POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Matematica

Technical report:
settings and performances of DiSMA on test cases

Davide Papapicco

DISCLAIMER

At the present release `DiSMA-v1.0` the library can only integrate differential problems over $2-$dimensional rectangular domains.

1 User interaction with the library

The present report shows the functionality of `DiSMA` (Differential Solver for Matlab Applications) for several parametric formulation of the same test case; the results described in the following sections are associated to the numerical simulations collected in the `Tutorials` subdirectory of the library and they follow the same naming convention.
The test cases discussed here all aim to discretise and approximate the analytical solution of a model equation with partial derivatives with parametric and time dependence.

1.1 $d-$dimensional unsteady model equation

The PDE is defined over a computational domain $\Omega \subset \mathbb{R}^d$ over which a mesh Ω_h is introduced for the approximation of the continuous solution; for unsteady problem we will consider the cartesian product $\Omega \times \mathbb{R}^+$ as the space-time domain of the problem. We consider a generic scalar function $u(\mathbf{x}, t)$ as the unknown solution of the differential model, whose most general strong formulation as a IBVP associated to the PDE in $d-$dimension, takes the following form (spatial and time dependence is implied for $u(\mathbf{x}, t)$ whenever possible for the sake of simplicity)

$$\begin{cases} \partial_t u - \nabla \cdot (\nu(\mathbf{x}, t) \nabla u) + \nabla \cdot (\boldsymbol{\beta}(\mathbf{x}, t) u) + \sigma(\mathbf{x}, t) u = f(\mathbf{x}, t) \ , \quad \forall \mathbf{x} \in \Omega \ , \quad \forall t \in [0, T] \\ u(\mathbf{x}, t) = g_D(\mathbf{x}, t) \quad \forall \mathbf{x} \in \Gamma_D \\ \partial_{\hat{\mathbf{n}}} u := \nabla u \cdot \hat{\mathbf{n}} = g_N(\mathbf{x}, t) \ , \quad \forall \mathbf{x} \in \Gamma_N \\ u(\mathbf{x}, 0) = u_0(\mathbf{x}) \ , \quad \forall \mathbf{x} \in \Omega \end{cases} ,$$

where $\nu(\mathbf{x}, t)$, $\boldsymbol{\beta}(\mathbf{x}, t)$, and $\sigma(\mathbf{x}, t)$ quantify the relative magnitude of the diffusive, convective and reactive phenomena respectively whereas $f(\mathbf{x}, t)$ models the source term function. We denote Γ_D and Γ_N as the mutually disjoint ($\Gamma_D \cap \Gamma_N = \emptyset$) portions of the boundary $\partial\Omega$ of the spatial domain over which we set Dirichlet ($g_D(\mathbf{x}, t)$) and Neumann ($g_N(\mathbf{x}, t)$) boundary conditions (BCs) respectively.

1.2 Input file

The structure of DiSMA is s.t. all the information necessary from the user for it to discretise and solve the desired model are passed via a single script called by default `inputs.m` but it can be renamed according to the user preference. One recommended practice is naming the file according to the simulation one is performing; for example if the users wants to run a series of simulations based on the same model but with different parameters then he/she can copy the file as many times as necessary and rename each one in accordance with the modifications made to the parameter functions.
The information that the user is required to specify are divided in **explicit** (the user must provide the information by hand) and **implicit** (the user can select the desired setting among a set of pre-existing selections). The former inputs are:

- Coordinates of the input vertices for the domain (2 for $1-$dimensional simulations and 4 for $2-$dimensional simulations);
- Explicit functional form of the coefficient $\nu(\mathbf{x}, t)$, $\boldsymbol{\beta}(\mathbf{x}, t)$, $\sigma(\mathbf{x}, t)$ and the source functions $f(\mathbf{x}, t)$;
- Explicit functional form of the analytical solution $u(\mathbf{x}, t)$ and its first order derivative ∇u;
- Explicit Dirichlet and Neumann boundary functions $g_D(\mathbf{x}, t)$ and $g_N(\mathbf{x}, t)$;
- Length of simulated time $T > 0$ and timestep δt for the solution of the associated time-dependent ODE.

The implicit information consist in selecting:

- Number of iterations for the grid convergence error estimation `Ns` $\in \{2,\ 4,\ 6,\ 8,\ 10\}$;
- Subspace of the basis function `subspace` $\in \{\mathbb{P}_1,\ \mathbb{P}_2\}$;
- Number of spatial dimensions `d` $\in \{1,\ 2\}$;
- Time dependence (unsteady) or independence (stationary) `time` $\in \{$True, False$\}$;
- Discretisation algorithm and solver for the time-dependence.

An example of the input file is shown below; in it we set the various terms in order to solve the $2-$dimensional test cases discussed below in the present report.

```matlab

% prepare the simulation
clear all
close all
clc

% including the triangulator library and solver library functions according
    to the choice of the subspace
addpath('bbtr30')
disp('bbtr30 added to the path')
addpath('P2elements')
disp(fprintf('P2 basis functions selected \n P2elements added to the path'))
    ;

% define coefficient functions of the PDE
global coefficient_functions;
coefficient_functions = {
    @(x,y) 1.0;...  % diffusion function (nu)
    @(x,y) 0.0;...  % convection function (beta)
    @(x,y) 0.0};    % reaction function (sigma)

% define analytical solution and its derivatives
global exact_functions;
exact_functions = {
    @(x,y) exp(-x^2 - y^2);...          % exact solution (U)
    @(x,y) -2*x*exp(-x^2 - y^2);...     % x-partial derivative (U_x)};
    @(x,y) -2*y*exp(-x^2 - y^2)};       % y-partial derivative (U_y)};

% define source function
global source_function;
source_function = @(x,y) -coefficient_functions{1}(x,y)*4*(x^2 + y^2 -1)*
    exact_functions{1}(x,y);

% construct boundary functions cell array
global boundary_functions;
boundary_functions = {
    % first row -> Dirichlet BCs
    @(x,y) exp(-x^2),...      % border 1
    @(x,y) exp(-(y^2+1)),...  % border 2
    @(x,y) exp(-(x^2+1)),...  % border 3
    @(x,y) exp(-y^2);         % border 4
    % second row -> Neumann BCs
    @(x,y) 2*y*exact_functions{1}(x,y),...              % border 1
    @(x,y) -2*x*exact_functions{1}(x,y),...             % border 2
    @(x,y) -2*y*exact_functions{1}(x,y),...             % border 3
    @(x,y) 2*x*exact_functions{1}(x,y)};                % border 4

% define computational domain
% BY CONVENTION: border 1 goes from first node (v1) to second node (v2);
    border 2 goes from (v2) to (v3); border 3 from (v3) to (v4) and so on
v1 = [0, 0];
v2 = [1, 0];
v3 = [1, 1];
v4 = [0, 1];

global vertices;
vertices = [v1; v2; v3; v4];
clear v1 v2 v3 v4;

% imposition of the BCs markers on each boundary of the domain
% BY CONVENTION: odd markers are for Dirichlet BCs while even markers are
    for Neumann BCs
b1 = 1;
b2 = 2;
```

```
b3 = 4;
b4 = 7;

global boundaries;
boundaries = [b1 b2 b3 b4];
clear b1 b2 b3 b4;

% imposition of the BCs markers on each input vertex of the domain
% BY CONVENTION: odd markers are for Dirichlet BCs while even markers are
    for Neumann BCs
global inputs;
inputs = [1 1 0 7];

% define basis functions subspaces in string array
subspace = ["P1", "P2"];

% define number of iterations for convergence (mesh refinement)
Ns = 3;

% launch the simulation
main(Ns, subspace(2));
```

1.3 Test cases

In the following sections we will show the outputs of the solver against different instances of the same test case in 1 and 2 spatial dimensions. Since the presence of advective and reactive phenomena directly contirbute to the mesh Peclet number, they have to be set carefully in order to derive accurate and stable simulations; as such, for the sake of simplicity, the test case considered throughout the report will be one of pure-diffusion, i.e. $\boldsymbol{\beta}, \sigma = 0\,,\, \nu(\mathbf{x}, t) \neq 0$.

1.3.1 $1-$dimensional setting

1.3.2 $2-$dimensional setting

We want our solution to be a multivariate normal distribution $e^{\mathbf{x}^T\mathbf{x}} = e^{-(x^2+y^2)}$ with the unitary square in the first quadrant of the $x-y$ plane as spatial domain, i.e. $\Omega = [0,1]\times[0,1]$. In order to achieve the desired result we must, given a certain function $\nu(\mathbf{x},t)$ calculate the correct source and boundary functions.
The calculation of the source function we start by computing the spatial partial derivatives of the desired solution and then plug both ∇u and u in the model equation

$$\nabla u = (\partial_x u, \partial_y u)^T = (-2x\, e^{-(x^2+y^2)}, -2y\, e^{-(x^2+y^2)})^T = -2u\begin{pmatrix} x \\ y \end{pmatrix} = -2u\,\mathbf{x}$$

$$f(\mathbf{x},t) = -\nabla\cdot(\nu\nabla u) = -\nabla\nu\cdot\nabla u - \nu\Delta u = 2u\,\nabla\nu\cdot\mathbf{x} - \nu\Delta u\,, \tag{1}$$

and complete the calculation according to $\nu(\mathbf{x},t)$.
The computation for the Dirichlet and Neumann boundary functions is less straightforward and we will refer to the graphical representation of the domain and the analytic function itself. Let's start by considering the Neumann boundary functions as they are somewhat less complicated to derive in a general fashion; as in Figure 1, the computational domain of our test case features 4 normal vectors (one for each edge of the domain) which are

$$\hat{\mathbf{n}}_\mathbf{1} = (0,-1)\,,\quad \hat{\mathbf{n}}_\mathbf{2} = (1,0)\,,\quad \hat{\mathbf{n}}_\mathbf{3} = -\hat{\mathbf{n}}_\mathbf{1}\,,\quad \hat{\mathbf{n}}_\mathbf{4} = -\hat{\mathbf{n}}_\mathbf{2}$$

To compute the Neumann functions associated to any given edge of the boundary we simply compute the dot product between the gradient of the analytical solution (which we already have following the calculation of the source function) and the normal vector of the edge on which the Neumann BC has been assigned. By putting in notation $N_j\,,\ j = 1,2,3,4$ s.t. uniquely $g_{N_j}(\mathbf{x},t)$ defines the Neumann function associated to the j^{th} edge of $\partial\Omega$, then we have

$$g_{N_1} = 2y\, e^{\mathbf{x}^T\mathbf{x}} = 2y\, u(\mathbf{x},t)\,,\quad g_{N_2} = -2x\, u\,,\quad g_{N_3} = -2y\, u\,,\quad g_{N_4} = 2x\, u\,.$$

For the calculation of the Dirichlet functions we must what values the analytical solution has along those edges where it is assigned; this is trivial for a function that goes uniformly quickly to zero (i.e. it features compact support). For the case of our multivariate distribution the process is more complex. In general one must compute the projection of the function onto the lower dimensional plane perpendicular to the flat surface domain with which it intersect along the desired edge. The 4 planes are depicted in Figure 2 alongside the multivariate analytical solution and its projection onto the planes themselves. By using the same notation for the edges of $\partial\Omega$, the 4 Dirichlet boundary function are

$$g_{D_1} = e^{-x^2}\,,\quad g_{D_2} = e^{-(y^2+1)}\,,\quad g_{D_3} = e^{-(x^2+1)}\,,\quad g_{D_4} = e^{-y^2}\,.$$

As a last remark we notice that only the source function in (1) depends explicitly on the parametric definition of the model equation and will thus be the main discrimination among the simulations of the test cases we will perform in the following section.

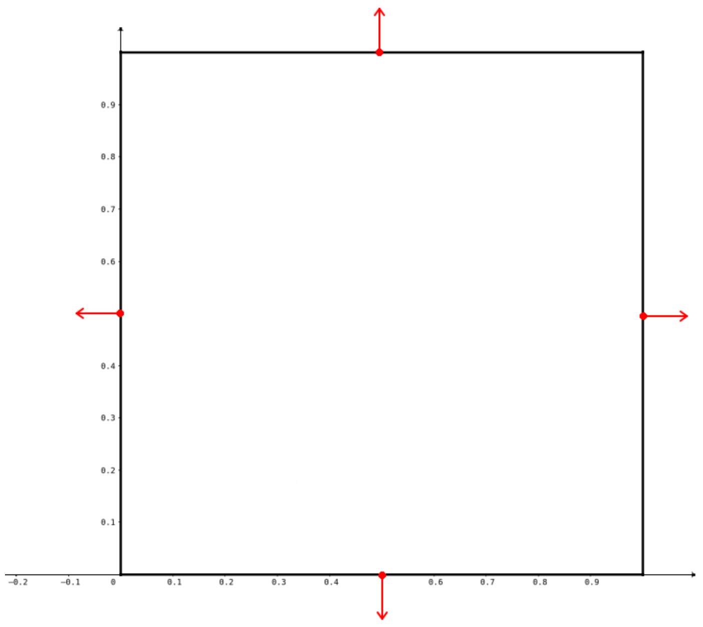


Figure 1: The 4 normal vectors of the edges of the computational domain for the 2−dimensional test case, needed for the calculation of the explicit form of Neumann boundary functions

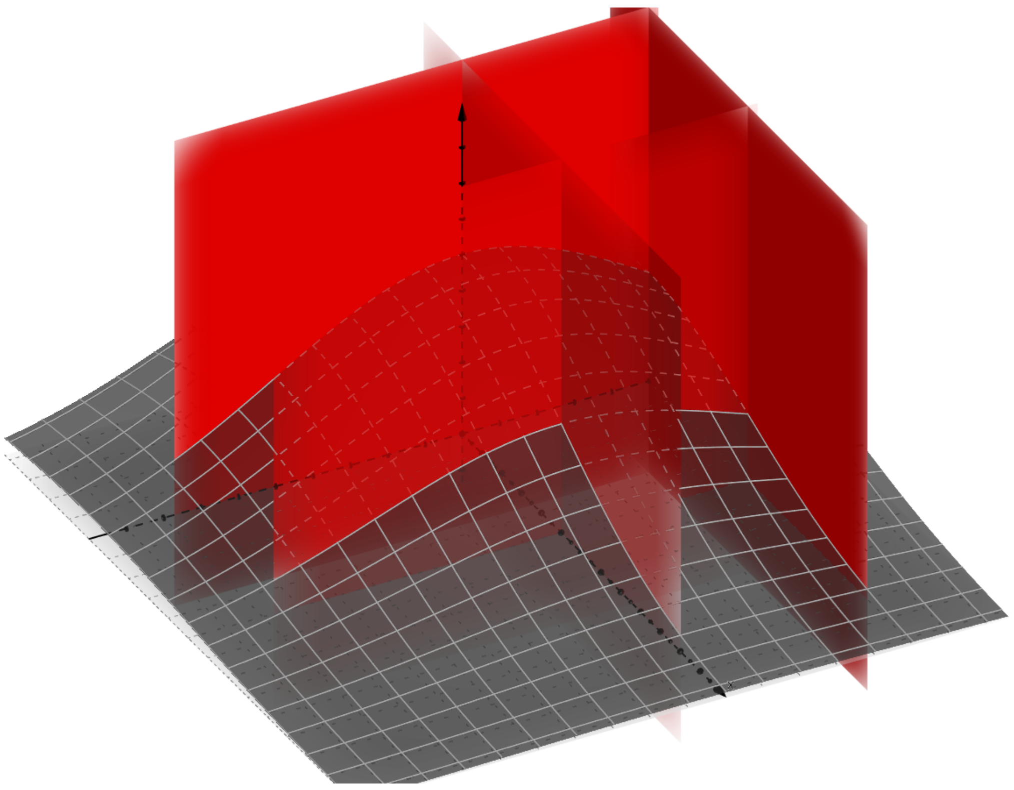


Figure 2: The 4 orthogonal planes to each edge of the computational domain for the 2−dimensional test case and the projected solution, needed for the calculation of the explicit form of Dirichlet boundary functions

2 FEM1D

We will show the application of the 1−dimensional FEM-based solvers of `DiSMA` to our test case.

2.1 Sim1: stationary uniform diffusion with Dirichlet BCs

2.1.1 Sim1.1: results with $\mathbb{P}_1$−basis functions

2.1.2 Sim1.2: results with $\mathbb{P}_2$−basis functions

2.2 Sim2: stationary uniform diffusion with mixed BCs

2.2.1 Sim2.1: results with $\mathbb{P}_1$−basis functions

2.2.2 Sim2.2: results with $\mathbb{P}_2$−basis functions

3 FEM2D

Here we perform 2−simulations for the test case with the application `FEM2D`, the FEM-based solvers of `DiSMA` for planar, rectangular domains using $\mathbb{P}_1$ and $\mathbb{P}_2$ basis functions.

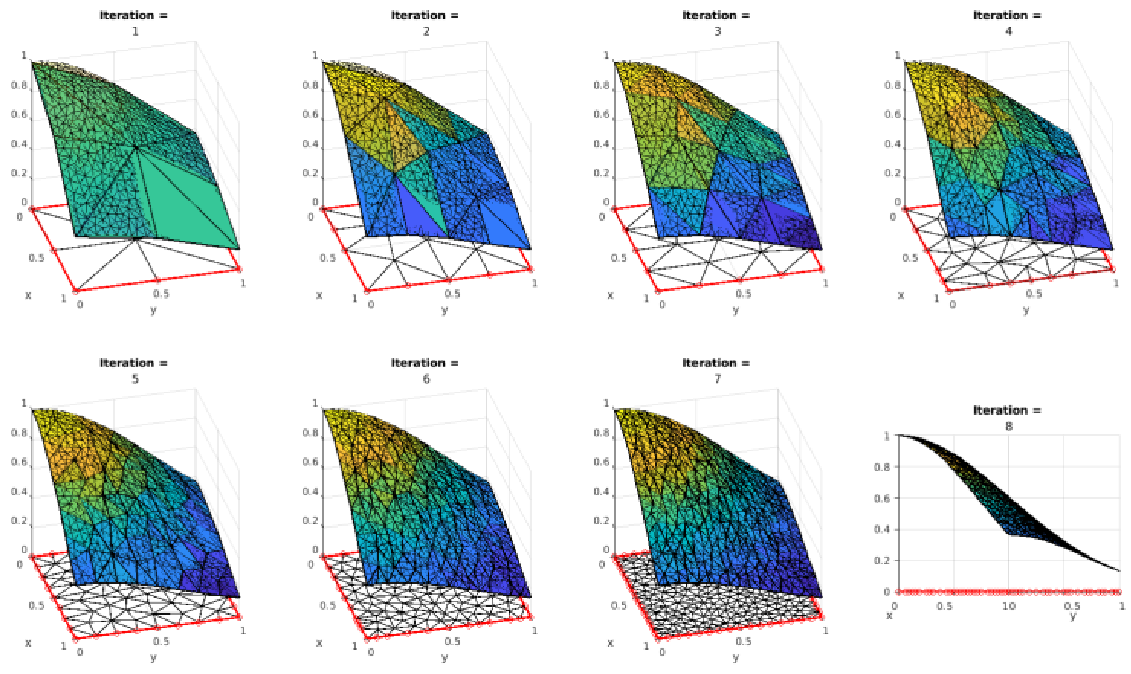
3.1 Sim3: stationary uniform diffusion with Dirichlet BCs

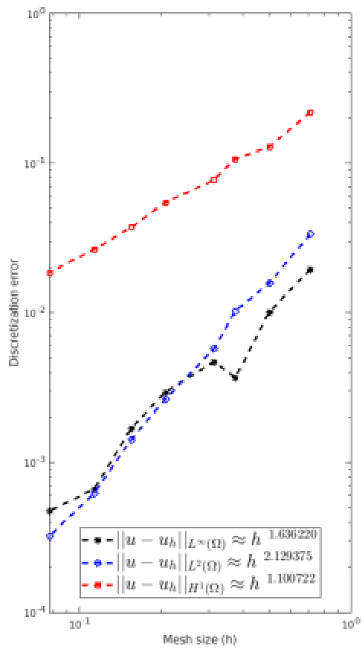
We set $\nu(\mathbf{x}) = 1$ throughout the entirety of Ω which implies that it has null gradient everywhere in the spatial domain. From (1) we immediately notice that only the solution's laplacian term is non-zero and as such we calculate the source function accordingly

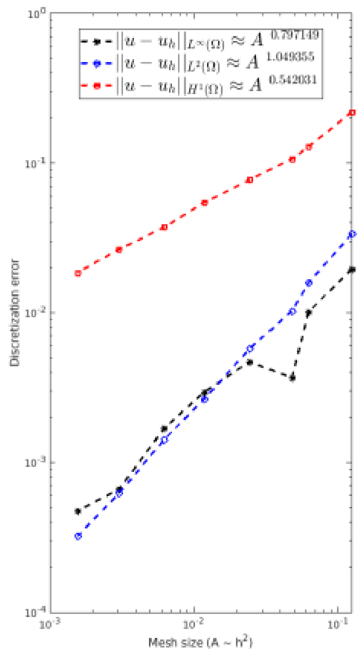
$$f(\mathbf{x}) = -4(x^2 + y^2 - 1)e^{\frac{1}{2}\mathbf{x}^T\mathbf{x}} = -4(x^2 + y^2 - 1)\, u(\mathbf{x})\,.$$

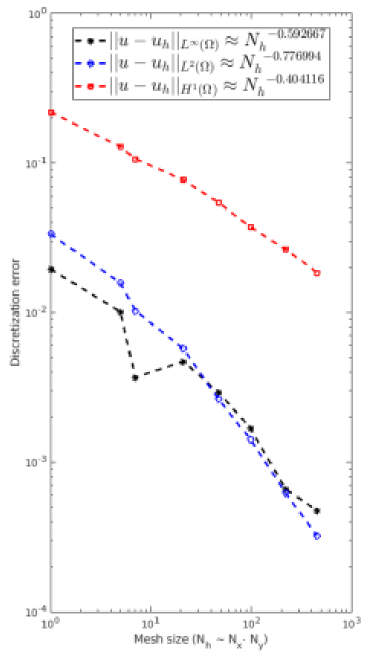
Finally we assign the non-homogeneous Dirichlet BCs we derived in **1.3.2** to all four edges of $\partial\Omega$.

3.1.1 Sim3.1: results with $\mathbb{P}_1$−basis functions

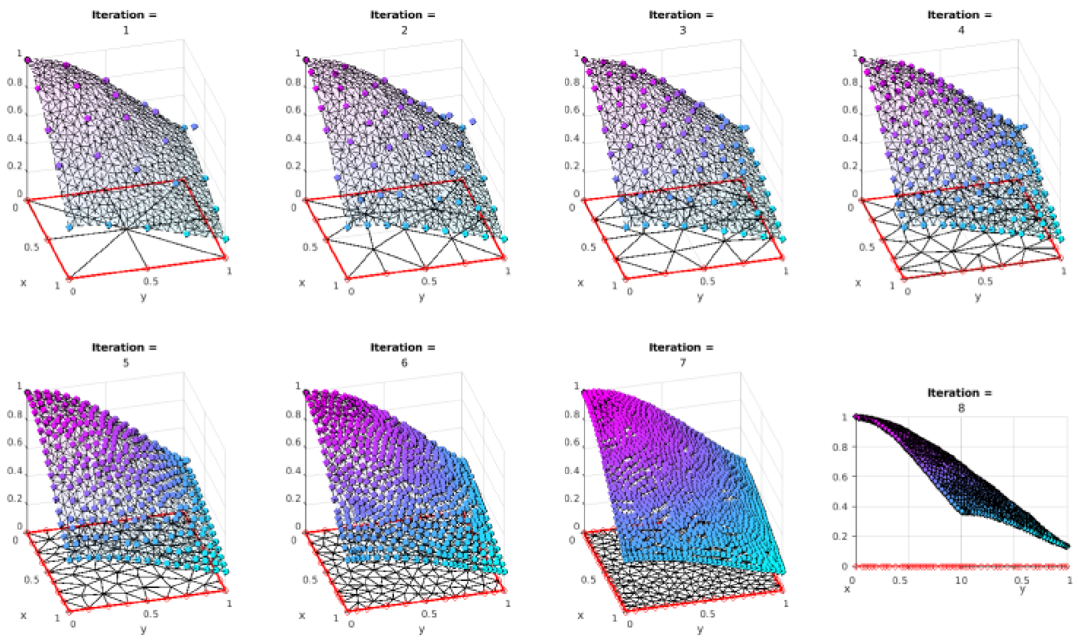








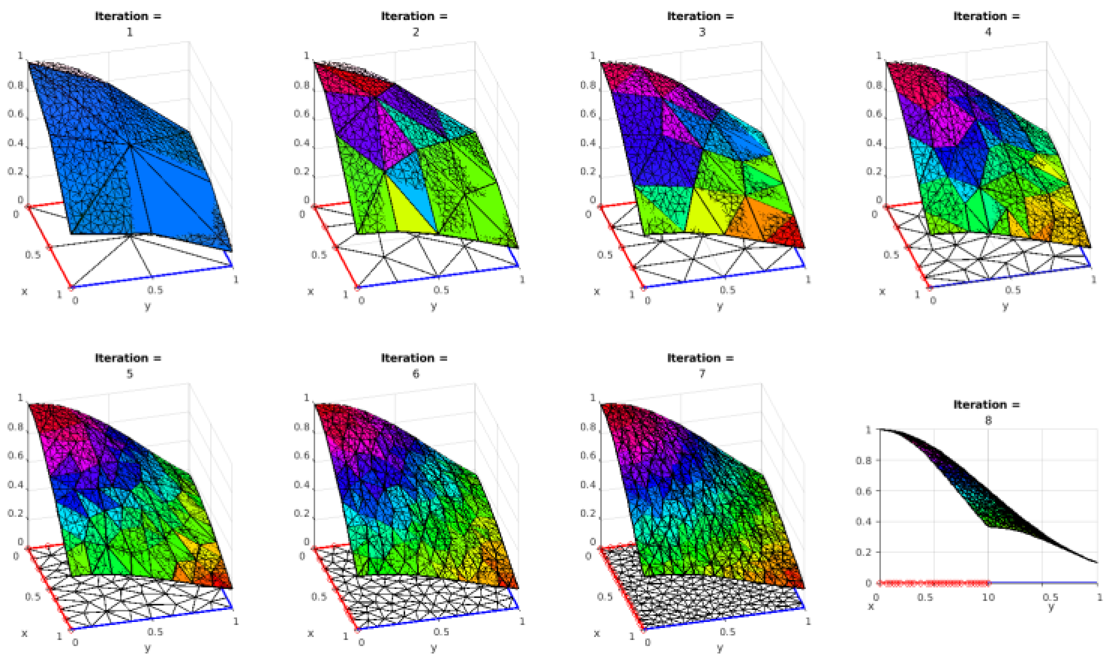
3.1.2 Sim3.2: results with $\mathbb{P}_2-$basis functions

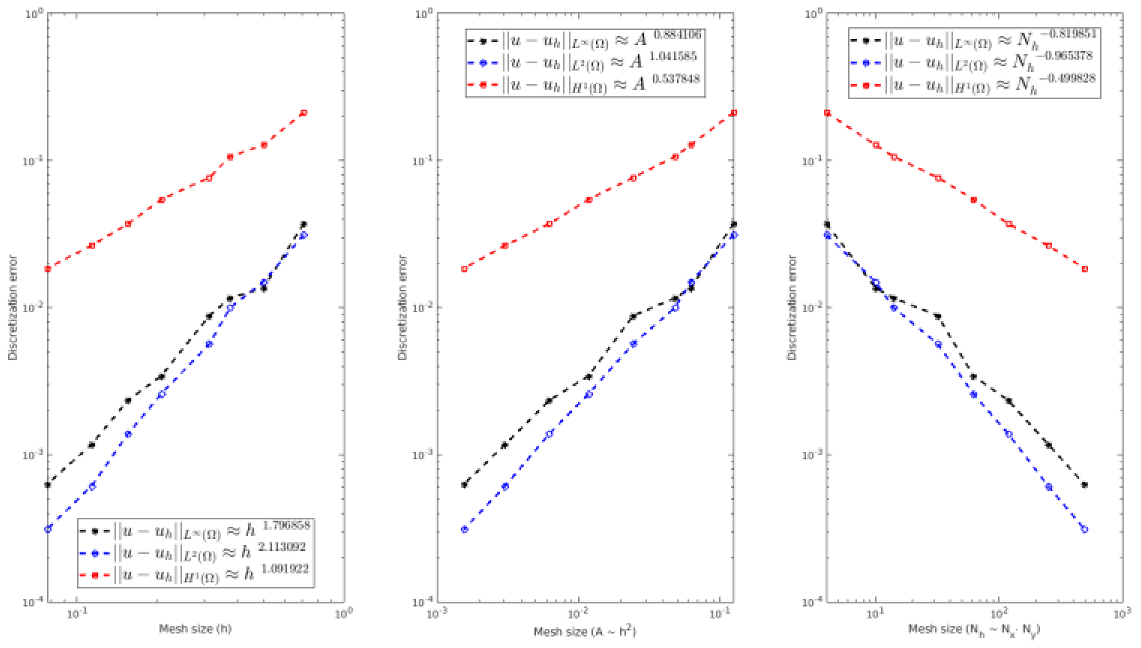


3.2 Sim4: stationary uniform diffusion with mixed BCs

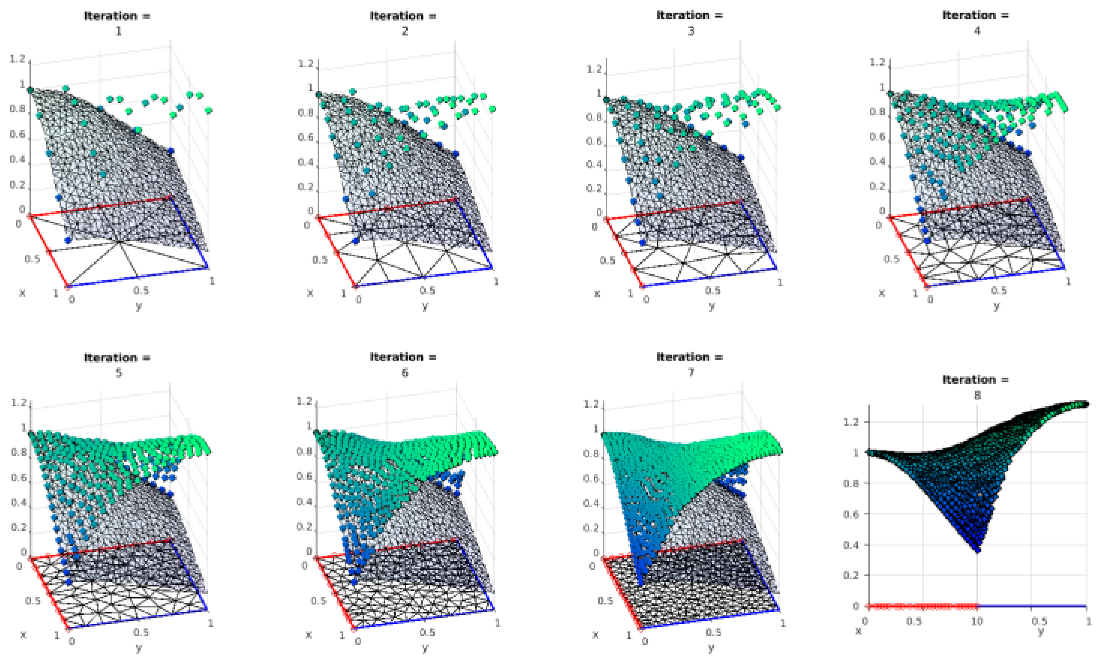
We keep the same value for ν as in **Sim3** and as such the source function stays unchanged. The assigned BCs this time however will be Dirichlet's on edges 1 and 4 and Neumann's on edges 2 and 3 of the border.

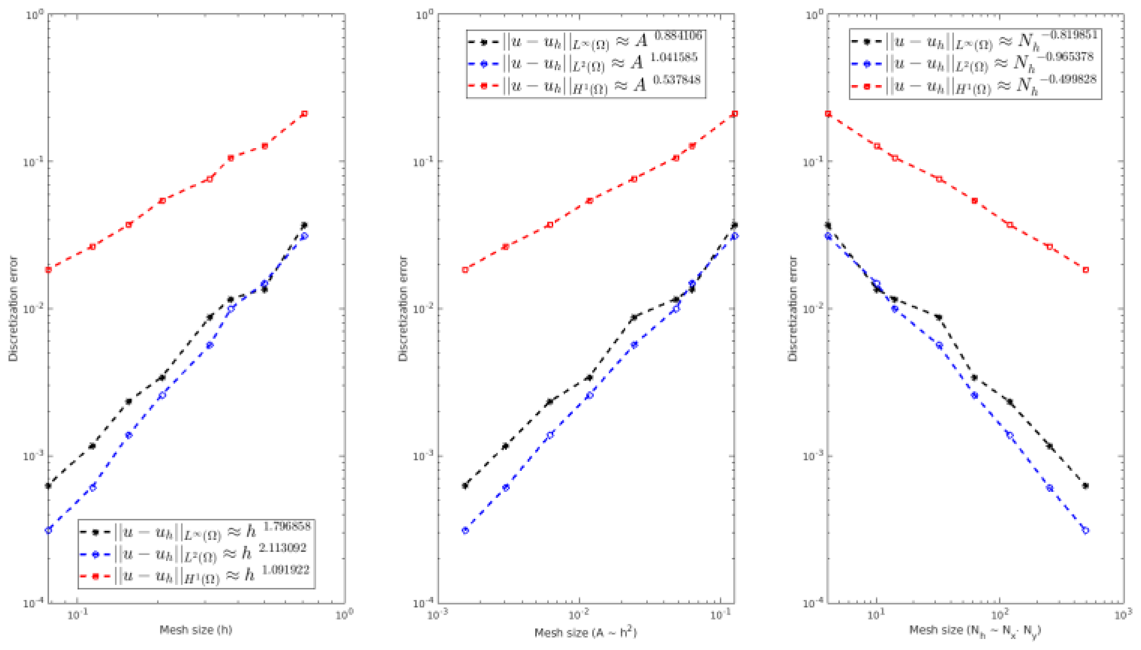
3.2.1 Sim4.1: results with $\mathbb{P}_1$−basis functions





3.2.2 Sim4.2: results with $\mathbb{P}_2-$basis functions





3.3 Sim5: stationary non-uniform diffusion

Here instead we choose a non-uniform spatial distribution for the diffusion $\nu(\mathbf{x}) = y\sin(x) + x\cos(y)\,,\ \ \forall \mathbf{x} = (x,y)^T \in \Omega$. As opposed to **Sim3** here the gradient of ν is not zero

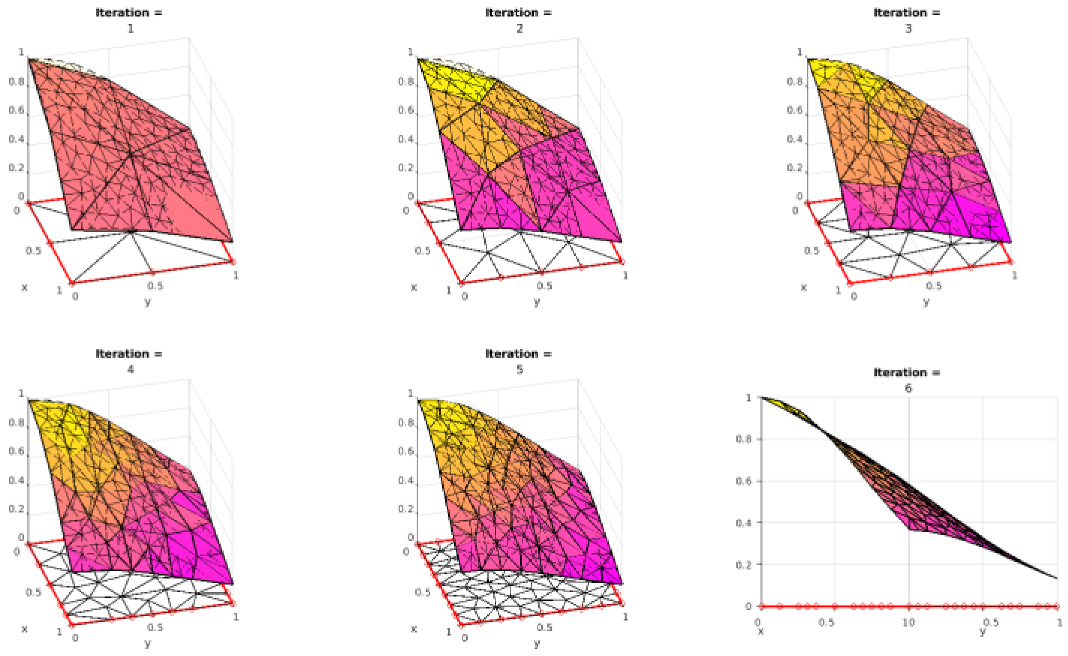
$$\nabla\nu = (\partial_x\nu, \partial_y\nu)^T = (y\cos(x) + \cos(y)\,,\ \sin(x) - x\sin(y))\,,$$

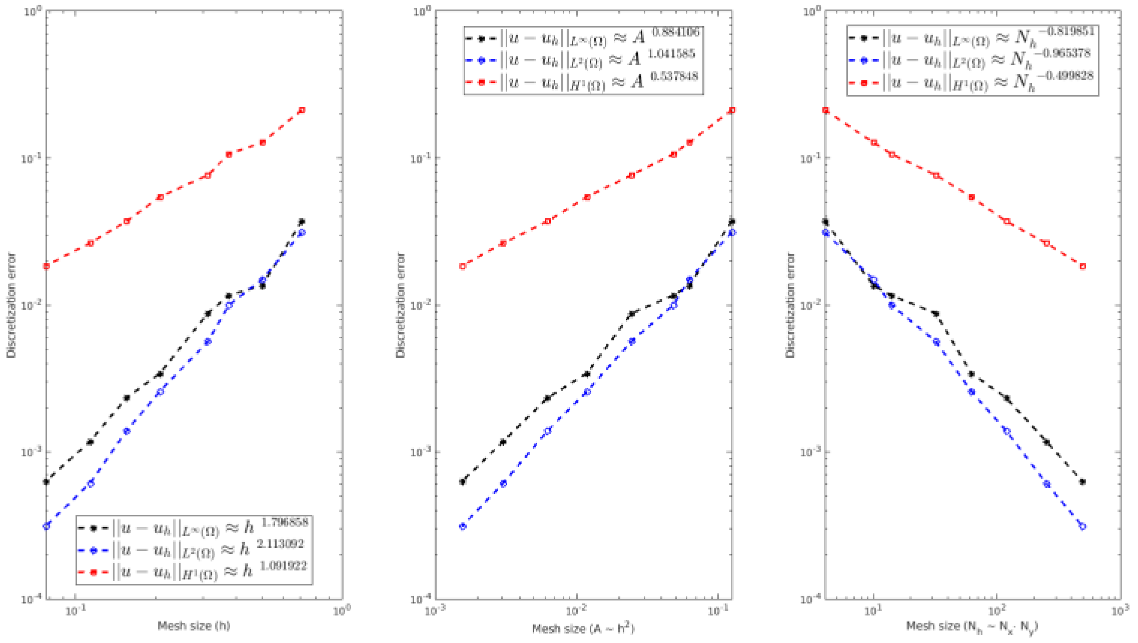
and thus the source function will contain the contributions from both the first the second term in (1) as derived below

$$f(\mathbf{x}) = 2xy\left(\cos(x) + \frac{\nu(\mathbf{x})}{xy} - \sin(y)\right)u(\mathbf{x}) - 4(x^2+y^2-1)\nu(\mathbf{x})\,u(\mathbf{x}) =$$
$$= 2xy\left(\cos(x) + \frac{\nu(\mathbf{x})}{xy} - 2\nu(\mathbf{x})(x^2+y^2-1) - \sin(y)\right)u(\mathbf{x})\,.$$

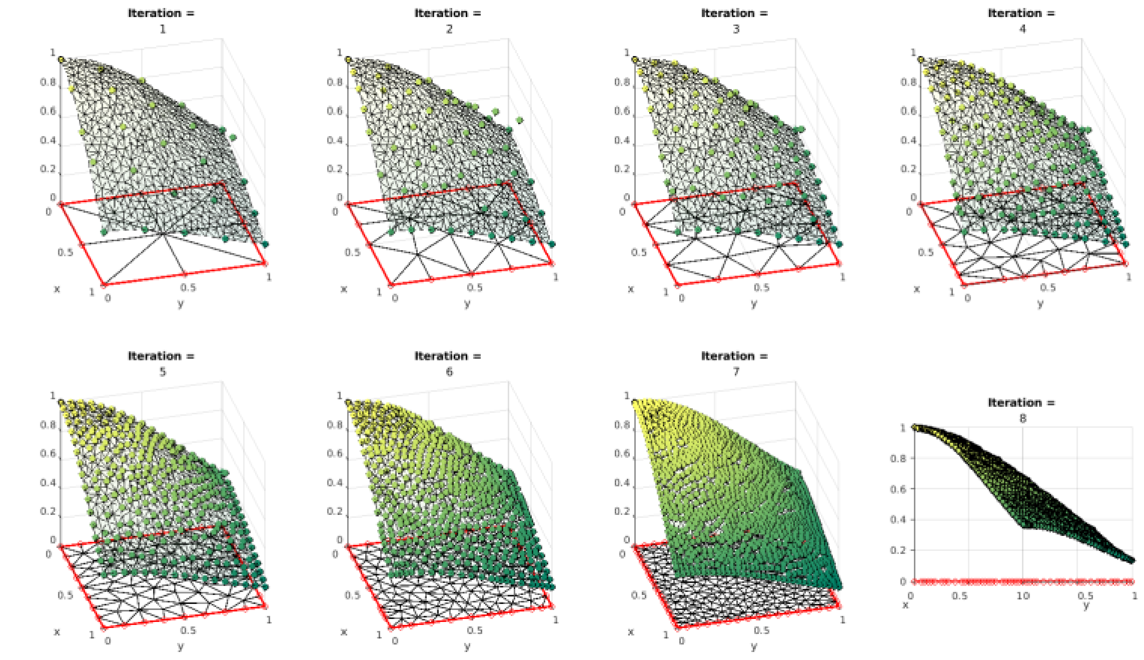
For the BCs we retain the same non-homogeneous Dirichlet BCs set in **Sim3** for the sake of simplicity.

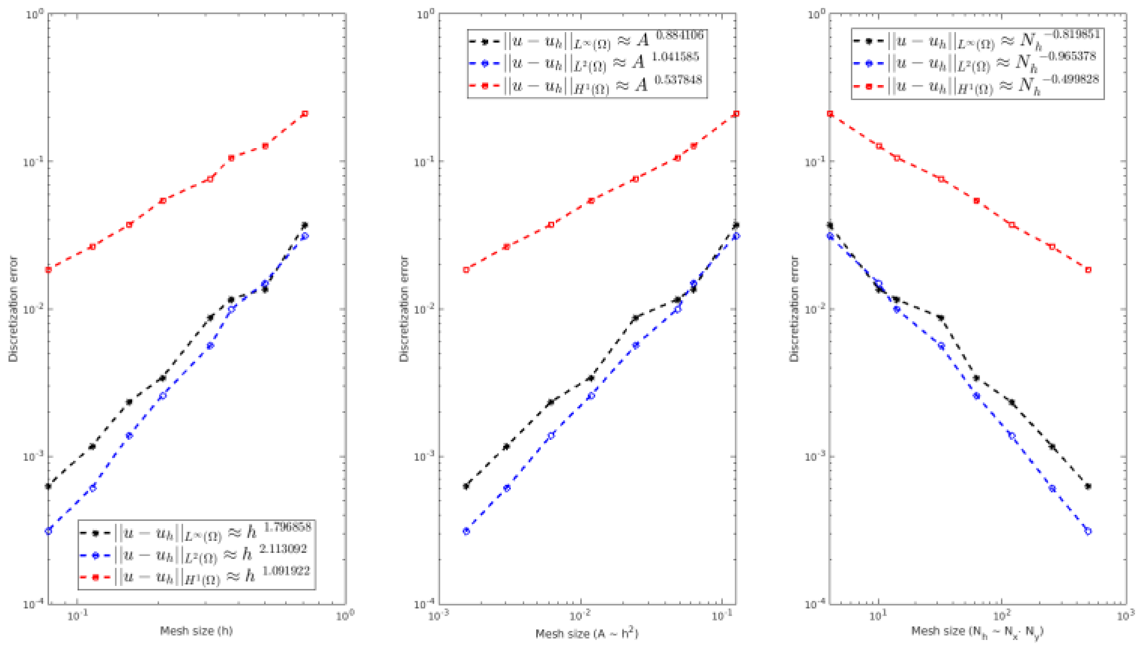
3.3.1 Sim5.1: results with $\mathbb{P}_1-$basis functions





3.3.2 Sim5.2: results with $\mathbb{P}_2-$basis functions





3.4 Sim6: unsteady problem with transient diffusion

As a case-test for a transient solution (one whose time behaviour starts with an initial oscillations that get dumped until a stationary state it's reached) we build the time-dependence for $u(\mathbf{x}, t)$ in the following way.

3.4.1 Sim6.1: results with $\mathbb{P}_1-$basis functions

3.4.2 Sim6.2: results with $\mathbb{P}_2-$basis functions

4 Introduction and motivation

Different techniques exist in numerical analysis to discretise and solve partial differential equations (PDEs). Among the most widespread we surely find the Finite Element Method (FEM) based on the construction of an ad-hoc finite-dimensional space whose basis is used to linearly derive the approximated solution of a given boundary value (partial) differential problem (IBVP).
In the following we shall address the main features of the aforementioned numerical method by implementing the algorithm on different IBVPs and post-processing its results by evaluating the performances and the accuracy of the former.

4.1 Variational (weak) formulation: Galerkin's problem

We shall hereby outline the fundamental assumptions in order for the FEM to converge to the desired solution as well as define the notation and terminology that will be used in the following sections.

Definition 1 *Let $d \in \mathbb{N}$ be an integer strictly greater than 0 defining the dimensionality of the PDE , $\Omega \subset \mathbb{R}^d$ be a compact subset and V an Hilbert space with functions valued in Ω then we define*

$$\textit{find}\, u \in V \ \textit{s.t.}\ \ a(u,v) = F(v)\,,\ \forall v \in V$$

a (abstract) variational problem on Ω.

Theorem 1 *Let $a(u,v) = F(v)$ be a variational problem on a Hilbert space V with $a : V \times V \mapsto \mathbb{R}$ and $F : V \mapsto \mathbb{R}$ some bi-linear and linear forms on V respectively. If a, F are s.t.*

- *they verify **Lax-Milgram theorem**;*
- *they depend continuously on the data in V;*

*then the IBVP associated to $a(u,v) = F(v)$ is a **well-posed problem** in the sense of Hadamard.*

Theorem 2 *Let $a(u,v) = F(v)$ satisfy 1 and let $V_h \subset V$ be a (finite-dimensional) subspace of V induced by a discretization of the domain (of "size" h) of the IBVP and the choice of a basis. If V_h is s.t.*

- *it satisfies **Cea's lemma**;*
- *it satisfies the consistency hypothesis, i.e.*

$$dist(u, V_h) \to 0\,,\ h \to 0^+$$

then the discretization is convergent i.e. $||u - u_h||_V \to 0\,,\ h \to 0^+$, with u_h being the discrete solution derived as a linear combination of the basis functions of V_h.

5 $\mathbb{P}_1$ finite elements

In the present section we shall impose

$$V = \left\{ v \in \mathcal{C}(\Omega) \ \text{ s.t. } \ v|_e \in \mathbb{P}_1\,,\ \forall e \in \Omega_h \right\},$$

Let's immediately notice that the continuity condition is a fundamental requirement for the formulation of Galerkin's problem however this condition can be relaxed in different techniques.

5.1 1−dimensional IBVPs

Definition 2 *Let $L > 0$ be a real number and $I = [0, L] \subset \mathbb{R}^+$ a finite, closed interval (domain); let also $f \in L^2(I)$ be a source function and $\mu, \beta, \sigma \in L^\infty(I)$ be coefficients functions, then the following*

$$\textit{find } u \in V \ \textit{s.t.} \begin{cases} (Du)(x) = -(\mu(x)u^{'}(x))^{'} + \beta(x)u^{'}(x) + \sigma(x)u(x) = f(x)\,,\ \forall x \in (0, L) \\ u(0) = g_0 \\ u(L) = g_L \end{cases}$$

*is an IBVP that will be henceforth referred to as **diffusion-advection-reaction equation** (DARe)*

Proposition 1 *The problem defined in 2 satisfies the hypothesis of 1 if and only if μ, β, σ are (uniformly limited) functions in I s.t.*

- *$\mu(x) \geq \mu_0 > 0$;*
- *$\beta^{'}$ is limited;*
- *$\sigma - \frac{1}{2}\beta^{'} \geq 0$*

Example 1 *Let $L = \pi$, $\mu(x) = \beta(x) = \sigma(x) = 1$ and $f = Du$ when $u(x) = sin(nx)$ (this imposes the BCs to be $g_0 = g_L = 0$). Implement a $\mathbb{P}_1-$based FEs algorithm and verify the convergence of 2.*

Solution
The implementation strategy has been performed by including an outer loop in which the number of FEs (i.e. subintervals) of partition $\mathcal{T}_h$ of subset I doubles at each iteration. In particular we wanted to control the discretization step h through an ad-hoc parametrisation that, at each iteration $k \in \mathbb{N}$ refines $\mathcal{T}_h^{(k)}$ in the following way

$$h^{(k)} = \frac{L}{2^k} = \frac{1}{2} \cdot \frac{L}{2^{k-1}} = \frac{1}{2} \cdot h^{(k-1)}$$

where $L = L - 0 = \mu(I)$ is the measure of subset I. It is trivial to observe that the number of FEs is $N_e := 2^k$, the number of $DOFs$ (inner nodes) is $N_h := N_e - 1$ and the total number of nodes is $N := N_h + 2 = \cdots = N_e + 1$.
We clearly see that at the fourth iteration ($k = 4$), with merely 15 nodes of estimation of $\mathcal{T}_h$ the approximation and its derivative almost identically follow the exact solution, as depicted in the below figure, with the behaviour becoming increasingly unstable with greater values of $n = 1, 2, \pi, ...$

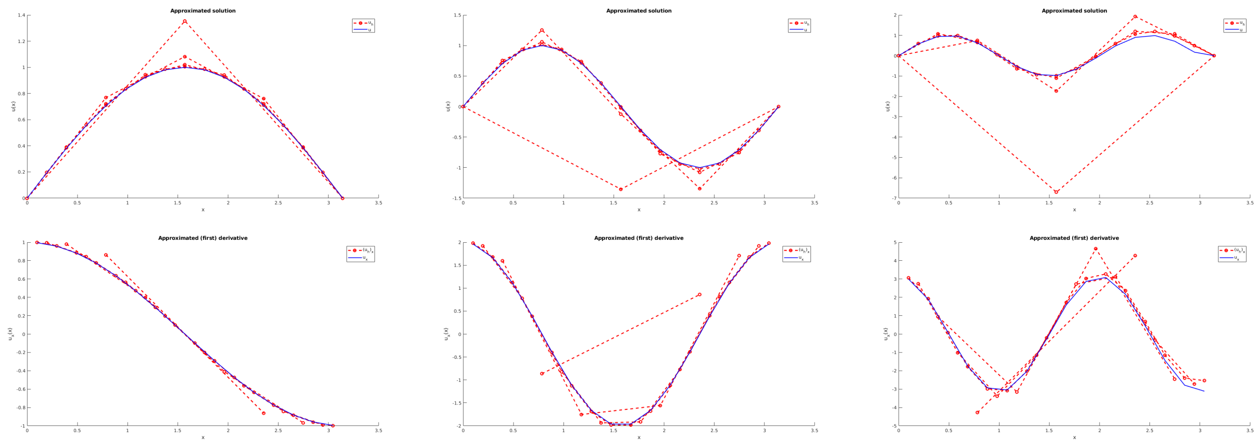
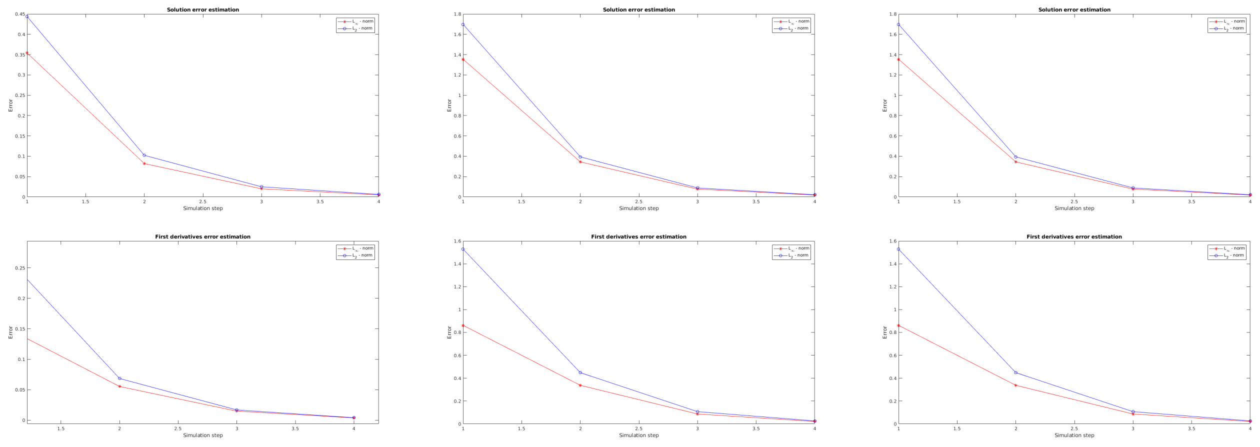


Figure 3: Convergence of the iterative approximations of the FEM solution to the exact solution (first row) and their respective (first order) derivatives (second row). From left to right we have different values of integer n in $sin(nx)$, specifically $n = 1$ (leftmost), $n = 2$ (centre), $n = \pi$ (rightmost)

The instability at greater values of n can be easily evaluated also from the errors' norm estimation, depicted below.



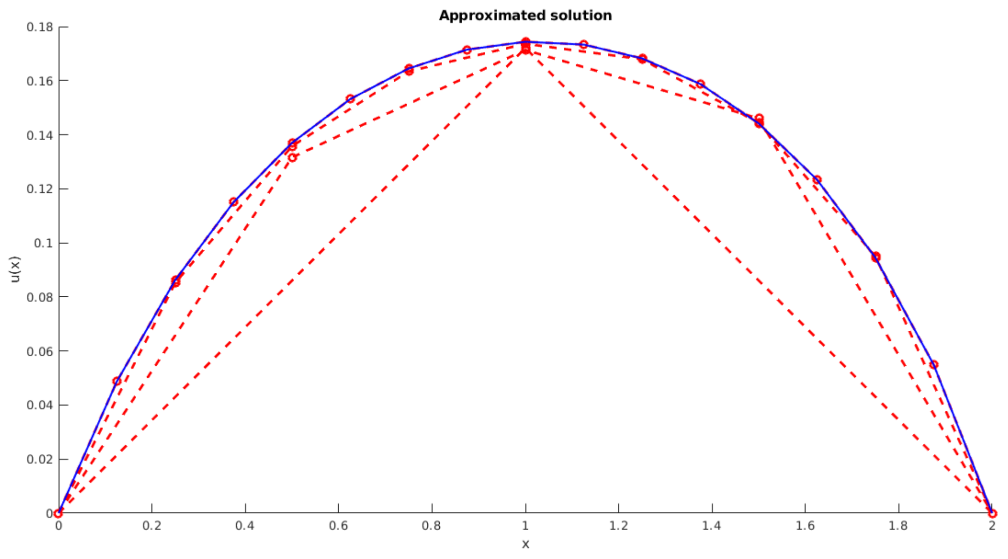
Example 2 *Let $L = 2$, $\mu(x) = 1 + x, \beta(x) = 3x, \sigma(x) = e^x$ and $f = 1$ with homogeneous Dirichlet BCs. Estimates the value of the solution of the DARe in $x = 1$.*

Solution

Let's implement the same strategy of and let's verify the convergence of u_h to u. The results are consistent with those of 5.1 as depicted in the following Figure.

We conclude that, as of the tenth iteration (i.e. $k = 10 \Rightarrow N_h = 1023$), the estimation of $u(x)$ in x=1 is as follows

$$\lim_{h \to 0^+} u_h(x = 1) = 0.1747 \approx u(x = 1)$$



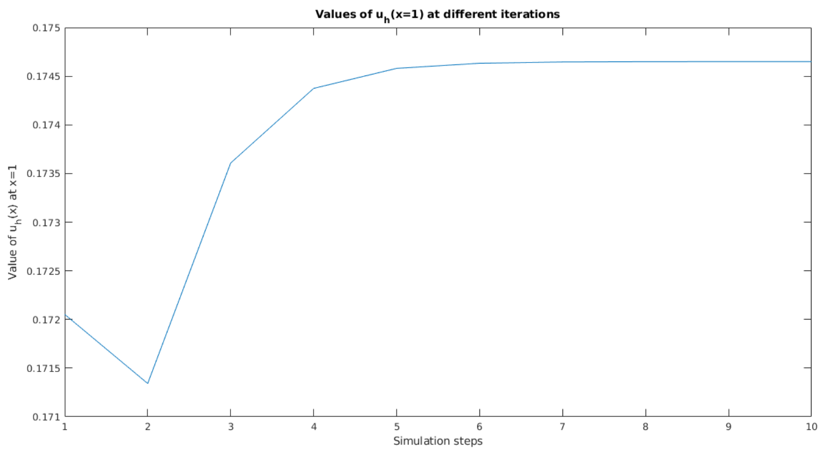


Figure 5: (Top) Iterative approximations of $u_h^{(k)}$; we can see that it does resemble the same convergence property of 5.1.

(Bottom) Iterative evaluations of $u_h^{(k)}$ in x=1 (midpoint of the computational domain

Example 3 *With the same data of 2, but in the case on non-homogeneous BCs* $g_0 = 2\,,\, g_L = 5$, *Estimates the value of the solution of the DARe in* $x = 1$.

Solution
Following the methodologies laid down in both 5.1 and 2 we concluded

$$\lim_{h \to 0^+} u_h(x = 1) = -1.2856$$