Numerical Methods II, Spring 2023
Assignment IV: Pseudospectral integrator for the KdV equation

Aleksandar Donev
Courant Institute, NYU, donev@courant.nyu.edu

February 25th 2023
Due: **5pm Tuesday ??? 2023**

The optional parts of this assignment (marked blue) do not carry any points; only do optional things if you have time and interest for your own benefit.

1 [100 points] Temporal discretization of KdV

In this assignment you will finally numerically solve the Korteweg de Vries PDE

$$\partial_t \phi = -\partial_{xxx}\phi - 3\partial_x\left(\phi^2\right) \equiv \mathcal{K}\left[\phi(\cdot,t)\right],$$

where $\mathcal{K}\left[\phi(\cdot,t)\right]$ denotes the functional on the right hand side, which only involves derivatives in x. Recall from the second homework that on an unbounded line, the KdV equation has traveling wave solutions in the forms of "soliton" waves

$$\phi_{\text{sol}}(x,t) = \frac{c}{2}\text{sech}^2\left(\frac{\sqrt{c}}{2}(x-ct)\right), \tag{1}$$

where c is the speed of the soliton moving to the right. While the soliton solution (1) is not periodic, it decays exponentially for large x so we can pretend it is periodic. Here we will consider the KdV equation on a periodic domain $x \in [-L/2, L/2)$ with $L = 60$.

We consider pseudo-spectral methods for solving this PDE, in which we represent the solution as a (truncated) Fourier series (recall that the wavenumber k is not an integer here)

$$\phi(x,t) = \sum_k \hat{\phi}_k(t) e^{ikx}.$$

The coefficients $\hat{\boldsymbol{\phi}}(t)$ are solutions of the system of ODEs

$$\frac{d\hat{\boldsymbol{\phi}}}{dt} = \boldsymbol{F}\left(\hat{\boldsymbol{\phi}}\right) = ik^3 \boxdot \hat{\boldsymbol{\phi}} - 3i\boldsymbol{k} \boxdot \boldsymbol{\mathcal{F}}\left(\left(\boldsymbol{\mathcal{F}}^{-1}\hat{\boldsymbol{\phi}}\right)^{\boxed{2}}\right) =$$
$$= \boldsymbol{A}\hat{\boldsymbol{\phi}} + \boldsymbol{B}\left(\hat{\boldsymbol{\phi}}\right) \tag{2}$$

where $\boldsymbol{\mathcal{F}}$ denotes a Fourier transform. Here the first term $\boldsymbol{A}\hat{\boldsymbol{\phi}} \equiv ik^3 \boxdot \hat{\boldsymbol{\phi}}$ corresponding to $-\partial_{xxx}\phi$ is *stiff* but linear, and the second term $\boldsymbol{B}\left(\hat{\boldsymbol{\phi}}\right)$ corresponding to $-3\partial_x\left(\phi^2\right)$ is a complicated nonlinear function (potentially involving anti-aliasing features as well). Note that the mode with wavenumber $k = 0$ does not evolve with time (this is because the KdV equation is a conservation law) and it should be omitted from the code.

In this homework you will consider two different method to solve the system of ODEs

$$\frac{d\hat{\boldsymbol{\phi}}}{dt} = \boldsymbol{A}\hat{\boldsymbol{\phi}} + \boldsymbol{B}\left(\hat{\boldsymbol{\phi}}\right), \tag{3}$$

one an implicit-explicit multistep method, and the other an exponential Runge-Kutta integrator. Both schemes are second order accurate and handle the linear term implicitly, but treat the nonlinear term

explicitly in order to avoid solving nonlinear systems of equations. If it turns out that the nonlinear term $\boldsymbol{B}\left(\hat{\boldsymbol{\phi}}\right)$ does not cause stiffness, then we have a hope of having a good method to solve the KdV equation.

Note: In this homework, just as in the second homework, you will continue to work with functions and function norms, not with vectors and vector norms. That is, we won't consider the solution to be the vector $\boldsymbol{\phi}(t) = \boldsymbol{\mathcal{F}}^{-1}\hat{\boldsymbol{\phi}}(t)$ but rather the function $\phi(x,t)$. In order to approximate function norms, you need to evaluate the solution on a finer grid using your routine *interpft*, just like you did in Homeworks 1 and 2.

Note: It is preferable if you treat the unmatched mode for even number of grid points that are powers of two carefully as explained in class. If you cannot then simply *use odd grid sizes that are powers of three.* Another often-used though imperfect alternative is to zero out that mode when computing odd derivatives; this however introduces a artificial conserved variable (the unmatched mode) so it can be dangerous if not used with some care (think about it).

[**Optional**] You do *not* have to do anti-aliasing for this homework, but it *may* improve things if you do it. A good way to test things is to employ a test function that is a sum of a small number of Fourier modes instead of a complicated nonlinear function like the soliton. Use only a small number of points (the minimal required, so that the highest Fourier mode is the unmatched one!) so that in principle the discretization should be *exact*. For the linear part of the PDE, you can write down an exact solution without ODE solvers! For the nonlinear part, just evaluate the nonlinear term in the r.h.s. for the test function and see if you can get it *exactly* correct. Doing things like handling the special mode or antialiasing does not guarantee that you will get a more accurate answer for the soliton, but it is a way to develop a modular code where pieces are tested and correct and the solver is as accurate as possible for very smooth solutions such as a sum of a (small) finite number of Fourier components (i.e., for band-limited functions).

1.1 [15 pts] Absolute stability of two methods

The first ODE integration method you will use is the **SBDF2** scheme explained in class. For an ODE of the form (3) a time step of duration Δt takes the form

$$\hat{\boldsymbol{\phi}}^{n+1} = \frac{4}{3}\hat{\boldsymbol{\phi}}^{n} - \frac{1}{3}\hat{\boldsymbol{\phi}}^{n-1} + \frac{2\Delta t}{3}\boldsymbol{A}\hat{\boldsymbol{\phi}}^{n+1} + \frac{2\Delta t}{3}\left(2\boldsymbol{B}\left(\hat{\boldsymbol{\phi}}^{n}\right) - \boldsymbol{B}\left(\hat{\boldsymbol{\phi}}^{n-1}\right)\right),$$

see also Eq. (35) in the paper of Cox and Matthews (CM) where it is called the AB2BDF2 scheme.

The second scheme is the exponential time differencing RK2 method. In class we gave a midpoint variant but more efficient (think about it) is to use a trapezoidal explicit RK2 formula, leading to the **ETDRK2** scheme

$$\begin{aligned}
\hat{\boldsymbol{\phi}}^{n+1,\star} &= e^{\boldsymbol{A}\Delta t}\hat{\boldsymbol{\phi}}^{n} + \boldsymbol{A}^{-1}\left(e^{\boldsymbol{A}\Delta t} - \boldsymbol{I}\right)\boldsymbol{B}\left(\hat{\boldsymbol{\phi}}^{n}\right) \quad \text{(predictor)} \\
\hat{\boldsymbol{\phi}}^{n+1} &= \hat{\boldsymbol{\phi}}^{n+1,\star} + \boldsymbol{A}^{-2}\left(\frac{e^{\boldsymbol{A}\Delta t} - \boldsymbol{I} - \boldsymbol{A}\Delta t}{\Delta t}\right)\left(\boldsymbol{B}\left(\hat{\boldsymbol{\phi}}^{n+1,\star}\right) - \boldsymbol{B}\left(\hat{\boldsymbol{\phi}}^{n}\right)\right) \quad \text{(corrector)},
\end{aligned}$$

see also Eqs. (20+22) in the paper by CM (hopefully now you see why you were advised to drop the $k = 0$ mode). Note that direct implementation of these formulas can suffer from roundoff errors for small Δt, more precisely, for small $|\lambda_{\min}|\,\Delta t$ where $\lambda_{\min}$ is the eigenvalue of $\boldsymbol{A}$ with smallest magnitude; make sure this is not the case or find a way to avoid catastrophic cancellation (e.g., by using Taylor series). *Make sure to write the code in a way that avoids evaluating things more than once. For example, evaluate* $\boldsymbol{B}\left(\hat{\boldsymbol{\phi}}^{n}\right)$ *only once per time step and store and reuse between the predictor and the corrector. In fact, observe that since* $\boldsymbol{A}$ *is constant many things can be computed once and only once at the beginning (and that computation is super cheap if done right). We will go through some sample good/bad codes in class later on.*

[15 pts] Explain why the term $\boldsymbol{A}\hat{\boldsymbol{\phi}}$ is stiff when there are lots of Fourier modes. Then explain whether you think SBDF2 and ETDRK2 are good choices to solve (2) and why [there is no single right or wrong answer].

1.2 [85 pts] Accuracy, Stability and Robustness for a Single Soliton

Take as initial condition the soliton $\phi(x, t=0) = \phi_{\text{sol}}(x, 0)$, and then solve the KdV equation to time $T = L/c$ — the solution should be unchanged since it has traveled around the periodic domain once, i.e.,

$\phi(x, t = T) = \phi_{\text{sol}}(x, 0)$. Recall that even though the soliton wave is not a periodic solution it decays sufficiently rapidly that it will be a solution to very high accuracy even in a periodic domain.

Implement both the ETDRK2 and AB2BDF2 schemes to integrate the KdV equation in time, and do each of the next tasks for each of the schemes.

[**Optional**] On the course homepage I have linked a code by A. K. Kassam and L. N. Trefethen that solves KdV with fourth-order ETDRK4 method, along with the paper that explains how to avoid roundoff problems. Try this method/code and compare to see how much improvement you can get from the higher-order scheme.

1.2.1 [15 pts] Empirical stability

Try several grid sizes (number of Fourier modes) that are powers of two (or powers of three), say from 32 up to 256 (remember that In homework 2 you figured out that you need to keep ~ 200 Fourier modes in order to evaluate $\boldsymbol{F}\left(\hat{\boldsymbol{\phi}}\right)$ to nine accurate digits for the soliton wave with $c = 1$). For each resolution, determine empirically by playing around (i.e., simply try increasing the time step size until you get unstable behavior of the solver; watching a movie of the solution can be very helpful) whether there is a stability limit on the time step size Δt, and if so, what that limit is (approximately). How does the stability limit depend on the resolution (number of grid points, or, equivalently, the number of modes)?

1.2.2 [20 pts] Accuracy of ODE solver

For this part choose (wisely) a certain number of points/modes to discretize the PDE in space. Solve the ODE (2) using the two schemes up to time T. Define the error from the temporal (ODE) integration

$$e_{\Delta t} = \left\| \hat{\boldsymbol{\phi}}_{\Delta t}(T) - \hat{\boldsymbol{\phi}}_{\Delta t=0}(T) \right\|_2,$$

where $\hat{\boldsymbol{\phi}}_{\Delta t}(t)$ is the numerical approximation for time step size Δt and $\hat{\boldsymbol{\phi}}_{\Delta t=0}(t)$ is the true solution. Theory says that $e_{\Delta t} = O(\Delta t^2)$ for sufficiently small time steps for both ODE solvers. Confirm this numerically. Observe that while we don't know $\hat{\boldsymbol{\phi}}_{\Delta t=0}(T)$, you can use the difference

$$\tilde{e}_{\Delta t} = \left\| \hat{\boldsymbol{\phi}}_{\Delta t}(T) - \hat{\boldsymbol{\phi}}_{\Delta t/2}(T) \right\|_2$$

to estimate the error, as covered in Appendix A.6.3 in LeVeque. How small does Δt have to be for you to see "clear" second-order convergence? If Δt has to be too small for you to be able to perform the computation comfortably with the computing resources you have, feel free to evolve the solution over a shorter time, say one quarter of the period T.

Note: This way of testing order of convergence is sometimes called the *method of successive refinements*. Note that this does not test that the ODE solver converges to the correct solution, it simply tests that the numerical solution converges to something (potentially wrong)! In practice, one would first test the ODE solver on a simpler system of ODEs for which maybe an analytical solution can be constructed, or use something called the *method of manufactured solutions* (which we will use later in the class). Since you already practiced ODE solvers in Homework 3, here you can confirm the ODE solver is doing the right thing by confirming convergence to the soliton solution for the PDE.

1.2.3 [25 pts] Accuracy of PDE solver

For this next part use ~ 256 modes. This ensures that the error is dominated by the error from integrating the ODEs, that is, the temporal discretization error is much larger than the spatial discretization error. It also gives us hope (but certainly no guarantees!) that if we accurately solve the system of ODEs we will also have accurately solved the PDE.

Confirm whether the (functional) error in $\phi(x, T)$ is $O(\Delta t^2)$ by comparing to the exact solution of the PDE. Here you can (should?) try different function norms, but also remember that plotting the error as a function is much more informative than looking at three numbers (norms). Compare the errors from the two schemes (ETDRK2 and AB2BDF2) and conclude which one is more accurate for the same time step size (but note that in practice what matters is which one is more accurate for the same overall computational cost).

If Δt has to be too small for you to be able to perform the computation comfortably with the computing resources you have, feel free to evolve the solution over a shorter time, say one quarter of the period T.

1.2.4 [25 pts] Robustness of PDE solver

Now gradually reduce the number of modes/points, and for each resolution set the time step size to be *one half* (if using grid sizes that are powers of two) or *one third* (if using powers of three) of the empirical stability limit you found in part 1.2.1. Plot the numerical estimate for the solution $\phi(x,T)$ together with the correct solution. Combine multiple curves on one plot in some intelligent and readable way (use different line styles, colors, symbols, etc., and put legends and captions). Comment on what you observe, i.e., discuss how the ways in which the solver fails (do the numerical errors make the wave move faster/slower, do they make it spread or shrink, do they cause oscillations, etc.) when the problem is *under-resolved*; this now relates to the *robustness* of the method.

Note: One lesson we will try to learn in this class is that a more accurate method is not necessarily more robust (usually the opposite!). Is one of the two ODE integration schemes clearly more robust? This is a judgment call, there is no right or wrong answer, but I hope you think about it carefully.

1.3 [Optional] Collisions between two solitons

Soliton waves have the spectacular property that they can pass through each other without changing their shape — this makes the useful for optical communication, for example. Try to construct a still figure that illustrates this — this is actually challenging and an animation may be more informative. Make a movie where you start with two solitons that are well-separated and move toward each other so that eventually the two solitons collide head on, for example, one has speed c and the other has speed $-c$, or, make a movie where one faster soliton of speed $2c$ passes another of speed c. Use what you learned from the previous parts of the homework to select the grid size and time step and method of temporal integration smartly. Do not use more computational resources than necessary, in fact, try to make a "nice movie" with as few points and as large a time step size as possible. Set the length of the domain L and the initial condition so that the solitons do not overlap initially. Report what you did and the reasoning behind your choices.

[Even more optional] If you add another small nonlinearity to the r.h.s. of the equations, e.g., $\epsilon\partial_x(\phi^3)$, then solitons will interact with each other and scatter from one another. This would not only make cool movies but may strain the numerical method as well. Try it if time permits.

Note: You can find lots of movies/images online, for example, check out `http://lie.math.brocku.ca/~sanco/solitons/kdv_solitons.php`.