Onno Assignment 3

Benjamin Dalby

December 2024

1

We are considering the problem of the Poisson equation

$$-\nabla^2 u = f \text{ on } (x,y) \in [0,1]^2$$
$$f(x,y) = 2\pi^2 sin(\pi x) cos(\pi y)$$

with combined Dirichlet and Neumann boundary conditions

$$u(0,y) = u(1,y) = 0$$
$$\partial_y u(x,y)|_{y=0} = \partial_x u(x,y)|_{x=0} = 0$$

The minimisation problem is:

$$F(u) = \int\int [\frac{1}{2}(\frac{\partial u}{\partial x}\frac{\partial u}{\partial y})^2 - f(x,y)u(x,y)]dxdy$$

giving rise to the Ritz-Galerkin integral

$$F(u) = \int\int \frac{1}{2}|\nabla u|^2 - ufdxdy$$

From the definition of a derivative,

$$\delta F(u) = \lim_{\epsilon \to 0} \frac{F(u+\epsilon\partial u) - F(u)}{\epsilon} = \frac{d}{dt}F(u+\epsilon\delta u) = 0$$

where $\epsilon << 1$, and $\epsilon = \Delta t$

Making a variation $\delta u(x,y)$, which is a pertubation,

$$F(u+\epsilon\delta u) = \int\int \frac{1}{2}\nabla|u+\epsilon\delta u|^2 - (u+\epsilon\delta u)fdxdy$$

So

$$\frac{d}{dt}F(u+\epsilon\delta u) = \frac{d}{dt}\int\int \frac{1}{2}\nabla|u+\epsilon\delta u|^2 - (u+\epsilon\delta u)fdxdy = 0$$

$$\frac{d}{dt}F(u+\epsilon\delta u) = \int\int \frac{d}{dt}(\frac{1}{2}\nabla|u+\epsilon\delta u|^2) - (\delta u)fdxdy = 0$$

Examining $\frac{d}{dt}(\frac{1}{2}\nabla|u+\epsilon\delta u|^2)$:

$$\frac{d}{dt}(\frac{1}{2}\nabla|u+\epsilon\delta u|^2) = \frac{d}{dt}(\frac{1}{2}\nabla(u^2 + 2u\cdot\epsilon\delta u + \epsilon^2\delta^2 u^2))$$

Looking at each term, and remembering that $\epsilon = \Delta t$, we realise that

$$\frac{d}{dt}(\frac{1}{2}(u^2) = 0$$
$$\frac{d}{dt}(2u \cdot \epsilon\delta u) = 2u \cdot \delta u$$
$$\frac{d}{dt}(\epsilon^2\delta^2u^2) = \frac{1}{2}\epsilon\delta^2u^2$$

$$\frac{d}{dt}(\frac{1}{2}\nabla|u + \epsilon\delta u|^2) = \nabla u \cdot \nabla\delta u + \frac{1}{4}\nabla\epsilon\delta^2u^2$$

But, as $\epsilon << 1$, we can linearise the system to the leading order term yielding

$$\frac{d}{dt}(\frac{1}{2}\nabla|u + \epsilon\delta u|^2) \approx \nabla u \cdot \nabla\delta u$$

therefore

$$\frac{d}{dt}F(u + \epsilon\delta u) = \int\int \nabla u \cdot \nabla\delta u - (\delta u)f dxdy = 0 \tag{1}$$

We can also obtain this "weak form" using a test function. We introduce the test function $\delta u(x, y)$. We take the inner product of the Poisson equation with the test function, which is equivalent to multiplying both sides by the test function and integrating over the domain. For the sake of compactness, we write $\Omega : (x, y) \in [0, 1]^2$, $\Gamma = \delta\Omega$.

The test function has the following conditions:

1. The test function belongs to the same function space as the trial function, i.e. it exists in the domain Ω and has outputs in the same codomain as $u(x, y)$
2. The test function satisfies the boundary conditions

$$-\iint_\Omega \delta u\nabla^2 u dxdy = \iint_\Omega \delta u f dxdy$$

Working on the left hand side, after expanding $\nabla^2 = \frac{\partial^2}{\partial x^2}\mathbf{e_x} + \frac{\partial^2}{\partial y^2}\mathbf{e_y}$via the product rule we obtain

$$\iint_\Omega \frac{\partial}{\partial x}(u_x\delta u) - u_x\frac{\partial}{\partial y}(\delta u) + \frac{\partial}{\partial y}(u_y\delta u) - u_y\frac{\partial}{\partial y}(\delta u)dxdy$$

$$\iint_\Omega \frac{\partial}{\partial x}(u_x\delta u) + \frac{\partial}{\partial y}(u_y\delta u) - \iint_\Omega u_x\frac{\partial}{\partial x}(\delta u) - u_y\frac{\partial}{\partial y}(\delta u)dxdy$$

Via Green's theorem,

$$\iint_\Omega \frac{\partial}{\partial x}(u_x\delta u) + \frac{\partial}{\partial y}(u_y\delta u) = \int_\Gamma u_x\delta u dx - u_y\delta u dy$$

so

$$\int_\Gamma u_x\delta u dx - u_y\delta u dy - \iint_\Omega u_x\frac{\partial}{\partial x}(\delta u) - u_y\frac{\partial}{\partial y}(\delta u)dxdy = \iint_\Omega \delta u f dxdy$$

$$\int_\Gamma u_x\delta u dx - \int_\Gamma u_y\delta u dy - \iint_\Omega u_x\frac{\partial}{\partial x}(\delta u) - u_y\frac{\partial}{\partial y}(\delta u)dxdy = \iint_\Omega \delta u f dxdy$$

Examining the the integrals $\int_\Gamma u_x \delta u dx, \int_\Gamma u_y \delta u dy$ with our boundary conditions, we realise that on the boundaries $\Gamma_2 = x(0), \Gamma_4 = x(1)$, $\int_\Gamma u_x \delta u dx = 0$ provided that the test function meets the boundary condition. This is because δu goes to zero at $x = 0, 1$. Similarly on the boundaries $\Gamma_1 = y(0), \Gamma_3 = y(1)$ we have $\partial_y u = 0$, therefore $\int_\Gamma u_y \delta u dy = 0$. Thus, as long as the test function meets the boundary condition we have

$$-\iint_\Omega u_x \frac{\partial}{\partial x}(\delta u) - u_y \frac{\partial}{\partial y}(\delta u) dx dy = \iint_\Omega \delta u f dx dy$$

Bringing the sign inside

$$\iint_\Omega u_x \frac{\partial}{\partial x}(\delta u) + u_y \frac{\partial}{\partial y}(\delta u) dx dy = \iint_\Omega \delta u f dx dy$$

and finally, rewriting in vector form

$$\iint_\Omega \nabla \delta u \cdot \nabla u = \iint_\Omega \delta u f dx dy$$

$$\iint_\Omega \nabla \delta u \cdot \nabla u - \delta u f dx dy = 0 \tag{2}$$

We can see that equations (1) and (2) are the same weak form.

Because the variation we took in obtaining (1) fulfils the conditions we made for a test function, we can say that the variation is a test function.

2

When we discretise the system we make the following substitutions

1. $u \approx u_h = \sum_i \tilde{u}_i(x, y) \cdot u_i$
2. $\partial u \approx \sum_i \tilde{u}_i(x, y)$

where $\tilde{u}_i(x, y)$ is the basis function.

From this we obtain that the discrete Ritz-Galerkin principle is

$$F(u_h) = \iint_\Omega \frac{1}{2} |\sum_i u_i \nabla \tilde{u}_i|^2 - \sum_i u_i \tilde{u}_i f dx dy$$

and the discrete weak form

$$\iint_\Omega (\sum_i \nabla \tilde{u}_i)(\sum_i u_i \nabla \tilde{u}_i) - \sum_i \tilde{u}_i f dx dy$$

We make an equivalent variation to the Ritz-Galerkin principle as we did in the continuous case, where now $\partial u \approx \sum_i \tilde{u}_i(x, y)$

$$F(u_h + \epsilon \sum_i \tilde{u}_i) = \iint_\Omega \frac{1}{2} \nabla |\sum_i u_i \tilde{u}_i + \epsilon \sum_i \tilde{u}_i|^2 - (\sum_i u_i \tilde{u}_i + \epsilon \sum_i \tilde{u}_i) f$$

Taking the time derivative

$$\frac{d}{}F(u_h + \epsilon \sum_i \tilde{u}_i) = \iint_\Omega \frac{\partial}{\partial t}(\frac{1}{2}\nabla|\sum_i u_i\tilde{u}_i + \epsilon \sum_i \tilde{u}_i|^2) - \sum_i \tilde{u}_i f$$

expanding the bracket and finding derivatives:

$$\frac{\partial}{\partial t}\sum_i (u_i\tilde{u}_i)^2 = 0$$

$$\frac{\partial}{\partial t}\epsilon^2 \sum_i (u_i)^2 = \frac{1}{2}\epsilon \sum_i (u_i)^2$$

$$\frac{\partial}{\partial t} 2\epsilon \sum_i u_i\tilde{u}_i \sum_i \tilde{u}_i = 2\sum_i u_i\tilde{u}_i \sum_i \tilde{u}_i$$

Therefore, again taking a linearised solution by disregarding the $O(\epsilon)$ term,

$$\frac{d}{dt}F(u_h + \epsilon \sum_i \tilde{u}_i) = \iint_\Omega \nabla(\sum_i u_i\tilde{u}_i \sum_i \tilde{u}_i) - \sum_i \tilde{u}_i f dxdy$$

distributing the ∇,

$$\frac{d}{dt}F(u_h + \epsilon \sum_i \tilde{u}_i) = \iint_\Omega (\sum_i \nabla\tilde{u}_i)(\sum_i u_i\nabla\tilde{u}_i) - \sum_i \tilde{u}_i f dxdy$$

Thus, we have shown that the discrete weak form can be obtained by variation from the discrete Ritz-Galerkin form.

3

Not attempted, optional

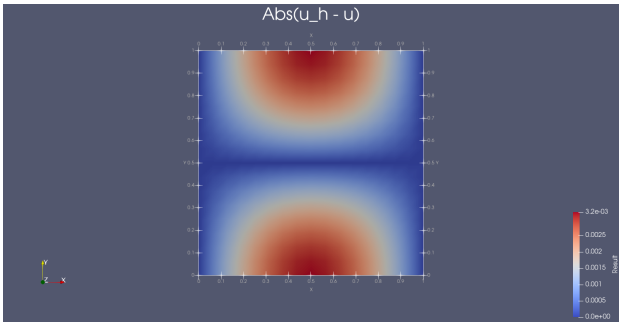
4

The exact solution was compared to the numeric solution in firedrake for a first order Discontinuous Galerkin function space. Figure 1 shows the error, given by $u_h - u$, as the mesh is refined from 16x16 to 64x64. The error reduces by a factor of 16 ($3.2e^{-3}$ to $2.0e^{-4}$) as the total number of mesh elements is increased by a factor of 16 (256 to 4096). Therefore we can see that the total mesh elements are inversely linearly related to the error, $E \approx \frac{1}{M}$ where M is the total number of mesh elements, $x_n \times y_n$.

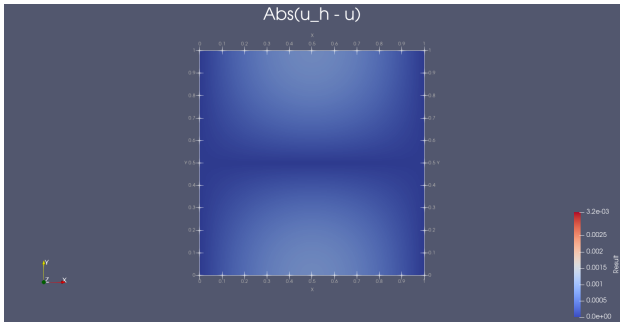
For a 64x64 mesh the order of the Discontinuous Galerkin method was increased from 1 to 3. Figure 2 shows the error as the order is increased. Due to the large differences in error values, the colour scales were reset for each graph as we would not be able to see any structure for $p = 2, 3$ otherwise. There is a relationship where the order of the error scales with $E \approx \frac{1}{M^p}$ where p is the order of the method. This is the same relationship we observed while refining the mesh, as there we fixed $p = 1$.

From this we can calculate which values of M, p will have similar error. We already know, from figure 2b, that the error for $p = 2, M = 64$ is $1 \times 10e^{-8}$. This, approximately, agrees with the theoretical prediction of $\frac{1}{(64x64)^2} = 5.96e^{-8}$. We can find the mesh required to obtain a similar value when $P = 1$ by solving for M. Using the theoretical value for the error, $M = \sqrt[p]{\frac{1}{5.96e^{-8}}}$, which gives $M \approx 4096 \times 4096 = 16,777,216$.

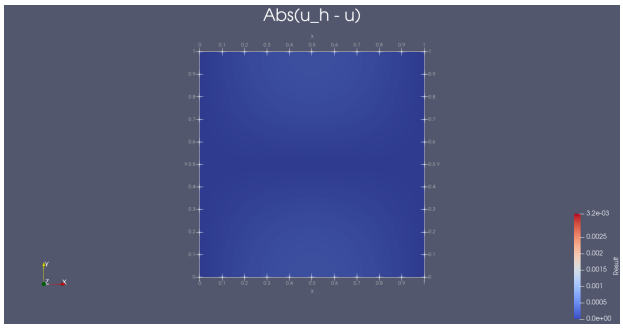
Figure 3 shows the errors for $p = 1, M = 4096 \times 4096$ and $p = 2, M = 64 \times 64$. The error results are (approximately) as the theory predicted but $p = 1, M = 4096 \times 4096$ took approximately half an hour to run. The method with $p = 2, M = 64 \times 64$ took only a few seconds and has slightly better error.



(a) The 16x16 mesh with a first order method. Maximum error $3.2e^{-3}$



(b) The 32x32 mesh with a first order method. Maximum error $8.0e^{-4}$

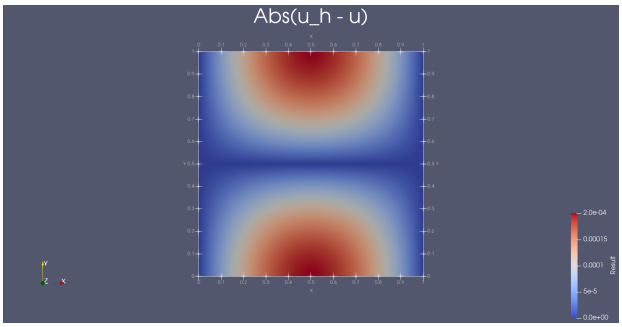


(c) The 64x64 mesh with a first order method. Maximum error $2.0e^{-4}$

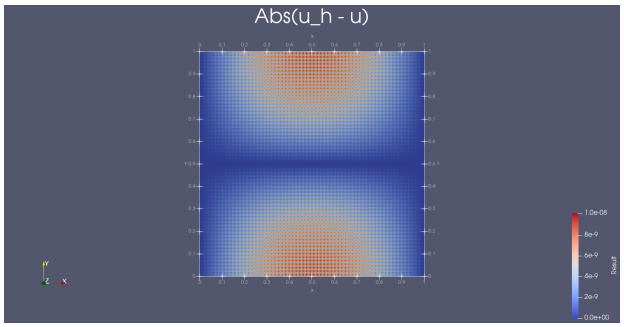
Figure 1: The error as the mesh is refined.

Ideally, we would also make the comparison using the actual error value $M = \sqrt[p]{\frac{1}{1.0e^{-8}}} \approx 7255 \times 7255 = 52,635,025$.This is too large to be computed on a desktop machine due to memory and time constraints. However, as this mesh is five times larger, our theory predicts that the error would be five times smaller and would therefore be $E \approx 1 \times e^{-8}$.

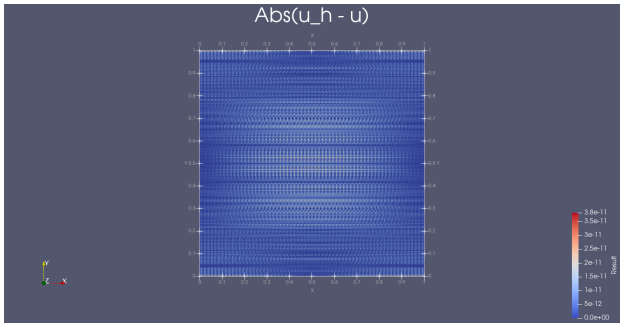
It is clear that it is generally favourable to increase the order rather than the mesh size. Using $M \approx \sqrt[p]{\frac{1}{E}}$, and $E \approx \frac{1}{M^p}$, it is possible to tune the mesh and order to find an acceptable compromise between error, runtime, and memory usage. This is an approximate set of relationships, however, which is correct to within an order of magnitude for the tested parameter values.



(a) The 64x64 mesh with a first order method. Maximum error $2.0e^{-4}$

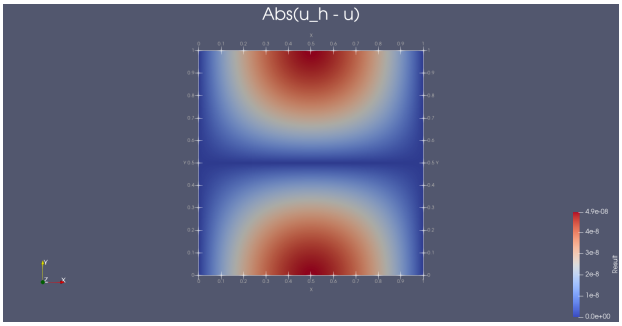


(b) The 64x64 mesh with a second order method. Maximum error $1.0e^{-8}$

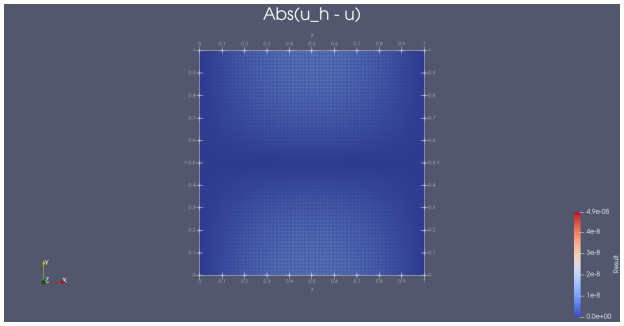


(c) The 64x64 mesh with a third order method. Maximum error $3.8e^{-11}$

Figure 2: The error as the order of the method is increased.



(a) The 4096x4096 mesh with a first order method. Maximum error $4.9e^{-8}$



(b) The 64x64 mesh with a second order method. Maximum error $1.0e^{-8}$

Figure 3: The errors in a first order method with a mesh of approximately 16 million elements are similar to that of a second order method with a mesh of 4096 elements.

5

Within firedrake there are several steps required to implement the method.

1. Create a mesh. In this case the mesh is chosen to be a square, so the the number of nodes is the same in x and y. A square mesh was used as we are in a square domain so it is possible to perfectly fit a structured mesh. But, other forms of mesh such as triangular (unstructured) meshes are available.
2. Define the function space. This is how we tell firedrake that we want to solve the Discontinous Galerkin problem on the mesh. We also define the order p here. Firedrake supports other solver methods apart from Discontinous Galerkin.
3. Create the trial function and test function. The trial function represents u in our equations. The test function represents the test function / variation ∂u. We will use these to solve the equation later.
4. Define $f(x, y)$. This is another component required in our solution, and is just as it was initially defined $f(x, y) = 2\pi^2 sin(\pi x)cos(\pi y)$.
5. We rewrite the weak form as $\iint_\Omega \nabla \delta u \cdot \nabla u = \iint_\Omega \delta u f dx dy$. We construct two variables representing the left hand side and right hand side of this equation. Everywhere we have u, δu, and f we use the trail function, test function, and $f(x, y)$ respectively which we created in steps 3 & 4. Importantly, we only write "*dx", not "*dx*dy", as firedrake takes the symbol "dx" to mean "a small piece of the internal domain regardless of the dimensions" i.e dA.
6. We create the Dirichlet boundary conditions as constant values. We do not need to create the Neuman boundary conditions are they are handled implicitly within the Ritz-Galerkin formulation.
7. We solve the system, asking firedrake to solve where the left hand side equals the right hand side of the weak form from step 5. We must be careful to pass the boundary conditions from step 6 as well.