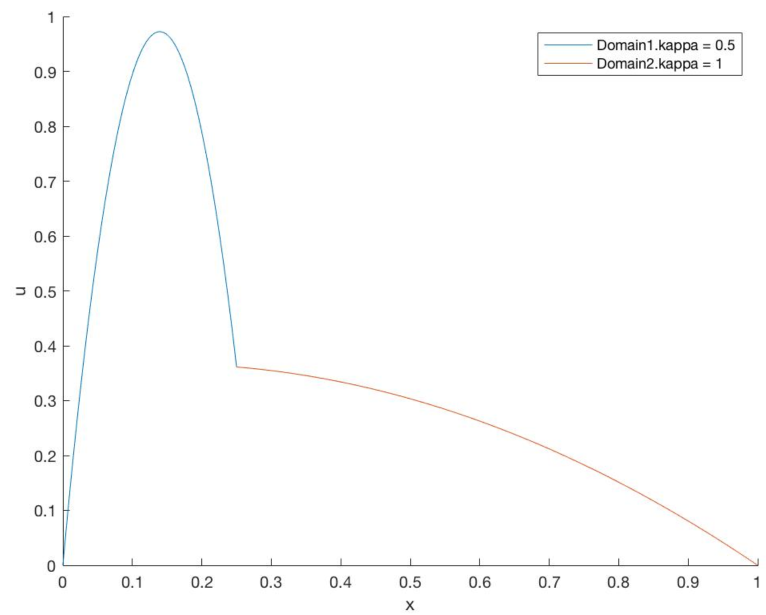

Figure 12: Relaxation scheme with w = 0.01 with kappa1 = 0.01

Also the Aitken method was implemented and the solutions found for the case of equal kappas in the subdomains is similar as the fixed w value in case it is stable. The advantage of the Aitken method in comparison with the fixed w value is that there is no need to change the w and find the stable value for the needed problem. The figures for the Aitken method are represented bellow.

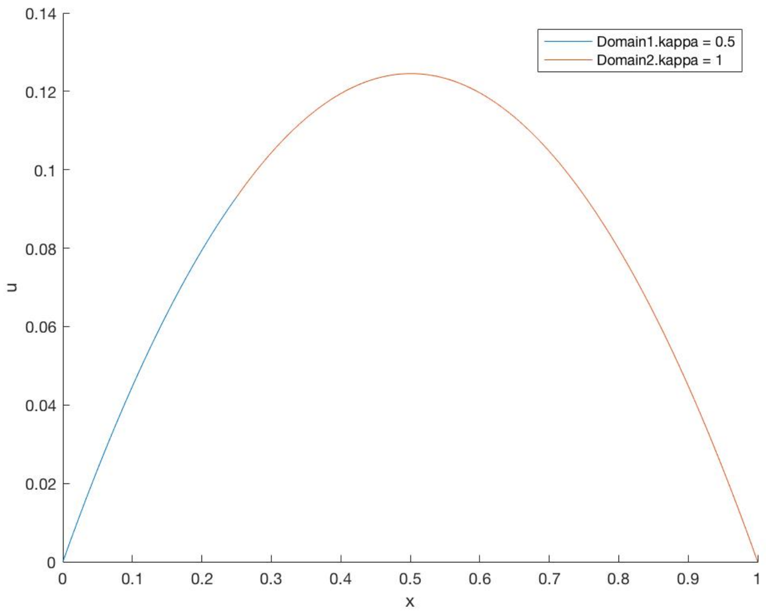

Figure 13: Aitken relaxation scheme with kappa1 = kappa2

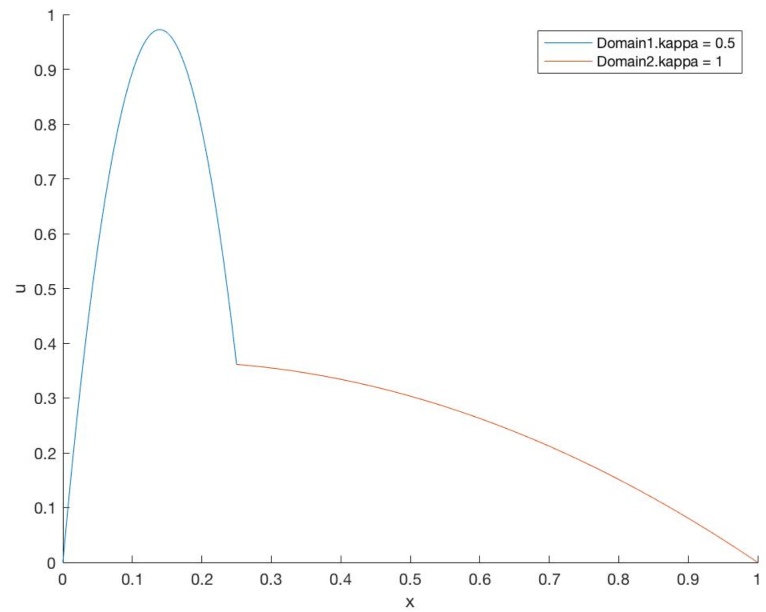

Figure 14: Aitken relaxation scheme with kappa1 = 0.01

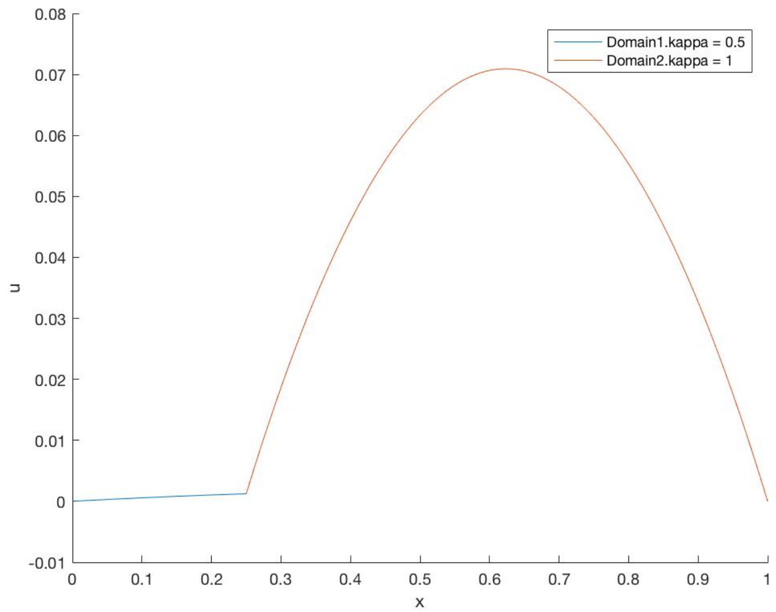

Figure 15: Aitken relaxation scheme with kappa1 = 0.01

In terms of convergence it is seen that the both algorithms with the relaxation scheme converge to the solution in 2 iterations whereas in the previous case it took on average about 10 iterations to reach convergence. In a model like the one given the difference between the two is not seen well but in a problem of big size the difference in the speed of convergence will show.

6 Codes

Function coupled1.m for problem 1:

```
clear all

kappai = [10,100,1000];

sol = [];


for i = 1:size(kappai,2)


    %Domain
    Data.inix = 0;
    Data.endx = 1;
    Data.nelem = 100;
    %Physical
    Data.kappa = kappai(i);
    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 = 25;


    HeatProblem = HP_Initialize(Data);
    HeatProblem = HP_Build(HeatProblem);
    HeatProblem = HP_Solve(HeatProblem);
    sol(i,:) = HeatProblem.Solution.U;

end

x = HeatProblem.Solution.coord;
plot(x,sol(1,:))
title('Change of the Source')

hold on

plot(x,sol(2,:))

plot(x,sol(3,:))

legend('Domain1','Domain2')
```

Function coupled2.m for problem 2:

```
clear all

nixi = [0,0.25];
endi = [0.25,1];
datarighti = [0,1];
datalefti = [1,0];
sol = [];
x = [];

for i = 1:size(nixi,2)


    %Domain
    Data.inix = nixi(i);
    Data.endx = endi(i);
```

```
    Data.nelem = 100;
    %Physical
    Data.kappa = 1;
    Data.source = 1;
    %Boundary conditions
    %Dirichlet
    Data.FixLeft = datalefti(i); %0, do not fix it, 1: fix it
    Data.LeftValue = 0;
    Data.FixRight =datarighti(i);
    Data.RightValue = 1;
    %Neumann
    Data.FixFluxesLeft = 0;
    Data.LeftFluxes = 0;
    Data.FixFluxesRight = 0;
    Data.RightFluxes = 25;


    HeatProblem = HP_Initialize(Data);
    HeatProblem = HP_Build(HeatProblem);
    HeatProblem = HP_Solve(HeatProblem);
    if i == 1
        x1 = HeatProblem.Solution.coord;
        sol1 =  HeatProblem.Solution.U;
    elseif i == 2
        x2 = HeatProblem.Solution.coord;
        sol2 =  HeatProblem.Solution.U;
    end
%     HP_Plot(HeatProblem,1);
end


plot(x1,sol1)
title('Change of the Source')

hold on

plot(x2,sol2)


legend('Domain1','Domain2')
```

Function coupled3.m for problem 3:

```
%Domain1
Data.inix = 0;
Data.endx = 0.25;
Data.nelem = 100;
%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;
Data.RightValue = 0;
%Neumann
Data.FixFluxesLeft = 0;
Data.LeftFluxes = 0;
Data.FixFluxesRight = 0;
Data.RightFluxes = 25;


%Domain2
Data2.inix = 0.25;
Data2.endx = 1;
Data2.nelem = 100;
%Physical
Data2.kappa = 1;
Data2.source = 1;
```

```
%Boundary conditions
%Dirichlet
Data2.FixLeft = 0; %0, do not fix it, 1: fix it
Data2.LeftValue = 0;
Data2.FixRight =1;
Data2.RightValue = 0;
%Neumann
Data2.FixFluxesLeft = 0;
Data2.LeftFluxes = 0;
Data2.FixFluxesRight = 0;
Data2.RightFluxes = 25;


[HeatProblem,HeatProblem2] = HP_SolveMonolithic(HeatProblem,HeatProblem2);
HP_Plot(HeatProblem,1);
HP_Plot(HeatProblem2,1);
legend('Domain1.kappa = 0.5','Domain2.kappa = 1')
```

Function coupled4.m for problem 4:

```
clear all

%Domain1
Data.inix = 0;
Data.endx = 0.25;
Data.nelem = 100;
%Physical
Data.kappa = 0.01;
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 = 1;
Data.RightFluxes = 0;

%Domain2
Data2.inix = 0.25;
Data2.endx = 1;
Data2.nelem = 100;
%Physical
Data2.kappa = 1;
Data2.source = 1;
%Boundary conditions
%Dirichlet
Data2.FixLeft = 1;
Data2.LeftValue = 0;
Data2.FixRight =1;
Data2.RightValue = 0;
%Neumann
Data2.FixFluxesLeft = 0;
Data2.LeftFluxes = 0;
Data2.FixFluxesRight = 0;
Data2.RightFluxes = 25;



tol = 10^-2;
differ = 1;
itr = 1;
maxitr = 40;
while itr<maxitr

    leftval2 = Data2.LeftValue;

```

```
    HeatProblem = HP_Initialize(Data);
    HeatProblem = HP_Build(HeatProblem);
    HeatProblem = HP_Solve(HeatProblem);
    Data2.LeftValue = HeatProblem.Solution.uRight;


    HeatProblem2 = HP_Initialize(Data2);
    HeatProblem2 = HP_Build(HeatProblem2);
    HeatProblem2 = HP_Solve(HeatProblem2);
    Data.RightFluxes = - HeatProblem2.Solution.FluxesLeft;


    differ = abs(Data2.LeftValue-leftval2);


    if itr>40
        break
    else
        itr = itr+1;
    end
end

HP_Plot(HeatProblem,1);
HP_Plot(HeatProblem2,1);
legend('Domain1.kappa = 0.5','Domain2.kappa = 1')
```

Function coupled5.m for problem 5:

```
clear all


relaxation = 1; % 0 for fixed w , 1 for aitken
w_fixedRelaxation = 0.01;


%Domain1
Data.inix = 0;
Data.endx = 0.25;
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 =0;
Data.RightValue = 0;
%Neumann
Data.FixFluxesLeft = 0;
Data.LeftFluxes = 0;
Data.FixFluxesRight = 1;
Data.RightFluxes = 0;

%Domain2
Data2.inix = 0.25;
Data2.endx = 1;
Data2.nelem = 100;
%Physical
Data2.kappa = 1;
Data2.source = 1;
%Boundary conditions
%Dirichlet
Data2.FixLeft = 1;
Data2.LeftValue = 0;
Data2.FixRight =1;
Data2.RightValue = 0;
%Neumann
Data2.FixFluxesLeft = 0;
Data2.LeftFluxes = 0;
```

```
Data2.FixFluxesRight = 0;
Data2.RightFluxes = 25;



tol = 10^-8;
differ = 1;
itr = 1;
maxitr = 40;
while itr < maxitr

    if relaxation == 1
        if itr == 1
            leftval2 = Data2.LeftValue;

            HeatProblem2 = HP_Initialize(Data2);
            HeatProblem2 = HP_Build(HeatProblem2);
            HeatProblem2 = HP_Solve(HeatProblem2);

            Data.RightFluxes = - HeatProblem2.Solution.FluxesLeft;

            HeatProblem = HP_Initialize(Data);
            HeatProblem = HP_Build(HeatProblem);
            HeatProblem = HP_Solve(HeatProblem);

            Data2.LeftValue = HeatProblem.Solution.uRight;

            HeatProblem2 = HP_Initialize(Data2);
            HeatProblem2 = HP_Build(HeatProblem2);
            HeatProblem2 = HP_Solve(HeatProblem2);

            Data.RightFluxes = - HeatProblem2.Solution.FluxesLeft;
        end

        U1prev = HeatProblem.Solution.uRight;

        prevleftval2 = leftval2;

        leftval2 = HeatProblem2.Solution.uLeft;
    else

        leftval2 = Data2.LeftValue;

    end

    HeatProblem = HP_Initialize(Data);
    HeatProblem = HP_Build(HeatProblem);
    HeatProblem = HP_Solve(HeatProblem);

    if relaxation == 0
        w = w_fixedRelaxation;
        Data2.LeftValue = w*HeatProblem.Solution.uRight + (1-w)*leftval2;

    elseif relaxation ==1
        w = (prevleftval2 - leftval2)/(prevleftval2 - leftval2 + ...
            HeatProblem.Solution.uRight - U1prev);
        Data2.LeftValue = leftval2 ...
            + w*(HeatProblem.Solution.uRight - leftval2);
    end

    HeatProblem2 = HP_Initialize(Data2);
    HeatProblem2 = HP_Build(HeatProblem2);
    HeatProblem2 = HP_Solve(HeatProblem2);


    Data.RightFluxes = - HeatProblem2.Solution.FluxesLeft;
    differ = abs(Data2.LeftValue - leftval2);

    if differ <= tol

```

```
            break
        end
            itr = itr + 1;
end



HP_Plot(HeatProblem,1);
HP_Plot(HeatProblem2,1);
legend('Domain1.kappa = 0.5','Domain2.kappa = 1')
```