NSSC 2 - Assignement 3

Bianchi Riccardo, Kapla Daniel, Kuen Jakob, Müller David

May 17, 2022

1 Numerical solution of the diffusion equation in a finite domain

We are given the 1D unsteady diffusion equation

$$\frac{\partial C}{\partial t} - D\frac{\partial^2 C}{\partial x^2} = 0 \tag{1}$$

with a scalar diffusion coefficient $D = 10^{-6}$. The domain size is h which is discretized with N_x points to give the grid space Δx with a time step Δt. The initial condition is $C = 0$ inside the domain.

Furthermore, denote with $N_t \in \mathbb{N}$ are the number of time steps of the simulation where $t_0 = 0$ is the initial time. All N_t discretized time points $t_n = n\Delta t$ leading to the final time point of the simulation as $t_{N_t} = N_t\Delta t$. The discretized x points of the domain are $x_i = (i-1)\Delta x = (i-1)h/(N_x - 1)$ for $i = 1, ..., N_x$. This gives exactly N_x grid points with equal distance between adjacent points on the domain $[0, h]$ with the first point $x_1 = 0$ and the last point $x_{N_x} = h$.

In the following we will also use the short hand notation

$$C_i^n = C(x_i, t_n), \qquad \partial_x^k C_i^n = \left.\frac{\partial^k C(x,t)}{\partial x^k}\right|_{x=x_i, t=t_n}, \qquad \partial_t^k C_i^n = \left.\frac{\partial^k C(x,t)}{\partial t^k}\right|_{x=x_i, t=t_n}$$

for $k \in \mathbb{N}$ as well as i and n are the space and time discretization indices, respectively. To distringuish between the exact solution and the approximation we add a "hat" $\widehat{\ }$ to the approximation. For example C_i^n is the true value and $\widehat{C}_i^n$ is the discretized solution with discretization errors at the evaluation point (x_i, t_n).

Furthermore, define

$$d = \frac{D\Delta t}{\Delta x^2}.$$

1.1 Explicit scheme with Dirichlet/Neumann BC

Given the Dirichlet boundary conditions $C(0, t) = 0$ and the Neumann boundary conditions $\partial_x C(h, t) = 0$. First we derive a 2^{nd} order *explicit* finite difference approach with a 1^{st} order discretization in time (see Figure 1).

To derive a second order scheme (assuming C is three time continuously differentiable as a function from $[0, h] \times \mathbb{R}^+ \to \mathbb{R}$) in space we first considure the Taylor expantion of C with respect to x given by

$$C(x + \Delta x, t) = C(x, t) + \partial_x C(x, t)\Delta x + \partial_x^2 C(x, t)\frac{\Delta x^2}{2} + \partial_x^3 C(x, t)\frac{\Delta x^3}{6} + \mathcal{O}(\Delta x^4).$$

Replace Δx with $\pm\Delta x$ and add the two equation together which gives

$$C(x + \Delta x, t) + C(x - \Delta x, t) = 2C(x, t) + \partial_x^2 C(x, t)\Delta x^2 + \mathcal{O}(\Delta x^4).$$

The first and third order terms drop out due to different signs. Finally, this results in

$$\partial_x^2 C(x, t) = \frac{C(x + \Delta x, t) - 2C(x, t) + C(x - \Delta x, t)}{\Delta x^2} + \mathcal{O}(\Delta x^2)$$

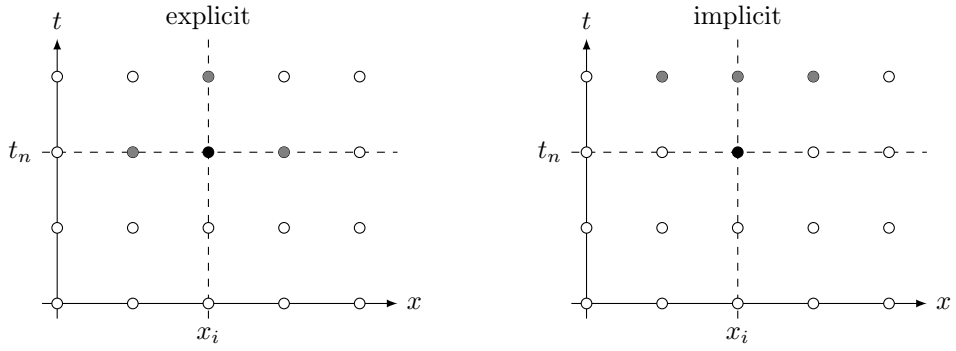

Figure 1: Dependency relation for the *explicit* (left) and *implicit* (right) finite difference approximation at a grid point indexed (i, n).

which is an approximation of the second derivative with error proportional to Δx^2. For the first order approximation in time the same trick can be used except that only the first two terms of the Taylor expantion need to be considured.

Therefore, the 2^{nd} order scheme for space and 1^{st} order in time at (i, n) using the explicit scheme derive as

$$\partial_x^2 \widehat{C}_i^n = \frac{\widehat{C}_{i-1}^n - 2\widehat{C}_i^n + \widehat{C}_{i+1}^n}{\Delta x^2}, \qquad \partial_t \widehat{C}_i^n = \frac{\widehat{C}_i^{n+1} - \widehat{C}_i^n}{\Delta t}.$$

Substitution into (1) yields after rearranging the update rule for internal points as

$$\widehat{C}_i^{n+1} = \widehat{C}_i^n + \frac{D\Delta t}{\Delta x^2}(\widehat{C}_{i-1}^n - 2\widehat{C}_i^n + \widehat{C}_{i+1}^n). \tag{2}$$

This holds at x_i where $i = 2, ..., N_x - 1$, or in other words, everywhere inside the space domain except the boundary. The boundary needs to be handled seperately. The left boundary condition is a Dirichlet constraint which is simply a constant giving $C_1^n = 1$ for all time while the Neumann condition on the right requires an additional discretization step for computing the next value. To achieve a 2^{nd} order discretization scheme it is usualy required to take the sum/difference of an Taylor expantion to the left and right direction at a given point.However, on the boundary there is no point at the right, which can be simply solved by adding a ghost point x_{N_x+1} to the system (which will not show up in the final scheme). This leads to the two 3^{rd} order expantions evaluated at the right boundary as

$$C_{N_x-1}^n = C_{N_x}^n - \partial_x C_{N_x}^n \Delta x + \partial_x^2 C_{N_x}^n + \mathcal{O}(\Delta x^3),$$
$$C_{N_x+1}^n = C_{N_x}^n + \partial_x C_{N_x}^n \Delta x + \partial_x^2 C_{N_x}^n + \mathcal{O}(\Delta x^3).$$

Taking the difference of both equations gives

$$0 = \partial_x C_{N_x}^n \Delta x = \frac{C_{N_x-1}^n - C_{N_x+1}^n}{2\Delta x} + \mathcal{O}(\Delta x^2).$$

which yields a 2^{nd} order equation for the Neumann BC on the right as

$$C_{N_x-1}^n = C_{N_x+1}^n.$$

This results in an system of equations containing the entire system (including the boundaryies) which reads

$$\begin{aligned}
&\widehat{C}_1^{n+1} = 1, \\
&\widehat{C}_i^{n+1} = d\widehat{C}_{i-1}^n + (1-2d)\widehat{C}_i^n + d\widehat{C}_{i+1}^n, && i = 2, ..., N_x - 1 \\
&\widehat{C}_{N_x}^{n+1} = 2d\widehat{C}_{N_x-1}^n + (1-2d)\widehat{C}_{N_x}^n.
\end{aligned}$$

or in matrix form

$$C^{n+1} = A_1 C^n$$

for a tridiagonal matrix A_1 of dimensions $N_x \times N_x$

$$A_1 = \begin{pmatrix} 1 & 0 & & & & & \\ d & 1-2d & d & & & & \\ & d & 1-2d & d & & & \\ & & d & 1-2d & \ddots & & \\ & & & \ddots & \ddots & d & \\ & & & & d & 1-2d & d \\ & & & & & 2d & 1-2d \end{pmatrix}.$$

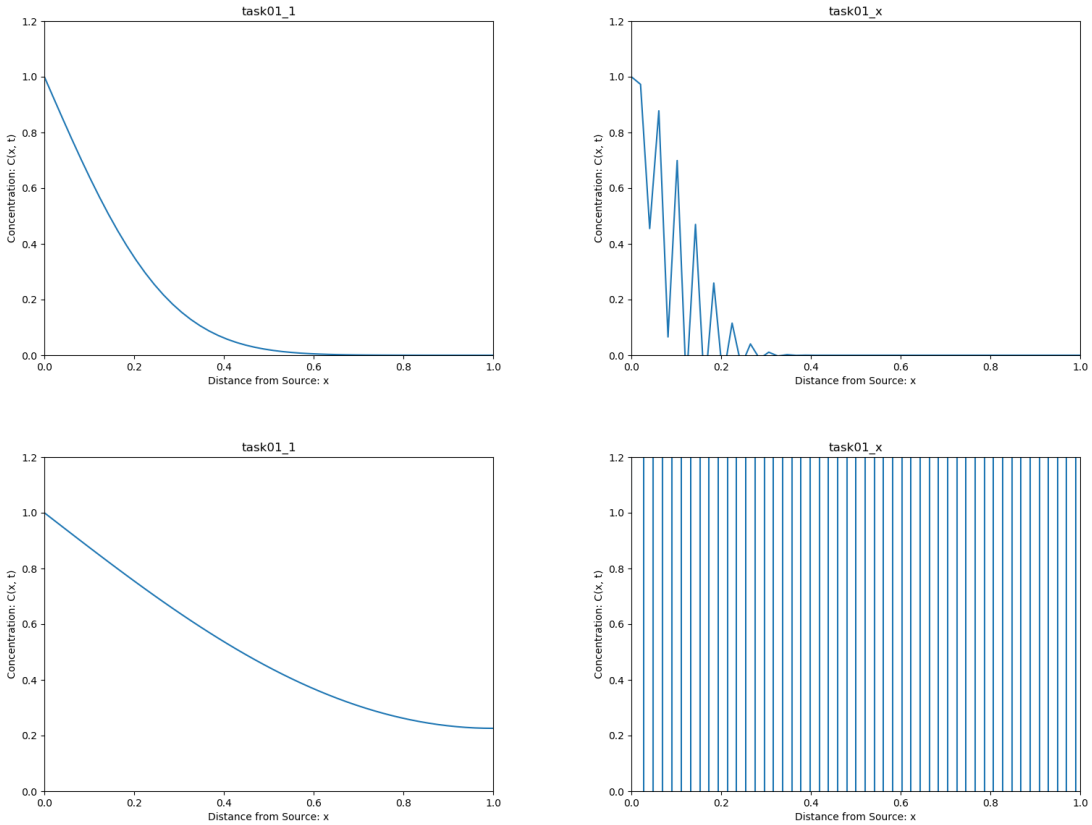

Figure 2: Comparison for stable (left, $d = 0.48 \leq 0.5$) vs. unstable (right, $d = 0.55 > 0.5$). At the top the simulation snapshot after 120 time steps and at the bottom after 1000 timesteps.

1.2 Explicit scheme with Dirichlet BC at both boundaries

In this task two Dirichlet boundary conditions are given. The new boundary condition is a Dirichlet BC with $C^n_{N_x} = 0$ which alters the equation system from Section 1.1 in the last two equations to

$$\begin{aligned} \widehat{C}_1^{n+1} &= 1, \\ \widehat{C}_i^{n+1} &= d\widehat{C}_{i-1}^n + (1-2d)\widehat{C}_i^n + d\widehat{C}_{i+1}^n, && i = 2, ..., N_x - 2 \\ \widehat{C}_{N_x-1}^{n+1} &= d\widehat{C}_{N_x-2}^n + (1-2d)\widehat{C}_{N_x-1}^n, \\ \widehat{C}_{N_x}^{n+1} &= 0. \end{aligned}$$

which yields to a new tridiagonal matrix A_2 of size $N_x \times N_x$ for the update rule $C^{n+1} = A_2 C^n$ as

$$A_2 = \begin{pmatrix} 1 & 0 & & & & & \\ d & 1-2d & d & & & & \\ & d & 1-2d & d & & & \\ & & d & 1-2d & \ddots & & \\ & & & \ddots & \ddots & d & \\ & & & & d & 1-2d & 0 \\ & & & & & 0 & 0 \end{pmatrix}.$$

See Figure 3 for $\lim_{t\to\infty} C(x,t)$ simulation (many timesteps) for subtask 1 and 2. In comparison both lead to a straight line solution (last energy) such that the Dirichlet or Neumann BC are fullfilled.

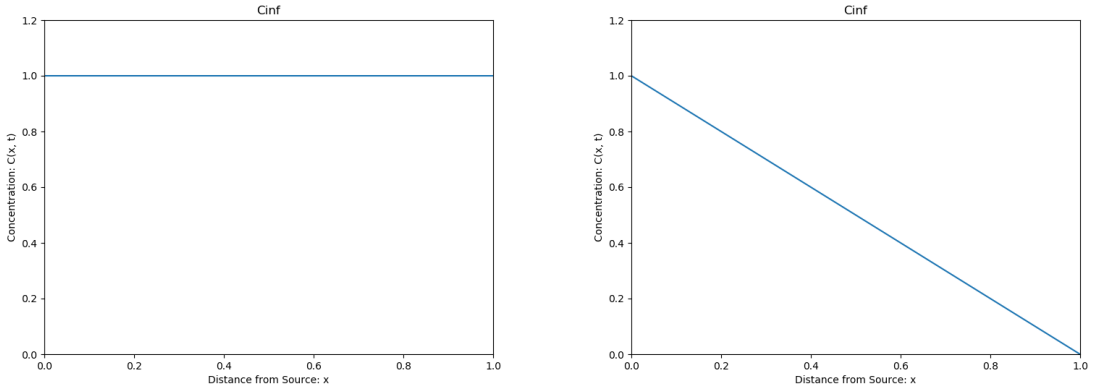

Figure 3: Solution from Section 1.1 (left) and Section 1.2 (right) for $t \to \infty$ simulated by setting number of time steps to $N_t = 20000$.

1.3 Implicit scheme with Dirichlet/Neumann BC

The following computations are similar in nature to the explicit scheme. The implicit discretization for the derivatives are (see: Figure 1)

$$\partial_x^2 \widehat{C}_i^{n+1} = \frac{\widehat{C}_{i-1}^{n+1} - 2\widehat{C}_i^{n+1} + \widehat{C}_{i+1}^{n+1}}{\Delta x^2}, \qquad \partial_t \widehat{C}_i^n = \frac{\widehat{C}_i^{n+1} - \widehat{C}_i^n}{\Delta t}.$$

Substitution into (1) gives the implicit (inverse) update rules

$$\begin{aligned} \widehat{C}_1^n &= \widehat{C}_1^{n+1} = 1, \\ \widehat{C}_i^n &= -d\widehat{C}_{i-1}^{n+1} + (1+2d)\widehat{C}_i^{n+1} - d\widehat{C}_{i+1}^{n+1} && i = 2, ..., N_x - 1 \\ \widehat{C}_{N_x}^n &= -2d\widehat{C}_{N_x-1}^{n+1} + (1+2d)\widehat{C}_{N_x}^{n+1} \end{aligned}$$

which leads again to an matrix equation

$$C^n = A_3 C^{n+1}$$

where the $N_x \times N_x$ tridiagonal matrix A_3 has the form

$$A_3 = \begin{pmatrix} 1 & 0 & & & & & \\ -d & 1+2d & -d & & & & \\ & -d & 1+2d & -d & & & \\ & & -d & 1+2d & \ddots & & \\ & & & \ddots & \ddots & -d & \\ & & & & -d & 1+2d & -d \\ & & & & & -2d & 1+2d \end{pmatrix}$$

and the update is performed by solving for C^{n+1}.

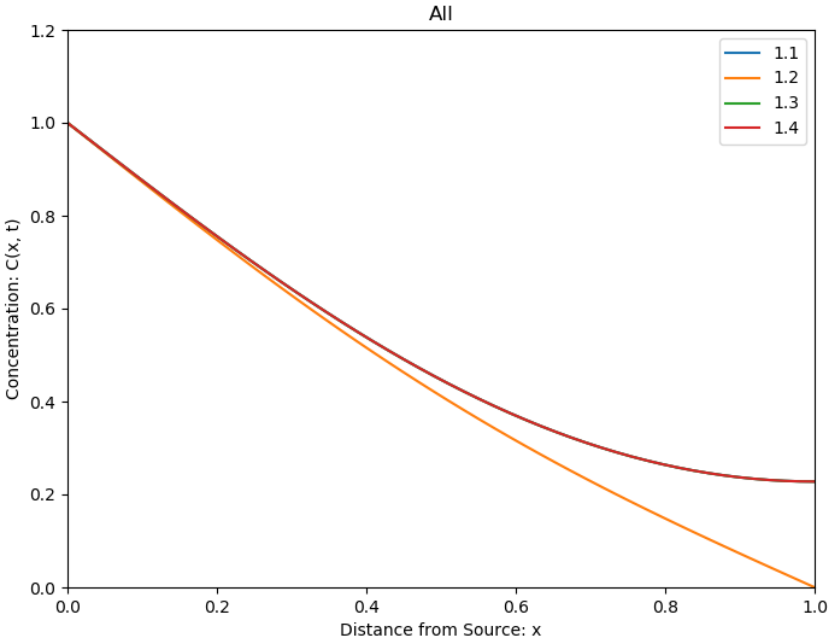

Figure 4: Comparison of all four subtasks. As expected 1.1, 1.3 and 1.4 solve the same probelm and as such have (basically) identical solution while the subtask 1.2 solves for different BC (Dirichlet) which is fullfilled.

1.4 Second order in time for implicit scheme with Dirichlet/Neumann BC

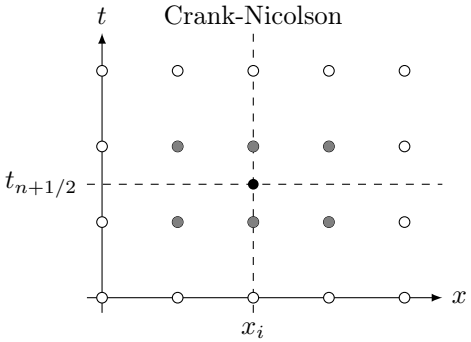

Figure 5: Crank-Nicolson dependency relations.

To derive the second order accurate scheme in time one uses the Crank-Nicolson approach to derive the descritization at the time midpoints $t_{n+1/2} = (n + 1/2)\Delta t$ which are indexed with $n + 1/2$.

By expantion into $\pm\Delta t/2$ at the midpoints one get

$$C_i^n = C_i^{n+1/2} - \partial_t C_i^{n+1/2} \frac{\Delta t}{2} + \partial_t^2 C_i^{n+1/2} \frac{\Delta t^2}{8} + \mathcal{O}(\Delta t^3),$$
$$C_i^{n+1} = C_i^{n+1/2} + \partial_t C_i^{n+1/2} \frac{\Delta t}{2} + \partial_t^2 C_i^{n+1/2} \frac{\Delta t^2}{8} + \mathcal{O}(\Delta t^3).$$

Taking the difference yields a 2^{nd} order approx at the time midpoints

$$\partial_t C_i^{n+1/2} = \frac{C_i^{n+1} - C_i^n}{\Delta t} + \mathcal{O}(\Delta t^2).$$

while the 2^{nd} order space discretization is analog to above but evaluated also at $(x_i, t_{n+1/2})$ to get

$$\partial_x^2 C_i^{n+1/2} = \frac{C_{i-1}^{n+1} - 2C_i^{n+1} + C_{i+1}^{n+1}}{\Delta x^2} + \mathcal{O}(\Delta x^2)$$

leading to an second order in time and space discretization scheme

$$\partial_t \widehat{C}_i^{n+1/2} - D\partial_x^2 \widehat{C}_i^{n+1/2} = \mathcal{O}(\Delta x^2 + \Delta t^2).$$

Next one need to ensure the evaluation to be done at the known grid points instead of the unknown time midpoints. Therefore, replacing the evaluation at the time midpoints by the time mean of the two direct neighbouring grid points

$$\widehat{C}_i^{n+1/2} = \frac{\widehat{C}_i^n + \widehat{C}_i^{n+1}}{2}.$$

After a bit of algebra and consideration of the Dirichlet and Neumann BC one end up with

$$\begin{aligned}
\widehat{C}_1^{n+1} &= \widehat{C}_1^n = 1, \\
\widehat{C}_i^{n+1} - \frac{d}{2}\widehat{C}_{i-1}^{n+1} + dC_i^{n+1} - \frac{d}{2}\widehat{C}_{i+1}^{n+1} &= C_i^n + \frac{d}{2}\widehat{C}_{i-1}^n - dC_i^n + \frac{d}{2}\widehat{C}_{i+1}^n, \qquad i = 2, ..., N_x - 1, \\
\widehat{C}_i^{n+1} - d\widehat{C}_{N_x-1}^{n+1} + dC_{N_x}^{n+1} &= C_i^n + d\widehat{C}_{N_x-1}^n - dC_{N_x}^n.
\end{aligned}$$

To write this again in a compact matrix equation system one defines the $N_x \times N_x$ matrix A_4 as

$$A_4 = \begin{pmatrix}
0 & 0 & & & & \\
-d/2 & d & -d/2 & & & \\
 & -d/2 & d & \ddots & & \\
 & & \ddots & \ddots & -d/2 & \\
 & & & -d/2 & d & -d/2 \\
 & & & & -d & d
\end{pmatrix}$$

then the entire system has the form

$$(I + A_4)C^{n+1} = (I - A_4)C^n$$

where I is the identity. Again, the update is performed by solving this tridiagonal system for C^{n+1}.

For comparison against the other methods see Figure 4.

1.5 Notes on Implementation

All four tasks follow the same basic sructure which can be written as a generalized linear system of the form

$$LC^{n+1} = RC^n$$

for updating the solution by one time step. The two matrices are dependent on the used scheme and the boundary conditions, see Table 1.

As a boild down solver the following is sufficient to compute the solution given an initial solution C_0 (argument `C`) with the appropriately defined matrices L, R (arguments `L, R`) which incorporate the scheme, space and time step size $\Delta x, \Delta t$ as well as the dispersion parameter D and the boundary conditions) at time $T = N_t\Delta$ (implicitly defined through the parameter `nx`).

```
def solve(C, L, R, nt):
    for t in range(nt):
        rhs = C if R is None else R @ C
        C = rhs if L is None else solve_banded((1, 1), L.data, rhs)
    return C
```

Task	L	R
1.1	I	A_1
1.2	I	A_2
1.3	A_3	I
1.4	$I + A_4$	$I - A_4$

Table 1: Left and right hand side matrices of the general update rule $LC^{n+1} = RC^n$ with A_i depending on the subtask as defined in Section 1.1 till 1.4 and I the identity matrix.

This code assumes `C` to be a 1D `numpy` array as well as L, R to by `scipy.sparse.spmatrix` sparse tridiagonal matrices or `None` which is equivalent to the identity matrix.

The additional code in the script `task01.py` contains setting up the needed matrices and plotting.

2 Numerical solution of the linear advection equation in a periodic domain

First a few words about the upwind scheme. With a positive U the upwind means that it is discretized by "going left", meaning into the opposite direction of the current position. In a similar faschion as above this leads to the discretization

$$\partial_x C_i^n = \frac{C_i^n - C_{i-1}^n}{\Delta x} + \mathcal{O}(\Delta x),$$
$$\partial_t C_i^{n+1} = \frac{C_i^{n+1} - C_i^n}{\Delta t} + \mathcal{O}(\Delta t).$$

One ends up with a first order discretized equation

$$0 = \frac{\widehat{C}_i^{n+1} - \widehat{C}_i^n}{\Delta t} + U\frac{\widehat{C}_i^n - \widehat{C}_{i-1}^n}{\Delta x}$$

leading to the update rule

$$C_i^{n+1} = (1 - C_0)C_i^n + C_o C_{i-1}^n, \qquad i = 2, ..., N_x$$

with the current $C_0 = \frac{U\Delta x}{\Delta t}$. The boundary condition enforces periodizity which means $C_1^n = C_{N_x}^n$ and therefore the update for the left most point is

$$C_1^{n+1} = (1 - C_0)C_1^n + C_o C_{N_x}^n.$$

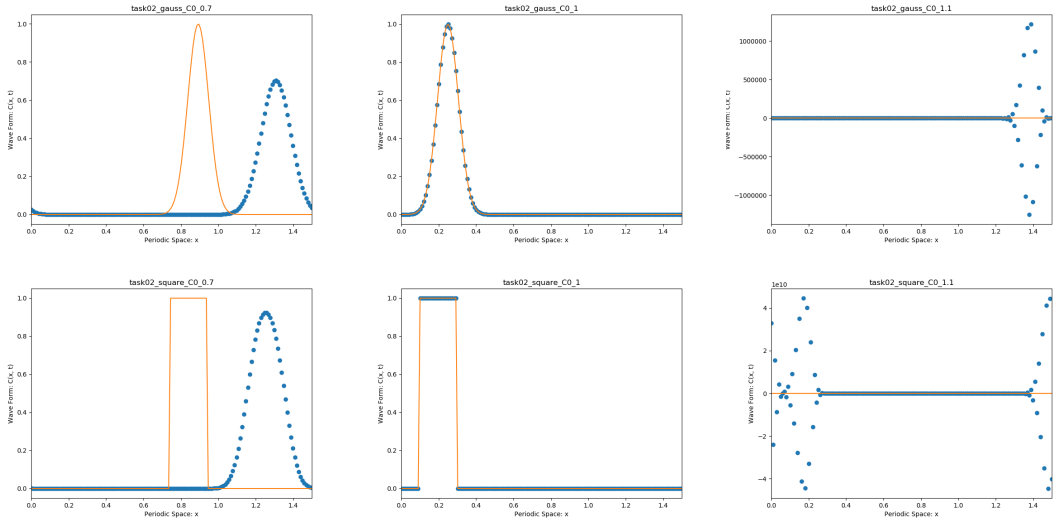

Figure 6: For initial Gauss wavelet (top) and Square wavelet (bottom) after 150 iterations (one leap around the boundary) with different current C_0. Left: $C_0 = 0.7$, Center: $C_0 = 1$ (perfect) and Right: $C_0 = 1.1$ (unstable).

For implementation one gets a very simple solver which computes the wave form at time $T = N_t\Delta t$ with $\Delta t = U\Delta x/C_0$ as the following;

```
def solve(C, C0, nt):
    nx = C.shape[0]
    im1 = np.array([nx - 1] + list(range(nx - 1)))
    for t in range(nt):
        C = (1 - C0) * C + C0 * C[im1]
    return C
```

where `C` is the initial wave form, the constant `C0` is the current C_0 and `nt` is the number of time steps to be performed. Again, the attached script `task02.py` implements this with additional code for setting up the initial conditions and plotting.