Fall 2019: Advanced Topics in Numerical Analysis: Finite Element Methods Assignment 3 (due Dec. 13, 2019)

Terrence Alsup

1. **A priori convergence rates.** We use the MATLAB finite element implementation I showed in class[1] for the Laplace problem and compute a priori convergence rates. We consider the two-dimensional domain $\Omega = [-1, 1] \times [-1, 1]$ and the problem

$$\begin{aligned} -\Delta u &= f \quad \text{on } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

Since we would like to compute L_2 a priori error estimates, it's practical to consider a problem where we know the exact solution. A standard trick to construct such a solution is the method of manufactured solutions: Assume a solution u_{exact} that satisfies the boundary conditions[2], and compute the corresponding right hand side forcing f. Then, solve the problem with this forcing, resulting in a finite element solution u_h and compare with u_{exact}.

(a) Following the method of manufactured solutions, compute the forcing f for the exact solution $u_{\text{exact}}(x, y) = \sin(2\pi x)\sin(\pi y)$.

Solution
We compute

$$\begin{aligned} -\Delta u_{\text{exact}} &= -\partial_{xx} u_{\text{exact}}(x, y) - \partial_{yy} u_{\text{exact}}(x, y) \\ &= 4\pi^2 \sin(2\pi x)\sin(\pi y) + \pi^2 \sin(2\pi x)\sin(\pi y) \\ &= 5\pi^2 \sin(2\pi x)\sin(\pi y). \end{aligned}$$

Thus, the forcing term for the exact solution is

$$f(x, y) = 5\pi^2 \sin(2\pi x)\sin(\pi y).$$

The git[3] repository `https://github.com/georgst/fem19-hw3` contains (some of the) MATLAB files from the above paper, but with a few modifications:

- The files `coordinates.dat` and `elements3.data` contains already the coordinates for a coarse mesh for Ω shown on the left in Figure 3. Since we will only use triangles, the `elements4.dat` file is empty.
- The boundary file `dirichlet.dat` contains all boundary segments and thus `neumann.dat` is empty.

[1] Alberty/Carstensen/Funken: *Remarks around 50 lines of Matlab: short finite element implementation*, Numerical Algorithms 20, (1999). Here's the PDF: `https://www.math.hu-berlin.de/~cc/cc_homepage/download/1999-AJ_CC_FS-50_Lines_of_Matlab.pdf`

[2] Or, you can simply impose the boundary conditions of the exact solution as Dirichlet boundary conditions on $\partial\Omega$.

[3] If you are not familiar with git, you can simply download all the files as a zip-file following this link in a browser.

- The main file, `fem2d.m` now contains a routine to refine the mesh:

  ```
  [coordinates,elements3,dirichlet,neumann]
  = refineRGB(coordinates,elements3,dirichlet,neumann,elemstorefine);
  ```

  This function refines the mesh for all elements whose number appears in the input vector `elemstorefine`. Since we currently do uniform refinement, we simply pass in a vector of all element numbers. The function `refineRGB` updates all mesh-related variables in place, i.e., overwrites them. Thus, it can be called several times to obtain meshes with different levels of refinement.[4]

- The file `sol.m` implements the exact solution we aim at, and after computing the solution, the main function `fem2d.m` computes and outputs

  $$\|u_{\text{exact}} - u_h\|_{L_2} = \left(\int_\Omega (u_{\text{exact}} - u_h)^2 \, dx \right)^{1/2}. \tag{1}$$

  This integral is computed similar to the other finite elements computations by, for each element, using a numerical quadrature rule. Note that simply comparing two solutions at the nodes is not an accurate estimation of the error.

(b) Update the right hand side forcing in `f.m` and solve for different levels of mesh refinement. Plot the resulting L_2-errors in a log-log plot, where you plot the error on the y-axis and the number of elements (or similarly, $1/h$, where h is the mesh size) on the x-axis. What convergence rate do you observe, and is this consistent with the theoretically expected rate? Note that the Nitsche trick allows to gain a power of h in the L^2-error compared to H^1-error.

Solution

Below are the plots of both the finite element solution on the refined mesh as well as the L_2 error for different mesh refinements. Looking at figure 2, we see that we indeed achieve the theoretical rate of $\|u - u_h\|_{L_2} \sim N^{-1}$ asymptotically, where N is the number of elements. Note that since this is a two-dimensional problem the number of elements N scales as $N \sim h^{-2}$. Therefore, we indeed observe that $\|u - u_h\|_{L_2} = O(h^2)$ as we expect.

[4]The function name indicates the kind of refinement the function is doing—here RGB stands for red-green-blue refinement. Since we use uniform refinement, you will not see the effect of different refinements in this example, but refinements differ in the way they deal with elements that get refined next to elements that do not get refined. One needs to be careful to avoid hanging nodes (the case that two triangles share an edge and one gets refined, the other one not), and to guarantee shape regularity, i.e., to avoid triangles with very sharp angles. As we have seen in class, shape regularity is important to guarantee nicely behaved constants in a priori estimates.

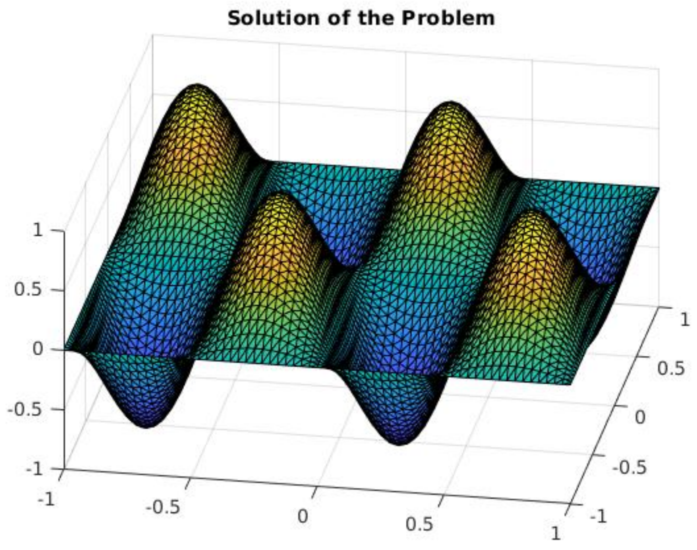


Figure 1: The computed finite element solution with $N = 8192$ triangle elements with linear basis functions.

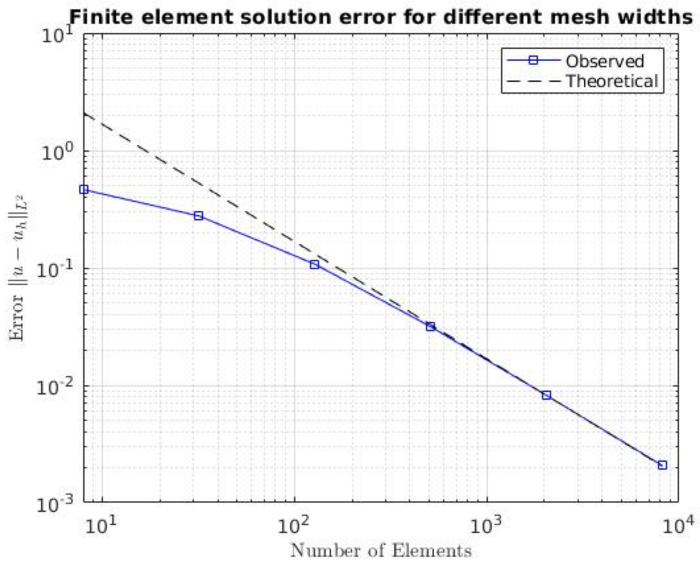


Figure 2: The error of the finite element solution for different uniform mesh refinements (blue) compared to the theoretical convergence rate of $O(N^{-1})$ (black).

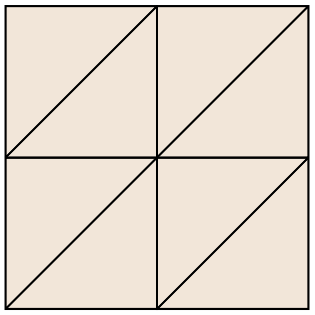
2. **The Motz problem.** Let us consider the following problem over the domain $\Omega = [-1, 1] \times [0, 1]$, where we impose different boundary conditions (see also the right image in Figure 3):

$$-\Delta u = 0 \qquad \text{on } \Omega, \tag{2}$$

$$u = 0 \qquad \text{on } \Gamma_1 := (-1, 0) \times \{0\}, \tag{3}$$

$$u = 500 \qquad \text{on } \Gamma_2 := \{1\} \times (0, 1), \tag{4}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega \setminus (\Gamma_1 \cup \Gamma_2). \tag{5}$$



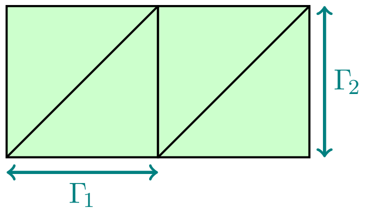


Figure 3: Initial discretizations for the domain $\Omega = [-1,1] \times [-1,1]$ (Problem 1) and domain $\Omega = [-1,1] \times [0,1]$ for (Problem 2).

The exact solution for this problem can be written as a series expansion with numerically approximated coefficients. I implemented that solution in the file `sol_motz.m`. Now your task is to update all files from the previous problem, solve the Motz problem on a sequence of refined meshes and plot the L_2 convergence rate. How does it compare with the theory? To update the files:

- Number the 6 corners in the right mesh in Figure 3, and update the `coordinates.dat` file. Then update the `elements3.dat` file to specify the 4 triangles. Be careful that you always order the corners anti-clockwise!
- Update the boundary condition files `neumann.dat` and `dirichlet.dat`. When you specify the edges, this must also be done in an anti-clockwise way as the code assumes that the outward pointing normal is obtained by a 90 degree anti-clockwise rotation!
- Update the Dirichlet boundary condition file `u_d.m` to reflect the boundary conditions of the Motz problem. Also, don't forget to update `f.m`.

Solution

The two figures below show the computed finite element solution for the Motz problem as well as the rate of convergence of the L_2 error. We can see that the computed solution is not smooth at the point $(0,0)$ since there is a sharp change in the gradient. Thus, $u \notin H^2(\Omega)$ and we cannot expect the same convergence rate of $O(h^2)$ as we did before. Instead, since u is only in $H^1(\Omega)$ we can expect that $\|u - u_h\|_{L^2} = O(h)$. This is indeed the observed rate since $h \sim N^{-1/2}$ due to being in two dimensions.

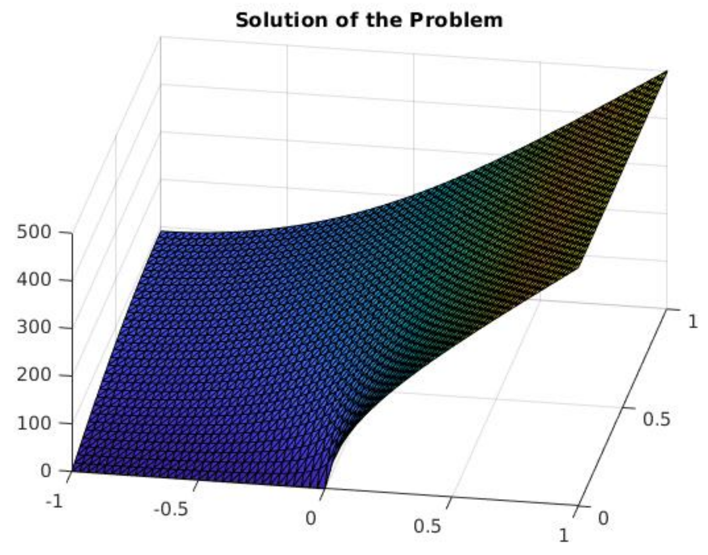


Figure 4: The computed finite element solution with $N = 4096$ triangle elements with linear basis functions.

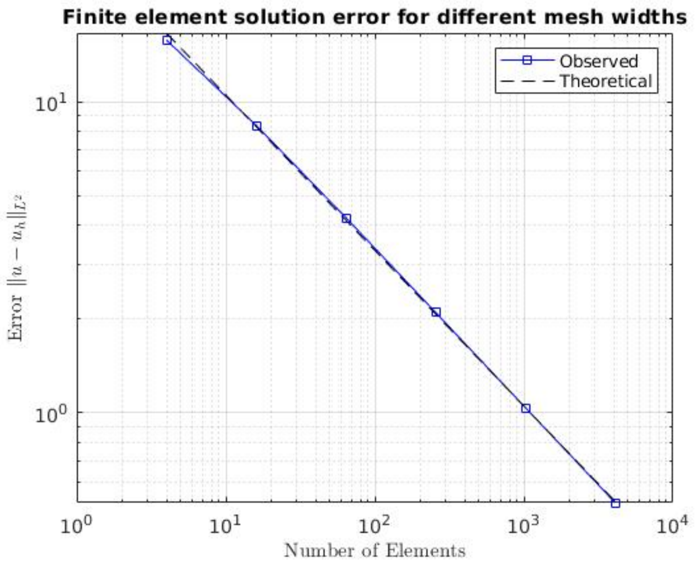


Figure 5: The error of the finite element solution for different uniform mesh refinements (blue) compared to the theoretical convergence rate of $O(N^{-1/2}) = O(h)$ (black).

3. **A nonlinear problem.** As example for a nonlinear problem, let's consider the Bratu problem from class, where $\lambda \geq 0$:

$$\begin{aligned} u'' + \lambda e^u &= 0 \quad \text{on } (0,1), \\ u(0) = 0, u(1) &= 0. \end{aligned}$$

The corresponding weak form is: Find $u \in V := H_0^1(0,1)$ such that for all $v \in V$ holds $G(u;v) = 0$, where $G(u;v)$ is the weak form for the above problem. To solve this nonlinear problem, we follow the linearize-then-discretize approach, i.e., we apply Newton's method for functions and then solve each linearized problem using the finite element method. To apply Newton's method, one can either linearize on the strong or the weak form level. Let's do the latter, i.e., we compute derivatives of G with respect to u, keeping v fixed. Differentiating with respect to functions is formally similar to standard differentiation, for instance, the directional derivative $\delta_u G$ of G in a direction $\hat{u} \in V$ is defined as:

$$\delta_u G(u;v)(\hat{u}) = \lim_{\varepsilon \to 0} \frac{G(u + \varepsilon\hat{u}; v) - G(u;v)}{\varepsilon}. \tag{6}$$

The Newton step, written in weak form, is then: Given $u^k \in V$, find the Newton update $\hat{u} \in V$ as the solution of

$$\delta_u G(u^k; v)(\hat{u}) = -G(u^k; v) \quad \text{for all } v \in V. \tag{7}$$

and perfom the Newton update, possibly with a damping factor $\mu \leq 1$:

$$u^{k+1} = u^k + \mu\hat{u}. \tag{8}$$

(a) Compute the weak form (7) explicitly for the Bratu problem, and give the corresponding strong form.

Solution
To find the weak form of the Bratu problem we multiply by a test function $v \in V$ and integrate.

$$\int_0^1 u''v + \lambda e^u v \; dx = \int_0^1 -u'v' + \lambda e^u v \; dx = 0$$

Therefore, we define

$$G(u,v) := \int_0^1 -u'v' + \lambda e^u v \; dx$$

for all $u, v \in V$. Now we compute the directional derivative in a direction $\hat{u} \in V$ by

$$\begin{aligned}
\delta_u G(u,v)(\hat{u}) &= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left\{ \int_0^1 -(u+\varepsilon\hat{u})'v' + \lambda e^{u+\varepsilon\hat{u}} v \; dx - \int_0^1 -u'v' + \lambda e^u v \; dx \right\} \\
&= \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left\{ \int_0^1 -\varepsilon\hat{u}'v' + \lambda e^u v \left(e^{\varepsilon\hat{u}} - 1 \right) \right\} \\
&= \int_0^1 -\hat{u}'v' + \lambda e^u v\hat{u} \; dx
\end{aligned}$$

where the last line follows from the dominated convergence theorem and the fact that

$$\lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \left(e^{\varepsilon\hat{u}} - 1 \right) = \hat{u}$$

for all $x \in (0,1)$. Thus, the weak form of the Newton update is: given $u^k \in V$, find $\hat{u} \in V$ such that

$$\int_0^1 -\hat{u}'v' + \lambda e^{u^k} v\hat{u} \; dx = -\int_0^1 -(u^k)'v' + \lambda e^{u^k} v \; dx$$

for all $v \in V$. This is now a linear equation for $\hat{u}$ and the right hand side does not depend on $\hat{u}$.

(b) Modify one of the implementations from one of the first homeworks (i.e., use either linear, quadratic or Hermite elements) to compute the corresponding Newton updates. For $\lambda = 2$ try to solve Bratu's problem with the two different initializations $u^0 \equiv 0$ and $u^0(x) = 14x(1-x)$.[5] Try to use full Newton steps (i.e., $\mu = 1$) and if you observe convergence problems, try damping, i.e., $\mu < 1$. Terminate the Newton iteration when the update $\hat{u}$ is small. Report your experience and plot the solution(s) you find.

Solution

The figure below shows the two computed solutions to Bratu's problem. To compute the solution I used 50 linear finite elements with a uniform mesh over the interval $(0, 1)$. Additionally I used 10 full Newton steps with $\mu = 1$. I did not run into any convergence issues due to lack of damping. My implementation in Python could have been much more efficient since I did not take advantage of the fact that the stiffness matrix A is sparse.

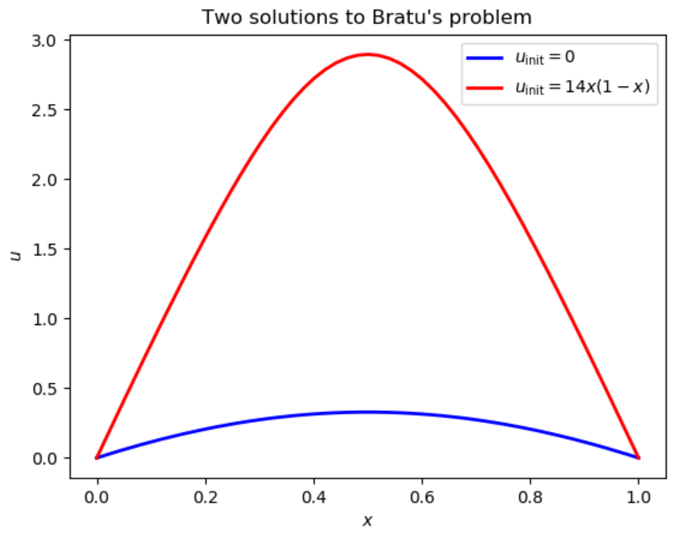


Figure 6: The two computed solutions to Bratu's problem when $\lambda = 2$.

[5]It is important that the initialization satisfies the Dirichlet boundary conditions since all Newton updates have zero Dirichlet boundary conditions. If we search for a solution that has non-zero Dirichlet conditions, one can choose an initialization that satisfies these boundary conditions and then all updates $\hat{u}$ satisfy zero Dirichlet conditions.