Continuous Assessment 2

MSc in Computer and Mathematical Engineering

Exercise 1

Modify the program L2Projector1D given in the book in order to compute the approximations L^2 for the case of the following functions:

a) $\boldsymbol{f(x) = 1 + 2x}$

b) $\boldsymbol{g(x) = x^3(x-1)(1-2x)}$

c) $\boldsymbol{h(x) = \arctan((x-0.5)/\epsilon)}$**, with** $\boldsymbol{\epsilon = 0.1}$ **and** $\boldsymbol{\epsilon = 0.01}$

Use uniform partitions in the interval $\boldsymbol{I = [-1,1]}$ **with** $\boldsymbol{n = 25}$ **and** $\boldsymbol{n = 100}$ **subintervals. Give also an estimation of the error obtained when using these approximations.**

In this exercise we are asked to modify the program for obtaining the L^2 projections of the different functions using a uniform partition on $[-1,1]$ with $n = 25$ and $n = 100$ and estimate the error obtained when approximating these functions compared with their true value.

The code of L2Projector1D is a code that allows to obtain the L^2 projection of the function $f(x) = x\sin(x)$ through computing its mass matrix $\boldsymbol{M}$ (how basis functions overlap) and load vector $\boldsymbol{b}$ (the projection of the function in each basis function) and obtain the system of equations $\boldsymbol{MP = b}$ that allows to determine the coefficients of the projection in $\boldsymbol{P}$. Because we are required to adapt the code for the different functions proposed, we offer the modification of this code in the Matlab file Ex1.m, available in the folder.

In fact, we do not need to adapt functions like the MassAssembler1D or the LoadAssembler1D, but just the L2Projector1D to obtain our desired results. The adaptation of the code consists in the following:

1) First, we extend the function to admit three different inputs: the function we want to project, the interval of projection, the number of subintervals and the name of the function. In this way, this function can also be able to deal with different combination of function, intervals and number of subintervals, while the name of the function is just used for identification purposes of the results.

   As the function is specified through the command @(x) in the original code, we can just pass an argument that will be a function written in this manner for the other functions and lines to use it and hence extend to other functions.

2) Second, we do not just plot the L^2 function, but we also plot the real function such that there can be a visual comparison between both (the original function just showed a blue line of the L^2 projection), having an adequate legend for identification purposes.

3) We finally added final lines for the function to print the error estimated for the desired function. As this is an L^2 projection, the error is computed through the discrete approximation of the L^2 norm:

$$\|f - P_f\|_{L^2} = \sqrt{\int_a^b [f(x) - P_f(x)]^2 \, dx} \approx \sqrt{\frac{1}{n}\sum_{i=0}^{n} [f(x_i) - P_f(x_i)]^2}$$

The resulting plots are shown in Figures 1, 2, 3 and 4, while the error estimates are shown in Figure 5.

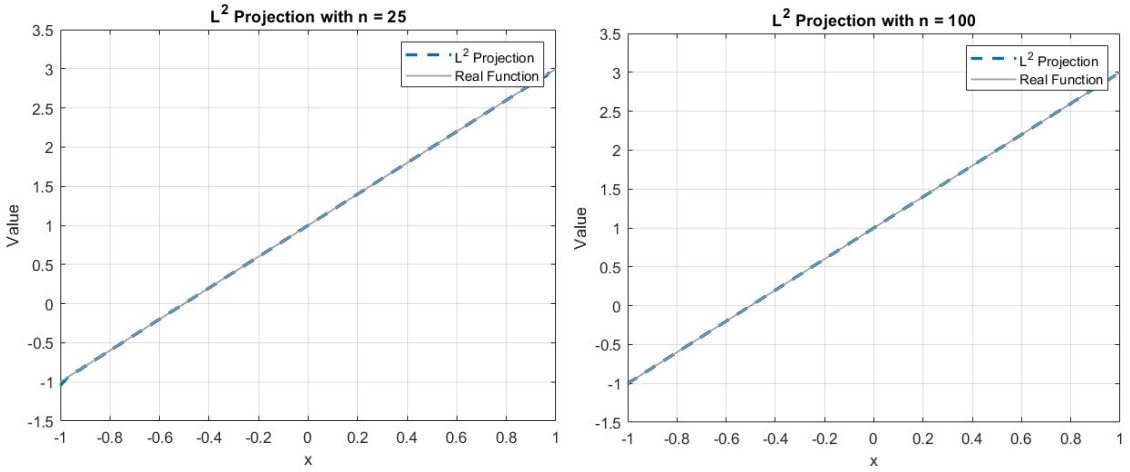

Figure 1. *Plots for the L^2 projection of $f(x)$ with $n = 25$ and $n = 100$*

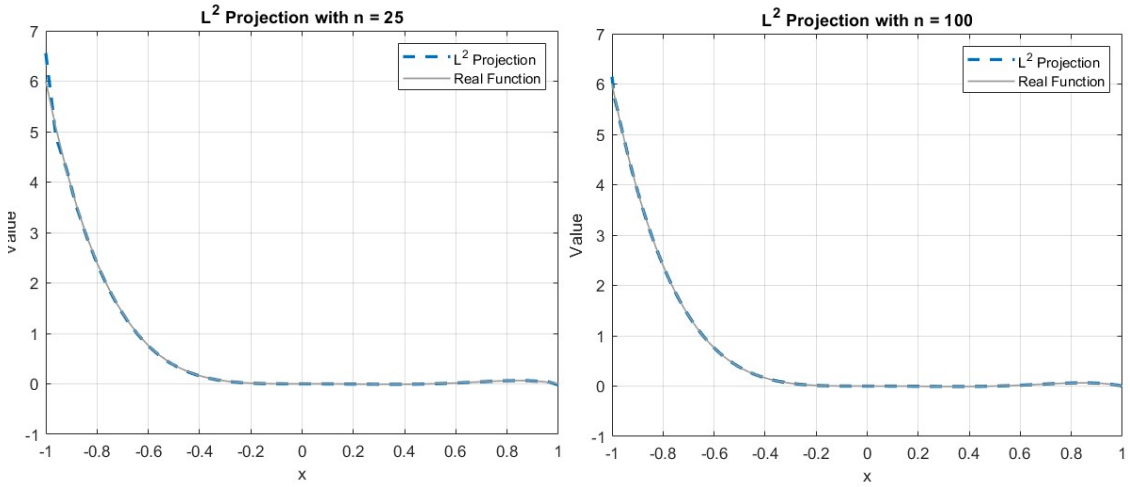

Figure 2. *Plots for the L^2 projection of $g(x)$ with $n = 25$ and $n = 100$*

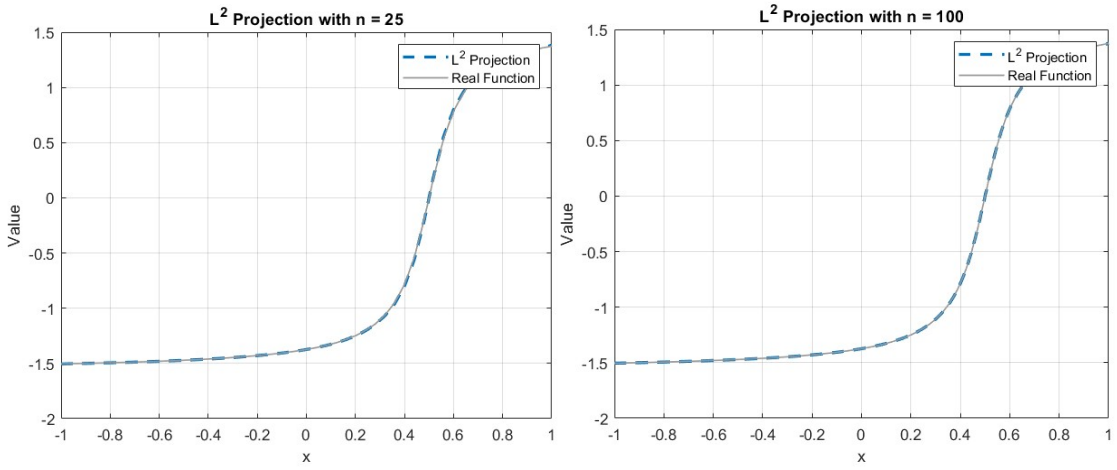

Figure 3. *Plots for the L^2 projection of $h(x)$ with $\epsilon = 0.1$, $n = 25$ and $n = 100$*

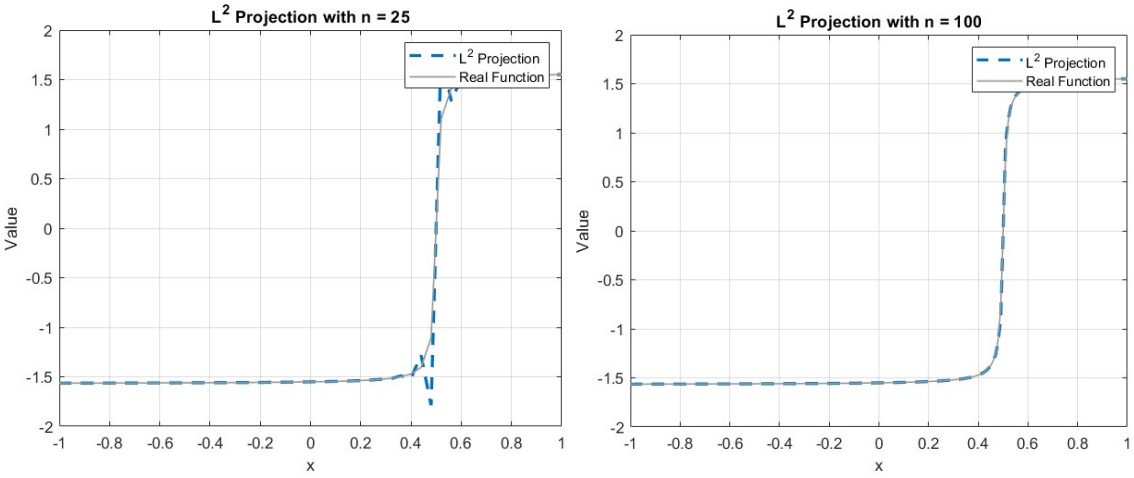

Figure 4. *Plots for the L^2 projection of $h(x)$ with $\epsilon = 0.01$, $n = 25$ and $n = 100$*

```
------------------------------------------------------------
Error estimate for f1 with 25 subintervals: 6.779893e-02
Error estimate for f2 with 25 subintervals: 5.845110e-01
Error estimate for f3_eps1 with 25 subintervals: 4.009564e-02
Error estimate for f3_eps2 with 25 subintervals: 9.778893e-01
------------------------------------------------------------
Error estimate for f1 with 100 subintervals: 1.694973e-02
Error estimate for f2 with 100 subintervals: 1.489983e-01
Error estimate for f3_eps1 with 100 subintervals: 5.233409e-03
Error estimate for f3_eps2 with 100 subintervals: 1.612514e-01
------------------------------------------------------------
```

Figure 5. *Error estimates for the different combination of functions and n values, where **eps1** corresponds to $\epsilon = 0.1$ and **eps2** corresponds to $\epsilon = 0.01$*

Exercise 2

a) Write down the equations needed to resolve numerically, using the Finite Elements Method (FEM), the 1D elliptic general problem

$$-(au')' + bu' + cu = f, \qquad x \in I = [0, L]$$
$$au'(0) = \kappa_0(u(0) - g_{0D}) + g_{0N}$$
$$-au'(L) = \kappa_L(u(L) - g_{LD}) + g_{LN}$$

Being $a = a(x), b = b(x), c = c(x)$ and $f = f(x)$ real functions with sufficiently high degrees in their derivatives.

To solve this boundary-value problem numerically, we must apply the steps used in Chapter 2 of the book, which allows us to derive the weak form of the differential equation.

The first step is to multiply a test function $v \in V$ where the space of functions V is $V = \{v: \|v\|_{L^2(I)} < \infty, \|v'\|_{L^2(I)} < \infty\}$ to obtain the variational formulation

$$\int_0^L fv\,dx = -\int_0^L (au')'v\,dx + \int_0^L bu'v\,dx + \int_0^L cuv\,dx =$$

$$= \int_0^L au'v'\,dx + \kappa_L(u(L) - g_{LD})v(L) + g_{LN}v(L) + \kappa_0(u(0) - g_{0D})v(0)$$
$$+ g_{0N}v(0) + \int_0^L bu'v\,dx + \int_0^L cuv\,dx$$

$$\Rightarrow \int_0^L fv\,dx + \kappa_L g_{LD}v(L) + \kappa_L g_{0D}v(0) = \int_0^L au'v'\,dx + \int_0^L bu'v\,dx +$$

$$+ \int_0^L cuv\,dx + \kappa_L u(L)v(L) + g_{LN}v(L) + \kappa_0 u(0)v(0) + g_{0N}v(0)$$

This is the weak form of the boundary-value problem (with its boundary conditions). By replacing the continuous space V with the discrete space of continuous piecewise linears V_h, we obtain a finite element approximation for $v \in V_h$

$$\int_0^L fv\,dx + \kappa_L g_{LD}v(L) + \kappa_L g_{0D}v(0) = \int_0^L au_h'v'\,dx + \int_0^L bu_h'v\,dx +$$

$$+\int_0^L cu_h v\,dx + \kappa_L u_h(L)v(L) + g_{LN}v(L) + \kappa_0 u_h(0)v(0) + g_{0N}v(0)$$

Because we have changed V for V_h, then the basis for V_h, which is the set of $n+1$ hat functions $\{\varphi\}_{i=0}^{n}$ defined as

$$\varphi_i = \begin{cases} (x - x_{i-1})/h_i & if\ x \in I_i \\ (x_{i+1} - x)/h_{i+1} & if\ x \in I_{i+1} \\ 0 & otherwise \end{cases}$$

can be used to approximate the solution $u(x)$ by finite elements. In this case, by using the ansatz

$$u_h = \sum_{i=0}^{n} \xi_i \varphi_i$$

and choosing $v = \varphi_i$ for $i = 0,1,2,\dots,n$, we end up with a linear system like the following:

$$(\boldsymbol{A} + \boldsymbol{R})\boldsymbol{\xi} = \boldsymbol{b} + \boldsymbol{r} \quad where$$

$$A_{ij} = \int_0^L a(x)\varphi_j'(x)\varphi_i'(x)\,dx + \int_0^L b(x)\varphi_j'(x)\varphi_i(x)\,dx + \int_0^L c(x)\varphi_j(x)\varphi_i(x)\,dx$$

$$R_{ij} = \kappa_L \varphi_j(L)\varphi_i(L) + +\kappa_0 \varphi_j(0)\varphi_i(0)$$

$$b_i = \int_0^L f(x)\varphi_j(x)\,dx$$

$$r_i = \kappa_L g_{LD}\varphi_i(L) - g_{LN}\varphi_i(L) + \kappa_L g_{0D}\varphi_i(0) - g_{0N}\varphi_i(0)$$

which is a $(n+1) \times (n+1)$ system of equations. Now, we can find the stiffness matrix $\boldsymbol{A}$ and the load vector $\boldsymbol{b}$ through the weak form of the problem. We will first start by deriving the entries of $\boldsymbol{A}$, then $\boldsymbol{R}$ and then the ones for $\boldsymbol{b}$ and $\boldsymbol{r}$.

For $|i-j| > 1$, $A_{ij} = 0$ because of the basis function's definition (do not have overlapping support). Then, we need to find $A_{i,i+1}$ and A_{ii}, just like the derivations from the book. In this case, we obtain A_{ii} by using the weak form for the relevant interval I_i and approximating the contributions from $a(x)$, $b(x)$ and $c(x)$ with the midpoint rule. For $a(x)$ we have

$$\int_{x_{i-1}}^{x_i} a(x)[\varphi_i'(x)]^2\,dx + \int_{x_i}^{x_{i+1}} a(x)[\varphi_i'(x)]^2\,dx =$$

$$= \int_{x_{i-1}}^{x_i} a(x)\frac{1}{h_i^2}\,dx + \int_{x_i}^{x_{i+1}} a(x)\frac{1}{h_{i+1}^2}\,dx \approx \frac{a_i}{h_i^2}h_i + \frac{a_{i+1}}{h_{i+1}^2}h_{i+1}$$

For $b(x)$ we have

$$\int_{x_{i-1}}^{x_i} b(x)\varphi_i(x)\varphi_i'(x)\,dx + \int_{x_i}^{x_{i+1}} b(x)\varphi_i(x)\varphi_i'(x)\,dx =$$

$$\approx b_i\varphi_i(x_{mid})\varphi_i'(x_{mid}) + b_i\varphi_i(x_{mid})\varphi_i'(x_{mid}) =$$

$$= b_i\frac{\frac{x_{i-1}+x_i}{2} - x_{i-1}}{h_i}\frac{1}{h_i}h_i - b_{i+1}\frac{x_{i+1} - \frac{x_i+x_{i+1}}{2}}{h_{i+1}}\frac{1}{h_{i+1}}h_{i+1} =$$

$$= b_i\frac{x_i - x_{i-1}}{2h_i} - b_{i+1}\frac{x_{i+1}-x_i}{2h_{i+1}} = b_i\frac{h_i}{2h_i} - b_{i+1}\frac{h_{i+1}}{2h_{i+1}} = \frac{b_i}{2} - \frac{b_{i+1}}{2}$$

And for $c(x)$ we have

$$\int_{x_{i-1}}^{x_i} c(x)[\varphi_i(x)]^2\,dx + \int_{x_i}^{x_{i+1}} c(x)[\varphi_i(x)]^2\,dx =$$

$$= \int_{x_{i-1}}^{x_i} c(x)\frac{(x-x_{i-1})^2}{h_i^2}\,dx + \int_{x_i}^{x_{i+1}} c(x)\frac{(x_{i+1}-x_i)^2}{h_{i+1}^2}\,dx \approx$$

$$\approx c_i\frac{\left(\frac{x_{i-1}+x_i}{2} - x_{i-1}\right)^2}{h_i^2}h_i + c_{i+1}\frac{\left(x_{i+1} - \frac{x_i+x_{i+1}}{2}\right)^2}{h_{i+1}^2}h_{i+1}$$

$$\approx c_i\frac{h_i^2}{h_i^2 4}h_i + c_{i+1}\frac{h_{i+1}^2}{h_{i+1}^2 4}h_{i+1} = \frac{c_i h_i}{4} + \frac{c_{i+1}h_{i+1}}{4}$$

Hence,

$$A_{ii} = \frac{a_i}{h_i} + \frac{a_{i+1}}{h_{i+1}} + \frac{b}{2} + \frac{b_{i+1}}{2} + \frac{c_i h_i}{4} + \frac{c_{i+1}h_{i+1}}{4}$$

For the superdiagonal elements (which are equal to the subdiagonal ones), we can do a similar approach and obtain

$$\int_{x_i}^{x_{i+1}} a(x)\varphi_i'(x)\varphi_{i+1}'(x)\,dx \approx -\frac{a_{i+1}}{h_{i+1}}$$

$$\int_{x_i}^{x_{i+1}} b(x)\varphi_i(x)\varphi_{i+1}'(x)\,dx \approx -\frac{b_{i+1}}{2}$$

$$\int_{x_i}^{x_{i+1}} c(x)\varphi_i(x)\varphi_{i+1}(x)\,dx \approx -\frac{c_{i+1}h_{i+1}}{4}$$

$$\Rightarrow A_{i,i+1} = -\frac{a_{i+1}}{h_{i+1}} - \frac{b_{i+1}}{2} - \frac{c_{i+1}h_{i+1}}{4}$$

Hence, the entries for the stiffness matrix $\boldsymbol{A}$ are

$$A_{ii} = \frac{a_i}{h_i} + \frac{a_{i+1}}{h_{i+1}} + \frac{b}{2} + \frac{b_{i+1}}{2} + \frac{c_i h_i}{4} + \frac{c_{i+1}h_{i+1}}{4}, \qquad \forall i = 0,1,\dots,n-1$$

$$A_{i,i+1} = -\frac{a_{i+1}}{h_{i+1}} - \frac{b_{i+1}}{2} - \frac{c_{i+1}h_{i+1}}{4}, \qquad \forall i = 0,1,\dots,n-1$$

$$A_{i+1,i} = -\frac{a_{i+1}}{h_{i+1}} - \frac{b_{i+1}}{2} - \frac{c_{i+1}h_{i+1}}{4}, \qquad \forall i = 0,1,\dots,n-1$$

$$A_{i,j} = 0, \qquad \forall i,j \;\; s.t. \;\; |i-j| > 1$$

Then, the entries of $\boldsymbol{R}$ are computed. In this case, $R_{ij} = 0$ for $\forall i,j \notin \{0,n\}$, as for any interval I_i which does not include 0 or L, φ_j and φ_i will be null. However, for R_{00} and R_{nn} we have

$$\varphi_0(0) = \frac{x_1 - x_0}{h_1} = \frac{h_1}{h_1} = 1, \qquad \varphi_n(0) = 0$$

$$\varphi_n(L) = \frac{x_n - x_{n-1}}{h_n} = \frac{h_n}{h_n} = 1, \qquad \varphi_0(L) = 0$$

$$\Rightarrow R_{00} = \kappa_0, \qquad R_{nn} = \kappa_0$$

$$R_{ij} = 0\,, \qquad \forall i \notin \{0,n\}$$

Finally, we can obtain the entries of $\boldsymbol{b}$ and $\boldsymbol{r}$ in a similar manner as in $\boldsymbol{A}$. For $i = 1,2,\dots,n-1$, we have

$$\int_{x_{i-1}}^{x_i} f(x)\varphi_i(x)\,dx + \int_{x_i}^{x_{i+1}} f(x)\varphi_i(x)\,dx =$$

$$= \int_{x_{i-1}}^{x_i} f(x)\frac{x - x_{i-1}}{h_i}\,dx + \int_{x_i}^{x_{i+1}} f(x)\frac{x_{i+1} - x_i}{h_{i+1}}\,dx \approx \frac{f_i h_i}{2} + \frac{f_{i+1}h_{i+1}}{2}$$

$$\Rightarrow\ b_i = \frac{f_i h_i}{2} + \frac{f_{i+1}h_{i+1}}{2}$$

and for $i = 0$ and $i = n$, we use

$$b_0 = \int_{x_0}^{x_1} f(x)\varphi_i(x)\,dx \approx \frac{f_0 h_1}{2} \qquad b_n = \int_{x_{n-1}}^{x_n} f(x)\varphi_i(x)\,dx \approx \frac{f_n h_n}{2}$$

so that the entries of the load vector $\boldsymbol{b}$ are

$$b_i = \frac{f_i h_i}{2} + \frac{f_{i+1}h_{i+1}}{2}, \qquad \forall i = 1,2,\dots,n-1$$

$$b_0 = \frac{f_0 h_0}{2} \qquad and \qquad b_n = \frac{f_n h_n}{2}$$

For the vector $\boldsymbol{r}$, we do the same reasoning as with $\boldsymbol{R}$ to see that $r_i = 0$ for $i \notin \{0, n\}$. For r_0 and r_n, $\varphi_0(0) = 1$ and $\varphi_n(L) = 1$, so that

$$r_0 = \kappa_0 g_{0D} - g_{0N} \quad and \quad r_n = \kappa_L g_{LD} - g_{LN}$$

$$r_i = 0\,, \qquad \forall i \notin \{0, n\}$$

Consequently, we have derived now all the entries of all the matrix involved in the linear system, so that we can obtain the equations to compute the solution numerically.

b) Modify accordingly the programs of the book to solve numerically the problem:

$$u'' + u = x^2, x \in I = [0,1]$$
$$u(0) = 0$$
$$u(1) = 0$$

Use the midpoint or trapezoidal approximation for the numerical integrations. Compare the numerical solution with the exact solution given by:

$$u(x) = \frac{\sin x + 2\sin(1-x)}{\sin 1} + x^2 - 2$$

Now that we have obtain the expressions for the stiffness matrix and the load vector, we can implement a code just like those in the book in order to obtain the numerical solution for the boundary-value problem. In this case, we use the trapezoidal approximation for the numerical integrations.

The modified code is available in the Matlab code Ex2b.m, available at the folder. The main modifications to the functions are the ones related to the StiffnessAssembler, the LoadAssembler, and the PoissonSolver1D functions. Firstly, we take into account that for this problem $a(x) = 1$, $b(x) = 0$, $c(x) = 1$ and $f(x) = x^2$, and that $a(x)$ has a positive sign in this differential equation, which is important because we will be required to change signs in order to use the results we obtained before, so that

$$u'' + u = x^2 \;\Rightarrow\; -u'' - u = -x^2 \;\Rightarrow\; a(x) = 1, c(x) = -1, f(x) = -x^2$$

$$u(0) = 0 \;\Rightarrow\; g_{0D} = g_{0N} = 0 \;\; and \;\; \kappa_0 = \infty \;\; or \; large \; value$$

$$u(1) = 0 \;\Rightarrow\; g_{1D} = g_{1N} = 0 \;\; and \;\; \kappa_1 = \infty \;\; or \; large \; value$$

In the StiffnessAssembler function we have added the contribution of the terms $b(x)$ and $c(x)$ in a general way for both the diagonal and the off-diagonal elements, so that it serves for all possible values of $b(x)$ and $c(x)$. We have done this using the midpoint rule for numerical integration. Additionally, we had to add the terms of g_{0n} and g_{1n} to $A(1,1)$ and $A(n+1, n+1)$ for taking into account possible Neumann conditions (in this case there are none).

When it comes to the LoadAssembler function, we change it to integrate through the midpoint rule (we are meant to choose just one method for numerical integration). The SourceAssembler function is not change because we take into account the Dirichlet conditions, even though we changed the name to identify them as g_{0D} and g_{LD}.

Finally, the PoissonSolver1D is changed by adding as inputs the different parameters of the problem such as interval, the spacing h, the kappas κ_0 and κ_1, the terms for the Dirichlet and Neumann conditions, and a boolean variable called $AntlcSol$ in order to take into account the analytical solution of the problem and compute its error when available. Finally, we also added a graph that allows to compare the analytical solution and the estimated solution, or just the latter if the analytical one is not available.

For this problem, the results of the approximated solution $u(x)$ with $h = 0.1$ is plotted against the analytical solution provided in Figure 6, and the error of the estimation is also shown in Figure 7. As we can see, the approximation works quite well, having a maximum error of order 10^{-3}.

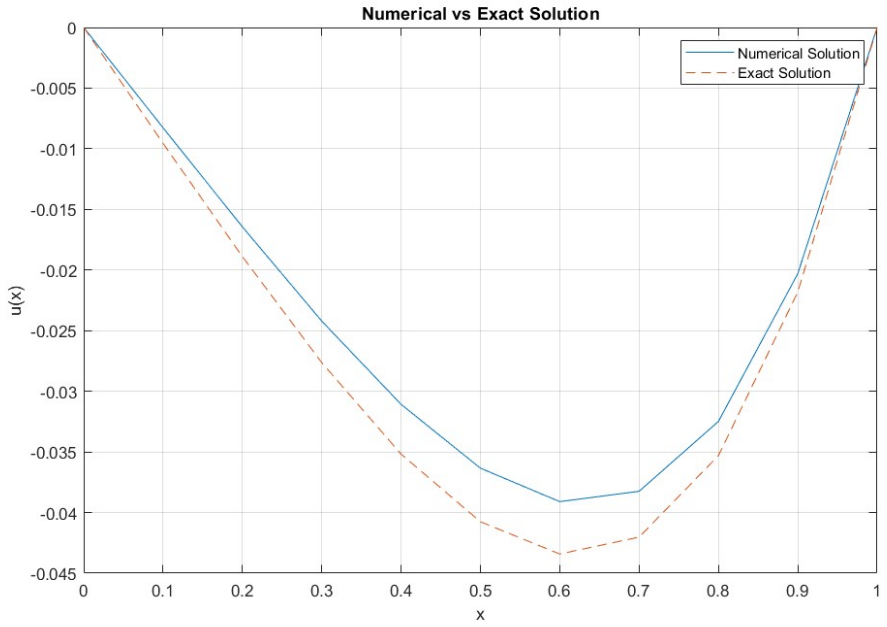

Figure 6. *Numerical solution $u(x)$ for the boundary-value problem and exact solution*

```
>> ex2b
Maximum error: 4.317546e-03
```

Figure 7. *Estimated error between the approximated and the analytical solution*

c) Solve now numerically the following boundary value problem

$$u'' + u' - 2u = x, x \in I = [0,1]$$
$$u(0) = 0$$
$$u'(1) = 1$$

In order to solve the boundary-value problem, we use the same code that we used in Ex2b.m, but now adapted to this problem in Ex2c.m. Because our code is general for this type of problems, we do not need to change anything but to recall the modifications explained in part b).

We obtain the parameters like before, so that

$$u'' + u' - 2u = x \;\Rightarrow\; -u'' - u' + 2u = -x$$

$$\Rightarrow \quad a(x) = 1, b(x) = -1, c(x) = -2, f(x) = -x$$

$$u(0) = 0 \quad \Rightarrow \quad g_{0D} = g_{0N} = 0 \quad and \quad \kappa_0 = \infty \quad or \; large \; value$$

$$u'(1) = 1 \quad \Rightarrow \quad g_{0D} = 0, g_{1N} = -1 \quad and \quad \kappa_1 = 0$$

By adapting the parameters and the functions that define them for this specific problem, we can obtain the approximated solution $u(x)$ with $h = 0.1$, shown in Figure 8.

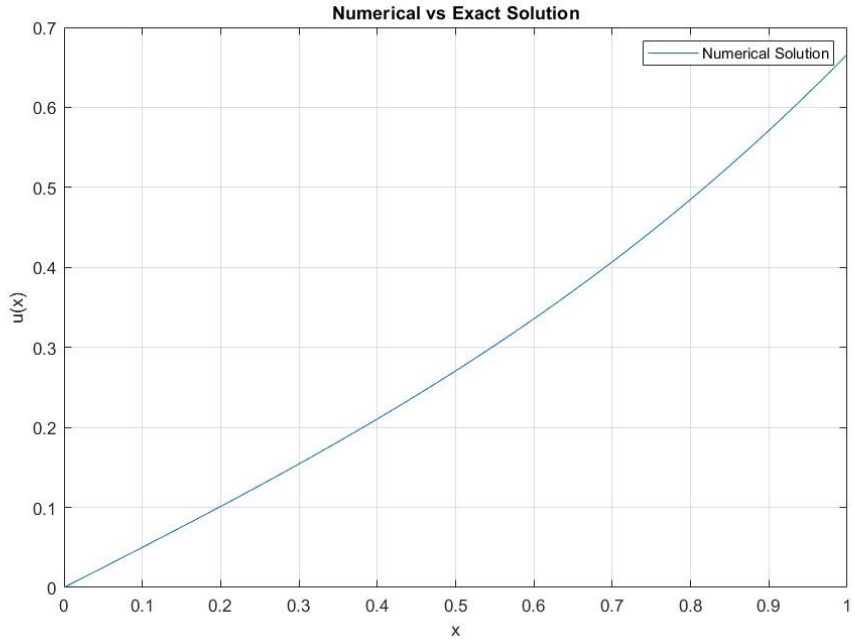

Figure 8. *Numerical solution $u(x)$ for the boundary-value problem*