Solution of the 1D unsteady transport equation with the finite difference method

October 16^{th}, 2024

Ben Wilfong

Files

```
HW3_root
├── CMakeLists.txt
├── homework.pdf
├── main.inp
├── make.sh
├── src
│   ├── m_global_parameters.f90
│   ├── m_helpers.f90
│   ├── m_linear_algebra.f90
│   └── p_main.f90
└── plots
    ├── *.csv
    └── plots.m
```

Compiling

This code can be compiled in a terminal by running `chmod u+x ./make.sh` followed by `./make.sh` in the `HW3_root/` directory. This will compile and link the source code in the `src/` directory and create an executable called `main` in the `HW3_root/` directory.

Running

The problem parameters are defined in the namelist file `main.inp` in `HW3_root/`. The input parameters are:

- gamma: Difusion coefficient
- U: advection speed
- bench: logical enabling multiple runs of the code for benchmarking runtime
- time_stepper: which time-stepper to use. 0 = Explicit Euler, 1 = Implicit Euler, 2 = Crank-Nicolson
- N: number of points
- dt: time-step
- t_start: start time
- t_stop: end time
- t_save: time interval between saves

Once the input parameters are set, the code is ran with `./main`. After execution, the solution is written to `HW3_root/output.csv` with the first row holding the x coordinates and each following row storing ϕ at a different instance in time.

Part 1: Numerical Schemes

The one-dimensional unsteady transport equation can be written with time-derivatives on the LHS and spatial derivatives on the RHS as

$$\frac{\partial \phi}{\partial t} = \frac{\partial}{\partial x}\left(\Gamma \frac{\partial \phi}{\partial x}\right) - U\frac{\partial \phi}{\partial x}.$$

For the case of constant U and Γ, the above simplifies to

$$\frac{\partial \phi}{\partial t} = f(\phi) \text{ where } f(\phi) = \Gamma \frac{\partial^2 \phi}{\partial x^2} - U\frac{\partial \phi}{\partial x}.$$

$f(\phi^{(n)})$ can be discretized using central differences as

$$\begin{aligned} f(\phi_i^{(n)}) &= \Gamma \frac{\phi_{i+1}^{(n)} - 2\phi_i^{(n)} + \phi_{i-1}^{(n)}}{(\Delta x)^2} - U\frac{\phi_{i+1}^{(n)} - \phi_{i-1}^{(n)}}{2\Delta x}, \\ &= \left(\frac{\Gamma}{(\Delta x)^2} - \frac{U}{2\Delta x}\right)\phi_{i+1}^{(n)} + \left(-\frac{2\Gamma}{(\Delta x)^2}\right)\phi_i^{(n)} + \left(\frac{\Gamma}{(\Delta x)^2} + \frac{U}{2\Delta x}\right)\phi_{i-1}^{(n)}. \end{aligned}$$

This discretization can be used to formulate temporal integration schemes using forward Euler, backward Euler, and Crank–Nicolson methods with the diffusion and Courant numbers

$$d = \frac{\Gamma \Delta t}{(\Delta x)^2} \quad \text{and} \quad c = \frac{U\Delta t}{\Delta x}$$

as follows:

- **Forward Euler** defines the solution at $t^{(n+1)}$ as $\phi^{(n+1)} = \phi^n + \Delta t f(\phi^{(n)}, t^n)$. Using the above discretization of $f(\phi^{(n)})$ gives the explicit update equation

  $$\phi_i^{(n+1)} = \left(d + \frac{c}{2}\right)\phi_{i-1}^{(n)} + (1-2d)\,\phi_i^{(n)} + \left(d - \frac{c}{2}\right)\phi_{i+1}^{(n)}.$$

  This explicit update, after taking into account boundary conditions, can be written as the tridiagonal matrix multiplication

  $$\underbrace{\begin{pmatrix} \phi_0^{(n+1)} \\ \phi_1^{(n+1)} \\ \vdots \\ \phi_{N-1}^{(n+1)} \\ \phi_N^{(n+1)} \end{pmatrix}}_{\Phi^{(n+1)}} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & & \\ d+c/2 & 1-2d & d-c/2 & & \\ & \ddots & \ddots & \ddots & \\ & & d+c/2 & 1-2d & d-c/2 \\ & & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \phi_0^{(n)} \\ \phi_1^{(n)} \\ \vdots \\ \phi_{N-1}^{(n)} \\ \phi_N^{(n)} \end{pmatrix}}_{\text{RHS}}.$$

- **Backward Euler** defines the solution at $t^{(n+1)}$ as $\phi^{(n+1)} = \phi^{(n)} + \Delta t f(\phi^{(n+1)}, t^{n+1})$. Using the above discretization of $f(\phi^{(n)})$ gives the implicit update equation

  $$\left(-\frac{c}{2} - d\right)\phi_{i-1}^{(n+1)} + (1+2d)\,\phi_i^{(n+1)} + \left(\frac{c}{2} - d\right)\phi_{i+1}^{(n+1)} = \phi_i^{(n)}.$$

This implicit update equation, after taking into account boundary conditions, can be written as the tridiagonal system of linear equations

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & & \\ -c/2-d & 1+2d & c/2-d & & \\ & \ddots & \ddots & \ddots & \\ & & -c/2-d & 1+2d & c/2-d \\ & & 0 & 0 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} \phi_0^{(n+1)} \\ \phi_1^{(n+1)} \\ \vdots \\ \phi_{N-1}^{(n+1)} \\ \phi_N^{(n+1)} \end{pmatrix}}_{\Phi^{(n+1)}} = \underbrace{\begin{pmatrix} 0 \\ \phi_1^{(n+1)} \\ \vdots \\ \phi_{N-1}^{(n+1)} \\ 0 \end{pmatrix}}_{\text{RHS}}$$

- **Crank–Nicolson** defines the solution at $t^{(n+1)}$ as

$$\phi^{(n+1)} = \phi^{(n)} + (\Delta t)/2 \left(f(\phi^{(n+1)}, t^{(n+1)}) + f(\phi^{(n)}, t^{(n)}) \right).$$

Using the discretization of $f(\phi^{(n)})$ gives the implicit–explicit update equation

$$\left(-\frac{c}{4} - \frac{d}{2}\right)\phi_{i-1}^{(n+1)} + (1+d)\,\phi_i^{(n+1)} + \left(\frac{c}{4} - \frac{d}{2}\right)\phi_{i+1}^{(n+1)} = \ldots$$
$$\left(\frac{d}{2} + \frac{c}{4}\right)\phi_{i-1}^{(n)} + (1-d)\,\phi_i^{(n)} + \left(\frac{d}{2} - \frac{c}{4}\right)\phi_{i+1}^{(n)}.$$

This implicit–explicit update equation, after taking into account boundary conditions, can be written as the tridiagonal system of linear equations

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & & \\ -c/4-d/2 & 1+d & c/4-d/2 & & \\ & \ddots & \ddots & \ddots & \\ & & -c/4-d/2 & 1+d & c/4-d/2 \\ & & 0 & 0 & 1 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} \phi_0^{(n+1)} \\ \phi_1^{(n+1)} \\ \vdots \\ \phi_{N-1}^{(n+1)} \\ \phi_N^{(n+1)} \end{pmatrix}}_{\Phi^{(n+1)}} =$$
$$\underbrace{\begin{pmatrix} 0 & 0 & 0 & & \\ d/2+c/4 & 1-d & d/2-c/4 & & \\ & \ddots & \ddots & \ddots & \\ & & d/2+c/4 & 1-d & d/2-c/4 \\ & & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \phi_0^{(n)} \\ \phi_1^{(n)} \\ \vdots \\ \phi_{N-1}^{(n)} \\ \phi_N^{(n)} \end{pmatrix}}_{\text{RHS}}.$$

Part 2: Problem 2

Figure 1 shows the solution using all three methods at $T = 20, 30$ and 40 in the first three rows. The first column shows a zoomed in view of the peak of the solution to highlight the minor differences between the three methods for case 1, which is shown in the second column. The solution for case 2 is given in the third column. The solutions to case 2 have spurious oscillations for all three temporal integrators. This is becuse there is no diffusion to damp these oscillations. The figure in the fourth row shows the soltuion at $T = 10, 20, 30$, and 40 for case 1 using Crank–Nicolson for time integration. The diffusion causes the signal's height to decrease while its' width increases. $N = 500$ was used for all solutions in this figure.

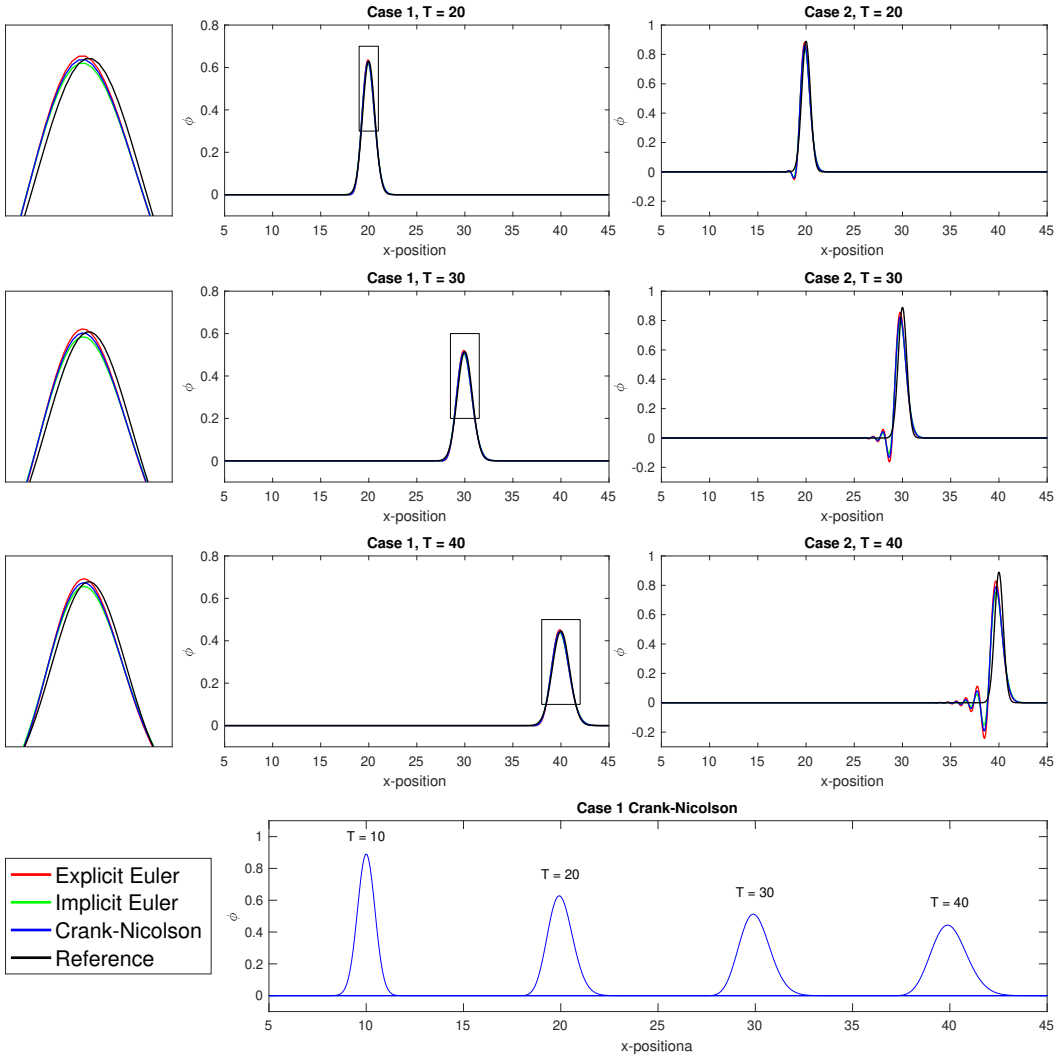


FIGURE 1. Comparision of the three numerical schemes

PART 3: PROBLEM 3

Figure 2 shows the value of ϕ as a function time at $x = 15$ and $x = 25$ for the three numerical methods. Explcit Euler overshoots the reference solution while implicit Euler undershoots the reference solution. Crank–Nicolson shows the best agreement of the three because it is essentially an average of implicit and explicit Euler.

PART 4: SPATIAL CONVERGENCE

Figure 3a shows the spatial convergence of Crank-Nicolson. The emperical spatial convergence rate is second order, which agrees with the second order central difference scheme used to discretize the spatial derivatives.

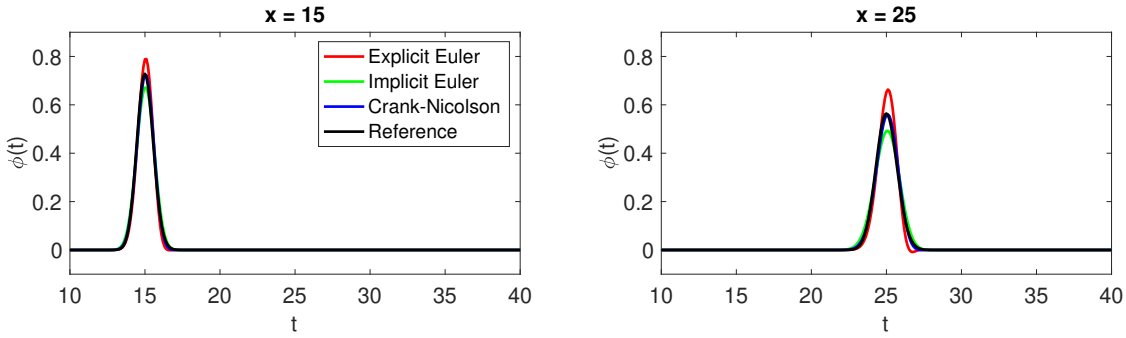
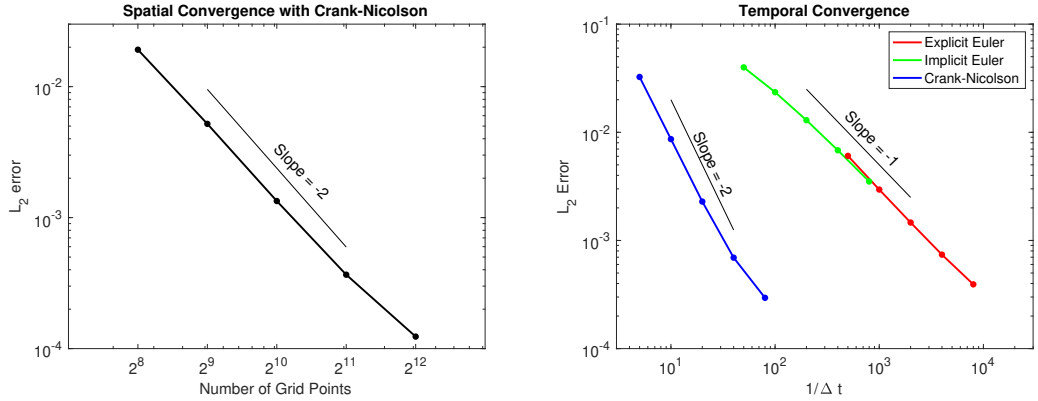


FIGURE 2. $\phi(t)$ at $x = 15$ and $x = 25$



(A) Spatial Convergence (B) Temporal Convergence

FIGURE 3. Convergence

PART 5: TEMPORAL CONVERGENCE

Figure 3b shows the temporal convergence of of all three time integegrators for $N = 4096$. Implicit and Explicit Euler both show first order accuracy in time while Crank–Nicolson exhibits second order accuracy in time. Larger time-steps were used for Crank-Nicolson to ensure that the temporal accuracy was being measured rather than the spatial accuracy. As the time-step decreases, the temporal convergence flatlines as spatial discretization error becomes dominant.

PART 6: STABILITY

Explicit Euler is conditionally stable when used to solve the 1D unsteady transport equation while Implicit Euler and Crank-Nicolson are unconditionally stable due to their implicit nature. The stability criterion for explicit Euler can be calculated using Von Neumann stability analysis by substituting the Fourier modes $\phi_j^{(n)} = e^{ij\Delta x\xi}$ and $\phi_j^{(n+1)} = g(\xi)e^{ij\Delta x\xi}$ into the update equation, which gives

$$g(\xi)e^{ij\Delta x\xi} = \left(d + \frac{c}{2}\right) e^{i(j-1)\Delta x\xi} + (1 - 2d)\, e^{ij\Delta x\xi} + \left(d - \frac{c}{2}\right) e^{i(j+1)\Delta x\xi}.$$

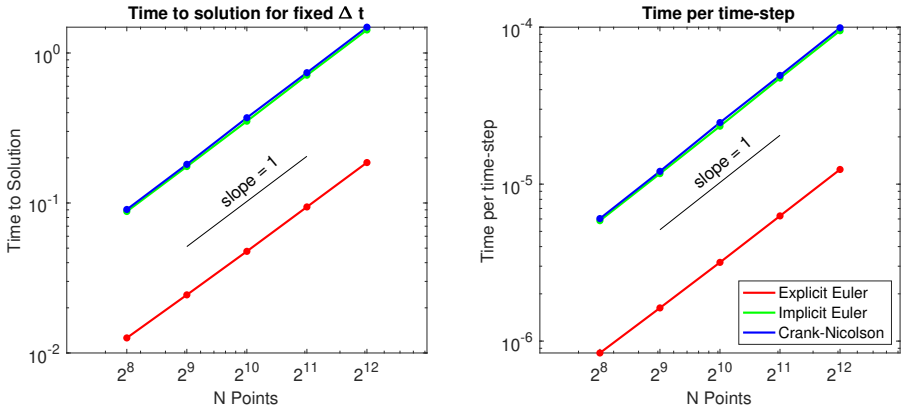


FIGURE 4. Time to solution and time per time-step for fixed Δt

TABLE 1. Time per time-step for each numerical method in μs

N	Explicit Euler	Implicit Euler	Crank–Nicolson
256	0.84	5.85	6.03
512	1.62	11.7	12.1
1024	3.17	23.4	24.7
2048	6.27	47.4	49.3
4096	12.4	94.9	99.1

Dividing each side by $e^{ij\Delta x\xi}$ and simplifying gives

$$g(\xi) = \left(d + \frac{c}{2}\right) e^{-i\Delta x\xi} + (1 - 2d) + \left(d - \frac{c}{2}\right) e^{i\Delta x\xi},$$
$$g(\xi) = \left(d + \frac{c}{2}\right) (\cos(\xi\Delta x) - i\sin(\xi\Delta x)) + (1 - 2d) + \left(d - \frac{c}{2}\right) (\cos(\xi\Delta x) + i\sin(\xi\Delta x)),$$
$$g(\xi) = 1 + 2d\,(\cos(\xi\Delta x) - 1) + c\sin(\xi\Delta x).$$

If there is no diffusion, then $d = 0$ and $|g(\xi)| \geq 1$ for all Δt mean the method is unconditinally unstable without difussion. If there is no convection, then the method is stable when $d < 1/2$, which gives the constraint $\Delta t < (\Delta x)^2/(2\Gamma)$. If $d \neq 0$ and the problem is dominated by convection, then $\Delta t < \Delta x/U$ must be satisfied.

PART 7: RUNTIME

Figure 4 shows the time to solution and time per time-step for each numerical method for a fixed Δt. Table 1 shows the exact runtime per time-step for each numerical method. All three time integrators show $\mathcal{O}(n)$ runtime as expected. Crank–Nicolson and Implicit Euler both take significantly longer due to the implicit solve required at each time-step. This implicit solve provides more value when using Crank–Nicolson because it greatly reduces temporal discretization error as shown in Figure 3b. This reduction in temporal discretization error ultimately means that fewer timesteps are required than for explicit Euler, meaning that the time to solution for the same target accuracy is actually less for Crank–Nicolson than for explicit Euler as a result of its higher order temporal accuracy.