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

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 $u_{\text{exact}}(x, y) = \sin(2\pi x)\sin(\pi y)$.

Solve:

Since $f = -\Delta u$, it's easily computed that $f = 5\pi^2 \sin(2\pi x)\sin(\pi y)$

(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.

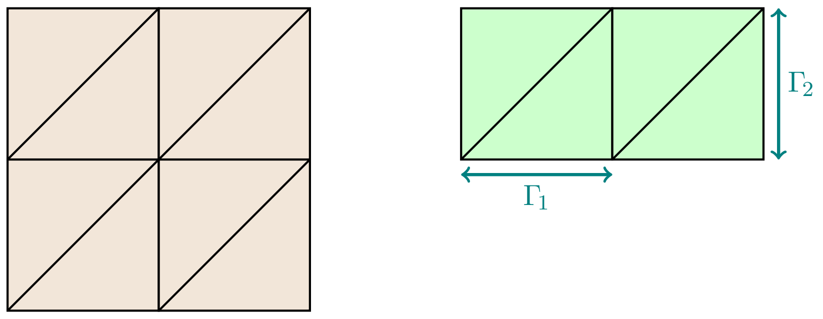

Figure 1: Initial discretizations for the domain $\Omega = [-1, 1] \times [-1, 1]$ (Problem 1) and domain $\Omega = [-1, 1] \times [0, 1]$ for (Problem 2).

Solve:

[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 $\partial\Omega$.

Generally, this whole Homework 3 is in my Github repo[3], which is public and can be found in footnote. Problem 1 is in folder 'problem1'. I order the nodes counter-clockwise in 'coordinates.dat', specify the elements in 'elements3.dat' and specify the dirichlet Boundary in 'dirichlet.mat'. In a series of mesh size $1/h = 1, 2, 4, 8, 16, 32, 64$, L_2 norm error with manufactured solution is computed and plotted in log-log scale in fig.2, where a clearly 2nd order convergence is observed. That's consistent with the theoretical prediction.

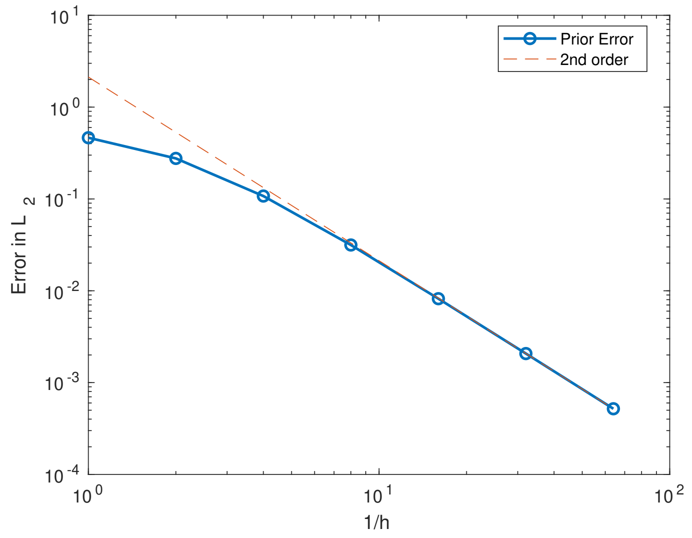

Figure 2: Mesh size $1/h = 1, 2, 4, 8, 16, 32, 64$, L_2 norm error with manufactured solution is plotted in log-log scale and shows 2nd order convergence.

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 1):

$$-\Delta u = 0 \qquad \text{on } \Omega, \tag{1}$$

$$u = 0 \qquad \text{on } \Gamma_1 := (-1, 0) \times \{0\}, \tag{2}$$

$$u = 500 \qquad \text{on } \Gamma_2 := \{1\} \times (0, 1), \tag{3}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \partial\Omega \setminus (\Gamma_1 \cup \Gamma_2). \tag{4}$$

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 1, 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!

[3] `https://github.com/CecilMartin/fem19-hw3`

- 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`.

Solve:

Code of this problem is in 'problem2' folder of my Github repo mentioned previously. Number of nodes is in 'coordinates.dat', elements are specified in 'elements3.dat' counter-clockwisely. Neumann BC is given in 'neumann.dat' and Dirichlet BC is in 'dirichlet.dat'.

f.m gives zero-force condition and 'u_d.m' specifies the value on Dirichlet Boundary. Mesh sizes are refined as $1/h = 1, 2, 4, 8, 16, 32, 64$, L_2 norm error with analytic solution (evaluated by numerical approximation) is computed and plotted in log-log scale in fig.3, where a perfect 1st order convergence is observed. That's consistent with the theoretical prediction.

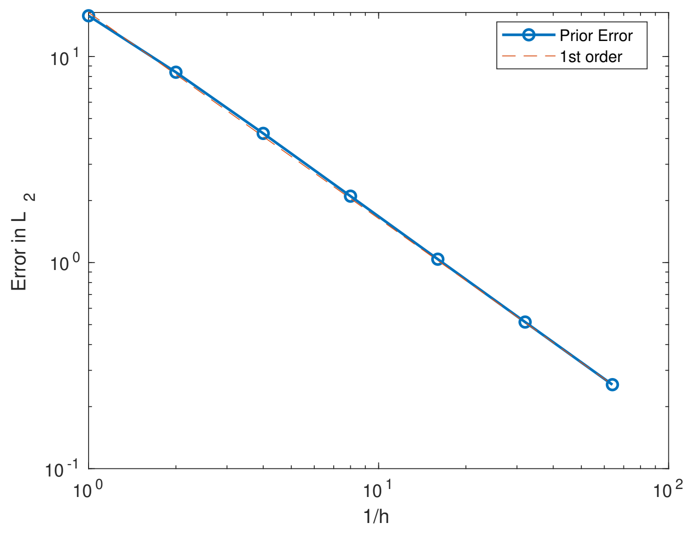

Figure 3: Mesh size $1/h = 1, 2, 4, 8, 16, 32, 64$, L_2 norm error with analytic solution (evaluated by numerical approximation) is plotted in log-log scale and shows 1st order convergence.

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

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

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:

$$\delta_u G(u;v)(\hat{u}) = \lim_{\varepsilon \to 0} \frac{G(u + \varepsilon \hat{u}; v) - G(u;v)}{\varepsilon}. \tag{5}$$

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

$$\delta_u G(u^k; v)(\hat{u}) = -G(u^k; v) \quad \text{for all } v \in V. \tag{6}$$

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

$$u^{k+1} = u^k + \mu\hat{u}. \tag{7}$$

(a) Compute the weak form (6) explicitly for the Bratu problem, and give the corresponding strong form.

Solve:

From class, we know that weak form of original nonlinear equation is

$$G(u, v) = \int_0^1 u'v'dx + \int_0^1 \lambda e^u v dx = 0$$

.

Thus, Newton update $\hat{u}$ would be $\delta_u G(u^k; v)(\hat{u}) = -G(u^k; v)$:

$$\int_0^1 \hat{u}'v'dx + \int_0^1 \lambda v e^u \hat{u} dx = -\int_0^1 u'v'dx - \int_0^1 \lambda e^u v dx, \tag{8}$$

whose strong form is

$$-\hat{u}'' + \lambda e^u \hat{u} = u'' - \lambda e^u$$

.

(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 $u^0(x) = 14x(1-x)$.[4] 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.

Solve:

First, from literature, we know that for $\lambda = 2$, there's two analytic solution

$$u_{real} = -2\log\left(\frac{((x - 0.5)\omega/2)}{\cosh(\omega/4)}\right),$$

where ω satisfies $\omega = \sqrt{2\lambda}\cosh(\omega/4)$. Numerically solve this equation and I got $\omega_1 = 2.3575510538774020425939799885899$ and $\omega_2 = 8.5071995707130261295811297868289$.

Then, to solve this nonlinear weak form equation, I use FEM quadratic elements from problem 4 in Hw1. All codes are in folder 'problem3' in my Github repo mentioned previously in footnotes.

[4] 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.

Denote bi-liear form $a_1(u,v) == \int_0^1 u'v'dx$ and $a_1(u,v) == \int_0^1 uvdx$, with corresponding stiffness matrix A_1 and A_2. Set $p=1,\ q=1$ in hw1 and we could get A_1 and A_2 easily. Hence, the weak form of Newton update could be written as

$$a_1(\hat{u},v) + \int_0^1 \lambda v e^u \hat{u} dx = -a_1(u,v) - \int_0^1 \lambda e^u v dx$$

Since there's annoying non-linear term, I tried to linearize it. For the LHS non-linear term, I treat it as constant e^u on nodes of $\hat{u}$. For the RHS, I approximate it as quadratic function so that I could write it as,

$$a_1(\hat{u},v) + a_2(e^u\hat{u},v) = -a_1(u,v) - \lambda a_2(\mathbf{Q}(e^u),v),$$

where $\mathbf{Q}$ is quadratic approximation.

So in matrix form it would be linear system,

$$(A_1 + \lambda A_2 diag(e.^u))x = -A_1 u - \lambda A_2 \cdot (e.^u)$$

Solve this linear system and update solution with $\mu\hat{u}$. Terminate this iteration when L_2 norm of $\hat{u}$ is less than eps$=1e-12$ or iteration reaches max_iter$=1000$. ('FEM1D.m')

Set $\mu=1$ first, i.e. no damping. And initialize it with $u_0=0$ and $u_0=14x(1-x)$ and compare it with analytical solution discussed above. Generally, it converges to one of the analytic solution as fig.4, 5 shows. The convergence speed is super fast as less than 10 iteration and $\hat{u}$ has L_2 norm less than eps. Newton method in FEM to solve non-linear problem is very convinient. All we need to do is to derive the weak form of Newton update step and compute it with finite elements.

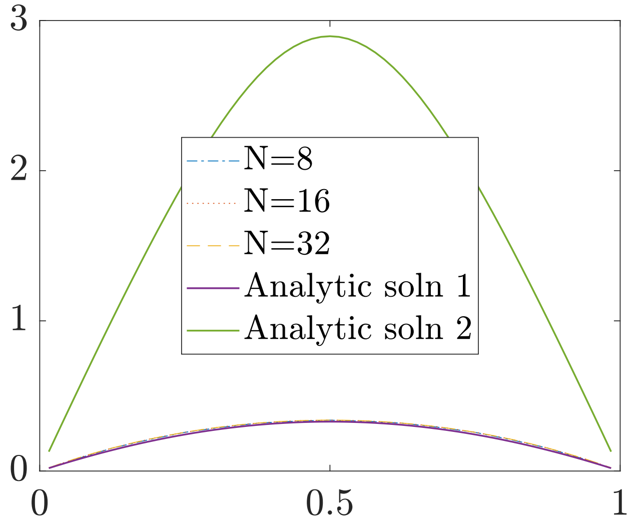

Figure 4: $u_0 = 0$

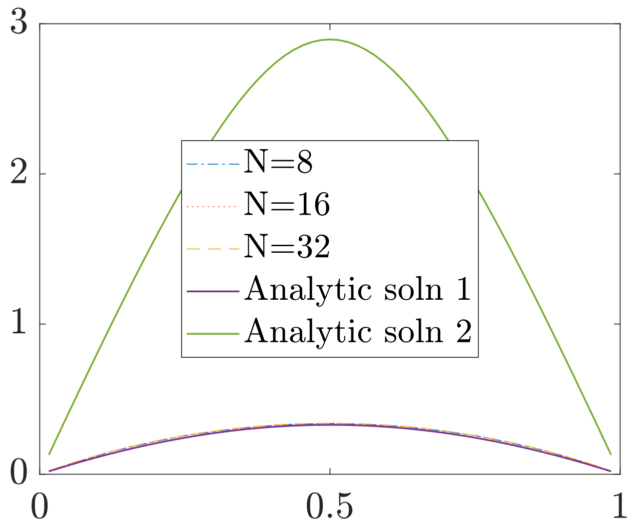

Figure 5: $u_0 = 14x(1-x)$