COMPUTATIONAL MECHANICS TOOLS

Master of Science in Computational Mechanics/ Numerical Methods

Fall Semester 2015

Assignment 2: PDE-Toolbox

December 1st, 2015

Albert Capalvo Viladot

Task 1· Solve the problem and refine the initial mesh up to 4 times. Verify that the theoretical convergence order holds.

As it says on Matlab Pdetool user's manual the expected error is of order $O(h^2)$ so the convergence rate is quadratic.

To verify that this holds, a mesh convergence analysis is carried out refining the mesh several times and computing an error associated to each mesh. In this case the error is defined as the maximum nodal difference between the numerical and analytical solutions, and the mesh size h associated to each mesh is obtained through averaging all element sizes (see function definition in Annex I).

Considering that when h is small enough the mesh size and the error are related by a function of the form:

$$e(h) = n\ h\text{^}m$$

Where m is the order of convergence and n is a constant independent of h.

We can apply the logarithm of both sides, obtaining:

$$\log\big(e(h)\big) = m\log(h) + \log\,(n)$$

Finally plotting $log(e)\ vs\ log(h)$, the order of convergence m can be obtained computing the slope. As a requisite is that h is small enough the last slope is the one considered for the order of convergence.

Solving the case for t=5, the obtained rate is 1.84 (figure 1) which would verify the theoretical order of convergence 2 as it refers to an optimal rate difficult to obtain while doing simulations.

Note: If instead of the average size we get the mesh size on the upper corners (where the error is maximum; figure 2), we get a convergence order much closer to 2. However using the average size it is still considered to be more appropriate. It would also be interesting to use another measure of the error that takes into consideration the error distribution among the entire domain.

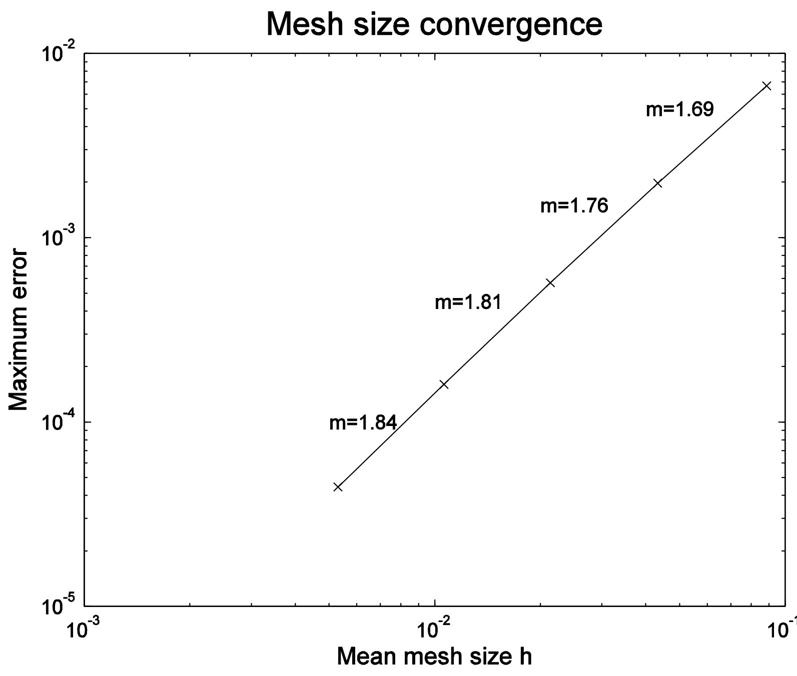

Figure 1. Mesh size convergence

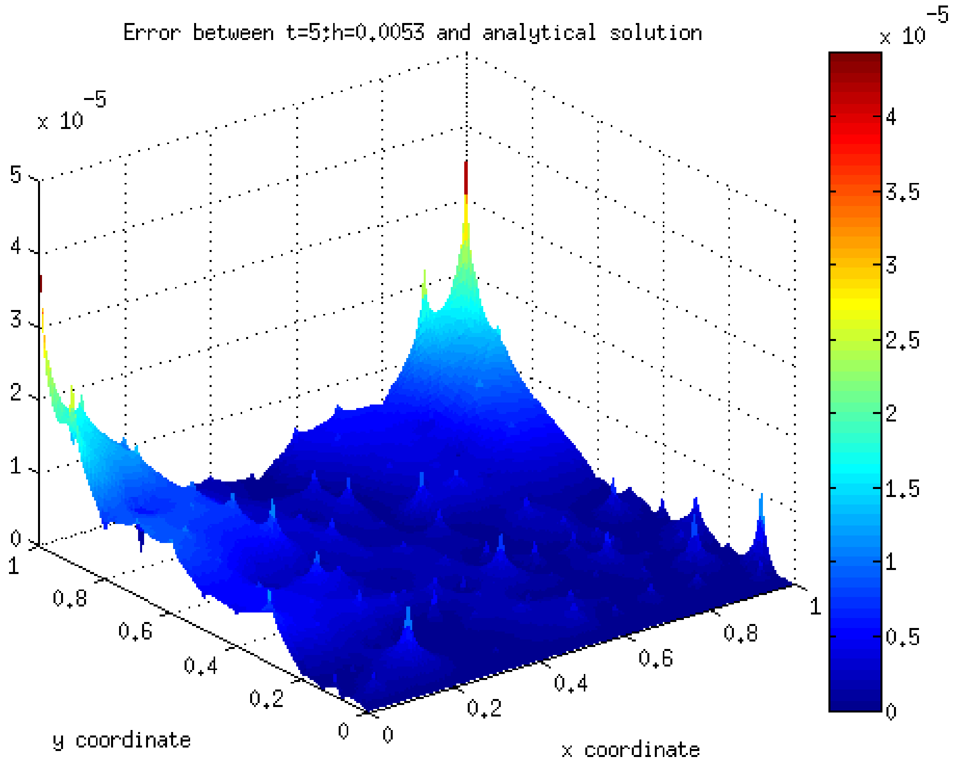

Figure 2. Error between simulation at t=5; h=0.053 and analytical solution

Task 2· How is the solution affected when we modify the final time?

As the source term and some boundary conditions depend on the time, it is expected that the solution also does. To check this out, the solutions at times t=1 and t=10 are plotted (figure 3).

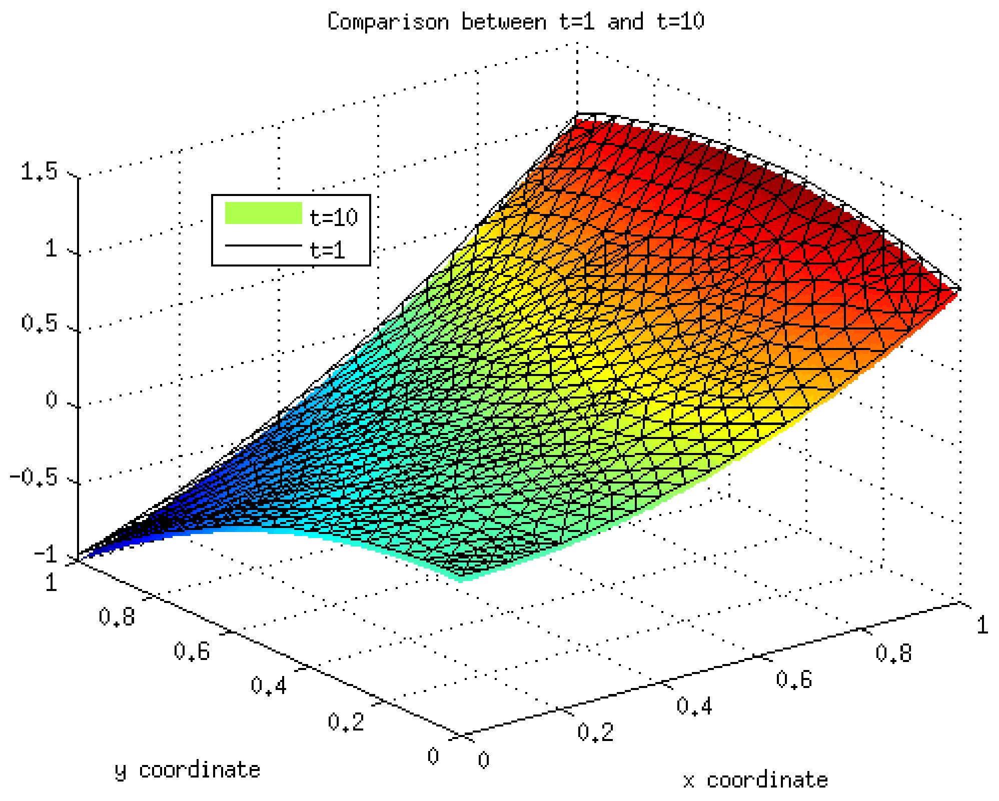

Figure 3. Comparison between t=1 And t=10

Computing the difference between the solutions at both times it results that in each node there is a difference of ~ -0.0497878. This value matches the one that we could obtain from the analytical solution $e^{-3\cdot 10} - e^{-3\cdot 1}$.

Task 3· We are interested in obtaining the solution at time $t_{end} = 50$. Find a more efficient manner to solve this problem. You do not need to prove the equivalence mathematically, but you need to provide numerical evidence of the new method.

Looking at the time dependant terms commented on previous task, it can be seen that they decrease exponentially as time advances; meaning that for advanced enough times, those terms can be neglected, and therefore the problem becomes a steady state equilibrium.

To verify that the transient solution converges to a steady state solution, a transient error evolution analysis is carried out, where the transient error is computed as

$$e_{tr}(t) = \sum_{i=1}^{N} \frac{|u(i,t) - u(i,t-1)|}{N}$$

It is chosen to compute the addition of the absolute value of the nodal differences because is a restrictive criteria that ensures that the transient error converges to 0 for all the nodes.

In figure 4 we can see that in fact the transient error reduces as time advances, however below a certain value of ~ 10^{-6} the evolution of the error using the two different meshes differs from the rate expected coming from the analytical solution. This fact could be explained as the consequence of the numerical error that comes from the mesh spacing and the solver tolerance, and taking into consideration that we are trying to capture differences not far from the actual accuracy of the solution.

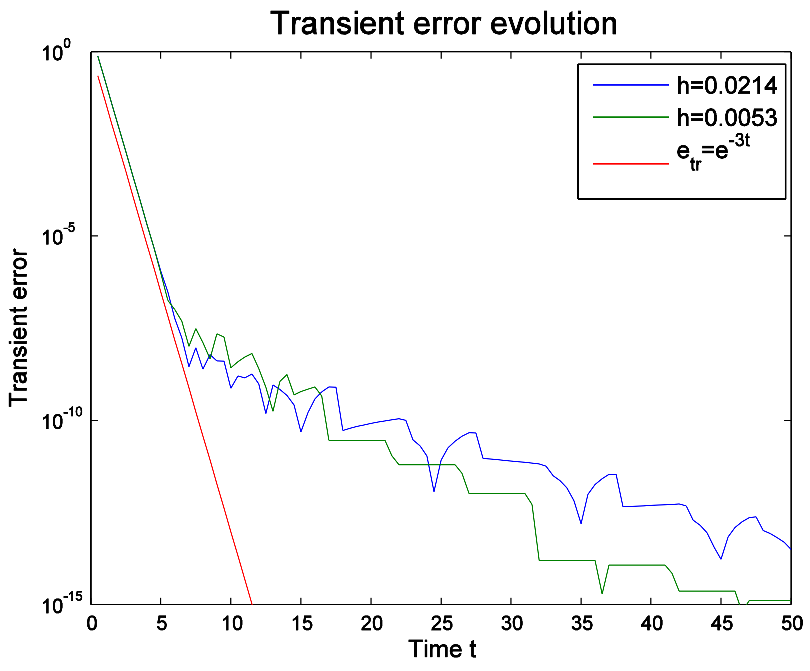

Figure 4. Transient error evolution

In any case, considering that for t >5 the transient error drops below 10^-6 we can say that for t>5 the solution reaches a steady state equilibrium.

From all the exposed above it derives that the most effective way to compute the solution for t=50, is solving the steady state problem. To do so, some modifications are needed:

- Eliminate the temporal terms in the source term and boundary condition.
- Eliminate the initial condition.
- Change the pde type from parabolic to elliptic so the time derivate vanishes.

As a measure for comparing the solution obtained at t=50 using a transient simulation and a steady state simulation it is decided to compute the difference of the integral of both solutions (using Matlab's command *convhull*). The result is 2.43e-10, so it can be said that following both ways the same solution is obtained, although using a steady state simulation is considerably cheaper computationally speaking.

Annex I

```
function [h] = h_size(p,e,t)

j=0;h=0;

for i=1:length(t)

    w1=0.5;     w2=0.5;     w3=0.5;

    n1=t(1,i);     n2=t(2,i);     n3=t(3,i);
    % Computing distances between nodes in triangle i
    d1=sqrt( (p(1,n2)-p(1,n1))^2 + (p(2,n2)-p(2,n1))^2 );  %% n1 to n2
    d2=sqrt( (p(1,n3)-p(1,n2))^2 + (p(2,n3)-p(2,n2))^2 );  %% n2 to n3
    d3=sqrt( (p(1,n1)-p(1,n3))^2 + (p(2,n1)-p(2,n3))^2 );  %% n3 to n1

    % If two elements are part of boundary, they are assigned weigh=1
    for k=1:length(e)
        if (e(1,k)==n1 && e(2,k)==n2) || (e(1,k)==n2 && e(2,k)==n1)
            w1=1;
        end
        if (e(1,k)==n2 && e(2,k)==n3) || (e(1,k)==n3 && e(2,k)==n2)
            w2=1;
        end
        if (e(1,k)==n3 && e(2,k)==n1) || (e(1,k)==n1 && e(2,k)==n3)
            w3=1;
        end
    end

   h=h+w1*d1+w2*d2+w3*d3;
   j=j+3 ;
end

h=2*h/j;

end
```