WI4011-17

Computational Fluid Dynamics

Assignment 1.1

Deadline - 23:59, March 17, 2024

Instructions and assessment criteria to keep in mind:

- A submission for Assignment 1.1 is **necessary for passing** WI4011-17.
- Group submission is accepted but not mandated for this assignment. Recommended group size is 2 and maximum group size is 3 in exceptional cases.
- The report should be **typed in LaTeX or Word** and should be submitted in PDF format. It must also contain the code to any computer program that you used for numerical computations.
- The **deadline** for uploading your solutions to Brightspace is 23:59, March 17, 2024.
- Late submissions will NOT be accepted.
- Provide **clear and motivated answers** to the questions. No/reduced points will be awarded if your solutions are unaccompanied by explanations.

1 Bubnov-Galerkin Finite Element Method: 1D (9 points)

Question 1.1

Question (1 point)

Derive a weak form of the following scalar convection-diffusion equation with appropriately defined function spaces. Given, $u, \varepsilon \in \mathbb{R}$, $f : (0, L) \to \mathbb{R}$, find $\varphi : [0, L] \to \mathbb{R}$ such that

$$\begin{aligned} \frac{\mathrm{d}(u\varphi)}{\mathrm{d}x} - \frac{\mathrm{d}}{\mathrm{d}x}\left(\varepsilon \frac{\mathrm{d}\varphi}{\mathrm{d}x}\right) &= f, \quad x \in (0, L) \\ \varphi(0) &= 0, \\ \varphi(L) &= 1. \end{aligned} \tag{1}$$

In order to write down the weak form, we need to select the correct function spaces for both the solution φ, $\mathcal{V}$, and the test functions, $\mathcal{W}$. Since one objective of the weak form is to reduce the regularity constraints of the solution, we already know that we will reduced the order of the derivatives on the second term by performing integration by parts. Therefore we will only have first order derivatives on our solution φ. For this reason we choose the following spaces:

$$\mathcal{V} := \{\varphi \in \mathcal{H}^1([0, L]) : \varphi(0) = 0, \varphi(L) = 1\}, \tag{2}$$

$$\mathcal{W} := \{w \in \mathcal{H}^1([0, L]) : w(0) = w(L) = 0\}. \tag{3}$$

Note that, as usual, we impose the essential boundary conditions directly on the solution space $\mathcal{V}$. Because we do this, the trial space, contains homogenous boundary conditions at the same locations where we enforce the essential boundary conditions on the solution space.

To derive the weak form, we multiply the strong form (1) by a test function $w \in \mathcal{W}$ and integrate over the domain as

$$\int_0^L w[\frac{\mathrm{d}}{\mathrm{d}x}(u\varphi) - \frac{\mathrm{d}}{\mathrm{d}x}(\varepsilon\frac{\mathrm{d}}{\mathrm{d}x}\varphi)]\,dx = \int_0^L wf\,dx. \tag{4}$$

Integrating the second-derivative term by parts and invoking homogenous Dirichlet boundary conditions for the trial function w, we get

$$\int_0^L w\frac{\mathrm{d}}{\mathrm{d}x}(u\varphi) + \frac{\mathrm{d}w}{\mathrm{d}x}(\varepsilon\frac{\mathrm{d}\varphi}{\mathrm{d}x})\,dx - \cancelto{0}{[w\varepsilon\frac{\mathrm{d}\varphi}{\mathrm{d}x}]_{x=0}^{x=L}} = \int_0^L wf\,dx. \tag{5}$$

Finally, we obtain a weak form for the strong form (1) as: Given $f : (0, L) \to \mathbb{R}$, find $\varphi \in \mathcal{V}$ such that

$$\int_0^L \left[w\frac{\mathrm{d}}{\mathrm{d}x}(u\varphi) + \frac{\mathrm{d}w}{\mathrm{d}x}(\varepsilon\frac{\mathrm{d}\varphi}{\mathrm{d}x})\right]\,dx = \int_0^L wf\,dx\,, \qquad \forall w \in \mathcal{W}. \tag{6}$$

Question 1.2

Question (0.5 points)

We will use a C^0 piecewise-linear function space denoted by $V_{1,h}^0$ for performing a Bubnov-Galerkin approximation of the weak form derived above. Define the trial and test function spaces $\mathcal{V}^h$ and $\mathcal{W}^h$ for your approximation.

The space of piecewise linear functions $V_{1,h}^0$ on a mesh with n elements and $(n+1)$ vertices is given by

$$V_{1,h}^0 := \mathrm{span}\{B_i\}_{i=1}^{n+1} \tag{7}$$

where the basis B_i are the usual hat functions, explicitly defined in the text of the next question.

The discrete solution space, $\mathcal{V}_h$, and the discrete test space, $\mathcal{W}_h$, are given by

$$\mathcal{V}_h := \left\{v \in V_{1,h}^0 : v(0) = 0, \quad v(L) = 1\right\}\,, \tag{8}$$

$$\mathcal{W}_h := \left\{v \in V_{1,h}^0 : v(0) = v(L) = 0\right\}\,. \tag{9}$$

Question 1.3

Question (2.5 points)

Derive the linear system corresponding to the Bubnov-Galerkin approximation as per the above setup using a basis of *hat* functions B_i over a uniform grid with spacing h. Specifically, show that the linear system that needs to be solved for computing $\varphi_h(x) = \sum_{i=0}^{N+1} \phi_i B_i(x)$ is given as,

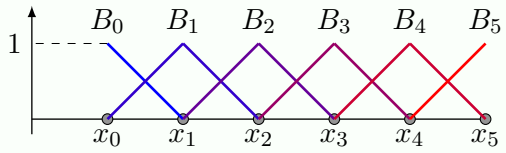
$$h\left[u\frac{\phi_{i+1}-\phi_{i-1}}{2h} - \epsilon\frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{h^2}\right] = \int_{x_{i-1}}^{x_{i+1}} B_i f \,\mathrm{d}x \,, i = 1,\ldots,N \tag{10}$$

where $\phi_0 = 0$ and $\phi_{N+1} = 1$.

Hint: Recall that $V_{1,h}^0$ is spanned by the hat basis functions $B_i(x), i = 0,\ldots,N+1$, defined over $[0, L]$ as

$$B_i(x) = \begin{cases} \frac{x-x_{i-1}}{h} & x \in [x_{i-1}, x_i], \\ \frac{x_{i+1}-x}{h} & x \in [x_i, x_{i+1}], \\ 0 & \text{otherwise}, \end{cases} \tag{11}$$

where $h = \frac{L}{N+1}$. See the figure below for an example of these functions for $N = 4$.



Thus, any $f_h \in V_{1,h}^0$ can be expressed as a linear combination of the basis functions as

$$f_h(x) = \sum_{i=0}^{N+1} f_i B_i(x) \,, \tag{12}$$

with $f_i \in \mathbb{R}, i = 0,\ldots,N+1$.

As noted above, our approximate solution, φ_h, can be expressed as a linear combination of the basis (together with the boundary conditions)

$$\varphi_h(x) = \sum_{i=1}^{n+1} \varphi_i B_i(x) = \sum_{i=2}^{n} \varphi_i B_i(x) + \varphi(0)\cdot B_1(x) + \varphi(L)\cdot B_{n+1}(x) = \sum_{i=2}^{n} \varphi_i B_i(x) + B_{n+1}(x) \,. \tag{13}$$

The Galerkin equations, i.e., the discrete version of the weak form presented above, (6), are then

$$\sum_{i=2}^{n} \varphi_i \left(\int_0^L B_j u \frac{\mathrm{d}B_i}{\mathrm{d}x}\,\mathrm{d}x + \int_0^L \frac{\mathrm{d}B_j}{\mathrm{d}x}\varepsilon\frac{\mathrm{d}B_i}{\mathrm{d}x}\,dx \right) = \int_0^L B_j f \,\mathrm{d}x - \\ - \int_0^L B_j u \frac{\mathrm{d}B_{n+1}}{\mathrm{d}x}\,\mathrm{d}x - \int_0^L \frac{\mathrm{d}B_j}{\mathrm{d}x}\varepsilon\frac{\mathrm{d}B_{n+1}}{\mathrm{d}x})\,dx \,, \quad \forall j = 2,\ldots,n. \tag{14}$$

Since this is a simple 1D case, we can figure out the explicit form of the system of equations on each vertex i. We know that the support of each basis associated to the node i is comprised of the elements $(i-1)$ and i. Therefore the integrals over most pairs of basis are zero, and only the integrals over adjacent basis are nonzero. This means that for an interior vertex, i, outside the influence of the boundary, i.e.,

$2 < i < n$, we have

$$\sum_{i=j-1}^{j+1} \varphi_i \left(\int_{x_{j-1}}^{x_{j+1}} B_j u \frac{\mathrm{d}B_i}{\mathrm{d}x}\,\mathrm{d}x + \int_{x_{j-1}}^{x_{j+1}} \frac{\mathrm{d}B_j}{\mathrm{d}x} \varepsilon \frac{\mathrm{d}B_i}{\mathrm{d}x}\,dx \right) = \int_{x_{j-1}}^{x_{j+1}} B_j f\,\mathrm{d}x\,, \quad \forall j = 2, \ldots, n. \tag{15}$$

We can expand this expression to get

$$\begin{aligned} \varphi_{j-1} \left(\int_{x_{j-1}}^{x_{j+1}} B_j u \frac{\mathrm{d}B_{j-1}}{\mathrm{d}x}\,\mathrm{d}x + \int_{x_{j-1}}^{x_{j+1}} \frac{\mathrm{d}B_j}{\mathrm{d}x} \varepsilon \frac{\mathrm{d}B_{j-1}}{\mathrm{d}x}\,dx \right) + \varphi_j \left(\int_{x_{j-1}}^{x_{j+1}} B_j u \frac{\mathrm{d}B_j}{\mathrm{d}x}\,\mathrm{d}x + \int_{x_{j-1}}^{x_{j+1}} \frac{\mathrm{d}B_j}{\mathrm{d}x} \varepsilon \frac{\mathrm{d}B_j}{\mathrm{d}x}\,dx \right) + \\ + \varphi_{j+1} \left(\int_{x_{j-1}}^{x_{j+1}} B_j u \frac{\mathrm{d}B_{j+1}}{\mathrm{d}x}\,\mathrm{d}x + \int_{x_{j-1}}^{x_{j+1}} \frac{\mathrm{d}B_j}{\mathrm{d}x} \varepsilon \frac{\mathrm{d}B_{j+1}}{\mathrm{d}x}\,dx \right) = \int_{x_{j-1}}^{x_{j+1}} B_j f\,\mathrm{d}x\,, \quad \forall j = 2, \ldots, n. \end{aligned} \tag{16}$$

To make our lives easier, we just introduce some notation. We introduce the local basis as

$$b_i^1(x) := \frac{x_{i+1} - x}{x_{i+1} - x_i}, \qquad b_i^2(x) := \frac{x - x_i}{x_{i+1} - x_i}\,. \tag{17}$$

and then we rewrite (11) as

$$B_i(x) = \begin{cases} b_{i-1}^2(x), & x \in [x_{i-1}, x_i], \\ b_i^1(x), & x \in [x_i, x_{i+1}], \\ 0 & \text{otherwise, .} \end{cases} \tag{18}$$

With this notation, and using the definition of the basis functions, we can rewrite our Galerkin equations as

$$\begin{aligned} \varphi_{j-1} \left(\int_{x_{j-1}}^{x_j} b_{j-1}^2 u \frac{\mathrm{d}b_{j-1}^1}{\mathrm{d}x}\,\mathrm{d}x + \int_{x_{j-1}}^{x_j} \frac{\mathrm{d}b_{j-1}^2}{\mathrm{d}x} \varepsilon \frac{\mathrm{d}b_{j-1}^1}{\mathrm{d}x}\,dx \right) + \\ \varphi_j \left(\int_{x_{j-1}}^{x_j} b_{j-1}^2 u \frac{\mathrm{d}b_{j-1}^2}{\mathrm{d}x}\,\mathrm{d}x + \int_{x_{j-1}}^{x_j} \frac{\mathrm{d}b_{j-1}^2}{\mathrm{d}x} \varepsilon \frac{\mathrm{d}b_{j-1}^2}{\mathrm{d}x}\,dx + \int_{x_j}^{x_{j+1}} b_j^1 u \frac{\mathrm{d}b_j^1}{\mathrm{d}x}\,\mathrm{d}x + \int_{x_j}^{x_{j+1}} \frac{\mathrm{d}b_j^1}{\mathrm{d}x} \varepsilon \frac{\mathrm{d}b_j^1}{\mathrm{d}x}\,dx \right) + \\ + \varphi_{j+1} \left(\int_{x_j}^{x_{j+1}} b_j^1 u \frac{\mathrm{d}b_j^2}{\mathrm{d}x}\,\mathrm{d}x + \int_{x_j}^{x_{j+1}} \frac{\mathrm{d}b_j^1}{\mathrm{d}x} \varepsilon \frac{\mathrm{d}b_j^2}{\mathrm{d}x}\,dx \right) = \int_{x_{j-1}}^{x_j} b_{j-1}^2 f\,\mathrm{d}x + \int_{x_j}^{x_{j+1}} b_j^1 f\,\mathrm{d}x\,, \quad \forall j = 2, \ldots, n. \end{aligned} \tag{19}$$

If we now substitute the expressions of the local basis and perform the integrals, we directly see that we obtain the desired expression.

Question 1.4

Question (2 points)

Consider the case $f = 0$. [a]

[a] Note that, in this case, system (10) has the same form as a standard central finite difference discretization of the convection-diffusion equation (1) on the chosen mesh.

Question 1.4.1

Question (0.5 points)

Show that each equation in system (10) can be expressed in the form $c_0\phi_{i+1} - 2c_1\phi_i + c_2\phi_{i-1} = 0$; give explicit expressions for c_0, c_1, c_2.

Simple manipulations lead to

$$c_0 = \frac{u}{2} - \frac{\epsilon}{h}\,, \tag{20}$$

$$c_1 = -\frac{\epsilon}{h}\,, \tag{21}$$

$$c_2 = -\frac{u}{2} - \frac{\epsilon}{h}\,. \tag{22}$$

We can get a simpler set of coefficients if we multiply the equation by $-\frac{h}{\epsilon}$

$$c_0 = 1 - \mathrm{Pe}_h\,, \tag{23}$$

$$c_1 = 1\,, \tag{24}$$

$$c_2 = 1 + \mathrm{Pe}_h\,. \tag{25}$$

with $\mathrm{Pe}_h := \frac{uh}{2\epsilon}$. Other possible results, multiple of this result are possible. I think this final result is one of the simplest, and simpler makes life more enjoyable.

Question 1.4.2

Question (0.5 points)

Assume that the solution of the linear system (10) takes the form $\phi_i \sim \zeta^i$ and find the two possible values $\zeta_1 := \{\zeta_1^1, \ldots, \zeta_1^{N+1}\}$ and $\zeta_2 := \{\zeta_2^1, \ldots, \zeta_2^{N+1}\}$ that satisfy the difference equations (10).

The characteristic polynomial of this difference equation is

$$c_0\lambda^2 - 2c_1\lambda + c_2 = 0\,, \tag{26}$$

and its roots are

$$\lambda_\pm = \frac{1 \pm \sqrt{1 - (1 - \mathrm{Pe}_h)(1 + \mathrm{Pe}_h)}}{1 - \mathrm{Pe}_h} = \frac{1 \pm \mathrm{Pe}_h}{1 - \mathrm{Pe}_h}\,, \tag{27}$$

i.e.,

$$\lambda_- = 1\,, \qquad \lambda_+ = \frac{1 + \mathrm{Pe}_h}{1 - \mathrm{Pe}_h}\,. \tag{28}$$

For this reason, the solutions to the difference equation are of the form

$$u_k = c_3\lambda_- + c_4\lambda_+^k\,. \tag{29}$$

If we now look back at the question, we see that

$$\zeta_1^k = 1\,, \quad \text{and} \quad \zeta_2^k = \left(\frac{1 + \mathrm{Pe}_h}{1 - \mathrm{Pe}_h}\right)^k\,. \tag{30}$$

This is a standard approach to solve difference equations. If you recall it is similar to how one solves ODEs.

Question 1.4.3

Question (0.5 points)

A general solution of the equations with $f = 0$ is then given by $\phi_i = c_3\zeta_1^i + c_4\zeta_2^i$; find the values of c_3 and c_4 using the boundary conditions.

Here we consider the case of N cells, and therefore $(N + 1)$ vertices, numbered from 0 to N. If we impose the left boundary condition we get

$$\varphi(0) := \varphi_0 = c_3 + c_4\lambda_+^0 = c_3 + c_4 = 0 \quad \Rightarrow \quad c_3 = -c_4\,. \tag{31}$$

If we then impose the right boundary condition we get

$$\varphi(L) := \varphi_N = c_3 - c_3\lambda_+^N = 1 \quad \Rightarrow \quad c_3 = \left(1 - \lambda_+^N\right)^{-1} . \tag{32}$$

If you decided to use a different total number of vertices, then this final result would look different. The point is that the exponent should be equal to the last index of your vertices.

Question 1.4.4

Question (0.5 points)

Find an expression for $\overline{\epsilon}$ such that the solution to the following modified equations is nodally exact, i.e., $\phi_i = \phi(x_i)$,

$$u\frac{\phi_{i+1} - \phi_{i-1}}{2h} - (\epsilon + \overline{\epsilon})\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} = 0 , \quad i = 1, \ldots, N . \tag{33}$$

Hint: Recall from the class that the exact solution is explicitly available in this case.

This question was tricky. I will spare you the details of the derivation because you can find them in the book by Donea. The idea is the following. We know the exact solution to this problem, which is

$$\varphi(x) = \frac{e^{\frac{u}{\epsilon}x} - 1}{e^{\frac{uL}{\epsilon}} - 1} . \tag{34}$$

We can now evaluate it at the vertices of the mesh, i.e., $x_k = hk$, $k = 0, \ldots, N$, with h the spacing between the vertices. These evaluations are $\varphi(x_k)$, which are the φ_k we have been discussing above. If we want to have a nodally exact numerical solution, we can substitute in the difference equation these values for the $\varphi(x_k)$. We can then try to find out the values of the coefficients c_0, c_1, and c_2 that enable us to exactly satisfy the equation.

This is done in instructive (if somewhat boring) detail in the book by Donea. If you follow the book, you see that the solutions for the coefficients satisfy the following equations

$$\begin{cases} c_2 - 2c_1 + c_0 = 0 , \\ -c_2 + c_0 = \dfrac{u}{h} , \\ c_2 e^{-2\mathrm{Pe}_h} - 2c_1 + c_0 e^{2\mathrm{Pe}_h} = 0 , \end{cases} \tag{35}$$

and the solution is

$$\begin{cases} c_2 = -\dfrac{u}{2h}\left(1 + \coth \mathrm{Pe}_h\right) , \\ c_1 = -\dfrac{u}{2h}\coth \mathrm{Pe}_h , \\ c_0 = \dfrac{u}{2h}\left(1 + \coth \mathrm{Pe}_h\right) . \end{cases} \tag{36}$$

This solution if for $f = 1$. If we do the same for our case $f = 0$ we get only two of the three equations

$$\begin{cases} c_2 - 2c_1 + c_0 = 0 , \\ c_2 e^{-2\mathrm{Pe}_h} - 2c_1 + c_0 e^{2\mathrm{Pe}_h} = 0 . \end{cases} \tag{37}$$

Since we still have three unknowns, the system is underdetermined. This means that we are free to add a third equation, as long as it is compatible with the other two. This means we can choose as third equation, the missing equation from the book and therefore borrow the solution (36).

If we now substitute these coefficient into the original difference equation and rearrange it to get it in the formed asked, we find

$$\overline{\epsilon} = \epsilon\left(\mathrm{Pe}_h \coth \mathrm{Pe}_h - 1\right) . \tag{38}$$

Question 1.5

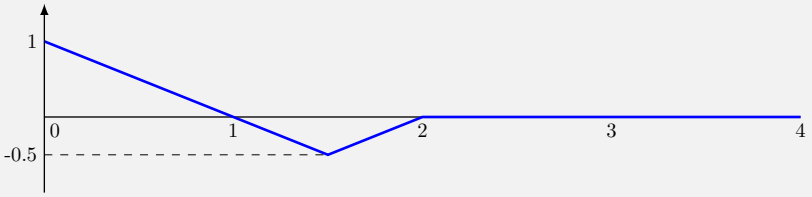
Question (3 points)

Choose $L = 4$, $u = 1$ and $N = 14$. Implement the Bubnov Galerkin approximation (10) in a computer code and compare it with the following discretization scheme modified from central differences.

$$u\frac{\phi_{i+1} - \phi_{i-1}}{2h} - (\epsilon + \bar{\epsilon})\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} = f(x_i) \ . \tag{39}$$

Present the solutions for the following cases.

- $\epsilon \in \{0.01, 0.1, 1\}$ and $f = 0$;
- $\epsilon \in \{0.01, 0.1, 1\}$ and f defined as below.



Present three plots for each subitem, i.e., one plot for each value of ϵ. Each plot should contain the numerical solution using the above two methods, as well as the exact analytical solution when available or the solution from the Bubnov-Galerkin method on a refined mesh of $N = 1000$ when the exact solution is not available.
Discuss the results.

The code is in the Jupyter notebook.

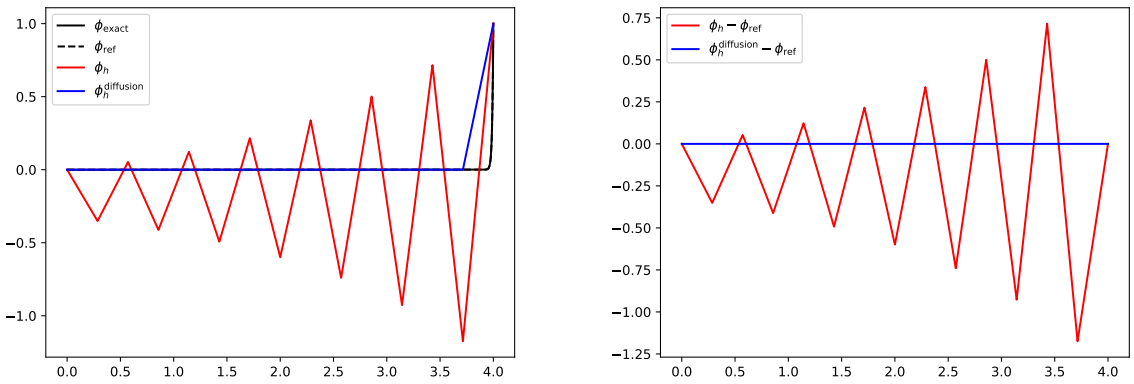


Figure 1: Left: comparison of solution with standard Galerkin formulation (φ_h) and solution with added diffusion ($\varphi_h^{\text{diffusion}}$), for $\epsilon = 0.01$, and $f = 0$. Right: nodal error for the two numerical solutions.

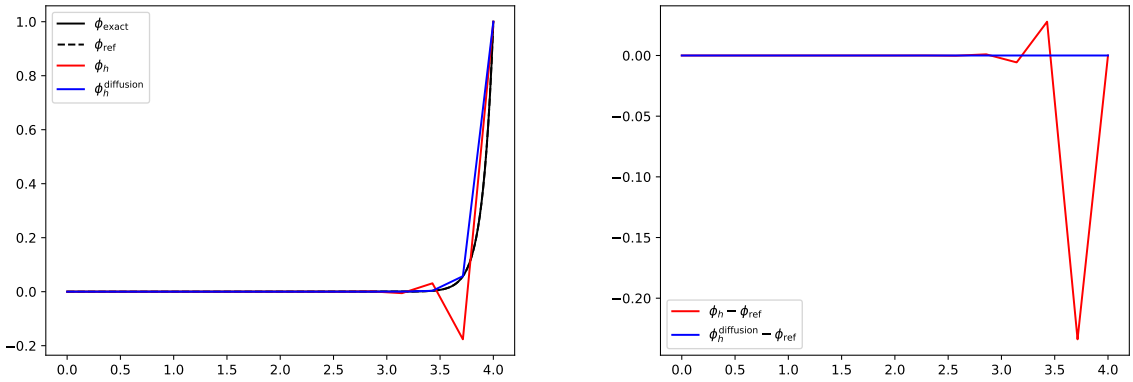


Figure 2: Left: comparison of solution with standard Galerkin formulation (φ_h) and solution with added diffusion ($\varphi_h^{\text{diffusion}}$), for $\epsilon = 0.1$, and $f = 0$. Right: nodal error for the two numerical solutions.

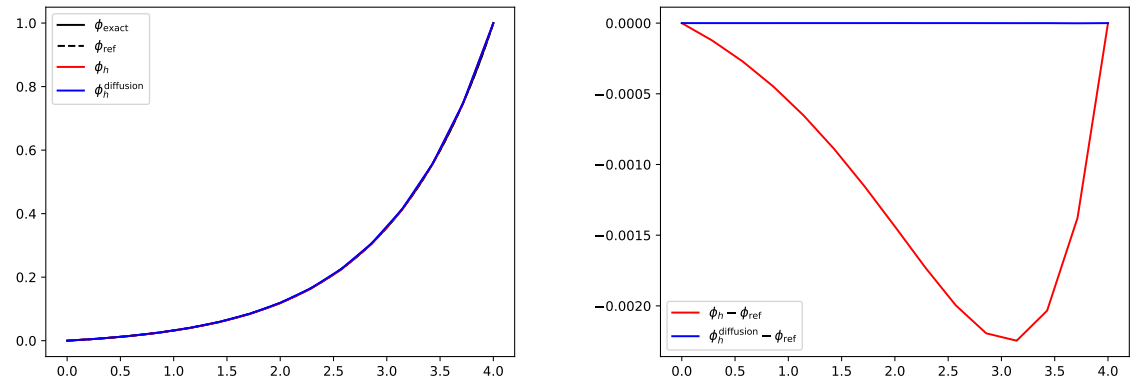


Figure 3: Left: comparison of solution with standard Galerkin formulation (φ_h) and solution with added diffusion ($\varphi_h^{\text{diffusion}}$), for $\epsilon = 1.0$, and $f = 0$. Right: nodal error for the two numerical solutions.

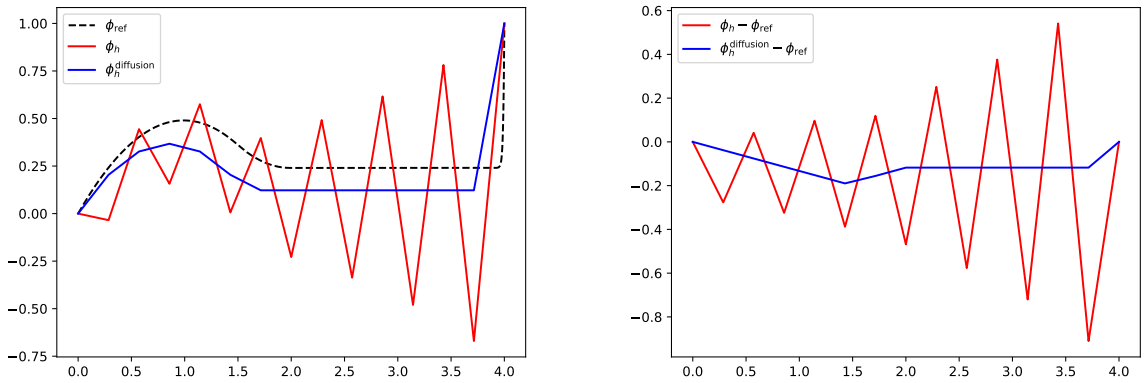


Figure 4: Left: comparison of solution with standard Galerkin formulation (φ_h) and solution with added diffusion ($\varphi_h^{\text{diffusion}}$), for $\epsilon = 0.01$, and variable f. Right: nodal error for the two numerical solutions.

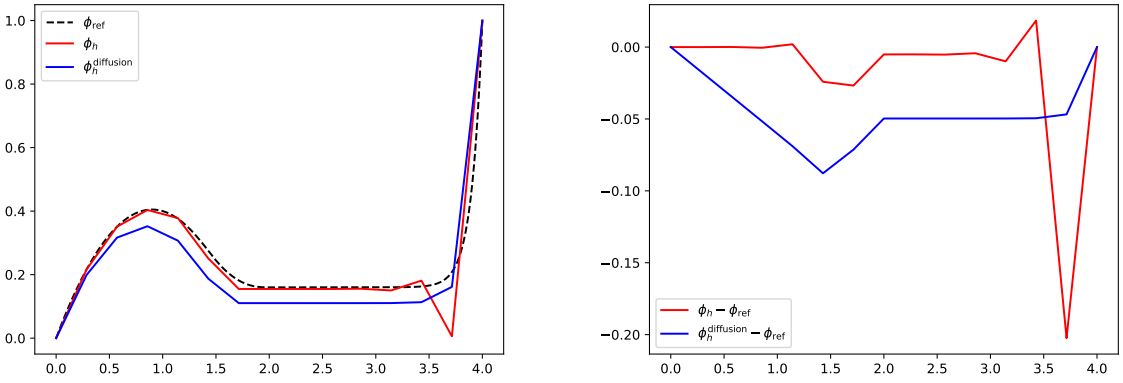


Figure 5: Left: comparison of solution with standard Galerkin formulation (φ_h) and solution with added diffusion ($\varphi_h^{\text{diffusion}}$), for $\epsilon = 0.1$, and variable f. Right: nodal error for the two numerical solutions.

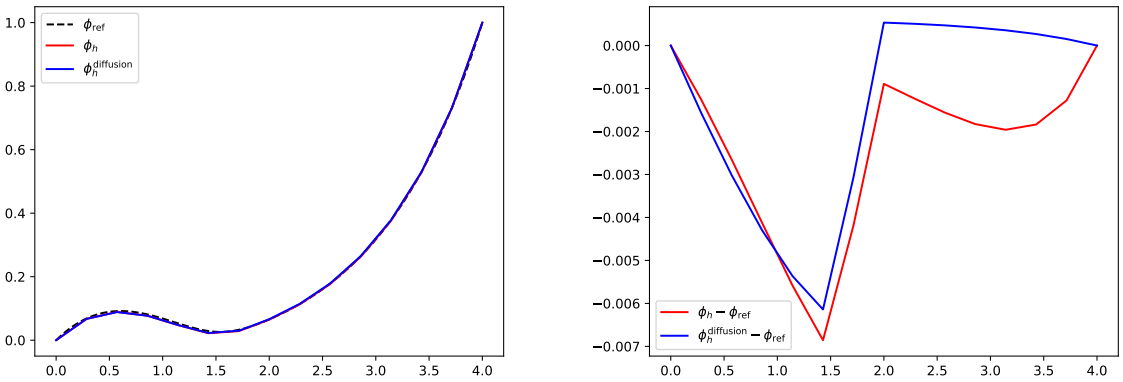


Figure 6: Left: comparison of solution with standard Galerkin formulation (φ_h) and solution with added diffusion ($\varphi_h^{\text{diffusion}}$), for $\epsilon = 1.0$, and variable f. Right: nodal error for the two numerical solutions.

Bubnov-Galerkin Finite Element Method: 2D (8 points)

Consider the following weak form[1] of the scalar convection-diffusion equation on $\Omega = (0, L)^2$ with homogeneous boundary conditions, i.e., $\varphi|_{\partial\Omega} = 0$.

Find $\varphi \in \mathcal{V} := \{v \in L^2(\Omega) \,|\, \nabla v \in L^2(\Omega)^2 \text{ and } \varphi|_{\partial\Omega} = 0\}$ such that

$$\int_\Omega -\nabla w \cdot \mathbf{u}\varphi + \nabla w \cdot \epsilon \nabla \varphi \, dx = \int_\Omega w f \, dx \,, \quad \forall w \in \mathcal{W} \,. \tag{40}$$

In this case, since the space $\mathcal{V}$ contains only essential boundary conditions along its whole boundary, we have have $\mathcal{W} = \mathcal{V}$. In the following, we will approximate the solution to weak form (40) using the Bubnov-Galerkin method and C^0 piecewise-bilinear finite element spaces $V^0_{1,h}$. The finite element space will be based on a *uniform cartesian mesh* with $(n \times n)$ elements of size $h = \frac{L}{n}$ in each direction.

The basis functions that span $V^0_{1,h}$ are denoted with B_{ij}, $0 \le i, j \le n$, and they are built as the tensor-product of univariate piecewise-linear basis functions. That is, for any $f \in V^0_{1,h}$, there exists coefficients $f_{ij} \in \mathbb{R}$, $0 \le i, j \le n$, such that

$$\begin{aligned} f &= \sum_{i=0}^{n} \sum_{j=0}^{n} f_{ij} B_{ij} \,, \\ B_{ij}(x, y) &:= B_i(x) B_j(y) \,, \end{aligned} \tag{41}$$

where B_i are defined similarly to eq. (11).

With the above choice of finite elements, the trial and test function spaces are given as,

$$\begin{aligned} \mathcal{V}_h &:= \left\{ f \in V^0_{1,h} \;:\; f|_{\partial\Omega} = 0 \right\} \,, \\ \mathcal{W}_h &:= \mathcal{V}_h \,. \end{aligned} \tag{42}$$

Note that $\mathcal{V}_h \subseteq \mathcal{V}$. Finally, we obtain the Galerkin approximation of the weak form (40) as follows. Find $\phi_h \in \mathcal{V}_h$ such that

$$\int_\Omega \left[-\nabla w_h \cdot \mathbf{u}\phi_h + \nabla w_h \cdot \epsilon \nabla \phi_h \right] \, d\Omega = \int_\Omega w_h f \, \mathrm{d}\Omega \,, \quad \forall w_h \in \mathcal{W}_h \,. \tag{43}$$

Since $\mathcal{W}_h$ is a finite dimensional space with a basis, i.e., $\mathcal{V}_h = \mathcal{W}_h = \text{span}\{B_{ij}(x, y)\}_{i,j=1}^{n-1}$, every element $w_h \in \mathcal{W}_h$ can be written as a linear combination of the basis. Therefore the previous expression is equivalent to

$$\int_\Omega \left[-\nabla B_{ij} \cdot \mathbf{u}\phi_h + \nabla B_{ij} \cdot \epsilon \nabla \phi_h \right] \, d\Omega = \int_\Omega B_{ij} f \, \mathrm{d}\Omega \quad i, j = 1, \ldots, n-1 \,. \tag{44}$$

Question 2.1 (2 points)

Question (3 points)

Derive the linear system corresponding to the Galerkin approximation (43) that needs to be solved for computing $\phi_h = \sum_{i,j} \phi_{ij} B_{ij}$. Specifically, without evaluating any integrals, find explicit expressions for the coefficients c_{kl} in equation (44), i.e.,

$$\sum_{k,l=0}^{n} c_{k,l} \phi_{k,l} = \frac{1}{h^2} \int_\Omega B_{r,s} f \, d\Omega \,. \tag{45}$$

Both equation (44) and (45) are interesting expressions because they explicitly show the tensor product nature of the basis. Nevertheless, computationally (and even analytically) they quickly become complex.

[1] Note that this weak form is slightly different from the one derived in class. Here, both terms in the LHS have been integrated by parts; a reminder that a weak form is not unique.

For this reason here we start by introducing a single indexing that can be derived easily from the double indexing. From now on, whenever we have a double indexing, say $\phi_{k,l}$ we will convert it into a single indexing, ϕ_i, of the form

$$i = k + l(n_x + 1)\,, \qquad k = 0, \ldots, n_x, \quad l = 0, \ldots, n_y\,. \tag{46}$$

Note that we assume two things here

1. Our grid is a grid of $n_x \times n_y$ cells (or elements), and therefore $(n_x + 1) \times (n_y + 1)$ vertices (and coefficients for ϕ_h, the ones at the boundary will be known and set to zero, in this case).
2. Our indexing starts at 0 (if we had decided to start at 1 then instead of l we would have to have $(l - 1)$).

With this numbering, the boundary coefficients for the left, ϕ^L, right, ϕ^R, bottom, ϕ^B, and top , ϕ^T, are

$$\phi^L_{l(n_x+1)}, \qquad \phi^R_{n+l(n_x+1)}, \qquad \phi^B_k, \qquad \phi^T_{k+(n_y+1)(n_x+1)} \tag{47}$$

for $k = 0, \ldots, n_x$ and $l = 0, \ldots, n_y$.

We can use the same single indexing for the basis $B_{k,l}(x, y)$ to transform them into $B_i(x, y)$, again with

$$i = k + l(n_x + 1)\,, \qquad k = 0, \ldots, n_x, \quad l = 0, \ldots, n_y\,. \tag{48}$$

With this, we can write the requested expression and then convert it to single indexing, which is more useful when implementing this in a computer program. The reason for this is because we need to setup a matrix, and matrices have only two indices. If we used double indexing we would end up with a four index structure, which then would be very complex to use. At least for me, how could you easily use matrix vector multiplication for such an object? Again, if we can make our lives easier, we should certainly do so, life is already complex enough.

We start by substituting $\phi_h = \sum_{k,l=0}^{n_x,n_y} \phi_{k,l} B_{k,l}(x, y)$ in (44) in order to get

$$\sum_{k,l=0}^{n_x,n_y} \phi_{k,l} \int_\Omega \left[-\nabla B_{r,s} \cdot \mathbf{u} B_{k,l} + \nabla B_{r,s} \cdot \epsilon \nabla B_{k,l}\right]\, d\Omega = \int_\Omega B_{r,s} f \, \mathrm{d}\Omega \quad r = 1, \ldots, n_x - 1, \quad s = 1, \ldots, n_y - 1\,. \tag{49}$$

We can now impose the boundary conditions, which in this case are all zero are rewrite the previous equation as

$$\sum_{k,l=1}^{n_x-1,n_y-1} \phi_{k,l} \int_\Omega \left[-\nabla B_{r,s} \cdot \mathbf{u} B_{k,l} + \nabla B_{r,s} \cdot \epsilon \nabla B_{k,l}\right]\, d\Omega = \int_\Omega B_{r,s} f \, \mathrm{d}\Omega \quad r = 1, \ldots, n_x - 1, \quad s = 1, \ldots, n_y - 1\,. \tag{50}$$

Now, we have $(n_x - 1) \times (n_y - 1)$ equations and the same number of unknowns, as we would expect in order to be able to solve the resulting system of equations: so far so good!

If we now look at the expression in the question's text, we can easily see that

$$c_{k,l} = \frac{1}{h^2} \int_\Omega \left[-\nabla B_{r,s} \cdot \mathbf{u} B_{k,l} + \nabla B_{r,s} \cdot \epsilon \nabla B_{k,l}\right]\, d\Omega\,, \quad k = 1, \ldots, n_x - 1, \quad l = 1, \ldots, n_y - 1\,. \tag{51}$$

To be formally correct, since each $c_{k,l}$ differs for each (r, s), we should use $c^{r,s}_{k,l}$. This was omitted, to avoid falling into the so called Christmas tree issue (which happens everytime we use too many subscripts and superscripts in a variable, sometimes it is unavoidable, but often it is better to avoid it to prevent our variable from looking like a Christmas tree full of decoration).

We still miss the coefficients associated to the boundary coefficients. They are the same as the ones above, we can just extend the indices. The reason we did not add them is because the coefficients $\phi_{k,l}$ are zero, so these terms are not present.

Question 2.2 (6 points)

Question 2.2.1 (5 points)

Question (3 points)

Implement the discrete method in a computer code. Choose $L = 1$, $\epsilon = 0.01$, $\mathbf{u} = [0, 1]$ and

$$f = \begin{cases} +1\,, & x < 0.5\,, \\ -1\,, & x \geq 0.5\,. \end{cases}$$

Present the solutions that you obtain with the method for $n \in \{40, 41\}$.
Hint: Use an element-by-element approach to decompose all two-dimensional integrals over the whole domain into summations of integrals over the cells $\Omega_{ij} \subset \Omega$ where the integrand is nonzero. Just go over the cells and compute the nonzero contribution of that cell to the integral. Additionally, use the tensor product structure of the B_{ij} to simplify your task by finding the unique one-dimensional integrals that will fully determine all two-dimensional integrals you need, i.e.,

$$\begin{aligned} \int_{\Omega_{rs}} B_{ij}(x,y)B_{kl}(x,y)\,\mathrm{d}x\mathrm{d}y &\overset{(41)}{=} \int_{\Omega_{rs}} B_i(x)B_j(y)B_k(x)B_l(y)\,\mathrm{d}x\mathrm{d}y \\ &= \int_{x_r}^{x_{r+1}} B_i(x)B_k(x)\,\mathrm{d}x \int_{y_s}^{y_{s+1}} B_j(y)B_l(y)\,\mathrm{d}y \end{aligned} \tag{52}$$

with $\Omega_{rs} = [x_r, x_{r+1}] \times [y_s, y_{s+1}]$. You can further simplify this recalling that

$$\int_a^b f(x)\,\mathrm{d}x = (b-a)\int_0^1 f(\xi(b-a)+a)\,\mathrm{d}\xi\,, \tag{53}$$

and further approximate it by a quadrature rule.

Let us again go step by step and see what we get for our system of equations at the discrete level. The first step is to understand how our basis functions look like and work. As we did for the 1D case, we will construct our basis functions as a combination of local basis whose support is over only one element. In Figure 7 you can see the shape of the local basis on a reference element

$$b^{1,1}(x,y) = b^1(x)b^1(y), \quad b^{1,2}(x,y) = b^1(x)b^2(y), \quad b^{2,1}(x,y) = b^2(x)b^1(y), \quad b^{2,2}(x,y) = b^2(x)b^2(y)\,, \tag{54}$$

note that we do not add a subscript indication to which element these local basis belong to because at this point they are associated to a generic element, they are just representative. As you can see, we use a tensor product of the 1D bases introduced above. In the code we will also use single indexing, and we must use the same approach, so $b^k(x,y)$ with

$$k = i + 2(j-1)\,. \tag{55}$$

As done before, we can construct a basis function associated to a vertex (i,j) as linear combination of the local basis over the adjacent elements

$$B_{i,j}(x,y) := b^{1,1}_{i,j}(x,y) + b^{2,1}_{i-1,j}(x,y) + b^{2,2}_{i-1,j-1}(x,y) + b^{1,2}_{i,j-1}(x,y)\,. \tag{56}$$

and recall that, as before,

$$b^{m,n}_{i,j}(x,y) := \begin{cases} b^m_i(x)b^n_j(y)\,, & (x,y) \in [x_i, x_{i+1}] \times [y_j, y_{j+1}]\,, \\ 0, & \text{elsewhere}\,. \end{cases} \tag{57}$$

In Figure 8 you can see how this looks like for the basis (i,j) associated to the vertex located at $(0.0, 0.0)$. Note that we kept the color code of the local basis so that they can be easily identified. The idea is that

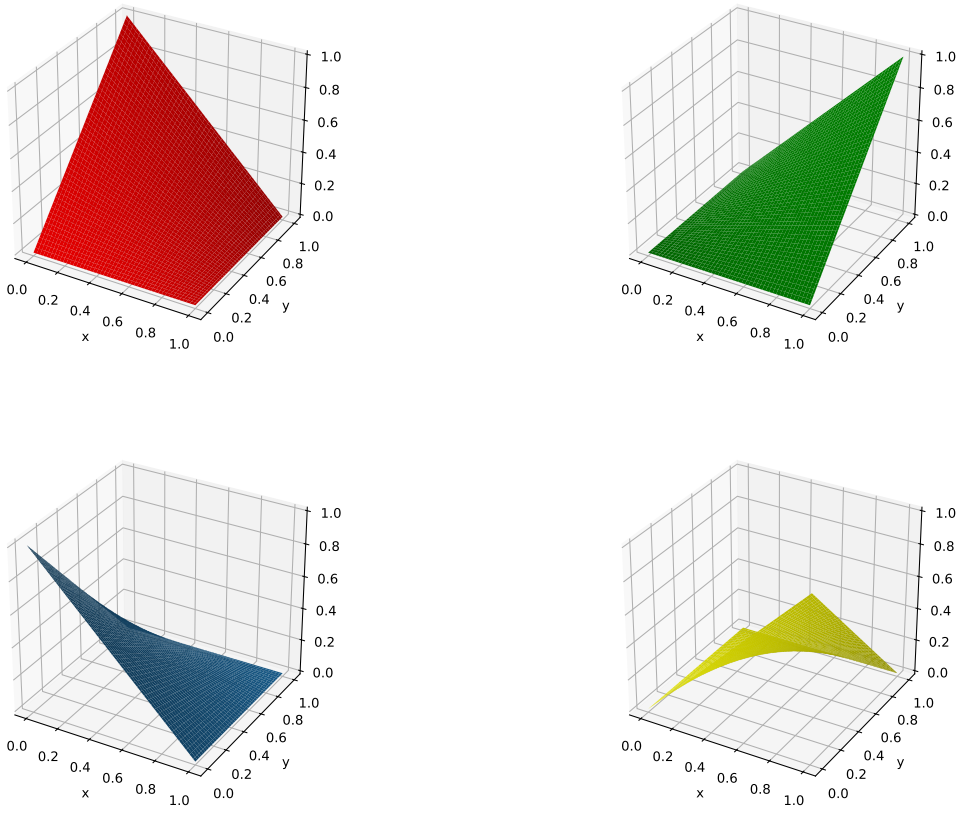


Figure 7: Representation of each of the four (local) bases on an element, as seen in the 1D case, the 2D basis elements can be expressed as linear combinations of these local bases over adjacent elements, as we will see below. Top left: local basis $b^{1,2}(x,y) = b^1(x)b^2(y)$, we will also number this basis as $b^3(x,y)$ when using single indexing. Top right: local basis $b^{2,2}(x,y) = b^2(x)b^2(y)$, we will also number this basis as $b^4(x,y)$ when using single indexing. Bottom left: local basis $b^{1,1}(x,y) = b^1(x)b^1(y)$, we will also number this basis as $b^1(x,y)$ when using single indexing. Bottom right: local basis $b^{2,1}(x,y) = b^2(x)b^1(y)$, we will also number this basis as $b^2(x,y)$ when using single indexing. Note that we have not specified any subscripting here, associating these local basis to a specific element. We have not done this here because at the moment we are only considering a generic element and we focus on the representation of these local basis. Below we will assign them to elements and therefore use subscripts.

vertex (i,j) will be present in elements (i,j), $(i-1,j)$, $(i-1,j-1)$, and, $(i,j-1)$ and the vertex will be the local vertex $(1,1)$, $(1,2)$, $(2,2)$, and $(1,2)$ of each of these elements, respectively. As you can see, we associated the local basis functions in each element in the same fashion as the vertex appears. This is the key idea behind the global assembly algorithm used in finite elements and what we used in the implementation. The advantage of using this more abstract algorithm is because (i) it is general, meaning that we can use it for meshes that are structured as ours and meshes that are unstructured, which are the general case, and (ii) because it is much simpler than keeping track of double indices and where basis overlap, the algorithm takes care of this automatically.

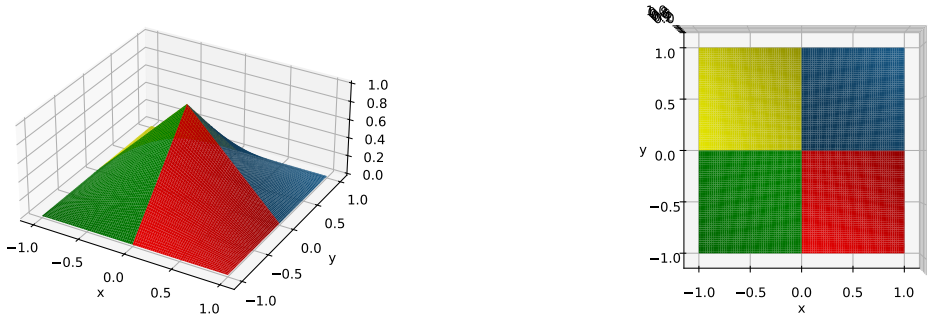


Figure 8: Representation of a 2D basis $B_{i,j}(x,y)$ associated to the vertex $(0.0, 0.0)$, with double index assumed (i,j). You can see that we can construct this basis element as a linear combination of the four local bases over the elements that share that vertex (recall the color code use in Figure 7). This linear combination for this case (basis associated to vertex (i,j), structured quadrilateral grid, cartesian numbering) is $B_{i,j}(x,y) = B_i(x)B_j(y) = \left(b^2_{i-1}(x) + b^1_i(x)\right)\left(b^2_{j-1}(y) + b^1_j(x)\right)$. Note that this expression is the same as (56).

We wish to compute equation (50), therefore we need to focus on evaluating the following integrals

$$\int_\Omega \nabla B_{r,s} \cdot \mathbf{u} B_{k,l} \,\mathrm{d}\Omega\,, \qquad \int_\Omega \nabla B_{r,s} \cdot \epsilon \nabla B_{k,l} \,d\Omega\,, \quad \text{and} \quad \int_\Omega B_{r,s} f \,\mathrm{d}\Omega\,. \tag{58}$$

We will first do it with double indexing and then will convert the double indexing into single indexing so that you can see it is possible (although very boring and tedious) to do the assembly of the matrices in this way.

Let us start with the first term. We will call these terms $C^{k,l}_{r,s}$ because they are the discrete version of the convection term (and we lack imagination). Note that here we have no choice but to go for the Christmas tree decoration. As we will see, all indices will be important. Let us replace the basis functions by their expressions (56) and (56)

$$\int_\Omega \nabla B_{r,s} \cdot \mathbf{u} B_{k,l} \,\mathrm{d}\Omega = \int_\Omega \nabla \left(b^{1,1}_{r,s}(x,y) + b^{2,1}_{r-1,s}(x,y) + b^{2,2}_{r-1,s-1}(x,y) + b^{1,2}_{r,s-1}(x,y)\right) \cdot \\ \cdot \mathbf{u} \left(b^{1,1}_{k,l}(x,y) + b^{2,1}_{k-1,l}(x,y) + b^{2,2}_{k-1,l-1}(x,y) + b^{1,2}_{k,l-1}(x,y)\right) \,\mathrm{d}\Omega\,. \tag{59}$$

We can now expand these terms to get sixteen terms (an exercise in patience)

$$\begin{aligned}\int_\Omega \nabla B_{r,s} \cdot \mathbf{u} B_{k,l} \,\mathrm{d}\Omega = \int_\Omega [\\ \nabla b^{1,1}_{r,s}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{1,1}_{r,s}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \nabla b^{1,1}_{r,s}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{1,1}_{r,s}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y) + \\ + \nabla b^{2,1}_{r-1,s}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{2,1}_{r-1,s}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \nabla b^{2,1}_{r-1,s}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{2,1}_{r-1,s}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y) + \\ + \nabla b^{2,2}_{r-1,s-1}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{2,2}_{r-1,s-1}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \nabla b^{2,2}_{r-1,s-1}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{2,2}_{r-1,s-1}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y) + \\ + \nabla b^{1,2}_{r,s-1}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{1,2}_{r,s-1}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \nabla b^{1,2}_{r,s-1}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{1,2}_{r,s-1}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y)\Big]\,. \end{aligned} \tag{60}$$

We will not expand this much further, but the point here is that, as in the 1D case, we have pairs of products of local basis functions at different elements. Since the local basis are only nonzero at the associated element, most of these products will be zero. Specifically, the only nonzero integrals are the ones of the form

$$\int_\Omega \nabla B_{k+i,l+j} \cdot \mathbf{u} B_{k,l} \,\mathrm{d}\Omega\,, \qquad i,j = -1, 0, 1\,. \tag{61}$$

Special care needs to be taken for basis adjacent or on the boundary. We will not expand all these expressions over the nine possible combinations of the indices (i, j), since this would be bordering the insane. What we wish to highlight here is the clever algorithm to actually compute this without having to explicitly figure out all the possible cases (including the ones near the boundary). Let us just consider three cases, for illustration, and to motivate the algorithm and check that it works: $(i, j) = (0, 0)$, $(i, j) = (1, 0)$, and $(i, j) = (1, 1)$. By inspection we can see that for these three cases we will have only four terms, two terms, and one term, respectively. All other cases are very similar to these three.

Let us start with the one with more terms, $(i, j) = (0, 0)$,

$$\begin{aligned}\int_\Omega \nabla B_{k,l} \cdot \mathbf{u} B_{k,l} \,\mathrm{d}\Omega = \int_\Omega [\\ \nabla b^{1,1}_{k,l}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{1,1}_{k,l}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \nabla b^{1,1}_{k,l}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{1,1}_{k,l}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y) + \\ + \nabla b^{2,1}_{k-1,l}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{2,1}_{k-1,l}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \nabla b^{2,1}_{k-1,l}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{2,1}_{k-1,l}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y) + \\ + \nabla b^{2,2}_{k-1,l-1}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{2,2}_{k-1,l-1}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \nabla b^{2,2}_{k-1,l-1}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{2,2}_{k-1,l-1}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y) + \\ + \nabla b^{1,2}_{k,l-1}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{1,2}_{k,l-1}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \nabla b^{1,2}_{k,l-1}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{1,2}_{k,l-1}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y) \Big] \,\mathrm{d}\Omega \,. \end{aligned} \tag{62}$$

This full expansion is an interesting exercise to clearly see that most terms are zero, because they pair local basis on different elements (different subscripts). As promised, on each line, only one term is nonzero, therefore this expression simplifles to

$$\begin{aligned}\int_\Omega \nabla B_{k,l} \cdot \mathbf{u} B_{k,l} \,\mathrm{d}\Omega = \int_\Omega \Big[& \nabla b^{1,1}_{k,l}(x,y) \cdot \mathbf{u} b^{1,1}_{k,l}(x,y) + \nabla b^{2,1}_{k-1,l}(x,y) \cdot \mathbf{u} b^{2,1}_{k-1,l}(x,y) + \\ & + \nabla b^{2,2}_{k-1,l-1}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) + \nabla b^{1,2}_{k,l-1}(x,y) \cdot \mathbf{u} b^{1,2}_{k,l-1}(x,y) \Big] \,\mathrm{d}\Omega \,. \end{aligned} \tag{63}$$

Two important aspects to highlight here

1. Only terms over the same element are present, this was expected given the support of each local basis.

2. These terms correspond to the common elements containing the vertices associated to the two basis functions that appear in the integral. In this case, basis $B_{k,l}$ is associated to vertex (k, l) which is present in elements (k, l), $(k - 1, l)$, $(k - 1, l - 1)$, and $(k, l - 1)$. If you look at the expression, these are the elements that appear.

3. These terms correspond to the local indexing of the vertex in the elements. For example, the vertex (k, l) appears in the element (k, l) as local vertex with index $(1, 1)$, it is the bottom left index of that element. Also, the vertex (k, l) appears in the element $(k - 1, l)$ as local vertex with index $(2, 1)$, it is the bottom right index of that element. You can clearly see that in the color code we have used for the basis functions to ease the visualisation of this fact.

4. There is a clear connection between the basis, the vertex, the elements, and the local position of the vertex in the elements.

Now we look at the second case, the one with two terms, $(i, j) = (1, 0)$,

$$\int_\Omega \nabla B_{k+1,l} \cdot \mathbf{u} B_{k,l} \,\mathrm{d}\Omega = \int_\Omega \Big[\nabla b^{1,1}_{k,l}(x,y) \cdot \mathbf{u} b^{2,1}_{k,l}(x,y) + \nabla b^{1,2}_{k,l-1}(x,y) \cdot \mathbf{u} b^{2,2}_{k-1,l-1}(x,y) \Big] \,\mathrm{d}\Omega \,. \tag{64}$$

Note that we have spared your of the full expression and just presented the nonzero terms, the ones with the same element.

Finally, we address the third case, the one with one term, $(i, j) = (1, 1)$,

$$\int_\Omega \nabla B_{k+1,l+1} \cdot \mathbf{u} B_{k,l} \,\mathrm{d}\Omega = \int_\Omega \nabla b^{1,1}_{k,l}(x,y) \cdot \mathbf{u} b^{2,2}_{k,l}(x,y) \,\mathrm{d}\Omega \,. \tag{65}$$

The question now is, how can we efficiently and easily compute these terms? This can be done with the *global assembly* algorithm of finite elements.

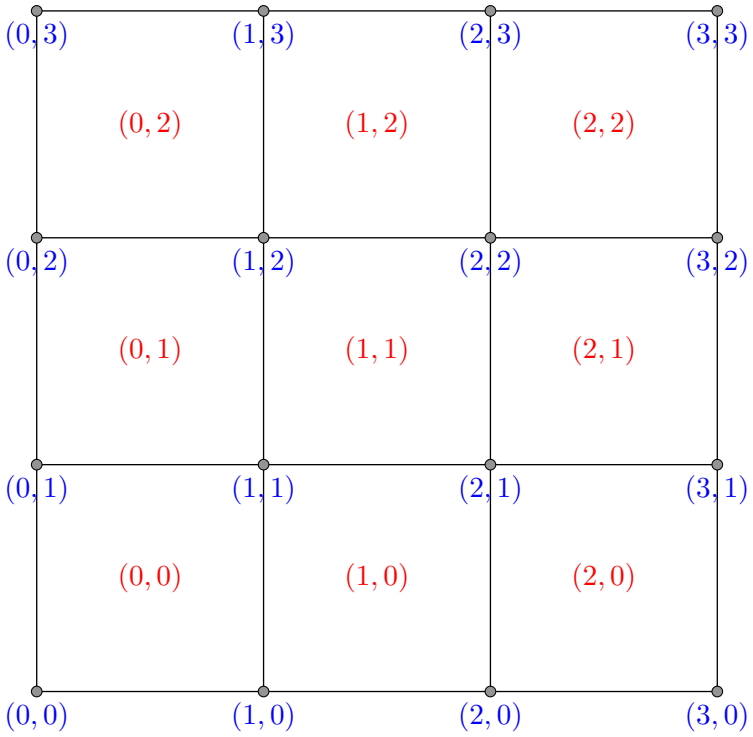


Figure 9: Global numbering of vertices (and basis functions) in blue, and elements in red.

Suppose now we have the data structure we discussed in class: a list of the indices of vertices present in each element, $E^{i,j}_{n,m}$, where (n,m) is the element and (p,t) is the index of the local vertex, i.e., $i,j=1,2$. Note that $E^{i,j}_{n,m}$ is a tuple, i.e., the pair of global indices of the local vertex (i,j) of element (n,m). For example, in the mesh with global numbering as in Figure 9 and local numbering as in Figure 10, we have

$$E^{0,0}_{0,0}=(0,0),\quad E^{1,1}_{0,0}=(1,1),\quad E^{0,0}_{1,1}=(1,1),\quad E^{1,0}_{1,1}=(2,1),\quad E^{0,1}_{1,2}=(1,3)\,. \tag{66}$$

We can now introduce the following notation for the local integral (note how multiple indexing leads to Christmas tree notation)

$$c^{(i,j),(p,t)}_{k,l}:=\int_\Omega \nabla b^{i,j}_{k,l}(x,y)\cdot \mathbf{u} b^{p,t}_{k,l}(x,y)\,\mathrm{d}\Omega\,,\quad i,j,p,t=1,2\,. \tag{67}$$

We now can see that if we loop over all elements (m,n) and sum the following terms

$$C^{E^{p,t}_{m,n}}_{E^{i,j}_{m,n}}:=\int_\Omega \nabla B_{E^{i,j}_{m,n}}\cdot \mathbf{u} B_{E^{p,t}_{m,n}}\,\mathrm{d}\Omega \mathrel{+}= c^{(i,j),(p,t)}_{m,n}\,, \tag{68}$$

we get the expressions above. Note that the terms $C^{E^{p,t}_{z,v}}_{E^{i,j}_{m,n}}=0$ for $(m,n)\neq(z,v)$, as expected.

Let us do this for the first case. In this first case, we wish to compute

$$\begin{aligned}C^{k,l}_{k,l}:=\int_\Omega \nabla B_{k,l}\cdot\mathbf{u}B_{k,l}\,\mathrm{d}\Omega=\int_\Omega\Big[&\nabla b^{1,1}_{k,l}(x,y)\cdot\mathbf{u}b^{1,1}_{k,l}(x,y)+\nabla b^{2,1}_{k-1,l}(x,y)\cdot\mathbf{u}b^{2,1}_{k-1,l}(x,y)+\\&+\nabla b^{2,2}_{k-1,l-1}(x,y)\cdot\mathbf{u}b^{2,2}_{k-1,l-1}(x,y)+\nabla b^{1,2}_{k,l-1}(x,y)\cdot\mathbf{u}b^{1,2}_{k,l-1}(x,y)\Big]\,\mathrm{d}\Omega\,.\end{aligned} \tag{69}$$

This means that $E^{i,j}_{m,n}=E^{p,t}_{m,n}=(k,l)$. This pair of indices, (k,l), is the pair of global indices of the vertex associated to the basis function $B_{k,l}(x,y)$, as we saw before. Where can we find (k,l) in E? This is simple, we can find this vertex only at $E^{1,1}_{k,l}$, $E^{2,1}_{k-1,l}$, $E^{2,2}_{k-1,l-1}$, and $E^{1,2}_{k,l-1}$. Therefore the summation of terms going element by element leads to

$$C^{k,l}_{k,l}:=\int_\Omega \nabla B_{k,l}\cdot\mathbf{u}B_{k,l}\,\mathrm{d}\Omega=c^{(1,1),(1,1)}_{k,l}+c^{(2,1),(2,1)}_{k-1,l}+c^{(2,2),(2,2)}_{k-1,l-1}+c^{(1,2),(1,2)}_{k,l-1}\,. \tag{70}$$

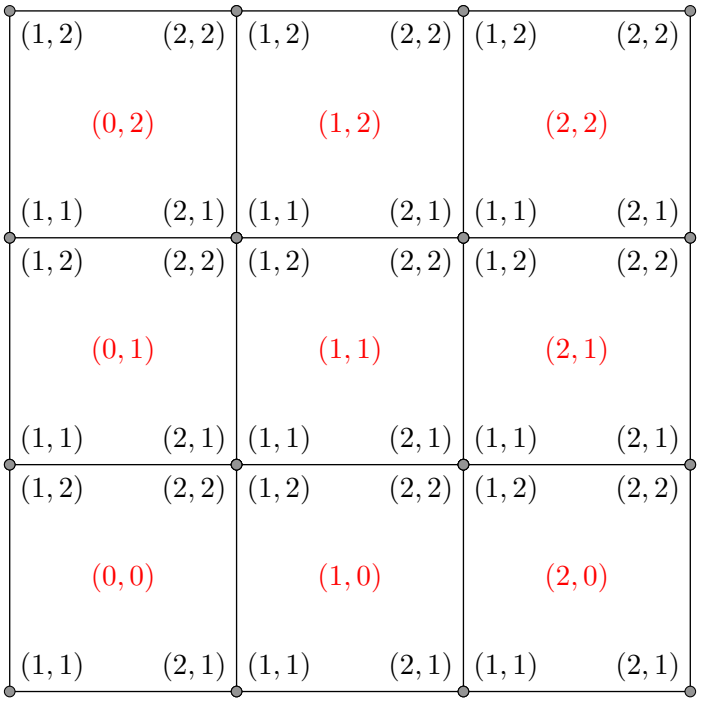


Figure 10: Local numbering of vertices (and local basis functions) in black, and elements in red.

The first term added when we pass by element (k, l), the second term added when we pass by element $(k-1, l)$, the third when we pass by element $(k-1, l-1)$, and the fourth and final term when we pass by element $(k, l-1)$.

To clarify, let us do this also for the third case. For this case we wish to compute

$$C^{k,l}_{k+1,l+1} := \int_\Omega \nabla B_{k+1,l+1} \cdot \mathbf{u} B_{k,l} \, \mathrm{d}\Omega = \int_\Omega \nabla b^{1,1}_{k,l}(x,y) \cdot \mathbf{u} b^{2,2}_{k,l}(x,y) \, \mathrm{d}\Omega \,. \tag{71}$$

In this case we have that $E^{i,j}_{m,n} = (k+1, l+1)$ and $E^{p,t}_{m,n} = (k, l)$. This means that only the element (m, n) that contains the vertices (k, l) and $(k+1, l+1)$ contributes. In the case of our mesh, this is the element (k, l) and the local indices are $(1, 1)$ and $(2, 2)$, respectively. Therefore the only term considered is

$$C^{k,l}_{k+1,l+1} := \int_\Omega \nabla B_{k+1,l+1} \cdot \mathbf{u} B_{k,l} \, \mathrm{d}\Omega = c^{(1,1),(2,2)}_{k,l} \,. \tag{72}$$

This term added when we pass by element (k, l).

We will not do this here, but this can be done for all other cases and will lead to the correct expressions. Also, this can also be applied to any of the other integrals we need to compute (the diffusion term, and the forcing term).

This is the *global assembly* algorithm used in finite elements. You can use double indexing, but we highly recommend using single indexing. Use double indexing but convert it to single indexing in the assembly process (see the computer implementation), as it makes everything simpler and in the end you get vectors and matrices, which is what is preferable to solve linear systems of equations.

One final note and we are done with this question. As we can see, besides this bookkeeping needed for the global assembly, we need to compute the terms $c^{ij}_{k,l}$ in (67). Let us focus on this integral and see how to compute it (the other integrals that appear follow the same approach). We need to compute

$$c^{(i,j),(p,t)}_{k,l} := \int_\Omega \nabla b^{i,j}_{k,l}(x,y) \cdot \mathbf{u} b^{p,t}_{k,l}(x,y) \, \mathrm{d}\Omega \,, \quad i,j,p,t = 1,2 \,. \tag{73}$$

If we substitute (57) into this equation, we get

$$c^{(i,j),(p,t)}_{k,l} := \int_\Omega \nabla \left(b^i_k(x) b^j_l(y) \right) \cdot \mathbf{u} b^p_k(x) b^t_l(y) \, \mathrm{d}\Omega \,, \quad i,j,p,t = 1,2 \,. \tag{74}$$

If we expand the terms we get

$$c_{k,l}^{(i,j),(p,t)} := \int_\Omega \left[\frac{\mathrm{d}b_k^i(x)}{\mathrm{d}x} b_l^j(y)\vec{e}_x + b_k^i(x)\frac{\mathrm{d}b_l^j(y)}{\mathrm{d}y}\vec{e}_y \right) \cdot \mathbf{u} b_k^p(x) b_l^t(y)\,\mathrm{d}\Omega\,, \quad i,j,p,t = 1,2\,. \tag{75}$$

We can compute the inner product and get

$$c_{k,l}^{(i,j),(p,t)} := \int_\Omega \left[\frac{\mathrm{d}b_k^i(x)}{\mathrm{d}x} b_l^j(y) u_x b_k^p(x) b_l^t(y) + b_k^i(x)\frac{\mathrm{d}b_l^j(y)}{\mathrm{d}y} u_y b_k^p(x) b_l^t(y) \right] \mathrm{d}\Omega\,, \quad i,j,p,t = 1,2\,. \tag{76}$$

Collecting all the terms in x and y together into separate integrals over the intervals over which the basis functions are nonzero, we get

$$c_{k,l}^{(i,j),(p,t)} := \int_{x_k}^{x_{k+1}} \frac{\mathrm{d}b_k^i(x)}{\mathrm{d}x} u_x b_k^p(x)\,\mathrm{d}x \int_{y_l}^{y_{l+1}} b_l^j(y) b_l^t(y)\,\mathrm{d}y + \int_{x_k}^{x_{k+1}} b_k^i(x) u_y b_k^p(x)\,\mathrm{d}x \int_{y_l}^{y_{l+1}} \frac{\mathrm{d}b_l^j(y)}{\mathrm{d}y} b_l^t(y)\,\mathrm{d}y\,, \quad i,j,p,t = 1,2\,. \tag{77}$$

Each of these one dimensional integrals can be computed as before for the 1D case. In fact, all the integrals have already been computed.

One special care is required. Note that in the question's text integration by parts was used in the convection term. This was not done in 1D. For this reason the local matrices used for the 1D integrals needs to be the transpose. If you decide to use the original 1D cases, then you need to drop the minus sign in the convection term. This is the route followed in the code: reuse the 1D matrices and drop the minus sign.

Question 2.2.2 (1 point)

Question (3 points)

Compare the above results to the "exact solution" obtained using $n = 100$ and discuss your observations on the following lines:

- Which solution features are being faithfully represented on the two meshes? Can you spot a pattern if you choose other even-odd values of n?
- If you had the flexibility to use a non-uniform Cartesian mesh (i.e., where the element sizes can vary in x and y directions), how would you choose the mesh spacings to best capture the solution of this problem? (You don't need to implement this.)

We present the requested solution in Figure 11 and a solution with $n_x = 41$ and $n_y = 40$ in Figure 12. We can observe the following:

- The higher resolution solution (the one we called reference or "exact" solution) shows the growth and decline near the top boundary, the so called boundary layer. The solution with lower resolution shows an abrupt decline to zero, i.e., it is monotonic and then zero. The high resolution shows the inflection point.
- Another aspect which is subtle is that by using an odd mesh in the x-direction we capture exactly the interface of the forcing function f, otherwise the interface is shifted one grid cell.
- If we could use different cell sizes, we should refine the mesh where the solution is changing more rapidly. This is near the top boundary. If we wish to be very accurate, then we could do the same near all the boundaries and also close to the vertical line at the center of the domain.

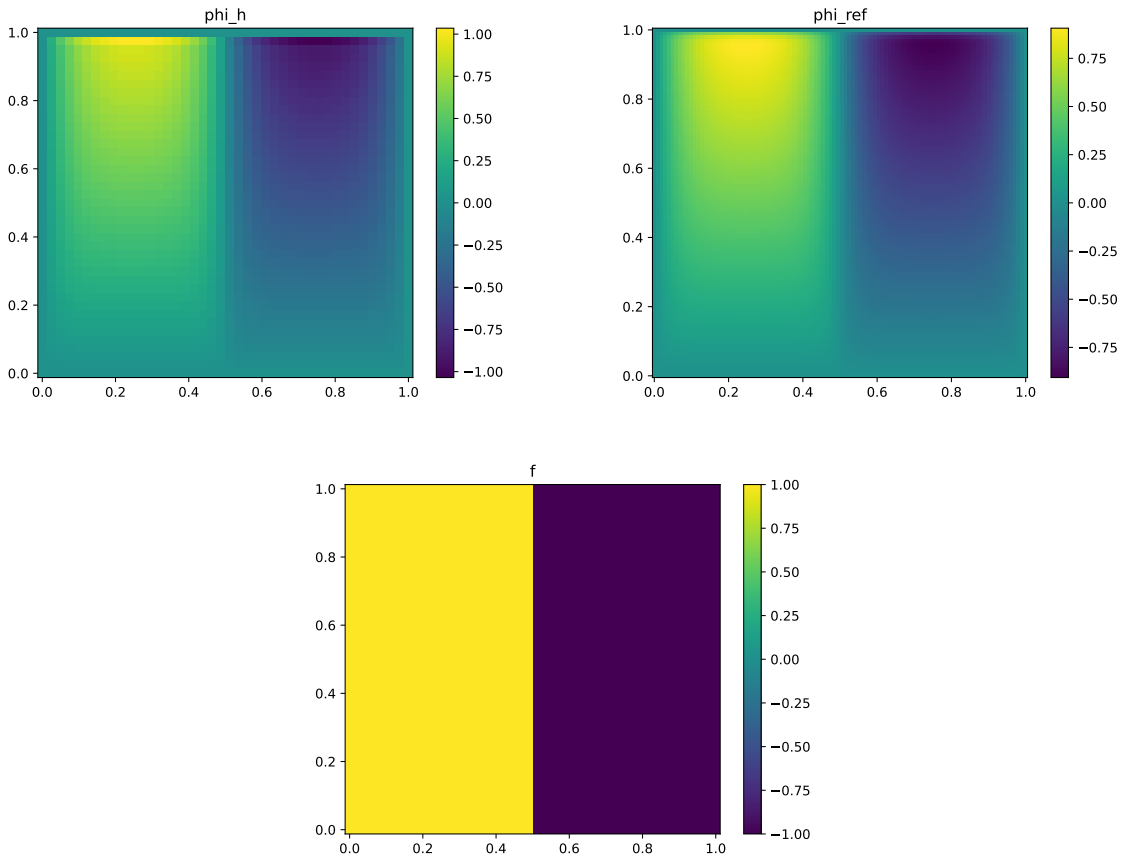


Figure 11: Top left: Numerical solution with $n_x = 40$ and $n_y = 41$. Top right: Reference solution with $n_x = 100$ and $n_y = 100$. Bottom: forcing term, f, on the computational mesh.

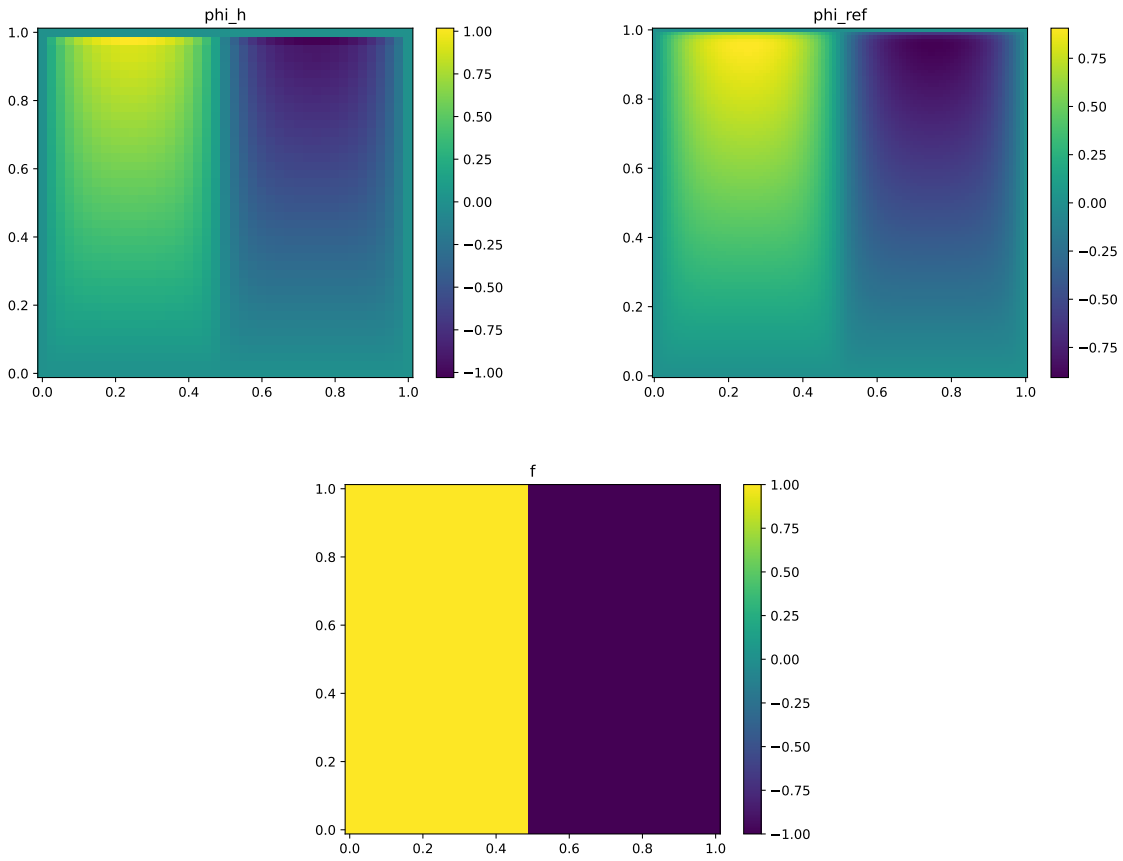


Figure 12: Top left: Numerical solution with $n_x = 41$ and $n_y = 40$. Top right: Reference solution with $n_x = 101$ and $n_y = 101$. Bottom: forcing term, f, on the computational mesh.