Universitat Politècnica de Catalunya

Coupled Problems

Master's Degree in Numerical Methods in Engineering

Iterative schemes for coupling in space

Authors:
Pau Márquez

Supervisor:
Prof. J. Baiges

Academic Year 2019-2020

Contents

1 Task 1

This first task is oriented to study which is the effect of the problem variables when solving the one-dimensional problem $k\frac{\partial^2 u}{\partial x^2} + f = 0$ by numerical methods. Specifically, it will be solved in a unique domain with 100 elements. For this, the two boundaries are fixed with a value of zero.

1.1 Effect of changing the thermal diffusion coefficient

The results are shown in Fig. 1. As clearly could be expected, if a higher diffusion coefficient κ is used, the gradient of temperature will have to be lower in order to reach the same source term. In other words, as the problem is symmetric, the temperature will experience a higher diffusivity and thus the temperature maps reflect lower values throughout the domain.

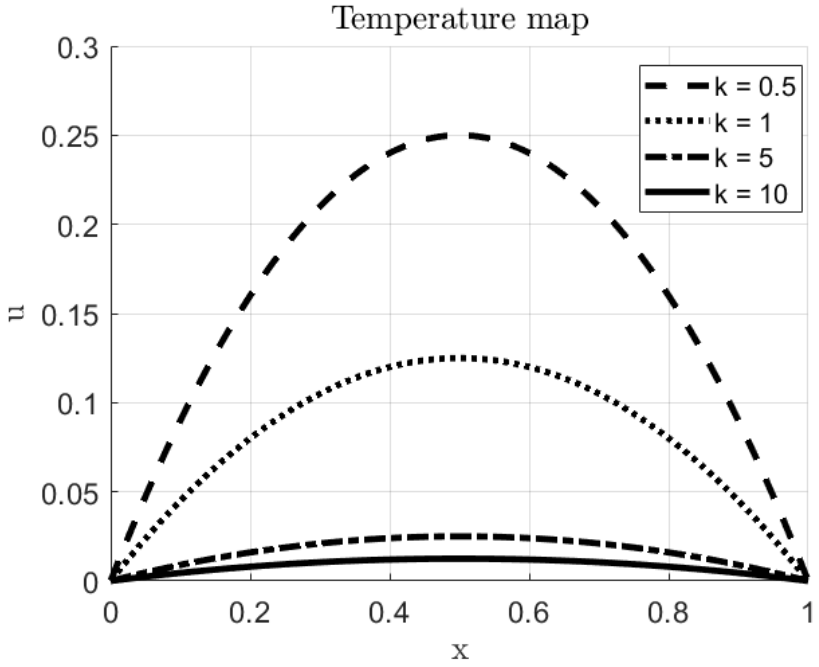


Figure 1: Map of temperatures with different values of κ.

1.2 Effect of changing the source term value

The result is shown in Fig. 2. Here, the same principle can be applied. If the conductivity is the same, the gradient of the temperature has to be higher in order to reach a higher source term in the equation, and a higher value of the solution is seen throughout the domain if a higher source term is added, as is would be expected from the analytical solution.

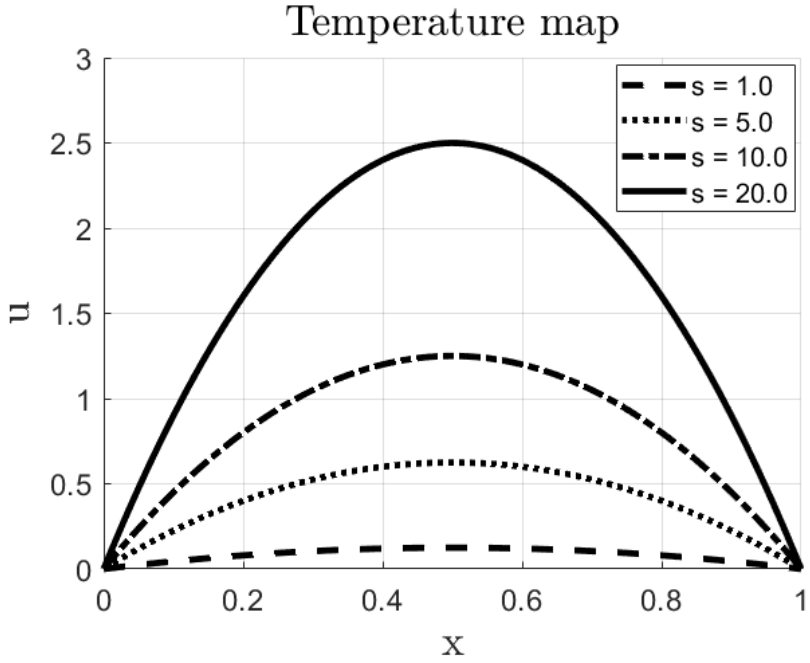


Figure 2: Map of temperatures with different values of source term.

1.3 Convergence rate of the error

Here it will be shown the evolution of the error as a function of the mesh size h. The analytical solution at the nodes may be computed resorting to the solution of the differential equation:

$$u(x) = \frac{f(1-x)x}{2k} \tag{1}$$

In particular, the analyical value is computed at the maximum heat value in the domain, that is, $x = 0.5$. Then, after solving this example with different mesh sizes h, it can verified that the evolution of the infinite norm, $e = |u(0.5) - u_h(0.5)|$ as a function of h, follows the trend in Fig. 3. Indeed, the decrease of the error with mesh size in a logarithmic plot should be a line. The exact value is $u(0.5) = 0.125$.

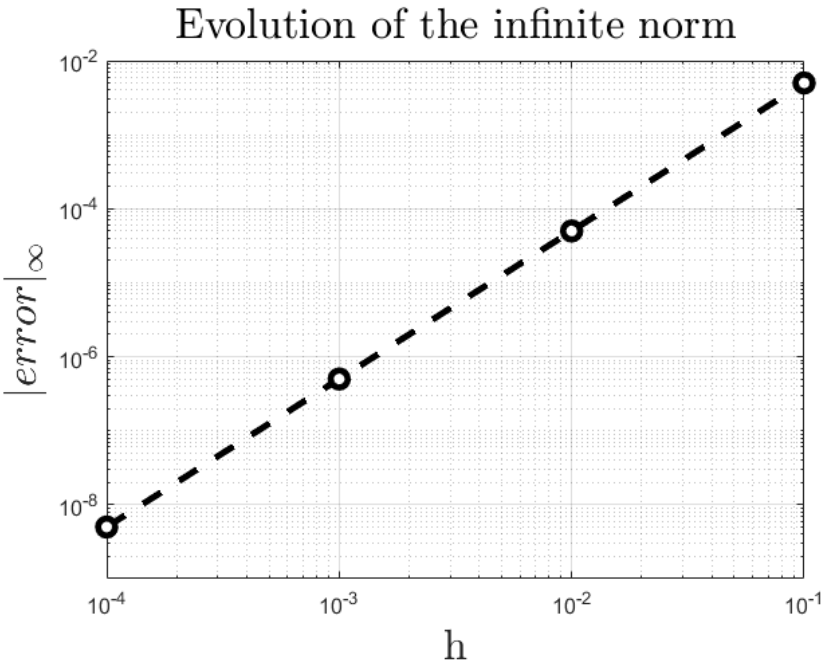


Figure 3: Logarithmic plot showing the linear reduction of the error with mesh size.

2 Task 2

2.1 Two independent heat transfer problems

This is the first case in which we consider two sub-domains. When this is done, care must be taken at the interface, where appropriate transmission conditions must be set if we want a coupled problem. This is an example of what happens if such transmission conditions are not set, or, in other words, if the two problems are solved independently regardless of the interface. Indeed, the only boundary conditions are set at the extreme values $u(0) = 0, u(1) = 0$, therefore observing a discontinuous solutions at the interface as there is no communication between both solvers whatsoever. The results are in Fig. 4.

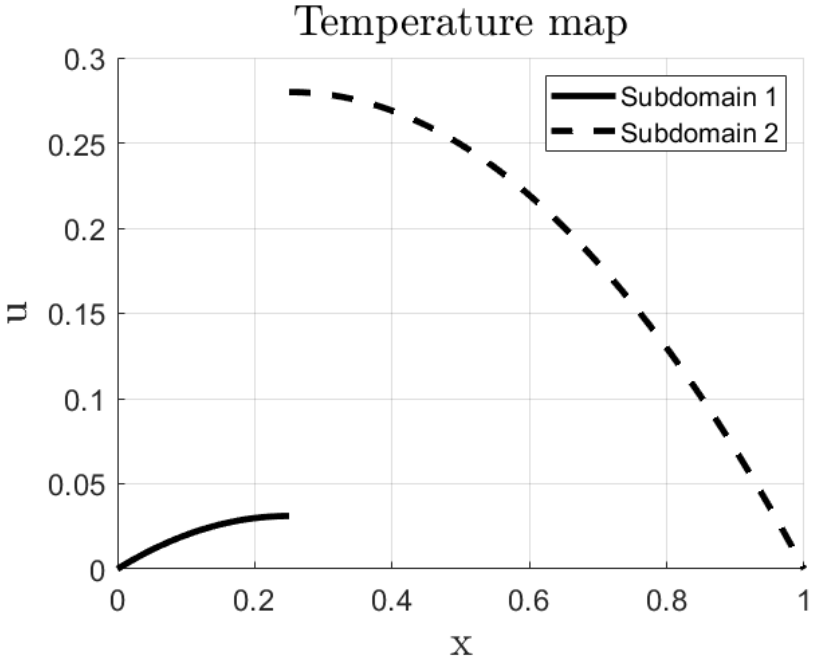


Figure 4: Outcome when solving two independent problems.

3 Task 3

3.1 Monolithic implementation on both sub-domains

In a monolithic implementation of the problem, the system is assembled globally so that there is only one coefficient matrix and one right-hand side matrix. The code takes the contributions from each subdomain and assembles each matrix K and each matrix F into a global matrix named K_{mono} and F_{mono} which are used to obtain the global solution vector U_{mono}, which contains the solution of both sub-domains. Then, as the problem at hand is linear, the solver can be a simple Matlab backslash, $K_{mono} \backslash F_{mono} = U_{mono}$, and there is no need to use an iterative solver, therefore the solution of the global matrix is computed directly.

If both sub-domains have the same properties as the single domain in task 1, then the results are expected to be the same, as the transmission conditions are met at the interface. This is seen in Fig. 5.

3.2 Effect of modifying the conductivity

When the conductivity is modified in one of the dub-domains, the other sub-domain will see its solution modified due to the coupling of the problem. In particular, if the sub-domain increases κ from 1 to 3, then it will be seen a change in the shape of the curve, since one material will dissipate heat more than the other. In the second sub-domain it is seen the parabolic shape with higher temperature distribution product of the lower κ and in the first sub-domain the slope is smaller compared to the case in which $\kappa = 1$, as mentioned before. Results are seen in

Fig. 5.

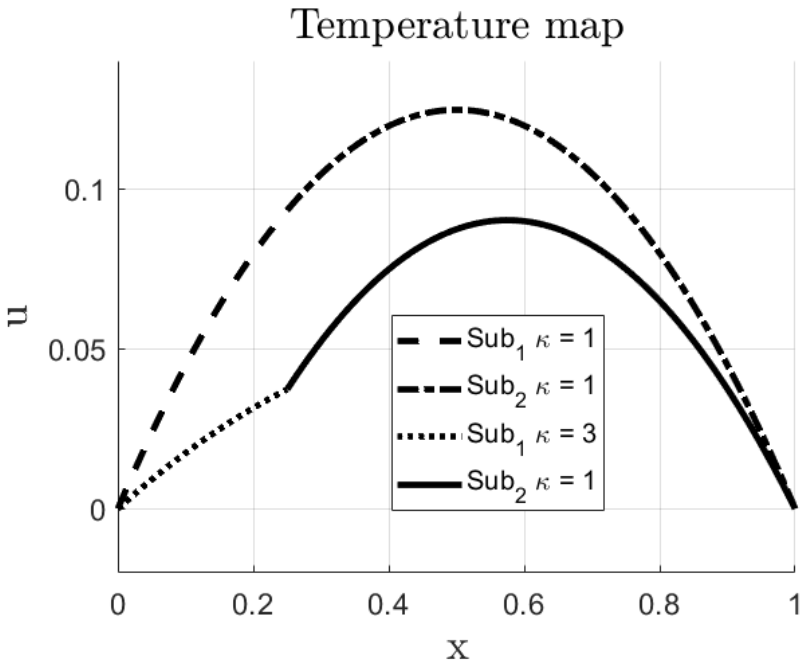


Figure 5: Different κ parameters and their effect on the monolithic approach.

4 Task 4

4.1 Convergence of the iterative scheme

The iterative scheme developed will have the following patters for each iteration:

1. Both problems are initialized and built with the updated problem Data.
2. The first sub-domain is solved. This first sub-domain will have a prescribed Neumann bc at the interface, therefore it has fixed the value at the left extreme and the flux at the right extreme. When the solution is computed for this domain, the solution at the interface is assigned to the prescribed value for the Dirichlet boundary of the second-subdomain.
3. The second sub-domain is solved with only Dirichlet boundary conditions, and the flux obtained at the left extreme is assigned as the Neumann flux for the right boundary of the first sub-domain.
4. The convergence is checked and if there is no convergence the cycle is repeated.

The convergence of this scheme should lead to the monolithic value if the same parameters are used. This is shown in Fig. 6, where after 11 iterations of the Dirichlet-Neumann scheme, the value at the middle converges to the Monolithic value. The convergence of the error will be

studied in the next section, where it is compared with different values of the κ parameter. The error chosen here is the relative, computed as (for the first or second subdomain),

$$e = 100 \frac{u_\Gamma^i - u_\Gamma^{i-1}}{u_\Gamma^{i-1}} \tag{2}$$

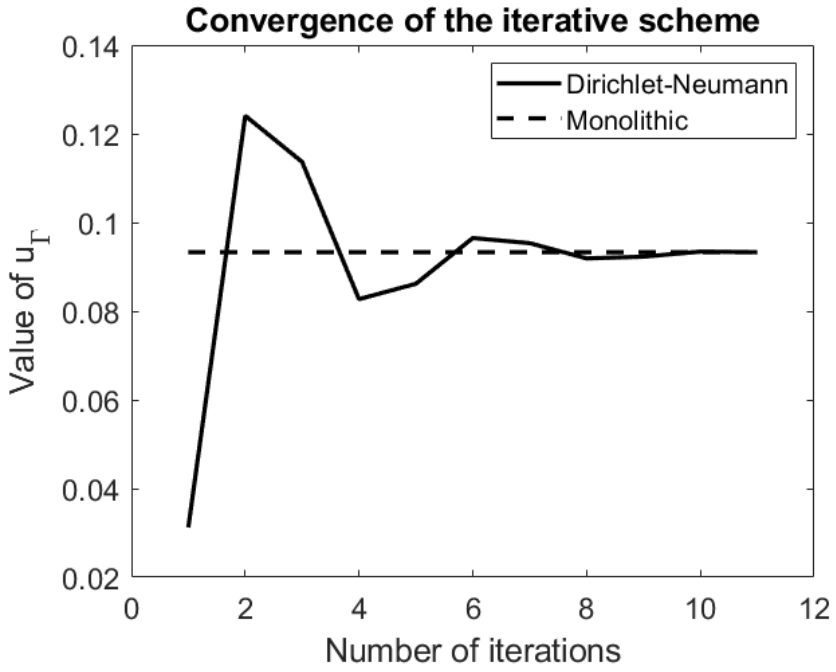


Figure 6: Convergence of the iterative scheme towards the monolithic value.

4.2 Augmented κ at sub-domain 1

When $\kappa_1 = 100, \kappa_2 = 1$, the convergence rate is seen to be approximately five times to that of $\kappa_1 = \kappa_2 = 1$ for a given tolerance, as seen in Fig. 7. The temperature map is seen in Fig. 8a, and at the first sub-domain the average temperature is close to zero given the high diffusivity of the solution.

4.3 Diminished κ at sub-domain 1

In this case it is interesting to see that the convergence rate for this problem, when $\kappa_1 = 0.01, \kappa_2 = 1$ is non-existing, as seen in Fig. 7. Whereas for moderate to high values of κ_1 was reached, the solution was reached in a number of iterations, here the relative error is not seen to reduce with the number of iterations. The temperature map for this problem subscribes this aspect, as there is no continuity of the solution at the interface, and a very high value of the temperature is observed throughout the first sub-domain, product of its low diffusivity.

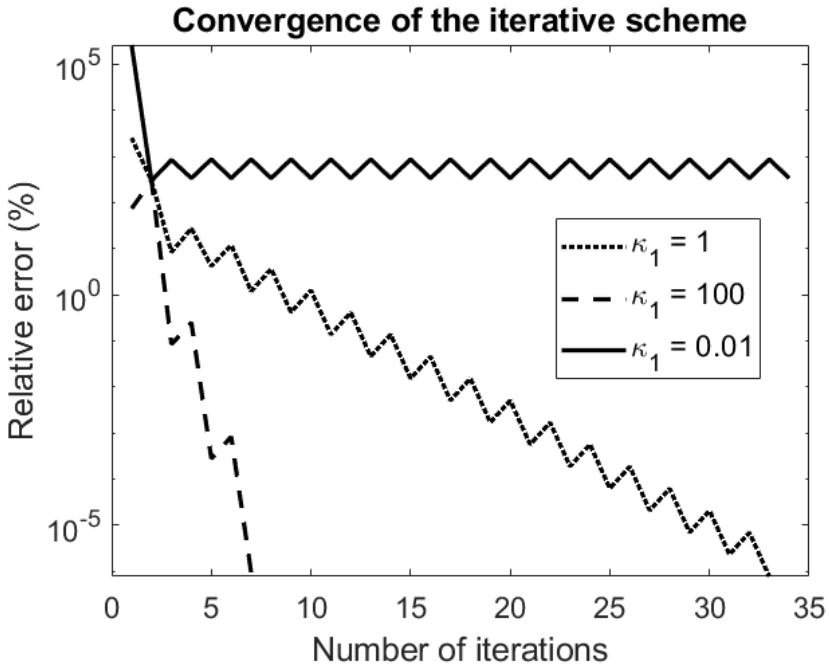
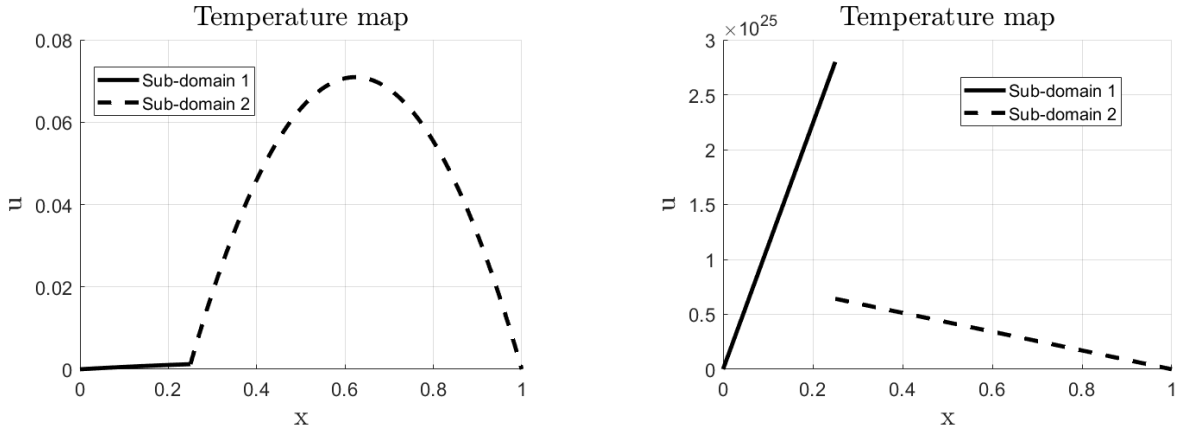


Figure 7: Convergence rates for different values of κ.



(a) Temperature map for $\kappa_1 = 100$. *(b) Temperature map for $\kappa_1 = 0.01$.*

Figure 8: Solution values for different values of κ on the sub-domains.

4.4 Stability of the coupling scheme

One may ask the question of whether when increasing 100 times κ_1 convergence was reached but not when it is diminished 100 times. If the theory of non-overlapping domain decomposition methods is employed on the homogeneous coupling of a heat transfer problem, then we reached that the following transmission conditions must hold point-wise

$$\begin{cases} u_1 = u_2 \\ \kappa_1 \frac{\partial u_1}{\partial n_1} = -\kappa_2 \frac{\partial u_2}{\partial n_2} \end{cases} \tag{3}$$

That it, there must be continuity of solution and fluxed. It has been seen that, for a Dirichlet-Neumann method, the solution of the flux on the left boundary of the second sub-domain is imposed as the Neumann flux on the right boundary of the first sub-domain. The stability problem remains when a flux is imposed on a domain with very low κ, which means that the gradient of temperature that this domain will hold will be very high. This is actually what has been observed in Fig. 8b, where the distribution of temperature is very pronounced. This is different as in the first case, when a flux is imposed in a domain with very high diffusivity, which is no problem as the gradient of temperatures will be small. The problem thus is that in the first iteration, the solution goes from zero to a very high value, and therefore instabilities appear and there is no convergence.

In order to overcome this issue, when such differences exist on the diffusivity of the sub-domains, the sub-domain with higher κ is the in which the Neumann condition has to be imposed.

5 Task 5

5.1 Fixed relaxation parameter

To overcome the instabilities observed in the previous section, a relaxation scheme can be implemented, which ensures that at each iteration the jump of the value of the solution at the interface is progressive and not drastic. For this, we say

$$u^i_{\Gamma 21} = w u^i_{\Gamma 12} + (1 - w) u^{i-1}_{\Gamma 21} \tag{4}$$

We will see which is the effect of such an scheme on a diverging problem $\kappa_1 = 0.01, \kappa_2 = 1$, and a problem which converges with no difficulty, $\kappa_1 = \kappa_2 = 1$. If we analyze Fig. 9, we see that, for the convergent case, the best results in terms of number of iterations to achieve a relative error of $1e-4$ is when we set $w = 0.5$, as it takes 17 iterations. Then, when setting w for other values, $w = 0.75$ takes 21 iterations, $w = 1$ takes 25 iterations and eventually $w = 0.25$ takes 29 iterations, which is the worst case. $w = 0$ gives interesting results, as convergence is achieved very fast but there is no continuity at the interface, which would be similar to solve two independent problems.

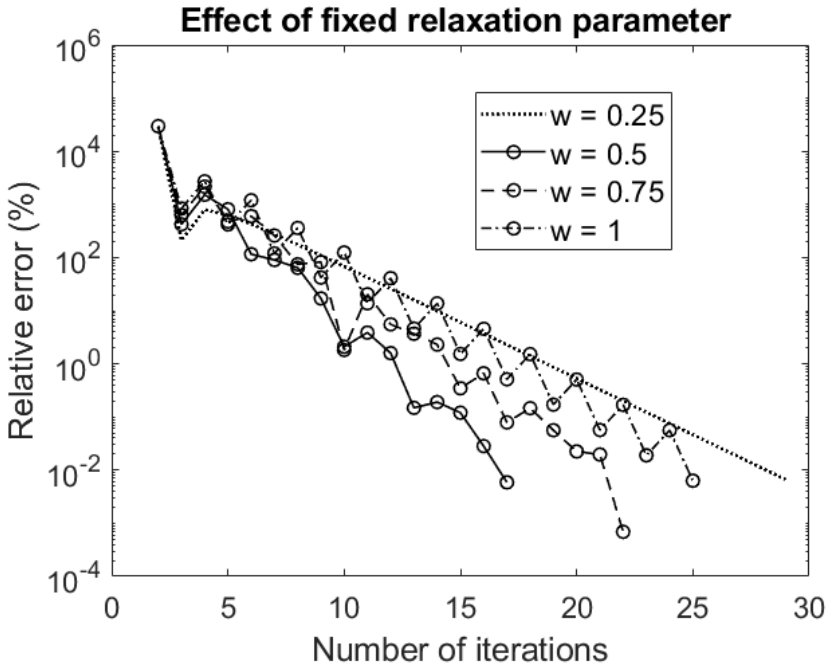


Figure 9: Convergence rates for different w parameters and $\kappa_1 = \kappa_2 = 1$.

Now let's see which is the effect of w on the divergent problem that has been seen. It is clear that the value of w has to be very small in order to ensure a progressive increase on the value at the interface. With that, interesting results are shown in Fig. 10. For $w = 0.01$ convergence is achieved after 32 iterations, although for smaller values $w = 0.005$, it takes more (69 iterations), and for higher values $w = 0.02$, it takes also more (74 iterations). For even higher values $w = 0.03$, it starts diverging.

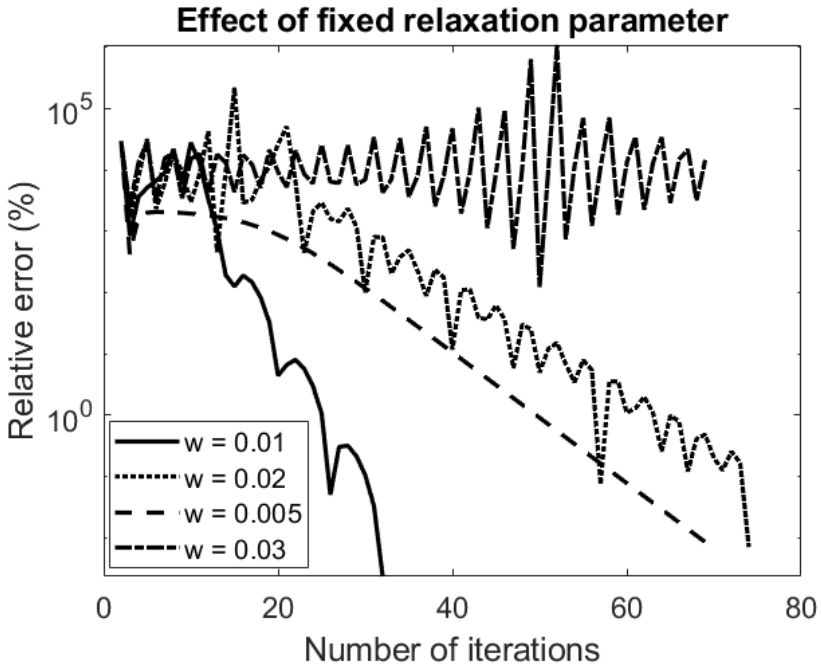


Figure 10: Convergence rates for different w parameters and $\kappa_1 = \kappa_2/100 = 0.01$.

Another interesting thing is that, in both Figures it is observed that, although very small values of w may take for iterations, the convergence of the error is smooth and do not present spurious oscillations as in the other cases.

5.2 Aitken relaxation scheme

The Aitken relaxation scheme provides a method to calculate w at each iteration, therefore it will not be fixed. It is calculated as

$$w = u^i_{\Gamma 21} = \frac{u^{i-1}_{\Gamma 21} - u^{i-2}_{\Gamma 21}}{u^{i-1}_{\Gamma 21} - u^{i-2}_{\Gamma 21} + u^{i-1}_{\Gamma 12} - u^{i-2}_{\Gamma 12}} \tag{5}$$

Which implies that we need at least two iterations in order to start computing this value. Before that, w can simply be 0.5. Let's see which is the effect on the previous problems, compared to the cases of no relaxation, $w = 0.5$ all the time and Aitken scheme. We see, for the case $\kappa_1 = \kappa_2 = 1$, the Aitken scheme achieves results in 16 iterations, with a smooth convergence. As for the case for $\kappa_1 = \kappa_2/100 = 0.01$, the Aitken scheme reaches convergence in less iterations and smoothly.

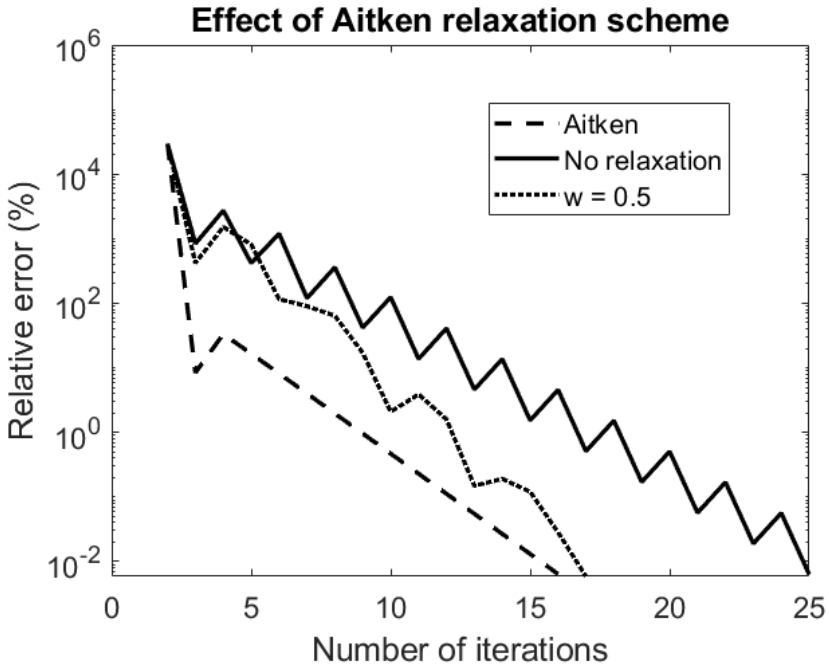


Figure 11: Different relaxation schemes for equal conductivity in the sub-domains.

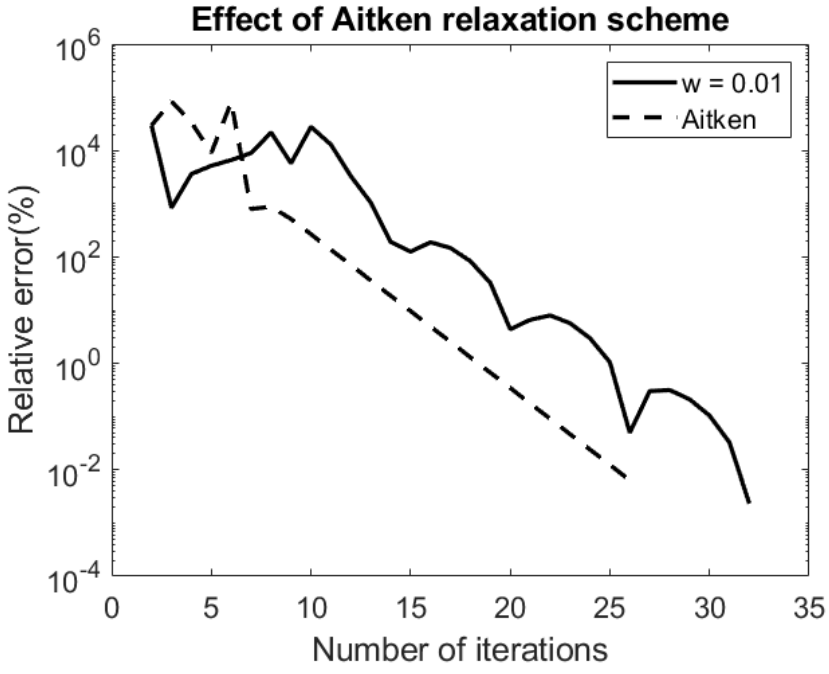


Figure 12: Different relaxation schemes for adiminished conductivity in the first sub-domain.

Aitken scheme also provides a convergence of the error which is very smooth compared with the other cases.

A Appendix: Codes

A.1 Script to solve task 1

```
clc; close all;

%% Solve task 1, part c)
Num_domains = 1;
Domain = [0 1];
legendText = {};
exact = -0.5*1*(0.5-1)*0.5;

vec = [10000 1000 100 10]; %this vector contains the values for the
%parameter to be studied, in this case the number of nodes

for i = 1:4
    Num_elem = vec(i);
    vec_h(i) = 1/vec(i);
    kappa = 1;
    source = 1;
    Fix_boundary = [1 1 0 0]; %[Dir_left Dir_right Neu_left Neu_right]
    Value_boundary = [0 0 0 25]; %[Dir_left Dir_right Neu_left Neu_right]

    Data = HP_Define(Domain,Num_elem,kappa,source,Fix_boundary,...
        Value_boundary);

    HeatProblem = HP_Initialize(Data); %Initializing
    HeatProblem = HP_Build(HeatProblem); %Building
    HeatProblem = HP_Solve(HeatProblem); %Solving
    vec_e(i) = abs(exact - HeatProblem.Solution.U(vec(i)/2));
end

HP_Plot_error(vec_h,vec_e);%Discomment to plot error
```

A.2 Script to solve task 2

```
clc; close all; clear all;
%% Solve task 2
Num_domains = 2;
Domains = [0 0.25; 0.25 1];
Num_elem = [100 100];
kappa = [1 1];
source = [1 1];
Fix_boundary = [1 0 0 0; 0 1 0 0]; %[Dir_left Dir_right Neu_left Neu_right]
Value_boundary = [0 0 0 0; 0 0 0 0]; %[Dir_left Dir_right Neu_left Neu_right]

Data1 = HP_Define(Domains(1,1:2),Num_elem(1),kappa(1),source(1),...
    Fix_boundary(1,1:4),Value_boundary(1,1:4));
Data2 = HP_Define(Domains(2,1:2),Num_elem(2),kappa(2),source(2),...
    Fix_boundary(2,1:4),Value_boundary(2,1:4));

HeatProblem = HP_Initialize(Data1); %Initializing
HeatProblem2 = HP_Initialize(Data2); %Initializing
HeatProblem = HP_Build(HeatProblem); %Building
HeatProblem2 = HP_Build(HeatProblem2); %Building
HeatProblem = HP_Solve(HeatProblem); %Solving
HeatProblem2 = HP_Solve(HeatProblem2); %Solving

HP_Plot(HeatProblem,1);
HP_Plot(HeatProblem2,1);
```

A.3 Script to solve task 3

```
clc; close all; clear all;
%% Solve task 3
Num_domains = 2;
Domains = [0 0.25; 0.25 1];
Num_elem = [100 100];
kappa = [100 1];
source = [1 1];
Fix_boundary = [1 0 0 0; 0 1 0 0]; %[Fix_left Neu_left Neu_right]
Value_boundary = [0 0 0 0; 0 0 0 0]; %[Dir_left Dir_right Neu_left Neu_right]

Data1 = HP_Define(Domains(1,1:2),Num_elem(1),kappa(1),source(1),...
    Fix_boundary(1,1:4),Value_boundary(1,1:4));
Data2 = HP_Define(Domains(2,1:2),Num_elem(2),kappa(2),source(2),...
    Fix_boundary(2,1:4),Value_boundary(2,1:4));

HeatProblem = HP_Initialize(Data1); %Initializing
HeatProblem2 = HP_Initialize(Data2); %Initializing
HeatProblem = HP_Build(HeatProblem); %Building
HeatProblem2 = HP_Build(HeatProblem2); %Building
HeatProblem = HP_Solve(HeatProblem); %Solving
HeatProblem2 = HP_Solve(HeatProblem2); %Solving

%Solve and plot
[HeatProblem,HeatProblem2] = HP_SolveMonolithic(HeatProblem,HeatProblem2);
HP_Plot(HeatProblem,1);
HP_Plot(HeatProblem2,1);
HeatProblem.Solution.uRight
```

A.4 Script to solve tasks 4 and 5

```
clc; clear all;
%% Solve task 4

Num_domains = 2;
Domains = [0 0.25; 0.25 1];
Num_elem = [100 100];
kappa = [1/100 1];
source = [1 1];
Fix_boundary = [1 0 0 1; 1 1 0 0];
%[Fix_left Fix_right Fix_fluxes_left Fix_fluxes_right]
Value_boundary = [0 0 0 0; 0 0 0 0]; %[Dir_left Dir_right Neu_left Neu_right]

Data1 = HP_Define(Domains(1,1:2),Num_elem(1),kappa(1),source(1),...
    Fix_boundary(1,1:4),Value_boundary(1,1:4));
Data2 = HP_Define(Domains(2,1:2),Num_elem(2),kappa(2),source(2),...
    Fix_boundary(2,1:4),Value_boundary(2,1:4));

iter = 2;
tol = 1e-4;
u_Gamma2 = 0; %Value at the interface for subdomain 2
u_Gamma1 = 0; %Value at the interface for subdomain 1
vect_error1 = [];
w = 1;
relaxation = 2; %2 for Aitken, 1 otherwise

while iter < 70
    HeatProblem = HP_Initialize(Data1); %Initializing
    HeatProblem2 = HP_Initialize(Data2); %Initializing
    HeatProblem = HP_Build(HeatProblem); %Building
    HeatProblem2 = HP_Build(HeatProblem2); %Building

    %Solve problem in Sub-domain1
    HeatProblem = HP_Solve(HeatProblem); %Solving
    u_Gamma1(iter) = HeatProblem.Solution.uRight;

    %Compute relaxation parameter for Aitken scheme
    if iter > 3
        if relaxation == 2
            wu = u_Gamma2(iter-2) - u_Gamma2(iter-1);
            wd = u_Gamma2(iter-2) - u_Gamma2(iter-1) + ...
                u_Gamma1(iter) - - u_Gamma1(iter-1);
```

```
            w = abs(wu/wd);
        end
    end

    %Apply Dir. bc at the interface in S-d2
    Data2.LeftValue = w*HeatProblem.Solution.uRight + (1-w)*u_Gamma2(iter-1);

    u_Gamma2(iter) = Data2.LeftValue;

    %Solve problem in Sub-domain2
    HeatProblem2 = HP_Solve(HeatProblem2); %Solving
    %Apply Neu. bc at the interface in S-d1
    Data1.RightFluxes = -HeatProblem2.Solution.FluxesLeft;

    % Check convergence
    delta1 = abs(abs(HeatProblem.Solution.uRight - u_Gamma1(iter-1))/...
    u_Gamma1(iter-1))*100;
    vect_error1 = [vect_error1, delta1];
    u_Gamma1(iter) = HeatProblem.Solution.uRight;
    fprintf('Increment=%8.6e \n',delta1);
    if delta1 < tol
        fprintf('\nConvergence achieved in iteration number %g\n',iter);
        break
    else
        iter = iter + 1;
    end

end

HP_Plot(HeatProblem,1);
HP_Plot(HeatProblem2,1);
figure
plot(1:iter,vect_error1*100);
figure;
semilogy(1:iter-1,vect_error1*100);
```