Universitat Politècnica de Catalunya

Coupled Problems

Master's Degree in Numerical Methods in Engineering

Iterative schemes for coupling in space

Author:
Diego Roldán

Supervisor:
Prof. J. Baiges

Academic Year 2019-2020

Contents

1 Task 1. Single heat transfer problem.

a) Study the effect of changing the value to the thermal diffusion coefficient (κ).

In Figure 1, it is shown the effect of changing the diffusion coefficient (κ) considering the source term as 1 and 100 elements.

As this coefficient increases, it produces that the maximum temperature in the domain is decreasing. In other words, the solution is more diffusive.

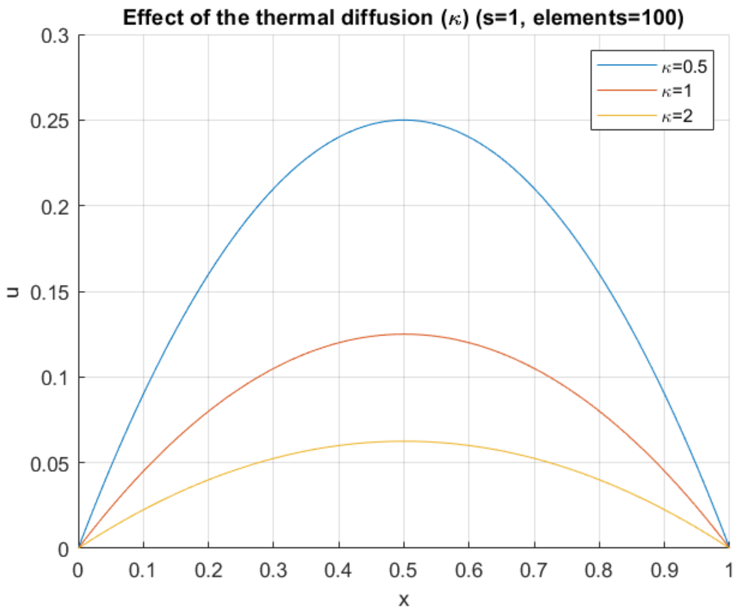


Figure 1: Effect of the thermal diffusion coefficient (κ, considering s=1 and 100 elements.

b) Study the effect of changing the source term value (s).

In Figure 2, it is shown the effect of changing the source term value (s) considering $\kappa = 1$ and 100 elements.

As the source term value increases, it increases the maximum temperature that the solution reaches. It is possible to notice that the same profiles of temperature are obtained as the previous case, concluding an inverse proportion to the diffusive coefficient.

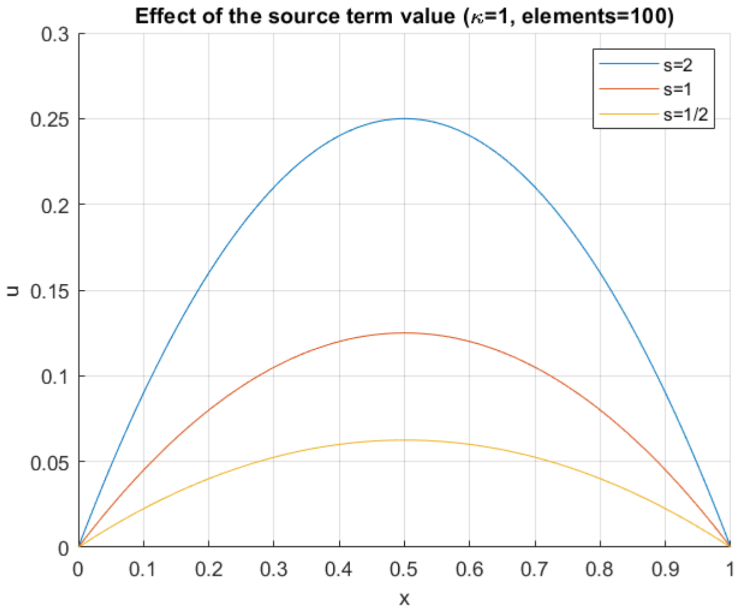


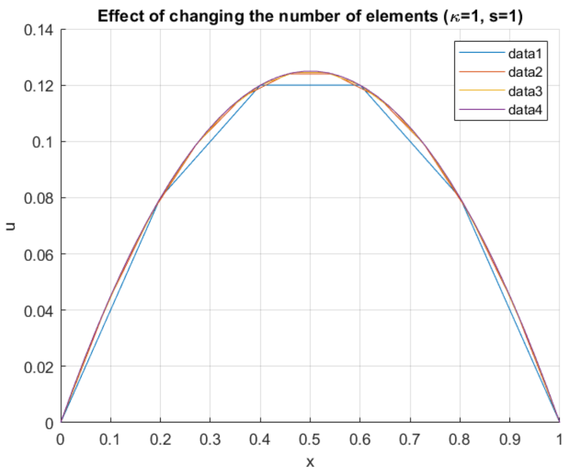
Figure 2: Effect of the source term, considering $\kappa = 1$ and 100 elements.

c) Study the effect of changing the number of elements.

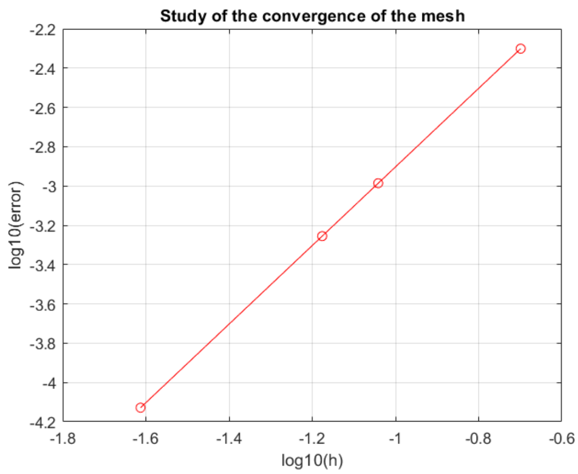
In Figure 3, it is shown the study of the convergence of the single heat transfer problem, considering $\kappa = 1$ and $s = 1$.

It is clearly shown in Figure 3a the effect of increasing the number of elements which leads to an accurate solution. For this study, it is used the following number of elements: $[5, 11, 15, 41]$. The 1D Poisson problem is exact in the nodes. Therefore, the solution will be exact if the number of elements are odd.

The study of the convergence of the mesh is carried out just considering the maximum temperature values that occur in the middle of the domain as it shown in Figure 3a. In Figure 3b, it is plotted the log-log graph showing the behaviour of the error of the maximum value against the element size (h).



(a) Effect of changing the number of elements.



(b) Convergence rate of the error.

Figure 3: Analysis of the convergence of the single heat transfer problem.

2 Task 2. Two independent heat transfer problems

In this section, it is solved two independent heat transfer problems for a domain of $[0, 1]$ split in two subdomains. The first problem subdomain is $[0, 0.25]$ with 100 elements. The second problem subdomain is $[0.25, 1]$ with 100 elements as well. It is fixed u in $x = 0$ and $x = 1$ as 0 value, leaving it free in the interface between subdomains. The diffusion coefficient is $\kappa = 1$ and the source term $s = 1$.

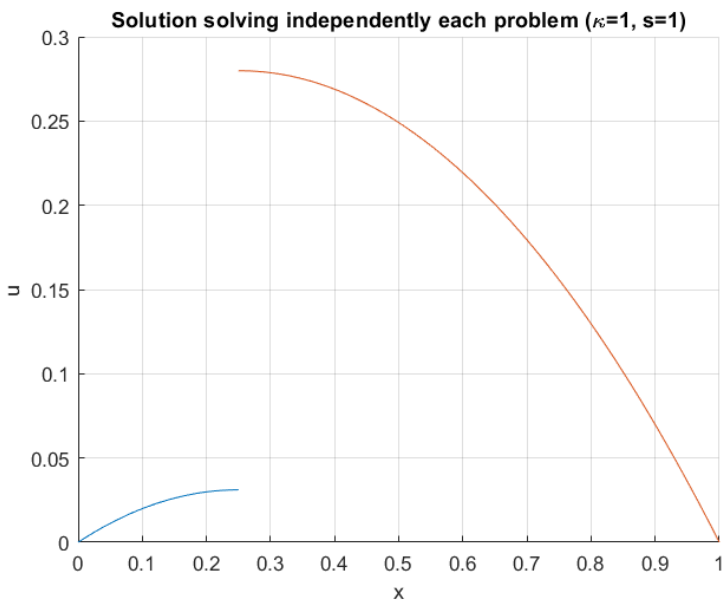


Figure 4: Solution of two independent heat transfer problems ($\kappa = 1, s = 1$).

The remaining boundary conditions (interface Dirichlet or Neumann) are not introduced. Therefore, as it is shown in Figure 4, there is a considerable jump between both subdomains in the interface. This is because both subdomains solve the heat transfer problem independently with only one Dirichlet boundary condition in each one and do not consider any transmission condition from the other subdomain.

In conclusion, the solution is not continuous without any physical meaning because there is no transmission conditions introduced.

3 Task 3. Monolithic solver

a) Solve previous problem in a Monolithic way.

In order to solve the previous problem with consistency is proposed the Monolithic solver, giving an accurate result as it is shown in Figure 5.

The main difference between the previous case and the monolithic way is that this solver gives a matching solution in the interface. This is due to the fact that the Monolithic solver computes the solution for both problems as one unique, combining the two degrees of freedom at the interface of the two subdomains as one as well.

This procedure assures the continuity of the solution and the fluxes at the interface. This problem is solved considering both diffusion coefficients for each subdomain equal to $\kappa_1 = \kappa_2 = 1$. And defining the flux defined as $\mathbf{n} \cdot \kappa \nabla u$, assures that the gradient of the solution (the slope) is continuous at the interface.

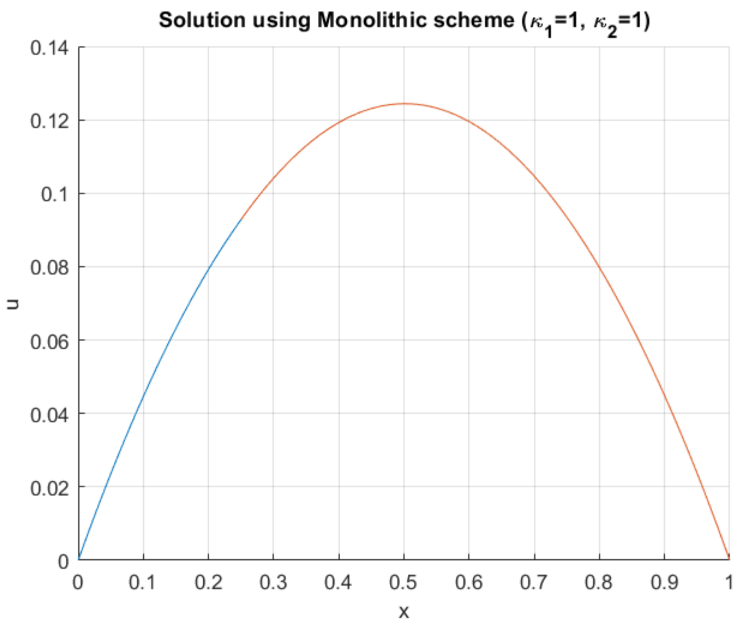


Figure 5: Solution using a monolithic solver, considering $\kappa_1 = 1, \kappa_2 = 1$.

b) Modify the kappa parameter of one of the subdomains.

In this section, it is required to modify the diffusion coefficient for one of the subdomains and comment on the results. Therefore, it is introduced a $\kappa_2 = 4$ for the second subdomain, resulting in the curve shown in Figure 6.

It is possible to appreciate that the solution is the same in the interface because there is still the condition of enforcing the continuity of fluxes. However, the slope of the solution (its gradient) is discontinuous at the interface due to the fact that the parameter κ is different for each subdomain. Therefore, the slopes of both subdomains alter in order to satisfy the continuity of the fluxes at the interface.

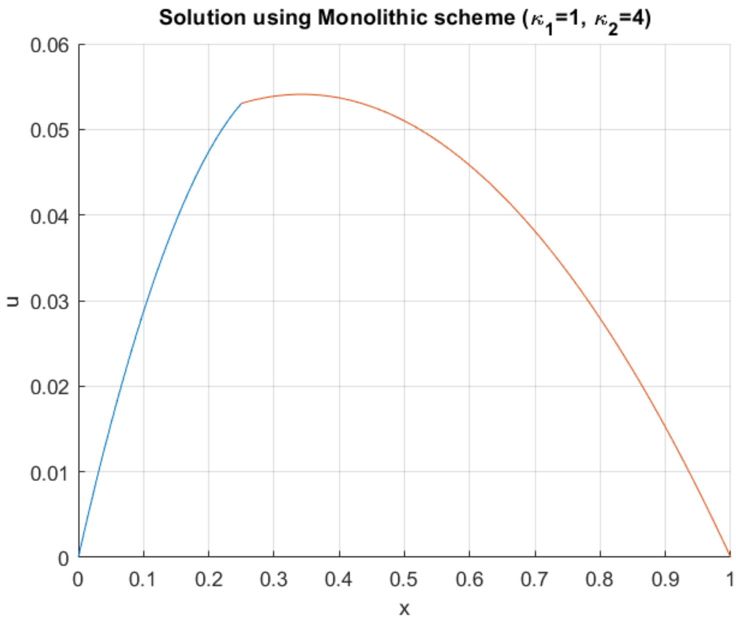


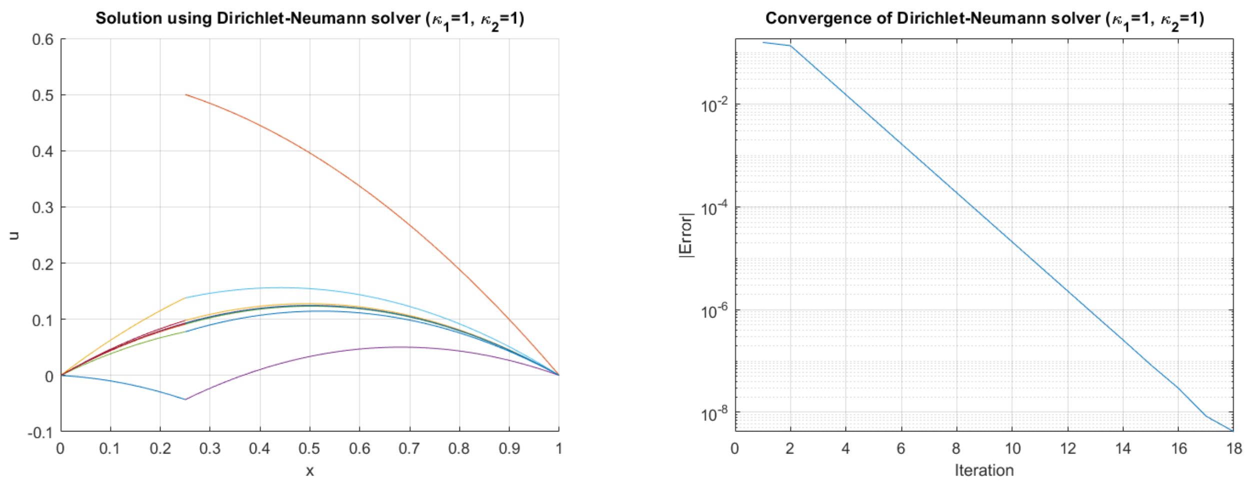
Figure 6: Solution using a monolithic solver, considering $\kappa_1 = 1, \kappa_2 = 4$.

4 Task 4. Solver using Dirichlet-Neumann iterations

a) Evaluate the convergence of the iterative scheme.

In this section, it is required to evaluate the convergence of the previous problem in an iterative manner (Dirichlet-Neumann). It is introduced $\kappa_1 = \kappa_2 = 1$ and it is applied Neumann boundary conditions at the interface in the first subdomain and Dirichlet boundary conditions at the interface in the second subdomain.

In Figure 7, it is shown the problem is solved an accurate way giving a good result and a linear convergence rate of the error until it is achieved a tolerance of 10^{-8} with 16 iterations. This iterative scheme gives a similar result as in the Monolithic solver.



(a) Solution of the problem.

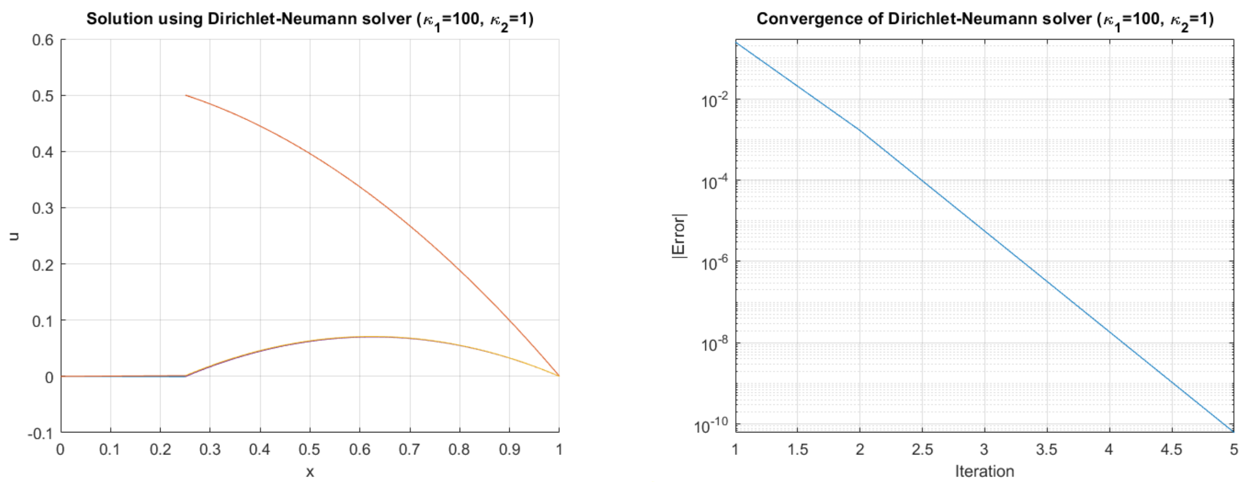
(b) Convergence rate of the error.

Figure 7: Solution and convergence study using iterative scheme, considering $\kappa_1 = 1$ and $\kappa_2 = 1$.

b) Increase the value for kappa at subdomain 1 (x100).

In this section, it is increased the diffusive coefficient of domain 1 to $\kappa_1 = 100$. As it is shown in Figure 8a, there is a jump in the solution produced for the same reason as it is explained in task 3.b.

There is a linear convergence and it is reached the tolerance of 10^{-8} in fewer iterations, just 4. Therefore, the convergence is reached faster than the previous case.



(a) Solution of the problem.

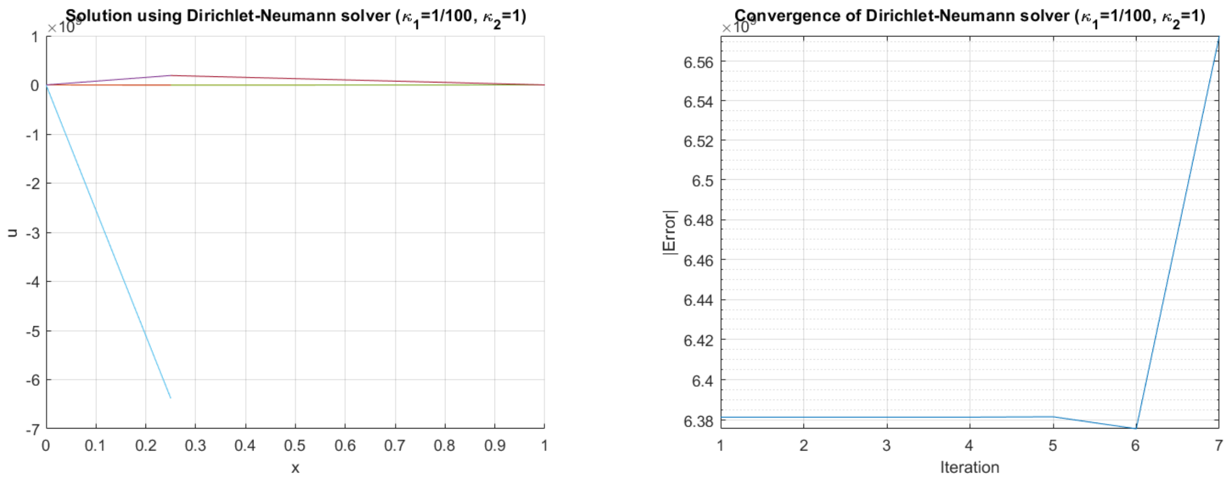
(b) Convergence rate of the error.

Figure 8: Solution and convergence study using iterative scheme, considering $\kappa_1 = 100$ and $\kappa_2 = 1$.

c) Diminish the value for kappa at subdomain 1(/100).

In this section, it is decreased the diffusive coefficient of domain 1 to $\kappa_1 = 1/100$.

As it is shown in Figure 9a, the solution blows up showing that the iterative scheme is unstable. In Figure 9b, it is shown that solution does not convergence, concluding that the iterative scheme does not work for this case presenting stability problems.



(a) Solution of the problem.

(b) Convergence rate of the error.

Figure 9: Solution and convergence study using iterative scheme, considering $\kappa_1 = 1/100$ and $\kappa_2 = 1$.

d) Stability of the coupling scheme.

To sum up the results, it is possible to conclude that, as it is shown in Figures 7 and 8, the Dirichlet-Neumann iteration scheme is stable only if the Dirichlet conditions are applied at the interface of the subdomain with lower diffusion coefficient (κ).

As for the other case where the solution does not converge and presents instabilities in the solution as it is shown in Figure 9, it is necessary to introduce a relaxation scheme to remove these instabilities. This would imply a slower convergence speed.

5 Task 5. Implementation of a relaxation scheme

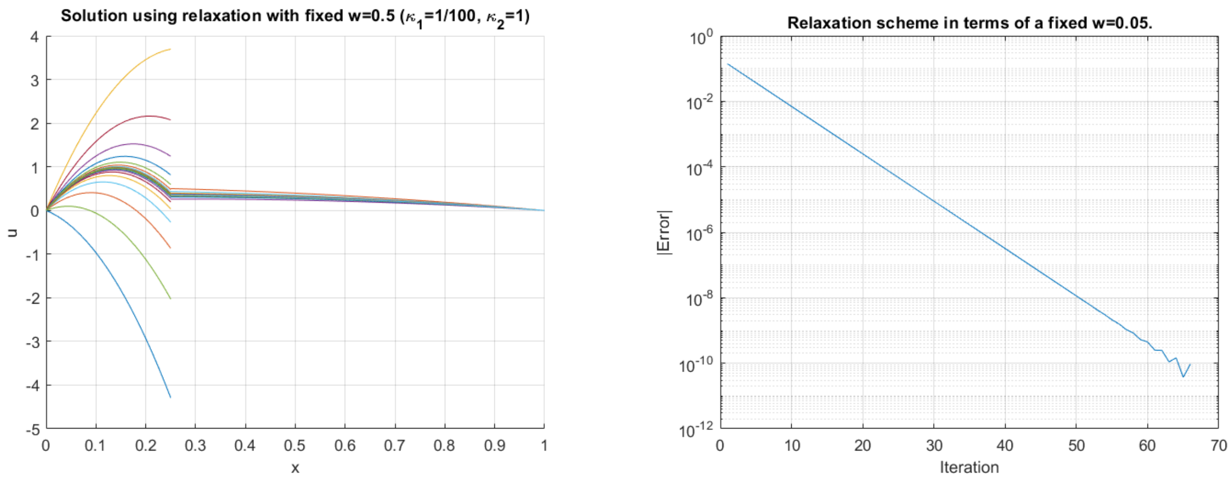
a) Relaxation scheme in terms of a fixed relaxation parameter w.

In this section, it is presented the solution for the instabilities presented previously for the iterative Dirichlet-Neumann scheme.

This solution implies to implement a relaxation scheme with a parameter fixed w which is the fixed relaxation. In this relaxation scheme, the Dirichlet condition applied to the second subdomain is relaxed. This means that the solution obtained from the first subdomain is not introduced automatically, instead it is applied the average of it and the solution is evaluated at the previous iteration.

$$u_{\Gamma 21}^{i} = \omega \cdot u_{\Gamma 12}^{i} + (1 - \omega) \cdot u_{\Gamma 21}^{i-1} \tag{1}$$

As it is shown in Figure 10, the previous problem that did not converge, with this scheme the solution converges, considering $\kappa_1 = 1/100$ and $\kappa_2 = 1$ with a relaxation parameter of $w = 0.05$ As it is shown in Figure 10b, the error tolerance of 10^{-10} is reached with 60 iterations. Therefore, it is shown that the convergence speed is slower than the previous case.



(a) Solution of the problem.

(b) Convergence rate of the error.

Figure 10: Solution and convergence study using relaxation scheme fixing $w = 0.05$.

b) Aitken relaxation scheme.

In this section, it is implemented the Aikten relaxation scheme and it is proved that it overcomes the disadvantages of the fixed relaxation scheme. In this scheme the relaxation parameter (w) is calculated automatically from the interface solution from both subdomains, considering the current iterations and the previous two iterations as it is shown in Equation 2.

$$\omega = \frac{u_{\Gamma 21}^{i-2} - u_{\Gamma 21}^{i-1}}{(u_{\Gamma 21}^{i-2} - u_{\Gamma 21}^{i-1}) - (u_{\Gamma 21}^{i} - u_{\Gamma 21}^{i-1})} \tag{2}$$

As it is shown in Figure 11, the previous problem that did not converge in Task 4, with this scheme the solution converges, considering $\kappa_1 = 1/100$ and $\kappa_2 = 1$ with a relaxation parameter of $w = 0.5$ As it is shown in Figure 11, the error tolerance of 10^{-10} is reached with 5 iterations. Therefore, it is shown that the convergence speed is much faster than the fixed relaxation scheme.

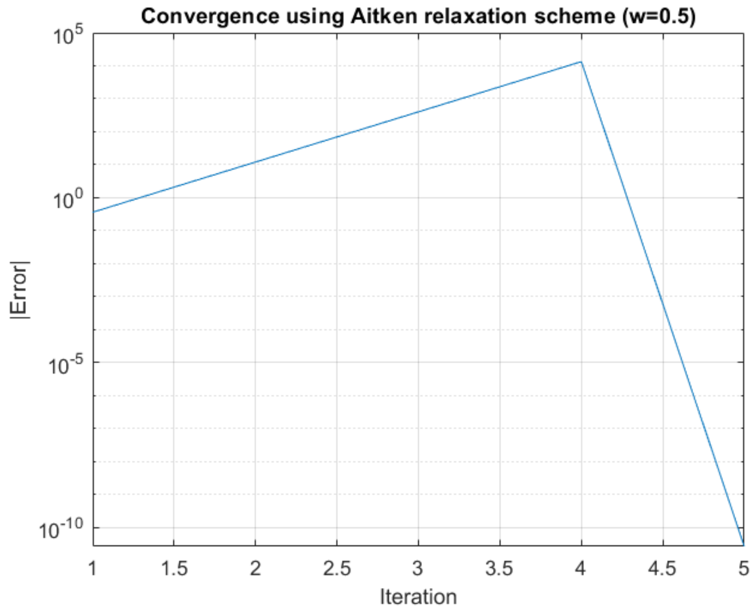


Figure 11: Solution and convergence study using Aitken relaxation scheme fixing $w = 0.5$.

A Appendix

Next, it is presented the implemented codes required for this assignment.

A.1 Code for Task 1

```
close all
clear variables


%% Effect of kappa

%Domain 1
Data.inix = 0;
Data.endx = 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;

%Effect of kappa
Data.nelem = 100;
Data.source = 1;

% Kappa = 0.5
Data.kappa = 0.5;

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

HeatProblem = HP_Solve(HeatProblem);
HP_Plot(HeatProblem,1);

% Kappa = 1
```

```
Data.kappa = 1;

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

HeatProblem = HP_Solve(HeatProblem);
HP_Plot(HeatProblem,1);

% Kappa = 2
Data.kappa = 2;

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

HeatProblem = HP_Solve(HeatProblem);
HP_Plot(HeatProblem,1);

title('Effect of the thermal diffusion (\kappa) (s=1, elements=100)')
legend('\kappa=0.5','\kappa=1','\kappa=2')


%% Effect of source term (s)

Data.nelem = 100;
Data.kappa = 1;

% s = 2
Data.source = 2;

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

HeatProblem = HP_Solve(HeatProblem);
HP_Plot(HeatProblem,2);

% s = 1
Data.source = 1;

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

HeatProblem = HP_Solve(HeatProblem);
HP_Plot(HeatProblem,2);
```

```
% s = 1/2
Data.source = 1/2;

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

HeatProblem = HP_Solve(HeatProblem);
HP_Plot(HeatProblem,2);

title('Effect of the source term value (\kappa=1, elements=100)')
legend('s=2','s=1','s=1/2')


%% Convergence study

%%Effect of number of elements
nOfElements = [5 11 15 41];

error = zeros(length(nOfElements),1);
h = zeros(length(nOfElements),1);

    for i = 1:length(nOfElements)

        %Domain
        Data.inix = 0;
        Data.endx = 1;
        Data.nelem = nOfElements(i);

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

        fignum = 3;
        legendElem = (['nelem = ',num2str(Data.nelem)]);
        HP_Plot(HeatProblem,fignum,legendElem);
        hold on

        title('Effect of changing the number of elements (\kappa=1, s=1)')
        legend('show')
        ylim([0 0.14])

        % Data needed to produce the mesh convergence plot
        Solmax = max(HeatProblem.Solution.U);
        error(i) = abs(0.125 - Solmax);
        sizeh(i) = (Data.endx - Data.inix)/nOfElements(i);
    end



% Plotting a mesh convergence plot
figure(4)
plot(log10(sizeh),log10(error),'-ro');
grid on
xlabel('log10(h)')
ylabel('log10(error)')
title('Study of the convergence of the mesh')
```

A.2 Code for Task 2

```
close all
clear variables

%Domain 1
DataL.inix = 0;
```

```
DataL.endx = 0.25;
DataL.nelem = 100;
%Physical
DataL.kappa = 1;
DataL.source = 1;
%Boundary conditions
%Dirichlet
DataL.FixLeft = 1; %0, do not fix it, 1: fix it
DataL.LeftValue = 0;
DataL.FixRight =0;
DataL.RightValue = 0;
%Neumann
DataL.FixFluxesLeft = 0;
DataL.LeftFluxes = 0;
DataL.FixFluxesRight = 0;
DataL.RightFluxes = 0;

%Domain2
DataR.inix = 0.25;
DataR.endx = 1;
DataR.nelem = 100;
%Physical
DataR.kappa = 1;
DataR.source = 1;
%Boundary conditions
%Dirichlet
DataR.FixLeft = 0; %0, do not fix it, 1: fix it
DataR.LeftValue = 0;
DataR.FixRight =1;
DataR.RightValue = 0;
%Neumann
DataR.FixFluxesLeft = 0;
DataR.LeftFluxes = 0;
DataR.FixFluxesRight = 0;
DataR.RightFluxes = 0;

% Monolithic
%Problem Left
    HeatProblemL = HP_Initialize(DataL);
    HeatProblemL = HP_Build(HeatProblemL);

%Problem Right
    HeatProblemR = HP_Initialize(DataR);
```

```
    HeatProblemR = HP_Build(HeatProblemR);

%Solve and plot
    HeatProblemL = HP_Solve(HeatProblemL);
    HeatProblemR = HP_Solve(HeatProblemR);
    HP_Plot(HeatProblemL,1);
    HP_Plot(HeatProblemR,1);

    title('Solution solving independently each problem (\kappa=1, s=1)')
```

A.3 Code for Task 3

```
close all
clear variables

%% Domain 1
DataL.inix = 0;
DataL.endx = 0.25;
DataL.nelem = 25;
%Physical
DataL.kappa = 1;
DataL.source = 1;
%Boundary conditions
%Dirichlet
DataL.FixLeft = 1; %0, do not fix it, 1: fix it
DataL.LeftValue = 0;
DataL.FixRight =0;
DataL.RightValue = 0;
%Neumann
DataL.FixFluxesLeft = 0;
DataL.LeftFluxes = 0;
DataL.FixFluxesRight = 0;
DataL.RightFluxes = 0;

%% Domain2
DataR.inix = 0.25;
DataR.endx = 1;
DataR.nelem = 75;
%Physical
```

```
DataR.kappa = 4;
DataR.source = 1;

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

%% Monolithic solver

%Problem Left
    HeatProblemL = HP_Initialize(DataL);
    HeatProblemL = HP_Build(HeatProblemL);
%Problem Right
    HeatProblemR = HP_Initialize(DataR);
    HeatProblemR = HP_Build(HeatProblemR);
%Solve and plot
    [HeatProblemL,HeatProblemR] = HP_SolveMonolithic(HeatProblemL,HeatProblemR);
    HP_Plot(HeatProblemL,1);
    HP_Plot(HeatProblemR,1);
    %ylim([0 0.06])
    title('Solution using Monolithic scheme (\kappa_1=1, \kappa_2=4)')
```

A.4 Code for Task 4

```
close all
clear variables

%Domain 1
DataL.inix = 0;
DataL.endx = 0.25;
DataL.nelem = 25;
```

```
%Physical
DataL.kappa = 1/100;
DataL.source = 1;
%Boundary conditions
%Dirichlet
DataL.FixLeft = 1; %0, do not fix it, 1: fix it
DataL.LeftValue = 0;
DataL.FixRight =0;
DataL.RightValue = 1;
%Neumann
DataL.FixFluxesLeft = 0;
DataL.LeftFluxes = 0;
DataL.FixFluxesRight = 1;
DataL.RightFluxes = 0;

%Domain2
DataR.inix = DataL.endx;
DataR.endx = 1;
DataR.nelem = 75;
%Physical
DataR.kappa = 1;
DataR.source = 1;
%Boundary conditions
%Dirichlet
DataR.FixLeft = 1; %0, do not fix it, 1: fix it
DataR.LeftValue = 0.25;
DataR.FixRight =1;
DataR.RightValue = 0;
%Neumann
DataR.FixFluxesLeft = 0;
DataR.LeftFluxes = 0;
DataR.FixFluxesRight = 0;
DataR.RightFluxes = 0;

% Dirichlet-Neumann
HeatProblemR = HP_Initialize(DataR);
HeatProblemL = HP_Initialize(DataL);
HeatProblemL.Solution.uRight = 0.5;
HeatProblemR.Solution.FluxesRight = 0;
u=0.25;

while true
%Problem R
```

```
    DataR.LeftValue = HeatProblemL.Solution.uRight;
    Solution_old = HeatProblemL.Solution.uRight;

    HeatProblemR = HP_Initialize(DataR);
    HeatProblemR = HP_Build(HeatProblemR);
    HeatProblemR = HP_Solve(HeatProblemR);

%Problem L
    DataL.RightFluxes = -HeatProblemR.Solution.FluxesLeft;

    HeatProblemL = HP_Initialize(DataL);
    HeatProblemL = HP_Build(HeatProblemL);
    HeatProblemL = HP_Solve(HeatProblemL);
    u = [u, HeatProblemL.Solution.uRight];
if abs(HeatProblemL.Solution.uRight - Solution_old) < 1E-8
   break
end
if abs(HeatProblemL.Solution.uRight) > 1E9
    break
end

HP_Plot(HeatProblemL,1);
HP_Plot(HeatProblemR,1);

end
HP_Plot(HeatProblemL,1);
HP_Plot(HeatProblemR,1);
figure(1)
title('Solution using Dirichlet-Neumann solver (\kappa_1=1/100, \kappa_2=1)')
x = 1:length(u);

figure(2)
semilogy(x, abs(u-u(end)))
xlabel('Iteration')
ylabel('|Error|')
title('Convergence of Dirichlet-Neumann solver (\kappa_1=1/100, \kappa_2=1)')
grid on
```

A.5 Code for Task 5.a)

```
close all
clear variables

%Domain 1
DataL.inix = 0;
DataL.endx = 0.25;
DataL.nelem = 25;
%Physical
DataL.kappa = 1/100;
DataL.source = 1;
%Boundary conditions
%Dirichlet
DataL.FixLeft = 1; %0, do not fix it, 1: fix it
DataL.LeftValue = 0;
DataL.FixRight =0;
DataL.RightValue = 0.5;
%Neumann
DataL.FixFluxesLeft = 0;
DataL.LeftFluxes = 0;
DataL.FixFluxesRight = 1;
DataL.RightFluxes = 0;

%Domain2
DataR.inix = DataL.endx;
DataR.endx = 1;
DataR.nelem = 75;
%Physical
DataR.kappa = 1;
DataR.source = 1;
%Boundary conditions
%Dirichlet
DataR.FixLeft = 1; %0, do not fix it, 1: fix it
DataR.LeftValue = 0.5;
DataR.FixRight =1;
DataR.RightValue = 0;
%Neumann
DataR.FixFluxesLeft = 0;
DataR.LeftFluxes = 0;
DataR.FixFluxesRight = 0;
DataR.RightFluxes = 0;

```

```
% Dirichlet-Neumann
HeatProblemR = HP_Initialize(DataR);
HeatProblemL = HP_Initialize(DataL);
HeatProblemL.Solution.uRight = 0.5;
HeatProblemR.Solution.FluxesRight = 0;

w = 0.05;
u = [0.5];
while true
%Problem R
    DataR.LeftValue = u(end);

    HeatProblemR = HP_Initialize(DataR);
    HeatProblemR = HP_Build(HeatProblemR);
    HeatProblemR = HP_Solve(HeatProblemR);

%Problem L
    DataL.RightFluxes = -HeatProblemR.Solution.FluxesLeft;

    HeatProblemL = HP_Initialize(DataL);
    HeatProblemL = HP_Build(HeatProblemL);
    HeatProblemL = HP_Solve(HeatProblemL);

    u = [u, HeatProblemL.Solution.uRight*w + u(end) * (1-w)];

if abs(u(end) - u(end-1)) < 1E-10
   break
end
if abs(HeatProblemL.Solution.uRight) > 1E10
    break
end
HP_Plot(HeatProblemL,1);
HP_Plot(HeatProblemR,1);

end
HP_Plot(HeatProblemL,1);
HP_Plot(HeatProblemR,1);
title('Solution using relaxation with fixed w=0.5 (\kappa_1=1/100, \kappa_2=1)')
x = 1:length(u);

figure(2)
semilogy(x, abs(u-u(end)))
title('Relaxation scheme in terms of a fixed w=0.05.')
```

```
xlabel('Iteration')
ylabel('|Error|')
grid on
```

A.6 Code for Task 5.b)

```
close all
clear variables

%Domain 1
DataL.inix = 0;
DataL.endx = 0.25;
DataL.nelem = 25;
%Physical
DataL.kappa = 1/100;
DataL.source = 1;
%Boundary conditions
%Dirichlet
DataL.FixLeft = 1; %0, do not fix it, 1: fix it
DataL.LeftValue = 0;
DataL.FixRight =0;
DataL.RightValue = 0.5;
%Neumann
DataL.FixFluxesLeft = 0;
DataL.LeftFluxes = 0;
DataL.FixFluxesRight = 1;
DataL.RightFluxes = 0;

%Domain2
DataR.inix = DataL.endx;
DataR.endx = 1;
DataR.nelem = 75;
%Physical
DataR.kappa = 1;
DataR.source = 1;
%Boundary conditions
%Dirichlet
DataR.FixLeft = 1; %0, do not fix it, 1: fix it
DataR.LeftValue = 0.5;
```

```
DataR.FixRight =1;
DataR.RightValue = 0;
%Neumann
DataR.FixFluxesLeft = 0;
DataR.LeftFluxes = 0;
DataR.FixFluxesRight = 0;
DataR.RightFluxes = 0;

% Dirichlet-Neumann
HeatProblemR = HP_Initialize(DataR);
HeatProblemL = HP_Initialize(DataL);

u = 0;
w = 0.05;
u_star = u;
while true
%Problem R
    DataR.LeftValue = u(end);

    HeatProblemR = HP_Initialize(DataR);
    HeatProblemR = HP_Build(HeatProblemR);
    HeatProblemR = HP_Solve(HeatProblemR);

%Problem L
    DataL.RightFluxes = -HeatProblemR.Solution.FluxesLeft;

    HeatProblemL = HP_Initialize(DataL);
    HeatProblemL = HP_Build(HeatProblemL);
    HeatProblemL = HP_Solve(HeatProblemL);

    u_star = [u_star, HeatProblemL.Solution.uRight];
    if length(u) < 4
        w = 1;
    else
        w =[w, (u(end - 1) - u(end)) / (u(end-1)-u(end)+u_star(end)-u_star(end-1))];
    end
        u = [u, u_star(end)*w(end) + (1-w(end))*u(end)];

if abs(u(end) - u(end-1)) < 1E-10
   break
end
HP_Plot(HeatProblemL,1);
HP_Plot(HeatProblemR,1);
```

```
end
HP_Plot(HeatProblemL,1);
HP_Plot(HeatProblemR,1);
title('Solution using Aitken relaxation scheme w=0.5 (\kappa_1=1/100, \kappa_2=1)')
x = 1:length(u);

figure(2)

semilogy(x, abs(u-u(end)))
xlabel('Iteration')
ylabel('|Error|')
title('Convergence using Aitken relaxation scheme (w=0.5)')
grid on
```