deal.II Example Code Documentation
AMSC663

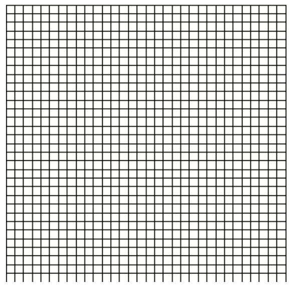
Gareth Johnson

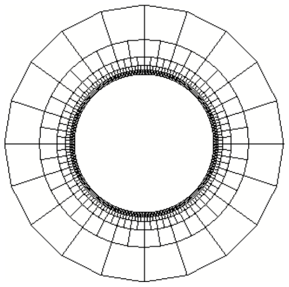
December 12, 2018

In order to learn the functionality of deal.II, I followed various example programs found in deal.II's tutorial [1]. Each tutorial focused on introducing various functionality in the library. For each tutorial that I completed, I will document the PDE that it solved as well as discuss some of the particular functionality that was learned and how it can be used in my own code.

Step-1

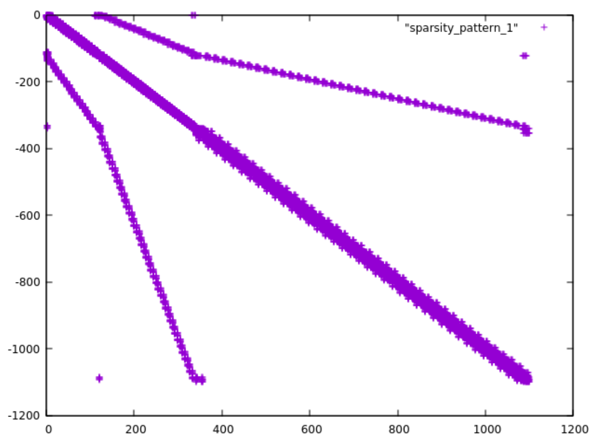
In this tutorial we learned how to generate both a rectangular and circular mesh. It also demonstrated how to globally refine the mesh and how to export a plot of the mesh to a file. Below are plots of each mesh after 5 levels of global refinement.

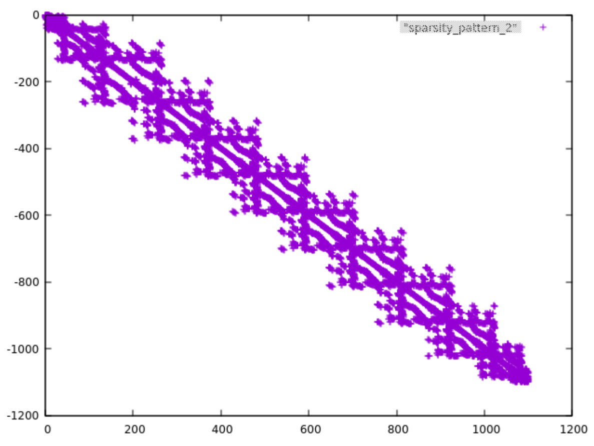




Step-2

In this tutorial we learned how to distribute degree of freedoms (DoFs) for a given finite element type and a given triangulation. In addition, we learned how to renumber the DoF using the Cuthill Mckee front marching algorithm in order for the DoFs to lie closer to the diagonal of the matrix, which can benefit certain solvers. Below are visual representations of the sparsity pattern with and without renumbering the DoFs.



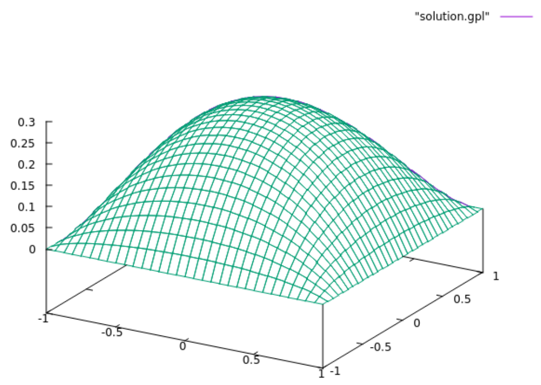


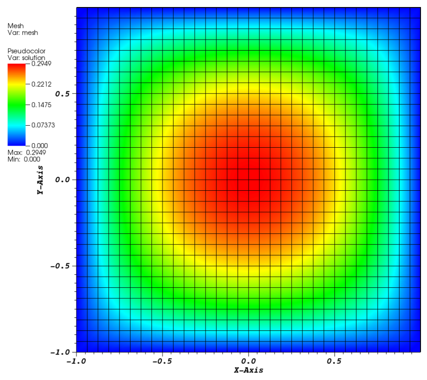
Step-3

In this tutorial we solved Poisson's equation:

$$\begin{cases} -\Delta u = f(x) & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

where $f(x) = 1$ and $\Omega = [0,1]^2$. This was the first program where we actually solved a PDE. We learned how to take the triangulation and DoFs that were computed in Step–2 and actually apply them to compute the bilinear and right hand side of the equation using Q_1 elements and how to incorporate dirichlet boundary conditions. We also learned how to interface with linear algebra routines, such as CG, in order to actually solve the system. Finally, it demonstrated how to export the solution into various graphical formats to view. Below are two such visual representations of the solution.





Step-4

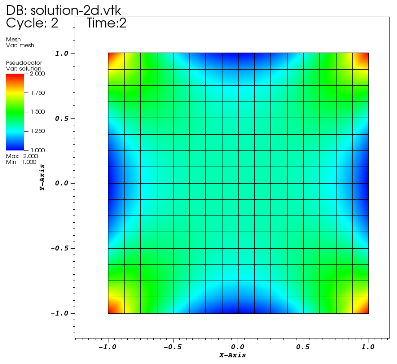
In this tutorial we solved Poisson's equation:

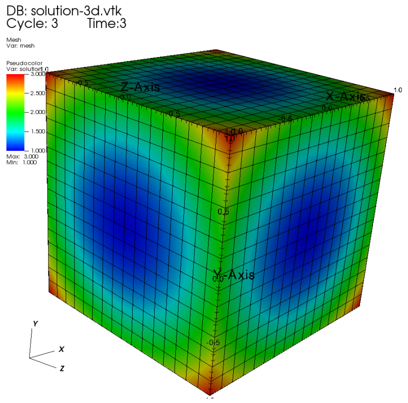
$$\begin{cases} -\Delta u = f(x) & \text{in } \Omega, \\ u = g(x) & \text{on } \partial\Omega, \end{cases}$$

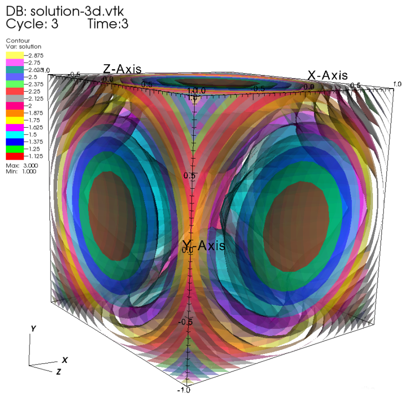
where

$$f(x) = \begin{cases} 4(x^4 + y^4) & \text{if } \Omega \subset \mathbb{R}^2 \\ 4(x^4 + y^4 + z^4) & \text{if } \Omega \subset \mathbb{R}^3 \end{cases}, \quad g(x) = \begin{cases} x^2 + y^2 & \text{if } \Omega \subset \mathbb{R}^2 \\ x^2 + y^2 + z^2 & \text{if } \Omega \subset \mathbb{R}^3 \end{cases},$$

and Ω is the unit square or cube. The new features of this example were how to take advantage of deal.II's dimensionless programming technique. By using C++'s template feature, it allows us to write the code where the dimension is a compile time parameter. This feature allows us to change the dimension of the PDE with almost no modifications to the actual code. In addition, it showed how to implement non-constant forcing functions and dirichlet boundary conditions. Below are plots of the solution in both 2D and 3D.







Step-5

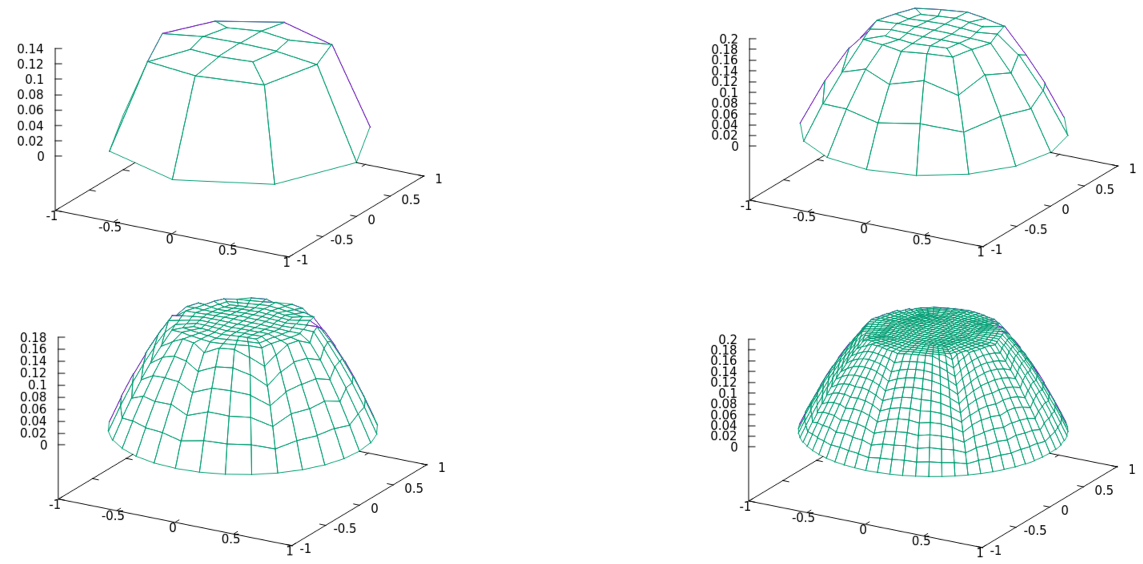
In this tutorial we solved an elliptic equation of the form:

$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = 1 & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

where $\Omega = B(0,1)$ and

$$a(x) = \begin{cases} 20 & \text{if } |x| < 0.5 \\ 1 & \text{otherwise} \end{cases}.$$

The new functionality introduced was how to incorporate non-constant coefficients into the bilinear form, how to use preconditioning (specifically SSOR) when using an iterative solver such as CG, and how to solve the problem on a sequence of globally refined meshes. Below are plots of the solution on increasingly refined meshes.



Step-6

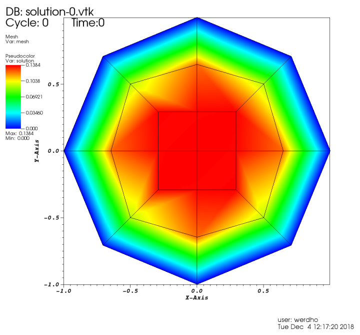
In this tutorial we solved the same PDE as in Step-5:

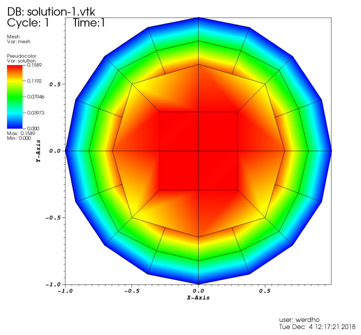
$$\begin{cases} -\nabla \cdot (a(x)\nabla u(x)) = 1 & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

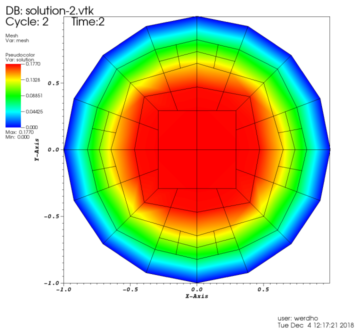
where $\Omega = B(0,1)$ and

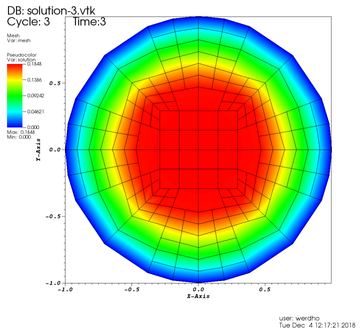
$$a(x) = \begin{cases} 20 & \text{if } |x| < 0.5 \\ 1 & \text{otherwise} \end{cases}.$$

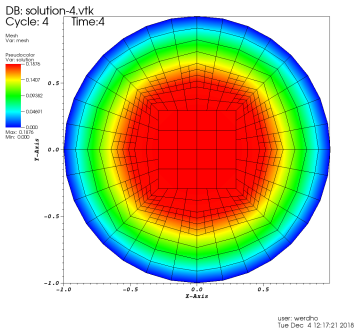
The main difference is that in this program we added the ability to adaptively refine/coarsen the mesh. This process works by first computing the Kelly error estimator on each element, then marking 30% of the elements with the highest error to be refined and 3% of the elements with the lowest error to be coarsened. There are some key things to point out about the refinement/coarsening procedure. First, the percentages chosen were arbitrary and can be easily tuned for specific problems. Second, the percentage of marked elements is not equivalent to the total percentage of elements that will be refined/coarsened. This is because elements are not allowed to be refined twice when neighbor elements are not refined. Lastly, since deal.II uses quadrilateral elements the process of refinement will cause there to be hanging nodes. The method of implementing these was also covered in the tutorial. Below are plots of the solution and mesh as we adaptively refined/coarsened the mesh.

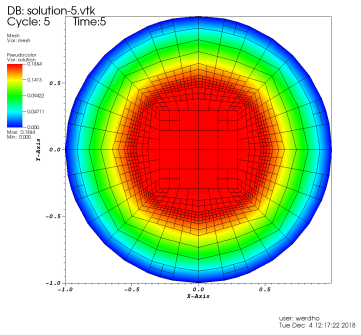


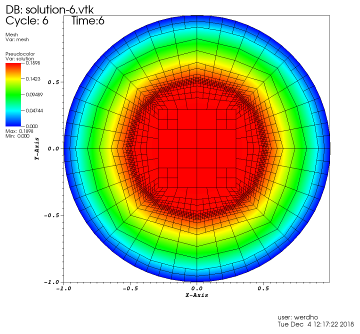


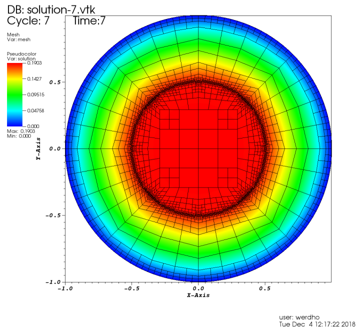












Step-7

In this tutorial we solved the Helmholtz equation:

$$\begin{cases} \Delta u + u = f & \text{in } \Omega \\ u = g_1 & \text{on } \Gamma_1 = \Gamma \cap \{\{x=1\} \cup \{y=1\}\}, \\ \mathbf{n} \cdot \nabla u = g_2 & \text{on } \Gamma_2 = \Gamma \setminus \Gamma_1 \end{cases}$$

where $\Omega = [-1,1]^2$. The solution was chosen to be

$$u(x) = \sum_{i=1}^{3} \exp\left(- \frac{|x - x_i|^2}{\sigma^2} \right),$$

with centers $x_1 = (-\frac{1}{2}, \frac{1}{2}), x_2 = (-\frac{1}{2}, -\frac{1}{2}), (\frac{1}{2}, -\frac{1}{2})$ and $\sigma = \frac{1}{8}$. This choice of u allowed us to compute the appropriate forcing function f and BC g_1, g_2. Using this particular solution, we computed the L^2, H^1-semi, and L^∞ error using global/adaptive refinement and Q_1 elements. Being able to compute the error and convergence rate will be useful in verifying our code for each of the three solvers individually before we try to incorporate them using the Picard iteration. Below are the error and convergence tables, and we note that the convergence rate matches what FEM theory tells us.

cycle	# cells	# dofs	L^2-error	H^1-error	L^∞-error
0	64	81	1.629e+00	2.858e+00	1.683e+00
1	256	289	3.563e-02	1.200e+00	1.324e-01
2	1024	1089	1.196e-02	7.568e-01	7.901e-02
3	4096	4225	3.049e-03	3.824e-01	2.343e-02
4	16384	16641	7.661e-04	1.917e-01	6.109e-03

Figure 1: Error using global refinement and Q_1 elements.

cycle	# cells	# dofs	L^2-error	H^1-error	L^∞-error
0	64	81	1.629e+00	2.858e+00	1.683e+00
1	124	158	5.585e-02	1.201e+00	1.402e-01
2	280	341	2.244e-02	7.925e-01	8.135e-02
3	571	682	1.904e-02	5.185e-01	4.862e-02
4	1087	1250	7.113e-03	3.097e-01	1.985e-02
5	2122	2383	7.559e-03	2.191e-01	1.411e-02
6	4051	4374	7.647e-03	1.543e-01	1.201e-02
7	7699	8215	8.954e-03	1.134e-01	1.321e-02
8	14749	15464	4.107e-03	8.716e-02	6.992e-03

Figure 2: Error using adaptive refinement and Q_1 elements.

n cells		H^1-error		L^2-error	
0	64	2.858e+00	-	1.629e+00	-
1	256	1.200e+00	1.25	3.563e-02	5.52
2	1024	7.568e-01	0.66	1.196e-02	1.58
3	4096	3.824e-01	0.99	3.049e-03	1.97
4	16384	1.917e-01	1.00	7.661e-04	1.99

Figure 3: Estimated convergence using global refinement and Q_1 elements.

Step-8

In this tutorial we solved the elastic equations:

$$\begin{cases} \partial_j(c_{ijkl}\partial_k u_l) = f_i, \quad i = 1, ..., d & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

where Ω is the unit square or cube and

$$f_x(x) = \begin{cases} 1 & x \in B((0,5,0),.2) \cup B((-0,5),.2), \\ 0 & \text{otherwise,} \end{cases} \qquad f_y(x) = \begin{cases} 1 & x \in B((0,0),.2), \\ 0 & \text{otherwise.} \end{cases}$$

The new technique introduced was how to handle vector valued solutions. This will be useful as both the magnetization and the velocity of the fluid are vector valued.

Step-12

In this tutorial we solved the linear advection equation:

$$\begin{cases} \nabla \cdot (\beta u) = 0 & \text{on } \Omega, \\ u = g & \text{on } \Gamma_1-, \end{cases}$$

where $\Omega = [0,1]^2$, $\beta = \frac{1}{|x|}(-x_2, x_1)$, $\Gamma_- = \{x \in \partial\Omega | \beta(x) \cdot \eta(x) < 0\}$, and $g = 1$. This problem was solved using discontinuous elements, in order to use an upwinding technique for the advection equation. This tutorial demonstrated how to handle integrals over faces of edges, which will be useful when solving the magnetization equation (even though the bilinear form will be different than the one presented in this tutorial).

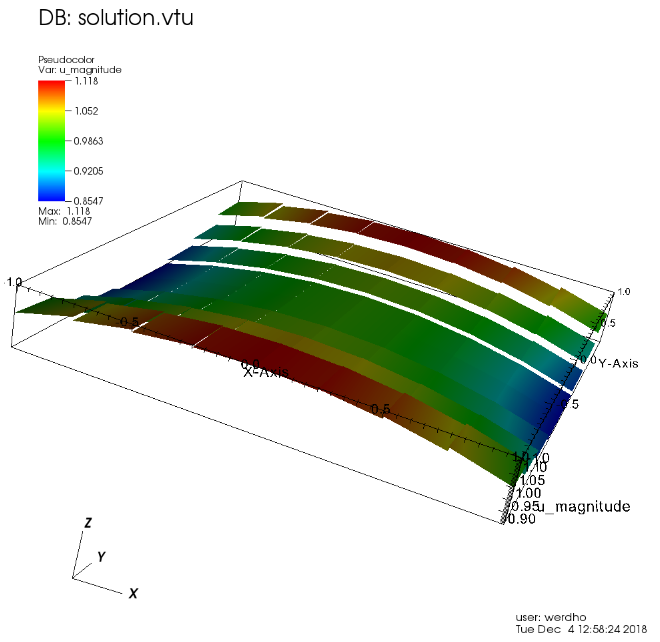
Step-20

In this tutorial we solved Stokes equation:

$$\begin{cases} K^{-1}\mathbf{u} + \nabla p = 0 & \text{in } \Omega \\ -\text{div}\mathbf{u} = -f & \text{in } \Omega \\ p = g & \text{on } \partial\Omega, \end{cases}$$

where K was chosen to be the identity. We chose the exact solution to be

$$p = -\left(\frac{\alpha}{2}xy^2 + \beta x - \frac{\alpha}{6}x^3\right), \quad \mathbf{u} = \begin{pmatrix} \frac{\alpha}{2}y^2 + \beta - \frac{\alpha}{6}x^2 \\ \alpha xy \end{pmatrix}.$$



The above system is a saddle point system and was solved using a Schur complement technique. The interesting technique shown was that we don't actually need to compute the Schur complement, but instead just need to know its action on a vector. Additionally, a method for preconditioning the Schur complement was discussed. This technique will be used to solve the Navier–Stokes system, albeit with a more advanced preconditioner. Below is a plot of the magnitude of the solution.

Step-26

In this tutorial we solved the heat equation:

$$\begin{cases} \partial_t u(x,t) - \Delta u(x,t) = f(x,t) & \text{in } \Omega \times (0,1] \\ u(x,0) = u_0(x) & \text{on } \Omega \times \{0\} \\ u(x,t) = g(x,t) & \text{on on } \partial\Omega \times (0,1]. \end{cases}$$

The solution was chosen to be

$$u(x,y,t) = \sin(3\pi x)e^{-y-2t}.$$

Two main techniques were introduced. The first was how to handle time dependent problems. By using a theta-scheme, we were able to discretize the PDE in time with $\theta = 1$ corresponding to Backward Euler, $\theta = 0$ corresponding to Forward Euler, and $\theta = .5$ corresponding to Crank-Nicholson. Second, a method for adaptive refinement/coarsening for time dependent problems was discussed. The mesh is refined/coarsened at a fixed interval of time steps, using the same refinement technique shown in previous examples. The main issue with this is that the DoFs on the two meshes will be different and thus you will need to interpolate the solution from one mesh to the other. Both of the techniques presented in this tutorial will be used almost exactly in our implementation for each of the three solvers. Movies of both the mesh and solution for the PDE are given in the folder Examples/step-26.

References

[1] W. Bangerth, T. Heister, and G. Kanschat, *Deal.ii differential equations analysis library, tutorial programs.*