Homework 1 - 2D Transfinite Interpolation

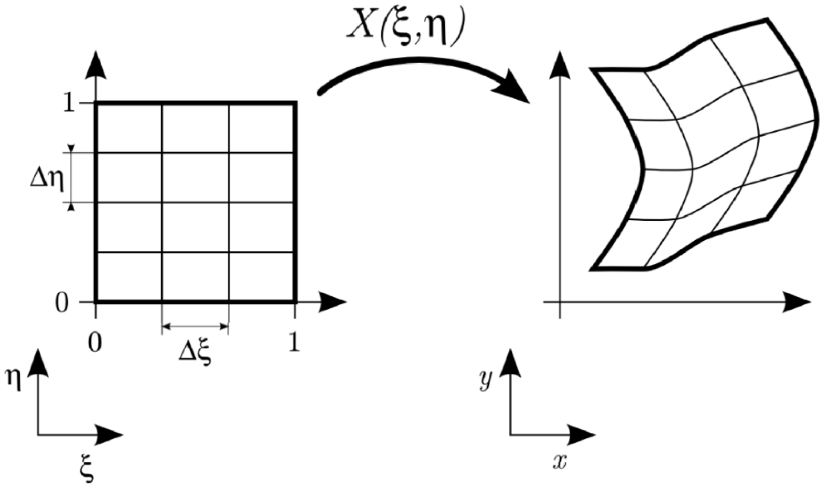
Computational mechanics tools

Andino Saint Antonin

Master in Numerical Methods in Engineering

Universitat Politècnica de Catalunya

Oct 2020



Contents

List of Figures

List of Tables

1 Assignment: 2D transfinite interpolation

Assignment

We are requested to implement in MatLab a 2D transfinite interpolation (from now TFI) workflow. The transfinite mapping from a computation domain (ϵ, η) to a physical domain (x, y) is defined as the Boolean sum of the two interpolations $\mathbf{U}$ and $\mathbf{U}$:

$$\mathbf{X}(u_I, v_J) = \mathbf{U}(u_I, v_J) \oplus \mathbf{V}(u_I, v_J) = \mathbf{U}(u_I, v_J) + \mathbf{V}(u_I, v_J) - \mathbf{UV}(u_I, v_J). \tag{1}$$

for $I = 1, 2...., M$ and $I = 1, 2...., M$, with the following definitions applying

$$\begin{aligned}\mathbf{U}(u_I, v_J) &= (1 - u_I)\mathbf{X}(0, v_J) + u_I\mathbf{X}(1, v_J)\\ \mathbf{V}(u_I, v_J) &= (1 - v_J)\mathbf{X}(u_I, 0) + v_I\mathbf{X}(u_I, 1)\end{aligned}$$

. and

$$\begin{aligned}\mathbf{UV}(u_I, v_J) = (1 - u_I)(1 - v_J)\mathbf{X}(0, 0) + (1 - u_I)v_J\mathbf{X}(0, 1) +\\ u_I(1 - v_J)\mathbf{X}(1, 0) + u_Iv_J\mathbf{X}(1, 1)\end{aligned}$$

The variables u and v constitute an intermediate domain, and are mapped from the domain (ϵ, η) using single-exponential functions

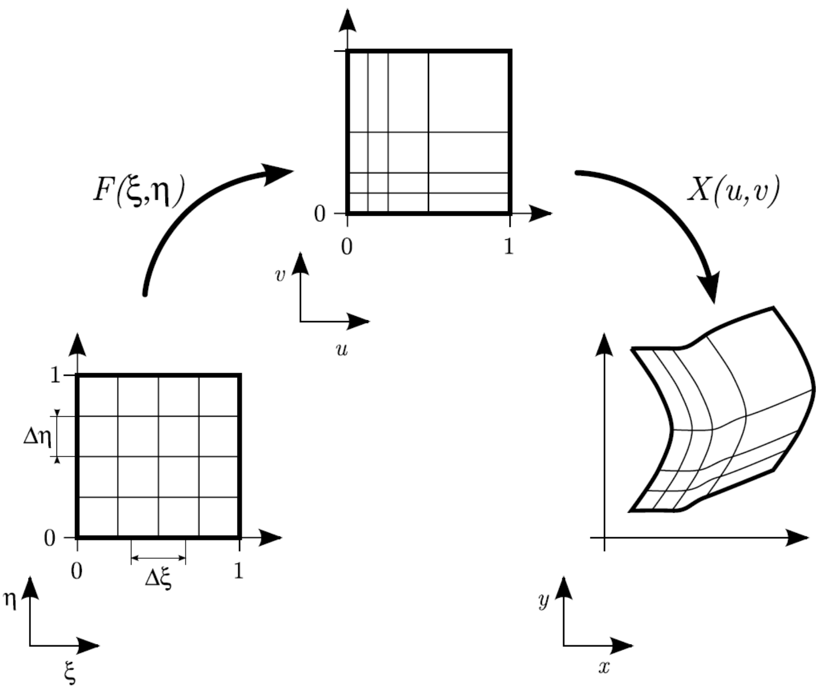


Figure 1: 2D transfinite interpolation with an intermediate domain

which will be used to increase the nodes' density in the desired positions as per figure 1. The mapping is

$$u = \frac{e^{A\epsilon - 1}}{e^{A-1}} \text{ and } v = \frac{e^{A\eta - 1}}{e^{A-1}} \tag{2}$$

. where the parameter A is chosen in order to increase the density of values on one end or the other of the domain, as shown

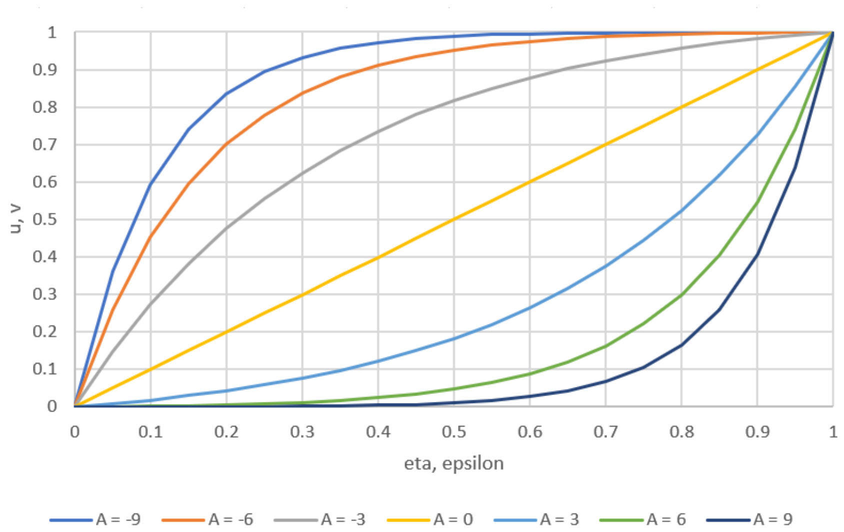


Figure 2: dependanve of u, v on the parameter A in intermediate mapping

The specific tasks for the homework are:

1. In file linearTFI.m write the code corresponding to functions: createInnerNodes, U, V, UV
2. In file gridControlSpacing.m write the code corresponding to function singleExp.
3. Generate a structured mesh using your application for:
   - a rectangular domain of height equals 4 and width equals 3 (example 1 in boundary.m file).
   - a quarter of circular ring of inner radii equals 4, outer radii equals 7 and angle 2 (example 2 in boundary.m file).
   - For both examples present the obtained mesh using A = 3 and A = 3 when function singleExp is used to concentrate nodes in the ϵ and η directions
4. Apply the developed application to a new geometry. To this end modify file boundary.m and create a new domain. Present three meshes concentrating nodes near different boundaries

2 MatLab Implementation

2.1 Write functions createInnerNodes, U, V, UV

The code that creates the inner nodes can be seen in Figure 3. A loop through computational domain variables (ϵ, η) creates first the intermediate domain inner nodes (u, v) using the grid control spacing function and then the physical domain nodes (x, y) using the Boolean sum defined above.

The terms of the Boolean sum are implemented as functions too. The code is simple enough, see Figure 4. Separate functions were written for U, V and the product UV as per the equations (1) shown above.

```matlab
function [phi]=createInnerNodes(phi)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Creates inner nodes
%   nOfChiElems, nOfEtaElems = Number of elements in each direction
%   phi = nOfChiNodes x nOfEtaNodes x 2 (temporary array)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
nOfChiNodes=size(phi,1);
nOfEtaNodes=size(phi,2);

% Compute computational coordinates chi and eta
chi=linspace(0,1,nOfChiNodes);
eta=linspace(0,1,nOfEtaNodes);

%Compute intermediate coordinates
for i=2:nOfChiNodes - 1
    % create the intermediate coordinates
    for j=2:nOfEtaNodes - 1
        [u,v]=gridControlSpacing(chi(i),eta(j));
        % create the physical coordinates
        phi(i,j,:) = U(u,j) + V(v,i) - UV(u,v);
    end
end

```

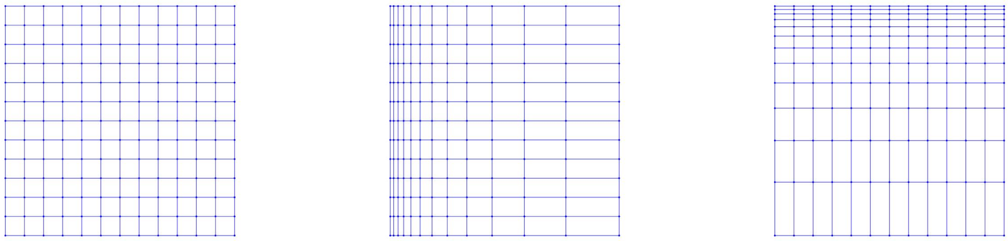
Figure 3: Matlab function that creates inner nodes

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  function U and V and UV for univariate blending
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [p]=U(u,j)
global phi
p(:,:) = (1-u)*phi(1,j,:) + u*phi(end,j,:);


function [p]=V(v,i)
global phi
p(:,:) = (1-v)*phi(i,1,:) + v*phi(i,end,:);

function [p]=UV(u,v)
global phi
p(:,:) = (1-u)*(1-v)*phi(1,1,:) + (1-u)*v*phi(1,end,:)
         + u*(1-v)*phi(end,1,:) + u*v*phi(end,end,:);

```

Figure 4: Matlab implementation of U, V and UV

2.2 Write function singleExp

The single exponential spacing is implemented also as a function using equation (2). The code caters separately for the case when $A = 0$ which would cause an error in the function otherwise.



(a) linear (b) denser on left (c) denser on top

Figure 6: Exponential spacing examples

```matlab
function psi=singleExp(psi_ini, A)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function to distribute point exponantially
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if A == 0
    psi = psi_ini;
else
    psi = (exp(A*psi_ini) - 1)/(exp(A) - 1);
end

```

Figure 5: Matlab single variable exponential mapping

2.3 Generate structured meshes

We tested the full program on multiple shapes. Beginning with the square physical domain, we tried $A = 3$ and $A = -3$ for both ϵ and η. Some of the results are presented in Figure 6

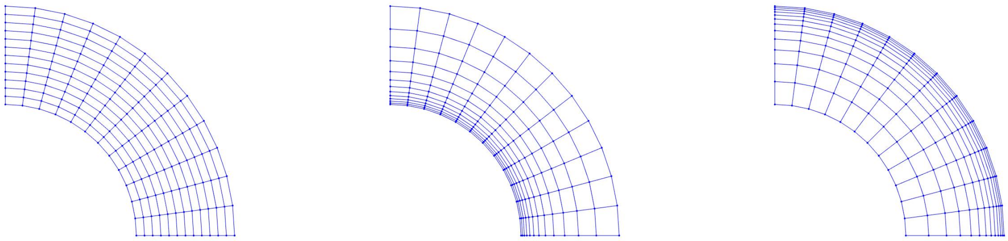
Then we implemented the quarter of an annular domain (see figure 7) and tested the exponential spacing again. The results can be appreciated in Figure 8.

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A quarter of circular ring
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [p]=boundaryChi0(eta)
    p=[0,radiIn+eta*(radiOut-radiIn)];
function [p]=boundaryChi1(eta)
    p=[radiIn+eta*(radiOut-radiIn) 0];
function [p]=boundaryEta0(chi)
    p=[radiIn*cos((1-chi)*sector) radiIn*sin((1-chi)*sector)];
function [p]=boundaryEta1(chi)
    p=[radiOut*cos((1-chi)*sector) radiOut*sin((1-chi)*sector)];
function [value]=radiIn()
    value=4;
function [value]=radiOut()
    value=7;
function [value]=sector()
    value=pi/2;
```

Figure 7: Matlab quarter of annular domain

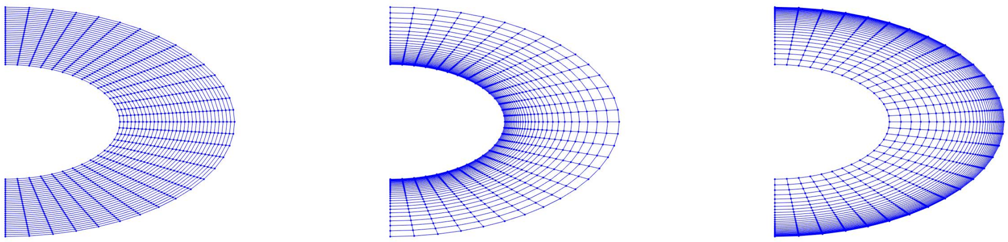
2.4 Apply on a new geometry

Finally, to cover the last point in the assignment, we implemented a horse-shoe domain. The code and the results can be found in Figures 9 and 10



(a) linear (b) denser on center (c) denser outside

Figure 8: Exponential spacing examples



(a) linear (b) denser on center (c) denser outside

Figure 10: Exponential spacing examples

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% EXAMPLE 3 Horse Shoe
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [p]=boundaryChi0(eta)
    p=[row*b0*cos(pi/2*(1-2*eta)) b0*sin(pi/2*(1-2*eta))];
function [p]=boundaryChi1(eta)
    p=[row*b1*cos(pi/2*(1-2*eta)) b1*sin(pi/2*(1-2*eta))];
function [p]=boundaryEta0(chi)
    p=[0 b0+(b1-b0)*chi];
function [p]=boundaryEta1(chi)
p=[0, -(b0+(b1-b0)*chi)];
function [value]=row()
    value=2.;
function [value]=b0()
    value=1.;
function [value]=b1()
    value=2.;
```

Figure 9: Matlab implementation of horse shoe mapping

References

[1] KSSV (2020). Transfinite Interpolation (https://www.mathworks.com/matlabcentral/fileexchange/40 transfinite-interpolation), MATLAB Central File Exchange. Retrieved October 28, 2020.

[2] Dyken, Christopher; Floater, Michael S.(2009). "Transfinite mean value interpolation". Computer Aided Geometric Design. 1 (26): 117–134. CiteSeerX 10.1.1.137.4822. doi:10.1016/j.cagd.2007.12.003.

[3] Gordon, William; Hall, Charles (1973). "Construction of curvilinear coordinate systems and application to mesh generation". International Journal for Numerical Methods in Engineering. 7 (4): 461–477. doi:10.1002/nme.1620070405.