CEE 557 Groundwater Modeling

COMPUTER PROBLEM #2

Author:
Peishi Jiang, Dongchen Wang

Presented to Prof. Albert Valocchi

Wednesday, March 30, 2016

Contents

The purpose of this assignment is to analyze and compare numerical solutions to the 1D solute transport equation. This problem is divided into two parts: Part A covers finite difference methods; Part B covers finite element methods. You should work in teams, as in the first computer assignment sd.

The model problem is:

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial^2 x} - u\frac{\partial c}{\partial x} \tag{1}$$

The initial and boundary conditions are:

$$c(x,0) = 0 \tag{2a}$$

$$(uc - D\frac{\partial c}{\partial x})_{x=0} = uc_{Feed}, c_{Feed} = 1.0 \tag{2b}$$

$$c(x->\infty, t) = 0 \tag{2c}$$

1 Part A – Finite Difference Methods

1.1 Explicit Finite Difference Solutions

1.1.1 AR and PE plots

The analytical amplification factor and phase shift of the equation 1 are

$$\Phi_n^{anal} = e^{-4\pi^2\mathcal{D}(\frac{\Delta x}{\Lambda_n})^2}e^{-2\pi\mathcal{U}\frac{\Delta x}{\Lambda_n}} \tag{3a}$$

$$\Delta\Theta_n^{anal} = -2\pi\mathcal{U}\frac{\Delta x}{\Lambda_n} \tag{3b}$$

where $\mathcal{D} = \frac{D\Delta t}{\Delta x^2}$ and $\mathcal{U} = \frac{U\Delta t}{\Delta x}$.

The numerical solutions of the amplification factor and phase shift of the upstream method are

$$\Phi_n^{fdm,explicit,up} = 1 - (2\mathcal{D} + \mathcal{U})(1 - \cos 2\pi\frac{\Delta x}{\Lambda_n}) - i\mathcal{U}\sin 2\pi\frac{\Delta x}{\Lambda_n} \tag{4a}$$

$$\Delta\Theta_n^{fdm,explicit,up} = \sin^{-1}[\frac{-\mathcal{U}\sin 2\pi\frac{\Delta x}{\Lambda_n}}{|\Phi_n^{fdm,up}|}] \tag{4b}$$

The numberical solutions of the amplification factor and phase shift of the central method are

$$\Phi_n^{fdm,explicit,central} = 1 - 2\mathcal{D}(1 - \cos 2\pi\frac{\Delta x}{\Lambda_n}) - i\mathcal{U}\sin 2\pi\frac{\Delta x}{\Lambda_n} \tag{5a}$$

$$\Delta\Theta_n^{fdm,explicit,central} = \sin^{-1}[\frac{-\mathcal{U}\sin 2\pi\frac{\Delta x}{\Lambda_n}}{|\Phi_n^{fdm,explicit,central}|}] \tag{5b}$$

Also, as we know that the amplification ratio $AR = (\frac{|\Phi_n^{num}|}{|\Phi_n^{anal}|})^{Nn}$ and the phase error $PE = -2\pi - Nn\Delta\Theta_n^{num}$, the amplification ratios and the phase errors of the two methods can be obtained as follows.

$$AR_{fdm,explicit,i} = (\frac{|\Phi_n^{fdm,explicit,i}|}{|\Phi_n^{anal}|})^{Nn} \tag{6a}$$

$$PE_{fdm,explicit,i} = -2\pi - Nn\Delta\Theta_n^{fdm,explicit,i} \tag{6b}$$

where $i = up, central$.

Settings According to the problem description, the settings are $\Delta x = 0.05$ m, $\Delta t = 1$ day, $D = 2.5 \times 10^{-5}$ m2/day and $u = 0.005$ m/day.

Result The results of AR and PE of the two methods (i.e., upstream and central methods) by usnig explicit finite difference are plotted in Figure 1a. The values are shown in the Appendix.

Discussion For phase error (P.E.), it can be observed from the figure that the small wavelengths of both the two methods lag far behind the analytical solution while there is almost no phase error for large wavelengths. Also, despite a little difference in small wavelengths between the two methods, the two P.E. curves are almost identical. However, the amplification ratio (A.R.) values of the two methods are quite different, especially for small wavelengths. The figure shows that for small wavelengths, the amplification factors of the central method are almost three times as that of the analytical solution (A.R. 3.0). Meanwhile, for the upstream method, the A.R. values are almost zeros for small wavelengths. Nevertheless, both of the two methods converge to 1.0 as wavelength increases.

1.1.2 Numerical solutions versus analytical solutions

Numerical solutions - upstream The scheme of the numerical solutions to the upstream - explicit - finite difference method can be fomulated as follows.

$$A_{fdm,explicit,up}\tilde{c}^{k+1} = B_{fdm,explicit,up}\tilde{c}^{k} + b_{fdm,explicit,up} \tag{7}$$

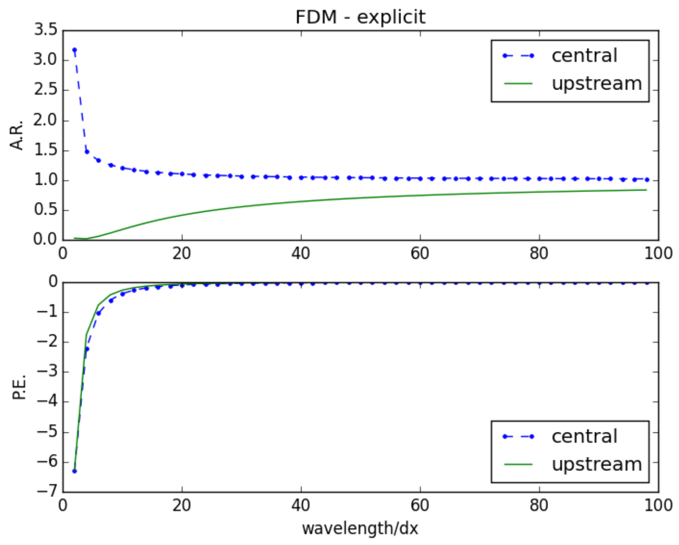
in which

$$A_{fdm,explicit,up} = I_{N\times N} \tag{8a}$$

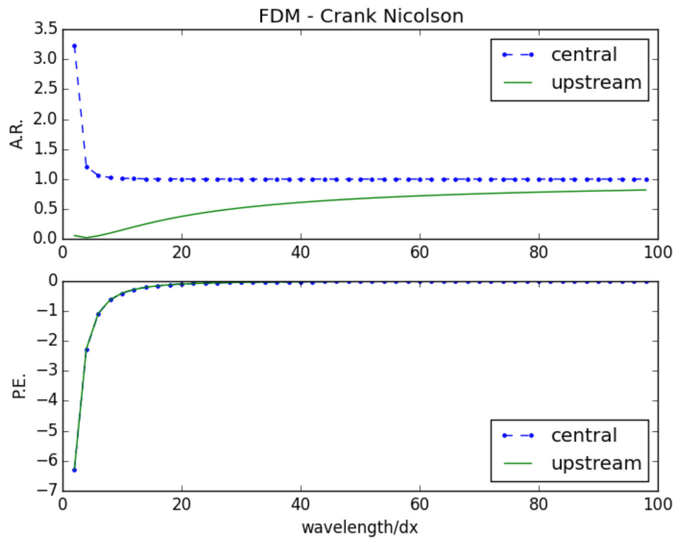
$$B_{fdm,explicit,up} = \begin{bmatrix} 1-(\alpha+1)\mathcal{D}-\alpha\mathcal{U} & \mathcal{D} & 0 & 0 & \cdots & 0 \\ \mathcal{D}+\mathcal{U} & -(2\mathcal{D}+\mathcal{U}-1) & \mathcal{D} & 0 & \cdots & 0 \\ 0 & \mathcal{D}+\mathcal{U} & -(2\mathcal{D}+\mathcal{U}-1) & \mathcal{D} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{D}+\mathcal{U} & -(2\mathcal{D}+\mathcal{U}-1) & \mathcal{D} \\ 0 & 0 & \cdots & 0 & \mathcal{D}+\mathcal{U} & -(2\mathcal{D}+\mathcal{U}-1 \end{bmatrix} \tag{8b}$$

$$b_{fdm,explicit,up} = \{\alpha(\mathcal{D}+\mathcal{U})c_{Feed}, 0, \ldots, 0\}^T_{1\times N} \tag{8c}$$

where $\alpha = \frac{\Delta x u}{D}$.



(a) Explicit



(b) Crank-Nicolson

Figure 1: AR and PE plots of finite difference solutions.

Numerical solutions - central The scheme of the numerical solutions to the central - finite difference method can be fomulated as follows.

$$A_{fdm,explicit,central}\tilde{c}^{k+1} = B_{fdm,explicit,central}\tilde{c}^{k} + b_{fdm,explicit,central} \tag{9}$$

in which

$$A_{fdm,explicit,central} = I_{N\times N} \tag{10a}$$

$$B_{fdm,explicit,central} = \begin{bmatrix} 1-2(\alpha+1)\mathcal{D}-\alpha\mathcal{U} & 2\mathcal{D} & 0 & 0 & \cdots & 0 \\ \mathcal{D}+\mathcal{U}/2 & 1-2\mathcal{D} & \mathcal{D}-\mathcal{U} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{D}+\mathcal{U}/2 & 1-2\mathcal{D} & \mathcal{D}-\mathcal{U} \\ 0 & 0 & \cdots & 0 & \mathcal{D}+\mathcal{U}/2 & 1-2\mathcal{D} \end{bmatrix}_{N\times N} \tag{10b}$$

$$b_{fdm,explicit,central} = \{\alpha(\mathcal{D}+\mathcal{U})c_{Feed}, 0, \ldots, 0\}^{T}_{1\times N} \tag{10c}$$

where $\alpha = \frac{\Delta x u}{D}$.

Settings The save as the settings in Section 1.1.1.

Result The numerial and analytical of solutions the two methods (i.e., upstream and central methods) by usnig explicit finite difference at T = 50 days and 100 days are plotted in Figure 2. The values are shown in the Appendix.

Discussion Both the results simulated at T=50 and T=100 show that compared with the analytical solutions, the numerical results of the central methods generate oscillations while numerical dispersions occur in the upstream method. These results are compatible with the Fourier analysis of part (a). As shown in part (a), the A.R. values of the central method are around 3 for small wavelengths, causing the numerical oscillation behaviors shown in the figure. Also, the low A.R. values of the upstream method for small wavelengths result in the numerical dispersion of the numerical solutions.

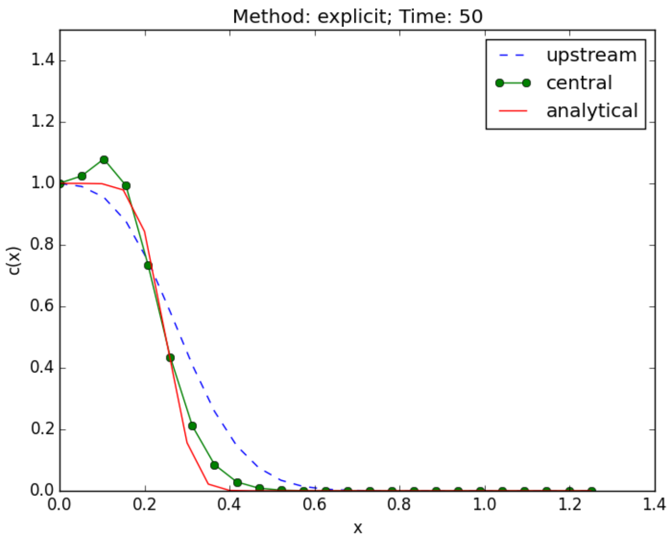
1.1.3 Stability examination with different Δt

The stability condition of the explicit-upstream-finite difference method is

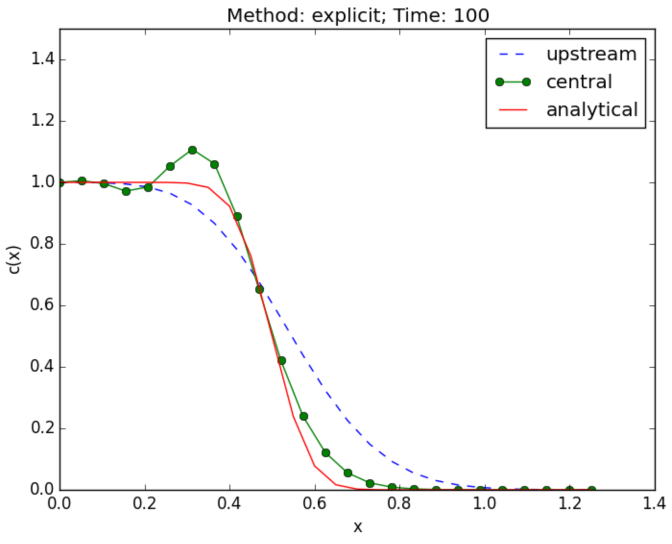
$$2\mathcal{D} + \mathcal{U} \leq 1 \tag{11}$$

We can get when $\Delta t \geq 8.33$, the numerical solution will be unstable.

Test 1 By assuming Δt = 8.5 days and the remaining setting remains unchanged, Figure 3a shows the numerical solutions at T = 8.5 days, 17 days, 25.5 days and 34 days. It can be seen from this figure that the numerical solution goes unstable very quickly.



(a) 50 days



(b) 100 days

Figure 2: Numerical and analytical solutions of explicit finite difference solutions at T = 50 days and 100 days.

Test 2 To further test the impact of Δt, we select $\Delta t = 2$ days. Figure 3b shows the numerical solutions at T = 2 days, 4 days, 6 days and 8 days. Similarly, it can be observed that the numerical solutions are unstable significantly.

Discussion Through the two tests with $\Delta t = 8.5$ days and 2 days, we conclude that the 3rd-type boundary condition would reduce the stability limit and enhance the unstable behavior with the increase of Δt. Even though in test 2, Δt (2 days) is within the stable condition (Eq. 11), the 3rd-type boundary condition still results in the unstable behavior of the numerical solutions.

1.2 Crank-Nicolson Solutions

1.2.1 AR and PE plots

The numerical solutions of the amplification factor and phase shift of the upstream method are

$$\Phi_n^{fdm,CN,up} = \frac{1 - (\mathcal{D} + \mathcal{U}/2)(1 - \cos 2\pi\frac{\Delta x}{\Lambda_n}) - i\frac{\mathcal{U}}{2}\sin 2\pi\frac{\Delta x}{\Lambda_n}}{1 - (\mathcal{D} + \mathcal{U}/2)(1 - \cos 2\pi\frac{\Delta x}{\Lambda_n}) + i\frac{\mathcal{U}}{2}\sin 2\pi\frac{\Delta x}{\Lambda_n}} \tag{12a}$$

$$\Delta\Theta_n^{fdm,CN,up} = \sin^{-1}[\frac{-\mathcal{U}\sin 2\pi\frac{\Delta x}{\Lambda_n}}{|\Phi_n^{fdm,CN,up}|}] \tag{12b}$$

The numberical solutions of the amplification factor and phase shift of the central method are

$$\Phi_n^{fdm,CN,central} = \frac{1 - \mathcal{D}(1 - \cos 2\pi\frac{\Delta x}{\Lambda_n}) - i\frac{\mathcal{U}}{2}\sin 2\pi\frac{\Delta x}{\Lambda_n}}{1 - \mathcal{D}(1 - \cos 2\pi\frac{\Delta x}{\Lambda_n}) + i\frac{\mathcal{U}}{2}\sin 2\pi\frac{\Delta x}{\Lambda_n}} \tag{13a}$$

$$\Delta\Theta_n^{fdm,CN,central} = \sin^{-1}[\frac{-\mathcal{U}\sin 2\pi\frac{\Delta x}{\Lambda_n}}{|\Phi_n^{fdm,CN,central}|}] \tag{13b}$$

Therefore, the amplification ratios and the phase errors of the two methods can be obtained as follows.

$$AR_{fdm,CN,i} = (\frac{|\Phi_n^{fdm,CN,i}|}{|\Phi_n^{anal}|})^{Nn} \tag{14a}$$

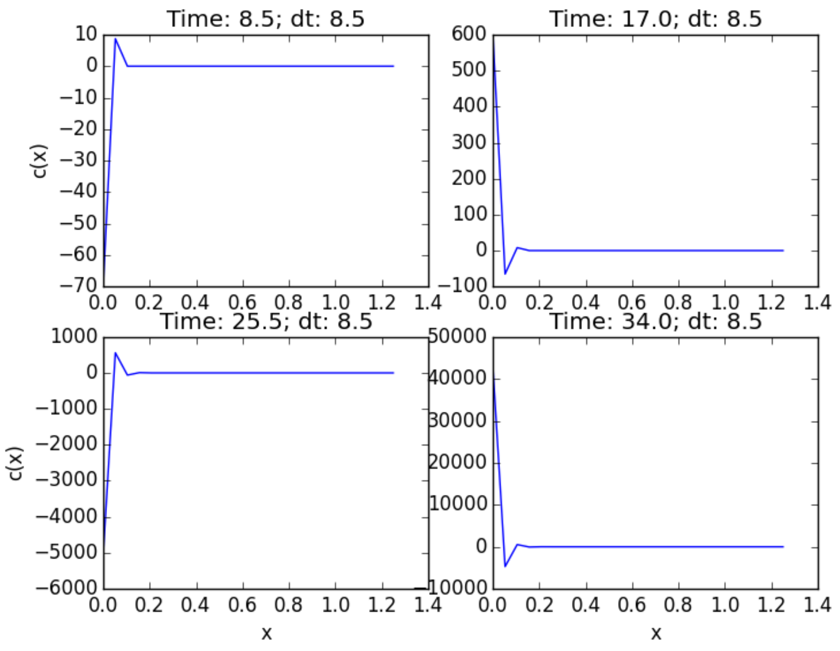
$$PE_{fdm,CN,i} = -2\pi - Nn\Delta\Theta_n^{fdm,CN,i} \tag{14b}$$

where $i = up, central$.

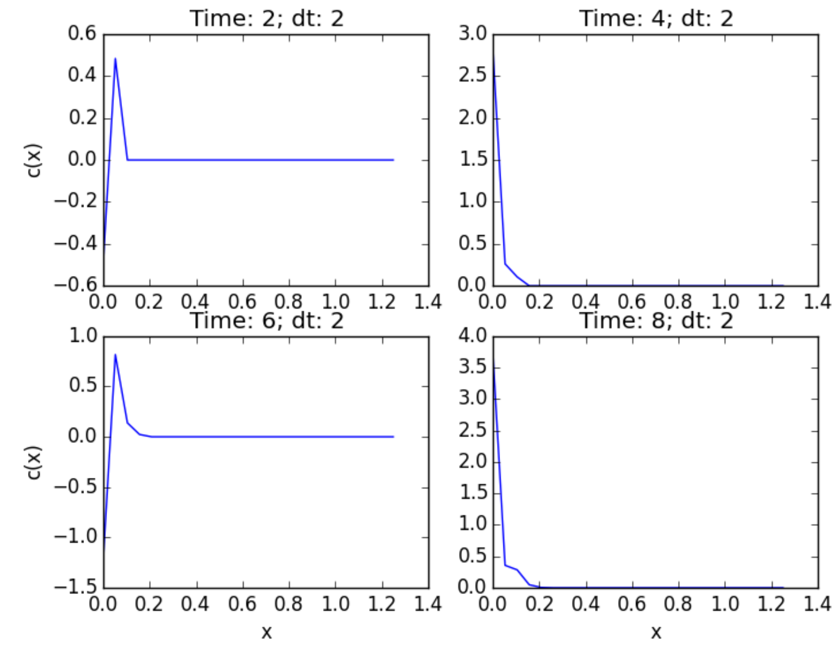
Settings The save as the settings in Section 1.1.1.

Result The results of AR and PE of the two methods (i.e., upstream and central methods) by usnig explicit finite difference are plotted in Figure 1b. The values are shown in the Appendix.

Discussion The A.R. and P.E. values of the Crank Nicolson method are almost the same as that of the explicit method for the central and upstream differences. Please refer to Section 1.1.1 for details.



(a) Explicit



(b) Crank-Nicolson

Figure 3: AR and PE plots of finite difference solutions.

1.2.2 Numerical solutions versus analytical solutions

Numerical solutions - upstream The scheme of the numerical solutions to the upstream - Crank Nicolson - finite difference method can be fomulated as follows.

$$A_{D,fdm,CN,up}\tilde{c}^{k+1} + A_{A,fdm,CN,up}\tilde{c}^{k+1} = B_{D,fdm,CN,up}\tilde{c}^{k} + B_{A,fdm,CN,up}\tilde{c}^{k} + b_{fdm,CN,up} \tag{15}$$

in which

$$A_{D,fdm,CN,up} = \begin{bmatrix} (\alpha+1)\mathcal{D}/2+1 & -\mathcal{D}/2 & 0 & 0 & \cdots & 0 \\ -\mathcal{D}/2 & \mathcal{D}+1 & -\mathcal{D}/2 & 0 & \cdots & 0 \\ 0 & -\mathcal{D}/2 & \mathcal{D}+1 & -\mathcal{D}/2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -\mathcal{D}/2 & \mathcal{D}+1 & -\mathcal{D}/2 \\ 0 & 0 & 0 & \cdots & -\mathcal{D}/2 & \mathcal{D}+1 \end{bmatrix}_{N\times N} \tag{16a}$$

$$B_{D,fdm,explicit,up} = \begin{bmatrix} 1-(\alpha+1)\mathcal{D}/2 & \mathcal{D}/2 & 0 & 0 & \cdots & 0 \\ \mathcal{D}/2 & 1-\mathcal{D} & \mathcal{D}/2 & 0 & \cdots & 0 \\ 0 & \mathcal{D}/2 & 1-\mathcal{D} & \mathcal{D}/2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{D}/2 & 1-\mathcal{D} & \mathcal{D}/2 \\ 0 & 0 & 0 & \cdots & \mathcal{D}/2 & 1-\mathcal{D} \end{bmatrix}_{N\times N} \tag{16b}$$

$$A_{A,fdm,CN,up} = \begin{bmatrix} \alpha\mathcal{U}/2 & 0 & \cdots & 0 \\ -\mathcal{U}/2 & \mathcal{U}/2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -\mathcal{U}/2 & \mathcal{U}/2 \end{bmatrix}_{N\times N} \tag{16c}$$

$$B_{A,fdm,explicit,up} = \begin{bmatrix} -\alpha\mathcal{U}/2 & 0 & \cdots & 0 \\ \mathcal{U}/2 & -\mathcal{U}/2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \mathcal{U}/2 & -\mathcal{U}/2 \end{bmatrix}_{N\times N} \tag{16d}$$

$$b_{fdm,explicit,up} = \{\alpha(\mathcal{D}+\mathcal{U})c_{Feed}, 0, \ldots, 0\}^T_{1\times N} \tag{16e}$$

where $\alpha = \frac{\Delta x u}{D}$.

Numerical solutions - central The scheme of the numerical solutions to the central - finite difference method can be fomulated as follows.

$$A_{D,fdm,CN,central}\tilde{c}^{k+1} + A_{A,fdm,CN,central}\tilde{c}^{k+1} = B_{D,fdm,CN,central}\tilde{c}^{k} + B_{A,fdm,CN,central}\tilde{c}^{k} + b_{fdm,CN,central} \tag{17}$$

in which

$$A_{D,fdm,CN,central} = \begin{bmatrix} (\alpha+1)\mathcal{D}+1 & -\mathcal{D} & 0 & 0 & \cdots & 0 \\ -\mathcal{D}/2 & \mathcal{D}+1 & -\mathcal{D}/2 & 0 & \cdots & 0 \\ 0 & -\mathcal{D}/2 & \mathcal{D}+1 & -\mathcal{D}/2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -\mathcal{D}/2 & \mathcal{D}+1 & -\mathcal{D}/2 \\ 0 & 0 & 0 & \cdots & -\mathcal{D}/2 & \mathcal{D}+1 \end{bmatrix}_{N\times N} \tag{18a}$$

$$B_{D,fdm,explicit,central} = \begin{bmatrix} 1-(\alpha+1)\mathcal{D} & \mathcal{D} & 0 & 0 & \cdots & 0 \\ \mathcal{D}/2 & 1-\mathcal{D} & \mathcal{D}/2 & 0 & \cdots & 0 \\ 0 & \mathcal{D}/2 & 1-\mathcal{D} & \mathcal{D}/2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{D}/2 & 1-\mathcal{D} & \mathcal{D}/2 \\ 0 & 0 & 0 & \cdots & \mathcal{D}/2 & 1-\mathcal{D} \end{bmatrix}_{N\times N} \tag{18b}$$

$$A_{A,fdm,CN,central} = \begin{bmatrix} \alpha\mathcal{U}/2 & 0 & 0 & \cdots & 0 \\ -\mathcal{U}/4 & 0 & \mathcal{U}/4 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -\mathcal{U}/4 & 0 & \mathcal{U}/4 \\ 0 & \cdots & 0 & -\mathcal{U}/4 & 0 \end{bmatrix}_{N\times N} \tag{18c}$$

$$B_{A,fdm,explicit,central} = \begin{bmatrix} -\alpha\mathcal{U}/2 & 0 & 0 & \cdots & 0 \\ \mathcal{U}/4 & 0 & -\mathcal{U}/4 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \mathcal{U}/4 & 0 & -\mathcal{U}/4 \\ 0 & \cdots & 0 & \mathcal{U}/4 & 0 \end{bmatrix}_{N\times N} \tag{18d}$$

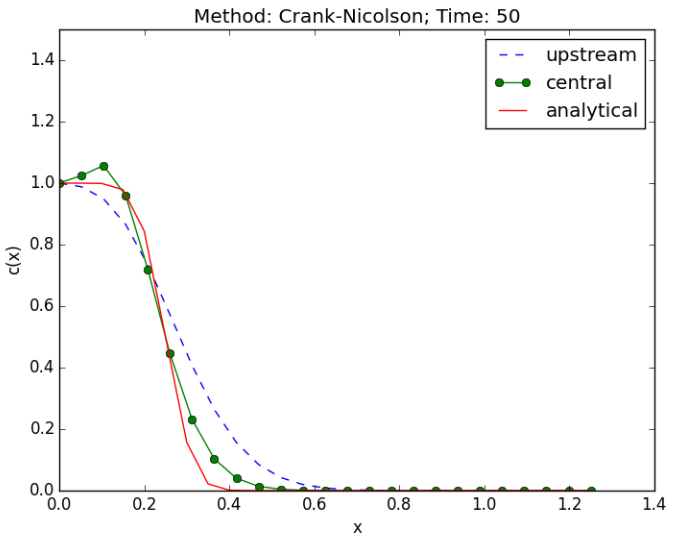
$$b_{fdm,explicit,central} = \{\alpha(2\mathcal{D}+\mathcal{U})c_{Feed}, 0, \ldots, 0\}^T_{1\times N} \tag{18e}$$

where $\alpha = \frac{\Delta x u}{D}$.

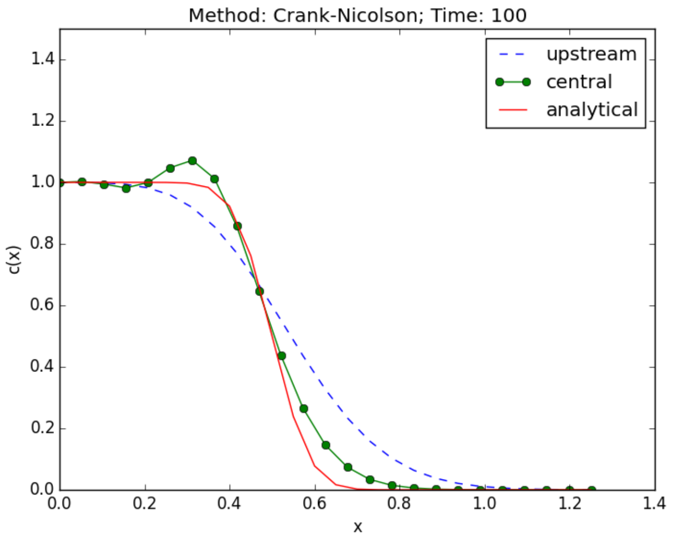
Settings The save as the settings in Section 1.1.1.

Result The numerial and analytical of solutions the two methods (i.e., upstream and central methods) by usnig Crank-Nicolson finite difference at T = 50 days and 100 days are plotted in Figure 4. The values are shown in the Appendix.

Discussion Similar to the comparisons of the explicit method discussed in part 1(b), for Crank Nicolson method, the upstream difference causes numerical dispersions and the central difference induces numerical oscillations. It complies with the Fourier analysis shown in part 2(a), as discussed in Section 1.1.2.



(a) 50 days



(b) 100 days

Figure 4: Numerical and analytical solutions of Crank-Nicolson finite difference solutions at T = 50 days and 100 days.

1.3 Mass Balance Analysis

Analytical solution The total mass of the analytical solution at time T is calculated as follows.

$$m_{anal,T} = c_{feed} \times u \times T \tag{19}$$

Numerical solution The total mass of the numerical solution at time T is calculated as follows

$$m^T_{fdm,i,j} = \sum_{i=1}^{N-1} \frac{\tilde{c_i^T} + \tilde{c_{i+1}^T}}{2} \times \Delta x \tag{20}$$

where i is explicit or central and j is up or central.

Result & Discussion The results of the mass of both the analytical and numerical solutions at T = 50 days and 100 days are shown in the Table 1. It can be seen from the table that in terms of the total mass, the solutions by using the central method outperform that using the upstream method. This is probably because despite the numerical oscillation generated by the central method, the central method is able to keep the total mass while the upstream method would result in a larger total mass because of the numerical dispersion. Meanwhile, Crank-Nicolson and the explicit method provide almost identical mass at a specific time.

Table 1: Nonlinear Model Results

time	analytical	explicit-up	explicit-central	CN-up	CN-central
50	0.25000	0.27993	0.25526	0.27992	0.25509
100	0.50000	0.52999	0.50492	0.52998	0.50493

2 Part B – Finite Element Methods for Solute Transport Problems

A Python code for Part A

```
#!/usr/bin/python
import csv
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv
from openpyxl import load_workbook

# parameters
dx = 0.05     # space resolution [m]
dt = 1.0      # time resolution [day]
```

```
D  = 2.5e-5  # dispersion coefficient [m2/day]
u  = 0.005   # advection speed [m/day]
N  = 25     # number of dx

cfeed = 1.


# Global settings
tprint = [50, 100]
xmin, xmax = 0.0, dx*N
x_pos = np.linspace(xmin, xmax, N)  # discretized x
# plot settings
linstyles = ['-', '--', 'o-']
colors = ['b', 'k', 'r']

def main():
    # Question 1 (b)
    c_explicit_upstream = finite_difference_c(dx, dt, u, D, tprint,
                                              time_method="explicit",
                                                  space_method="upstream")
    c_explicit_central = finite_difference_c(dx, dt, u, D, tprint,
                                             time_method="explicit",
                                             space_method="central")
    plot_c(tprint, c_explicit_central, c_explicit_upstream, "explicit")
    saveresult(tprint, c_explicit_central, c_explicit_upstream, "explicit")

    # Question 1 (c)
    # dt_unstable = 1./(2.*D/(dx**2)+u/dx) + 0.1
    dt_unstable = 2
    tprint_2 = 10*[1*dt_unstable, 2*dt_unstable, 3*dt_unstable, 4*dt_unstabl
    c_explicit_upstream_unstable = finite_difference_c(dx, dt_unstable, u, I
                                                       time_method="explicit
                                                       space_method="upstrea
    plot_c_unstable(tprint_2, c_explicit_upstream_unstable, dt_unstable)

    # Question 2 (b)
    c_CN_upstream = finite_difference_c(dx, dt, u, D, tprint,
                                        time_method="Crank-Nicolson",
                                        space_method="upstream")
    c_CN_central = finite_difference_c(dx, dt, u, D, tprint,
                                       time_method="Crank-Nicolson",
                                                space_method="central")
    plot_c(tprint, c_CN_central, c_CN_upstream, "Crank-Nicolson")
    saveresult(tprint, c_CN_central, c_CN_upstream, "Crank-Nicolson")

    # Question 3
    check_mass_balance(tprint, c_explicit_upstream, c_explicit_central,
                             c_CN_upstream, c_CN_central)

    plt.show()

def tridiag(a, b, c, n, k1=-1, k2=0, k3=1):
    a, b, c = a*np.ones(n-abs(k1)), b*np.ones(n-abs(k2)), c*np.ones(n-abs(k
    return np.matrix(np.diag(a, k1) + np.diag(b, k2) + np.diag(c, k3))

def ifin(t, tprint):
    for tcheck in tprint:
        if np.isclose(t, tcheck):
            return True
    return False

def analytical_c(n, x, t):
    h_n = lambda n, x, t: 1.0/n**2 * np.exp(-(n**2)*(np.pi**2)*t) * np.sin(
    h_n_num = 0
    for i in xrange(n):
        i_n = 2*(i+1) - 1
        h_n_num += h_n(i_n, x, t)
    return h_n_num * 8 / (np.pi**2)

def prepare_matrices(dx, dt, u, D, N, method_name):
    SD = D*dt/(dx**2.)
    SU = u*dt/dx
    alpha = dx*u/D
    #alpha=1.1
    if method_name == "Explicit (upstream)":
        A = np.identity(N)
        B = tridiag(SD+SU, -(2*SD+SU-1), SD, N)
        B[0, 0] = 1-(alpha+1)*SD-alpha*SU
        b = np.zeros(N)
        b[0] = alpha*(SD+SU)*cfeed
        A, B, b = np.matrix(A), np.matrix(B), np.matrix(b).T
    elif method_name == "Explicit (upstream_1st)":
        A = np.identity(N-1)
        B = tridiag(SD+SU, -(2*SD+SU-1), SD, N-1)
        b = np.zeros(N-1)
        b[0] = (SD+SU)*cfeed
        A, B, b = np.matrix(A), np.matrix(B), np.matrix(b).T
    elif method_name == "Explicit (central)":
        A = np.identity(N)
        B = tridiag(SD+SU/2., 1-2*SD, SD-SU/2., N)
        B[0, 0] = 1-2*(alpha+1)*SD-alpha*SU
        B[0, 1] = 2.*SD
        b = np.zeros(N)
        b[0] = alpha*(2.*SD+SU)*cfeed
        A, B, b = np.matrix(A), np.matrix(B), np.matrix(b).T
    elif method_name == "Crank-Nicolson (upstream)":
        Ad = tridiag(-SD/2., 1.+SD, -SD/2., N)
        Ad[0, 0] = 1.+(alpha+1.)/2.*SD
        Aa = tridiag(-SU/2., SU/2., 0., N)
        Aa[0, 0] = alpha*SU/2.
        A = Aa + Ad
        Bd = tridiag(SD/2., 1.-SD, SD/2., N)
        Bd[0, 0] = 1.-(alpha+1.)/2.*SD
        Ba = tridiag(SU/2., -SU/2., 0., N)
        Ba[0, 0] = -alpha*SU/2.
        B = Bd + Ba
        b = np.zeros(N)
        b[0] = alpha*(SD+SU)*cfeed
        A, B, b = np.matrix(A), np.matrix(B), np.matrix(b).T
    elif method_name == "Crank-Nicolson (central)":
        Ad = tridiag(-SD/2., 1.+SD, -SD/2., N)
        Ad[0, 0] = 1.+(alpha+1.)*SD
        Ad[0, 1] = -SD
        Aa = tridiag(-SU/4., 0, SU/4., N)
        Aa[0, 0] = alpha*SU/2.
        Aa[0, 1] = 0.
        A = Aa + Ad
        Bd = tridiag(SD/2., 1.-SD, SD/2., N)
        Bd[0, 0] = 1.-(alpha+1)*SD
        Bd[0, 1] = SD
        Ba = tridiag(SU/4., 0, -SU/4., N)
        Ba[0, 0] = -alpha*SU/2.
        Ba[0, 1] = 0.
        B = Bd + Ba
        b = np.zeros(N)
        b[0] = alpha*(2.*SD+SU)*cfeed
        A, B, b = np.matrix(A), np.matrix(B), np.matrix(b).T
    elif method_name == "Crank-Nicolson (compare)":
        Ad = tridiag(-SD/2., 1.+SD, -SD/2., N)
        Aa = tridiag(-SU/4., 0, SU/4., N)
        A = Aa + Ad
        Bd = tridiag(SD/2., 1.-SD, SD/2., N)
        Ba = tridiag(SU/4., 0, -SU/4., N)
        B = Bd + Ba
        b = np.zeros(N)
        b[0] = (SD+SU)*cfeed
        A, B, b = np.matrix(A), np.matrix(B), np.matrix(b).T
    else:
        raise Exception("Unknown method")

    return [A, B, b]

def finite_difference_c(dx, dt, u, D, tprint, tmax=101.0,
                        space_method="upstream", time_method="explicit"):

    # discretized x
    x_pos = np.linspace(xmin, xmax, N+1)
    # initial conditions
    c0 = np.zeros(N)

    # Script D & Script U
    SD = D*dt/(dx**2.)
    SU = u*dt/dx

    # x value
    c = [c0]
    c_print = []

    # build up the matrices
    if (space_method == "upstream") and (time_method == "explicit"):
        method_name = "Explicit (upstream)"
        [A, B, b] = prepare_matrices(dx, dt, u, D, N, method_name)
        Ainv = inv(A)
    elif (space_method == "upstream_1st") and (time_method == "explicit"):
        method_name = "Explicit (upstream_1st)"
        [A, B, b] = prepare_matrices(dx, dt, u, D, N, method_name)
        Ainv = inv(A)
    elif (space_method == "central") and (time_method == "explicit"):
        method_name = "Explicit (central)"
        [A, B, b] = prepare_matrices(dx, dt, u, D, N, method_name)
        Ainv = inv(A)
    elif (space_method == "upstream") and (time_method == "Crank-Nicolson")
        method_name = "Crank-Nicolson (upstream)"
        [A, B, b] = prepare_matrices(dx, dt, u, D, N, method_name)
        Ainv = inv(A)
    elif (space_method == "central") and (time_method == "Crank-Nicolson"):
        method_name = "Crank-Nicolson (central)"
        [A, B, b] = prepare_matrices(dx, dt, u, D, N, method_name)
        Ainv = inv(A)
    else:
        raise Exception("Unknown method")

    print method_name
    # loop
    t = 0.0
    c_old = np.matrix(c0).T
    while (t <= tmax):
        # calculate x values at time step k+1
        c_new = Ainv * (B * c_old + b)
        # append x_new to x
        c.append(c_new)
        # save the values to be printed in c_print
        if ifin(t, tprint):
            c_print.append(c_new)
        # update t
        t += dt
        # assign x_new to x_old
        c_old = c_new

    return c_print

def plot_c(tprint, c_central, c_upstream, method):

    pos_anal, val_anal = [], []
    # load the analytical solution
    # T = 50 days
    wbsheet = load_workbook('cp2_t_50_3rd.xlsx')['Sheet1'].columns
    pos_anal.append([cell.value for cell in wbsheet[1][1:]])
    val_anal.append([cell.value for cell in wbsheet[2][1:]])

    # T = 100 days
    wbsheet = load_workbook('cp2_t_100_3rd.xlsx')['Sheet1'].columns
    pos_anal.append([cell.value for cell in wbsheet[1][1:]])
    val_anal.append([cell.value for cell in wbsheet[2][1:]])

    for i in xrange(len(tprint)):
        fig, ax = plt.subplots()
        ax.plot(x_pos, c_upstream[i], '--', label="upstream")
        ax.plot(x_pos, c_central[i], 'o-', label="central")
        ax.plot(pos_anal[i], val_anal[i], label="analytical")
        ax.legend(loc="upper right")
        ax.set_ylim([0, 1.5])
        ax.set_ylabel('c(x)')
        ax.set_xlabel('x')
        ax.set_title('Method: %s; Time: %s' % (method, str(tprint[i])))
        plt.savefig(method + "_" + str(tprint[i]) + ".png")

def plot_c_unstable(tprint, c_unstable, dt_unstable):
    fig, axaar = plt.subplots(2, 2)
    c_unstable[0]
    axaar[0, 0].plot(x_pos, c_unstable[0])
    axaar[0, 0].set_title('Time: %s; dt: %s' % (tprint[0], dt_unstable))
    axaar[0, 0].set_ylabel('c(x)')
    axaar[0, 1].plot(x_pos, c_unstable[1])
    axaar[0, 1].set_title('Time: %s; dt: %s' % (tprint[1], dt_unstable))
    axaar[1, 0].plot(x_pos, c_unstable[2])
    axaar[1, 0].set_ylabel('c(x)')
    axaar[1, 0].set_xlabel('x')
    axaar[1, 0].set_title('Time: %s; dt: %s' % (tprint[2], dt_unstable))
    axaar[1, 1].plot(x_pos, c_unstable[3])
    axaar[1, 1].set_title('Time: %s; dt: %s' % (tprint[3], dt_unstable))
    axaar[1, 1].set_xlabel('x')
    plt.savefig('stable_check_dt' + '_' + str(dt_unstable) + ".png")
```

```
def saveresult(tprint, c_central, c_upstream, method):

    pos_anal, c_anal = [], []

    # load the analytical solution
    # T = 50 days
    wbsheet = load_workbook('cp2_t_50_3rd.xlsx')['Sheet1'].columns
    pos_anal.append([cell.value for cell in wbsheet[1][1:]])
    c_anal.append([cell.value for cell in wbsheet[2][1:]])

    # T = 100 days
    wbsheet = load_workbook('cp2_t_100_3rd.xlsx')['Sheet1'].columns
    pos_anal.append([cell.value for cell in wbsheet[1][1:]])
    c_anal.append([cell.value for cell in wbsheet[2][1:]])

    for i in xrange(len(tprint)):
        t = tprint[i]
        c_anal_v = c_anal[i]
        c_central_v = c_central[i]
        c_upstream_v = c_upstream[i]
        with open(method + "_T" + str(t) + "_values.csv", 'w') as csvfile:
            fieldnames = ['location', 'analytical', 'central', 'upstream']
            writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
            writer.writeheader()
            for j in xrange(len(x_pos)):
                writer.writerow({
                    'location':      '%.2f' % x_pos[j],
                    'analytical':    '%.5f' % c_anal_v[j],
                    'central':       '%.5f' % c_central_v[j],
                    'upstream':      '%.5f' % c_upstream_v[j]
                })


def check_mass_balance(t_set, c_explicit_upstream, c_explicit_central,
                       c_CN_upstream, c_CN_central):
    def calculate_mass(c):
        m = 0
        for i in xrange(len(c)-1):
            m += (c[i]+c[i+1])/2*dx
        return m
    with open('partA_mass_balance_check.csv', 'w') as csvfile:
        fieldnames = ['time', 'analytical', 'explicit-up', 'explicit-centra
                      'CN-up', 'CN-central']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        writer.writeheader()

        for i in xrange(len(t_set)):
            writer.writerow({
                'time':              '%d' % t_set[i],
                'analytical':        '%.5f' % (cfeed*u*t_set[i]),
                'explicit-up':       '%.5f' % calculate_mass(c_explicit_upst
                'explicit-central': '%.5f' % calculate_mass(c_explicit_cent
                'CN-up':             '%.5f' % calculate_mass(c_CN_upstream[i
                'CN-central':        '%.5f' % calculate_mass(c_CN_central[i]
            })

if __name__ == "__main__":
    main()
```