Project 3: Semilinear elliptic equation

Numerical approximation of PDEs

June 21th, 2024

Julie Charlet
SCIPER: 314454

Mathilde Simoni
SCIPER: 371423

1. Introduction

This project's goal is to compute the numerical solution of the following nonlinear problem:

$$\begin{cases} -\Delta u + \alpha u^3 = f, & \text{in } \Omega \\ u = 0, & \text{on } \delta\Omega \end{cases} \tag{1}$$

The computational domain is $\Omega = [0,1]^2$, $f = 100$ and $\alpha > 0$. Because of the non-linear αu^3 term, this equation can not be solved with the usual methodology, based on Lax-Milgram's Lemma. Instead, it can be discretized as in Equation 2 and solved iteratively, providing an initial guess u_0:

$$\begin{cases} -\Delta u_{n+1} + \alpha u_n^2 u_{n+1} = f, & \text{in } \Omega \\ u = 0, & \text{on } \delta\Omega \end{cases} \tag{2}$$

The new iterate is updated each time by solving Equation 2 for the unknown u_{n+1}. The algorithm stops once the error $\|u_{n+1} - u_n\|$ is smaller than a set tolerance threshold (tol). In this project, three iterative methods will be compared: *fixed point iteration*, *Anderson acceleration* and *Newton's method*.

The code for this project is available on GitHub.

2. Parameters

The domain $\Omega = [0,1]^2$ was discretized using a triangular mesh of size 0.1 resulting in 142 vertices. It has been generated with the code available on the class GitHub repository. A visualization of the resulting mesh in displayed in Figure 1. Regarding the Finite Element Method, $\mathbb{P}_1$ basis functions were used, as well as a Gauss quadrature scheme of order 6. The tolerance threshold for the final error was set at $\text{tol} = 10^{-6}$. Finally, an initial guess $u_o = 0$ was provided for every element. The three iterative methods were tested for $\alpha = \{0.1, 2, 5\}$ (except for the *fixed-point iteration* that was tested only for $\alpha = \{0.1, 2\}$).

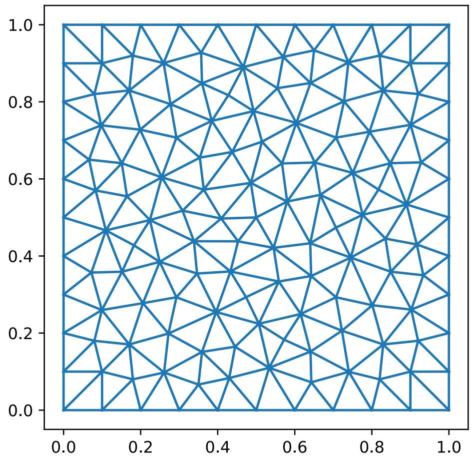

Figure 1: Tetrahedral mesh with domain $\Omega = [0, 1]^2$

3. Questions

3.1. Question 1

Weak Form

We derive the weak form of Equation 2, setting $c = \alpha u_n^2 > 0$ since $\alpha > 0$.

We seek to find $u \in H_0^1(\Omega)$ such that:

$$-\Delta u + cu = f. \tag{3}$$

Let $v \in H_0^1(\Omega)$ be a test function. We multiply Equation 3 by v and integrate over Ω:

$$\int_\Omega (-\Delta u + cu)v \; d\Omega = \int_\Omega fv \; d\Omega$$
$$-\int_\Omega (\Delta u)v \; d\Omega + \int_\Omega cuv \; d\Omega = \int_\Omega fv \; d\Omega.$$

We use integration by parts to simplify the first part of the left hand side:

$$-\int_\Omega (\Delta u)v \; d\Omega = -\int_{\delta\Omega} (\nabla u)v \cdot \boldsymbol{n} \; d\Gamma + \int_\Omega \nabla u \nabla v \; d\Omega$$
$$-\int_\Omega (\Delta u)v \; d\Omega = \int_\Omega \nabla u \nabla v \; d\Omega,$$

since $v \in H_0^1(\Omega)$ cancels out on the boundary. Thus, we are left with the following weak form:

$$\int_\Omega \nabla u \nabla v \; d\Omega + \int_\Omega cuv \; d\Omega = \int_\Omega fv \; d\Omega, \quad \forall v \in H_0^1(\Omega). \tag{4}$$

3.2. Question 2

Fixed-Point Method

Let $f(u)$ the function that solves Equation 4 for u. The fixed point method iterates as follows: $u_{n+1} = f(u_n)$ starting with an initial guess u_0.

To solve Equation 4, we use the FEM code provided for the class. In particular, we compute the stiffness matrix using `stiffness_with_diffusivity_iter()` without diffusivity term. Regarding the mass matrix, we slightly modify the function `mass_with_reaction_iter()` to handle a reaction term that does not depend on the position in Ω but on the value function value u at a vertex, since our reaction term is $c = \alpha u_n^2$. Finally, we use `poisson_rhs_iter()` for the right hand side. These functions, as well as the assembly routines, were provided as part of exercise session 5.

The algorithm converges in 12 iterations for $\alpha = 0.1$ and 353 iterations for $\alpha = 2$. The obtained solutions are displayed in Figure 2.

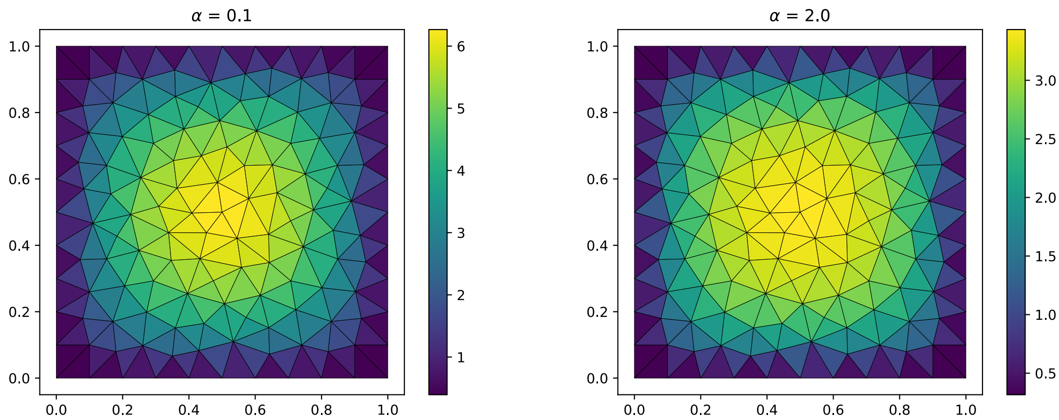

Figure 2: Discrete Solution for $\alpha = 0.1$ and $\alpha = 2$

3.3. Question 3

Anderson Acceleration

Anderson acceleration is used to speed up the fixed-point scheme. We use the `scipy.optimize.anderson` function that seeks the root of the functional $F : \mathbb{R}^n \to \mathbb{R}^n$. In our case, the functional to pass is $F(u) = f(u) - u$ (since it is equivalent to the convergence of the fixed point method: $u = f(u)$).

The number of iterations required for the algorithm to converge is 7 for $\alpha = 0.1$ and 24 for $\alpha = 2$. Thus, the accelerated scheme is far better, especially for $\alpha = 2$ with a speedup of almost 15.

3.4. Question 4

Newton's Method

The Newton's method iterates as follows: $u_{n+1} = u_n + \partial u_n$ where ∂u_n satisfies:

$$\int_\Omega \nabla\phi \cdot \nabla\partial u_n d\Omega + \int_\Omega 3\alpha u_n^2 \phi \partial u_n d\Omega = \int_\Omega \nabla\phi \cdot \nabla u_n d\Omega - \int_\Omega \phi(\alpha u_n^3 - f) d\Omega, \forall \phi \in H_0^1(\Omega) \quad (5)$$

Similarly to the fixed-point scheme, we can reuse some of the functions from the exercise session 5. The first term of the LHS is a stiffness matrix, so we use the function `stiffness_with_diffusivity_iter()`. The second term of the LHS is a mass matrix which is slightly different form the one use previously, as it is multiplied by a factor of 3.

The RHS of Equation 5 requires more changes. The second term can be calculated with a slightly modified version of `poisson_rhs_iter()`. For the first term, we use the `stiffness_with_diffusivity_iter()` and multiply it by the available scalar values u_n obtained at the previous iteration.

Solving the system with newton's method for the α values of $\{0.1, 2, 5\}$ yields the results presented in Figure 3. The algorithm converges in 5 iterations for $\alpha = 0.1$, 7 for $\alpha = 2$ and 8 for $\alpha = 5$.

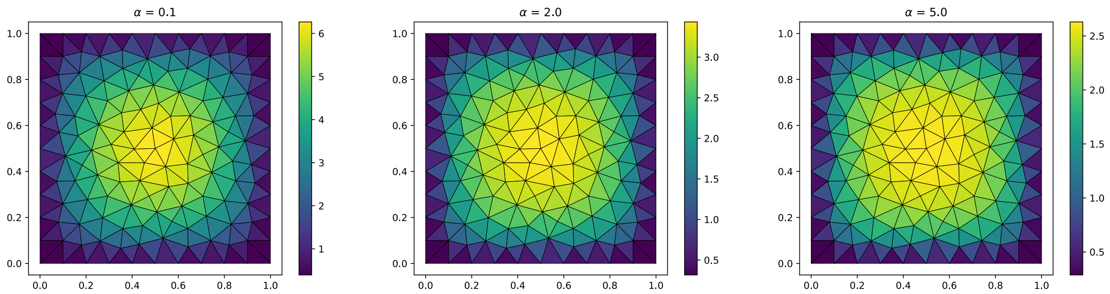

Figure 3: Discrete Solution of Newton's method for $\alpha = 0.1, 2, 5$

When comparing the number of iterations required by each of the three methods (*fixed point iterations*, *Anderson acceleration* and *Newton's method*) in Figure 4, one can observe that Newton's method fare better than the two other. The algorithm converges in 5 iterations for $\alpha = 0.1$, 7 for $\alpha = 2$ and 8 for $\alpha = 5$. Nevertheless, the number of iterations in Newton's method and Anderson acceleration scale in a similar way as a function of alpha. By contrast, the fixed-point iteration scales faster with α, making the computation for $\alpha = 5$ unacceptable.

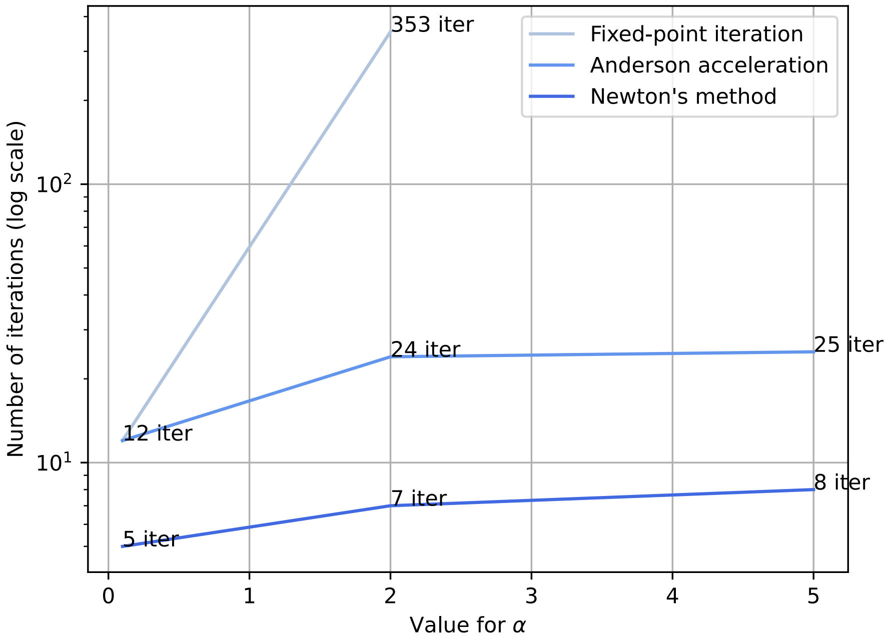

Figure 4: Required number of iterations for the three methods as a function of α