MTH 654-2024 Homework A1. **Noah Prentice**
Date: October 18, 2024
Prof. M. Peszynska

Note.

Code for this assignment was done in Python. Only important code snippets and outputs are shown here, but complete files can be found at this GitHub repository.[a] All code for Exercise N is in `ExN.py`.

[a] URL: https://github.com/NoahPrentice/Nonlinear-Coupled-PDE-MTH654-F24/tree/main/A1

Exercise 1. Consider the ODE

$$u' = u - u^3,\ u(u) = u_{init}. \tag{1}$$

Explore the stability of FE, BE and CS schemes for (1) allowing variable time step τ.

(i) Write your code that allows variable τ (show me snippets for time stepping). For nonlinear solver, use `fzero` or see Pbm 2. Test your code on some ODE with an easy analytical solution. Also, you can compare to `ode45` (do not turn these tests in).

(ii) Experiment with $u_{init} = 2$, 0.1 or 0.9, with $\tau = 0.1, \tau = 0.9$ or even $\tau = 2$. (For each case run until $T = 5$.)

Report if the numerical solution for FE, BE, and CS is monotone and energy stable (should it be?). Illustrate with plots or tables and theory if possible.

Solution. **(i)** Time stepping for FE is:

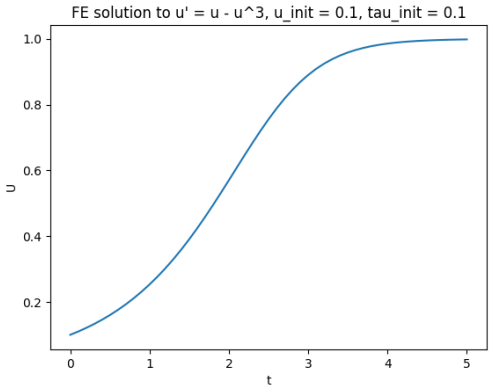
```
while True:
    update_tau()
    t += tau
    if t >= T:
        break
    t_values.append(t)

    # Solve (u - u_prev)/tau = u_prev - (u_prev)^3 for u.
    u = u_prev + tau * u_prev - tau * math.pow(u_prev, 3)
    u_values.append(u)
    u_prev = u
```

Time stepping for BE is:

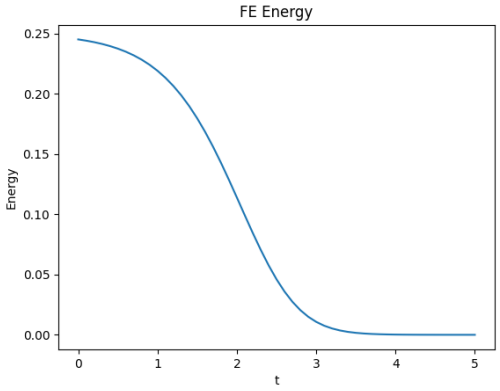
```
while True:
    update_tau()
    t += tau
    if t >= T:
        break
    t_values.append(t)

    # Solve F(u) = 0 for u.
    u = fsolve(F, u_prev)[0]
    u_values.append(u)
    u_prev = u
```

Time stepping for CS has the same structure as that for BE. The two differ only in how τ changes and which function F we use to solve $F(u) = 0$ at each time step; for BE $F(u) := u - u_{prev} - \tau u + \tau u^3$, while for CS $F(u) := u - u_{prev} - \tau u_{prev} + \tau u^3$.[1]

(ii) For $u_{init} = 0.1$, $\tau = 0.1$, all three methods exhibit energy stability:



(a) FE numerical solution, $\tau = 0.1$



(b) Energy of FE solution

However, when τ is larger (for instance with $\tau = 0.9$ and $u_{init} = 2$), we do not achieve stability for FE:

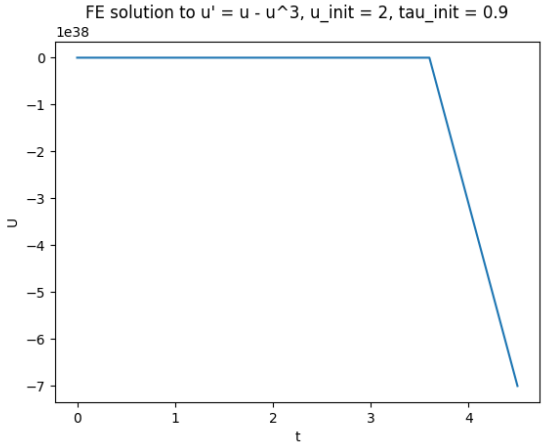


Figure 2: FE numerical solution, $\tau = 0.9$

This is expected, as we do not have a theoretical guarantee of energy stability for FE or BE for the nonlinear Cahn-Allen problem. Luckily, CS continues to be energy stable even for large τ, as proved in lecture on 10/16.

□

[1] Note that the code uses `fsolve()` from `scipy.optimize` to solve $F(u) = 0$ in place of MATLAB's `fzero`.

Exercise 2. Consider $F(u) = u^3 - u$. Use the Newton iteration to find its roots, i.e, the points where $F(u) = 0$. These roots can (perhaps) be also found with a fixed point set-up $u = G(u)$ (choose $G(u)$ wisely). Apply the theory and check what happens in practice.

A region of attraction is a region (set of initial guesses $u^{(0)}$) for which the algorithm for Newton converges quadratically. For fixed point iteration, this is a region in which the fixed point iteration converges. The attraction regions can be empty.

(i) Is the problem $F(u) = 0$ or $u = G(u)$ well-posed?

(ii) Find the attraction region for Newton iteration around the root $u^* = 1$ with initial guess $u^{(0)}$ by experimentation; compare with the theoretical estimates.

(iii) Do the same for fixed point iteration.

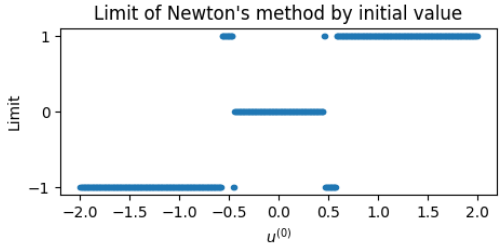
(iv) **Extra:** Attempt the "pseudo-transient continuation", e.g., apply Forward Euler method to the ODE $u' + F(u) = 0, u(0) = 2$ to advance the initial guess u_0 towards the root. Compare to what happens if you try instead the ODE $u' - F(u) = 0, u(0) = 2$? Explain what you see in practice, and what theory can explain what you see. How would you choose the time step in the Forward Euler method to succeed?

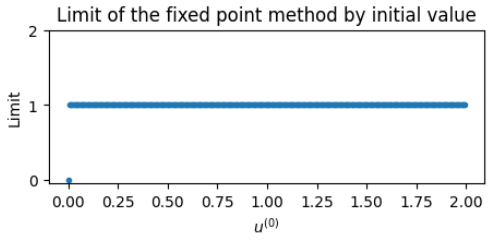
Solution. **(i)** The problems $F(u) = 0$, $u = G(u)$ are not globally well-posed. That is, if we are looking for solutions $u \in \mathbb{R}$, then uniqueness does not hold: $u = -1, 0, 1$ are all solutions. So, to achieve well-posedness, we must restrict the domain of F or G to a subset of $\mathbb{R}$ containing exactly one of $-1, 0, 1$.

(ii) Plotting the limit of Newton iteration against the initial guess yields the result in Figure 3a. This indicates that we have convergence to $u^* = 1$ whenever $u^{(0)} > 0.7$. This is a bit better than theoretical estimates may guarantee: Theorem I.13 in lecture notes guarantees q-quadratic convergence in $B(u^*, \varepsilon)$ for $\varepsilon = \min\{r, \frac{1}{2\gamma\beta}\}$ where β is an upper bound on $\|F'(u^*)^{-1}\|$ and $F' \in \mathrm{Lip}_\gamma(B(u^*, r))$.

In order to guarantee this convergence for $u^{(0)} > 0.7$, we need $\varepsilon \geq 0.3$. Since $F'(u^*) = 2$, we can take $\beta = 1/2$, but then we need $r \geq 0.3$ and $\gamma \leq 3.\overline{3}$ in order for $\varepsilon \geq 0.3$. But, unfortunately, $F' \notin \mathrm{Lip}_{3.\overline{3}}(B(u^*, 0.3))$: in fact, $|F''(1.25)| = 7.5 \geq 3.\overline{3}$. Thus, the best we can do for quadratic convergence in this case is $\gamma = 8$ which guarantees q-quadratic convergence for $u^{(0)} \in B(0.125, 1)$.[2]

Plotting the limit of fixed point iteration with $G(u) = \sqrt[3]{u}$ against the initial guess yields the result in Figure 3b. This indicates that we have convergence to $u^* = 1$ whenever $u^{(0)} > 0$. This is again better than theoretical estimates may guarantee: Theorem I.11 in lecture notes guarantees q-linear convergence when G is a contraction on $U \subseteq^{cl} \mathbb{R}$, which is not true on $(0, \infty)$ as $G'(0.1) = 1.547 > 1$. We *are* guaranteed convergence on $[0.2, N]$ for any $N \in \mathbb{R}$ as $G' < 1$ on this interval. □





(a) Result of 15 Newton iterations by initial guess (b) Result of 15 fixed point iterations by initial guess

[2] It is likely also provable that $u^{(0)}$ larger than 1.125 will converge to $u^* = 1$, as $u^{(0)} = 1.125$ yields $u^{(1)} \in B(0.125, 1)$ and F' is strictly increasing on $[1.125, \infty)$.

Exercise 4. Use the provided code `ELLIPTIC1d.m` for FV in 1D for

$$-(ku_x)_x = f, x \in (0,1) \tag{2}$$

which allows Dirichlet or Neumann boundary conditions. (Get acquainted with the code, try some easy examples when $k = 1$ and $u(x)$ is known: do not turn this in).

(i) Let $u(x) = e^{10x^2}/e^{10}$. Test using Dirichlet and Neumann boundary conditions with an appropriate "manufactured" f. Test convergence in $\|\cdot\|_{\Delta,2}$ on a sequence of uniform grids $M = 5, 10, 50, 100$. Determine an optimal non-uniform grid. (Be brief. I do not need to see the code unless you have difficulties).

(ii) Describe how to extend the algorithm (give the discrete equations) from (2) to

$$-(ku_x)_x + cu = f, x \in (0,1) \tag{3}$$

Implement and test as in (i).

(iii) Calculate an analytical solution $u^{exact}(x)$ to the flow problem (2) when $u(0) = 1$, $u(1) = 0$, $f \equiv 0$ and when $k(x) = k_{left} = 1$, $x < a$ and $k(x) = k_{right} = 10$, $x > a$ with $a = 1/3$. Implement the numerical solution using well chosen grid with $M = 5$ and $M = 50$, and compare with u^{exact}. What do you see? What does the theory say?

(iv) **Extra:** calculate the flux out of the domain at $x = 1$ (you need to code this). This flux gives *an upscaled* value $\hat{k}$, a weighted harmonic average of k_{left}, k_{right}. Check!

Solution. **(i)** Since the problem is not well-posed for Neumann boundary conditions at both boundaries, we test convergence in $\|\cdot\|_{\Delta,2}$ with Dirichlet and Mixed BC (Neumann at $x = 0$, Dirichlet at $x = 1$):

M	Error for Dirichlet BC	Error for Mixed BC
5	0.09970	0.13924
10	0.08784	0.12887
50	0.00535	0.00778
100	0.00135	0.00197

For $M = 5$, a non-uniform grid of cell-boundaries of $\{0, 0.5, 0.82, 0.9, 0.95\}$ reduced the error for Dirichlet BC to 0.01532 and the error for Mixed BC to 0.03279. This was from trial-and-error testing, but I additionally developed a rough algorithm for creating a non-uniform grid for arbitrary M by making $h_i = x_{i+1} - x_i$ inversely proportional to $|u'(x_i)|$:[3]

(1) Start at $i = 0$, $x_i = 0$. Set R, a 'step-size reduction factor', to be 1.

(2) While $x_i \leq 1$, do:

(a) Evaluate $|u'(x_i)|$. If this is 0, set $h_i = 1/8$. Otherwise, set

$$h_i = \min\left\{\frac{1}{4R}, \frac{1}{16R|u'(x_i)|}\right\}, \quad x_{i+1} = x_i + h_i.$$

(b) Increment i.

(3) The result of (2) will be a non-uniform grid $\{x_i\}_{i=0}^{N-1} \subseteq [0,1]$. There are then 3 cases:

(a) If $N = M$ is the desired discretization, return $\{x_i\}_{i=0}^{N-1}$.

(b) If $N < M$, double R and go back to (1).

[3] Developing this algorithm was a personal challenge to develop some algorithmic solution to the problem of creating a non-uniform grid. It is inefficient and suboptimal.

(c) If $N > M$, remove $N - M$ points from the grid, yielding $\{x_i\}_{i=0}^{M-1}$, and uniformly distribute the "lost" portion of the domain across all of the subintervals:

$$\text{For } i \in \{1, 2, \ldots, M\}, \text{set } x_i' = x_i + \frac{X_{N-1} - X_{M-1}}{M-1}.$$

Then return $\{x_0\} \cup \{x_i'\}_{i=1}^{M}$.

(ii) Note that the flux $q = -ku_x$ is the same as in (2), but now $q_x + cu = f$. Thus, approximating

$$q_x(x_j^*) \approx \frac{1}{h_j}(Q_{j+1/2} - Q_{j-1/2}) \overset{\text{yields}}{\longrightarrow} h_j f_j = Q_{j+1/2} - Q_{j-1/2} + h_j c u.$$

This is the discrete equation we get from approximating the flux on the cell boundaries. In the context of modifying the original system $AU = F$ for (2), we now solve $A'U = F$ where $A' = A + \text{diag}(h_i c)$. Implementationally, this only involves a modification to the matrix `diffmat` as:

```
for i in range(diffmat.shape[0]):
        diffmat[i, i] = diffmat[i, i] + dx[i] * c
```

Testing as in (i)—using the known exact solution from (i) and modifying f accordingly for the reaction term—yields the following grid-norm error:

M	Error for Dirichlet BC	Error for Mixed BC
5	0.09510	0.11458
10	0.08420	0.10489
50	0.00515	0.00638
100	0.00130	0.00161

(iii) With the problem described above, we now seek to solve

$$-(k(x)u'(x))' = 0, \quad k(x) = \begin{cases} 1 & \text{if } x \le 1/3 \\ 10 & \text{otherwise.} \end{cases}$$

Integrating both sides yields

$$k(x)u'(x) = C_1 \in \mathbb{R} \quad \Longrightarrow \quad u'(x) = \frac{C_1}{k(x)}.$$

Thus, integrating both sides from 0 to x and applying FTC, we have

$$u(x) = \begin{cases} C_1 x + C_2 & \text{if } x \le 1/3 \\ \frac{C_1}{10} x + C_3 & \text{otherwise.} \end{cases}$$

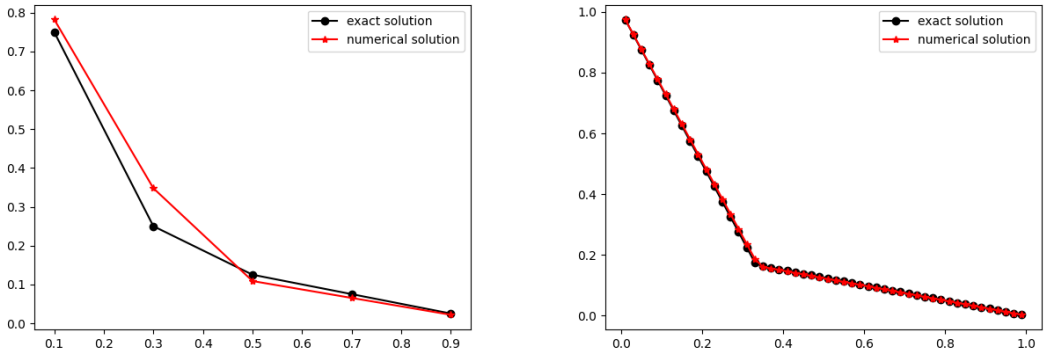
Using the boundary conditions, we have $C_2 = u(0) = 1$, $C_3 = u(1) - \frac{C_1}{10} = -\frac{C_1}{10}$. Assuming u is continuous, we have

$$\begin{aligned} &\lim_{x \to 1/3^-} u(x) = \lim_{x \to 1/3^+} u(x) \\ \Longrightarrow \quad & C_1\left(\frac{1}{3}\right) + 1 = \frac{C_1}{10}\left(\frac{1}{3}\right) - \frac{C_1}{10} \\ \Longrightarrow \quad & C_1 = -\frac{5}{2}. \end{aligned}$$

Thus our exact solution is

$$u(x) = \begin{cases} -\frac{5}{2}x + 1 & \text{if } x \leq 1/3 \\ -\frac{5}{20}x + \frac{5}{20} & \text{otherwise.} \end{cases}$$

Testing the numerical solver with $M = 5, 50$ yields a decent numerical solution which struggles with the corner at $x = 1/3$:



(a) $M = 5$ (b) $M = 50$

Figure 4: The numerical solution to the problem with non-constant coefficients.

□