전산유체역학

PJ#2: Hyperbolic Type

항공우주 및 기계공학부

담당교수 : 김문상 교수님

제출일 : 2021-05-11

2016121150 윤준영

1. 서론

대표적인 포물선 특성(Hyperbolic Type)의 편미분 방정식인 Wave Equation을 여러 수치해석 방법을 이용하여 해석하였다.

time $t = 0$일 때 Step Function과 Sine Wave가 각각 $a = 1m/s$의 속도로 오른쪽 방향으로 전파되어 가는 형상을 1-D Wave equation을 사용하여 해석하라.

(*) 1-D Wave Equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \qquad (a > 0, wave\ speed)$$

Wave Equation은 선형 편미분 방적식으로, 특성곡선이 $x - at = const.$ 인 직선 식이다.

(1) Step Function Propagation:

$$u(x,0) = 10, \qquad for\ x \le 0$$
$$u(x,0) = -10x + 10, \qquad for\ 0 \le x \le 1.0$$
$$u(x,0) = 0, \qquad for\ x \ge 1.0$$

(2) Sine Wave Propagation:

$$u(x,0) = \sin 2\pi x\,, \qquad for\ 0 \le x \le 1$$
$$u(x,0) = 0, \qquad for\ otherwise$$

하첨자 i는 격자점의 위치를 의미하고 상첨자 n은 현재 시간, $n+1$은 미지수가 존재하는 다음 시간 단계(time step)을 의미한다. 격자의 개수는 $gridn$, 그래프의 개수는 gno, 그래프 사이의 시간 간격은 $tint$으로 표기하고, CFL은 다음과 같이 정의한다.

$$CFL = a\frac{\Delta t}{\Delta x}, \qquad \Delta t = \frac{CFL \cdot \Delta x}{a}$$

여기서 Δx : space interval, Δt : time interval이다.

2. Explicit Method

2.1. Euler's FTFS Method

Euler's FTFS Method는 미지수가 오직 1개 존재하는 대표적인 양해법이다. 시간에 대해 전방 차분법(Forward Time)을 사용하고, 공간에 대해서도 전방 차분법(Forward Space)을 사용하는 방법으로, 1차 정확도를 가진다. Euler's FTFS Method로 풀이한 Wave Equation은 다음과 같다.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^n - u_i^n}{\Delta x} = 0$$

위 식을 미지수 u_i^{n+1}에 대해 표현하면 다음과 같다.

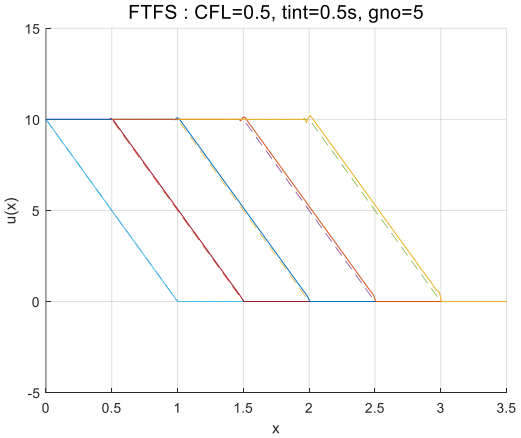
$$u_i^{n+1} = u_i^n - \frac{a\Delta t}{\Delta x}(u_{i+1}^n - u_i^n)$$

작성한 MATLAB CODE는 다음과 같다.

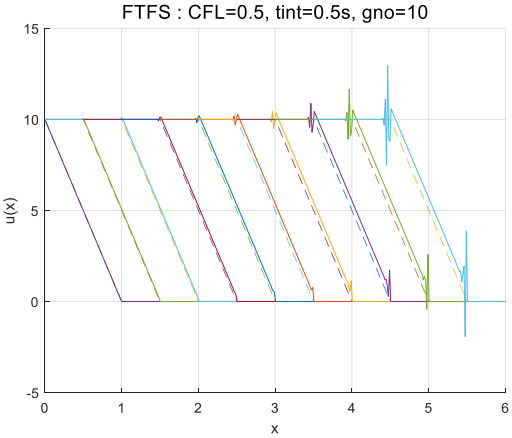
```
%% Euler's FTFS Method
clear; close all;

%% data input
clear; close all;
prompt = {'gridn', 'gno', 'tint', 'CFL'};
dlgtitle = 'Data'; dims = [1 35];
definput = {'200','5', '0.5', '0.8'}; % set default value
inputdata = inputdlg(prompt,dlgtitle,dims,definput); % input data
dialogue
gridn = str2double(inputdata{1}); % no. of grid
gno = str2double(inputdata{2}); % no. of graph
tint = str2double(inputdata{3}); % time interval between graphs
CFL = str2double(inputdata{4}); % CFL number

a=1; % speed of wave
width=3; % width of wave
x1=linspace(-1,2,gridn);
x(:,1)=x1;

%% exact solution
for i=2:1:gno
    x(:,i)=x(:,i-1)+a*tint;
end
for i=1:1:length(x1)
    if 0>x(i)
        u(1,i)=10;
    elseif 0<x(i) && x(i)<=1
        u(1,i)=-10*x(i)+10;
    else
        u(1,i)=0;
    end
end

%% FTFS method
u1(1,:)=u(1,:);
for n=1:1:gno-1
    for i=1:1:length(x1)-1
        u1(n+1,i)=u1(n,i)-CFL*(u1(n,i+1)-u1(n,i));
    end
end

%% plot
figure(1); hold on; grid on;
plot(x,u,'--') % plot exact solution
for i=1:1:gno % plot FTFS
    plot(x(:,i),u1(i,:))
end
axis([0 1+gno*tint -5 15])
title(['FTFS : CFL=',num2str(CFL),', tint=',num2str(tint),'s, gno=',
num2str(gno)], 'Fontsize', 14)
xlabel('x')
ylabel('u(x)')
```

〈 Table. 1 - Euler's FTFS Method MATLAB CODE 〉

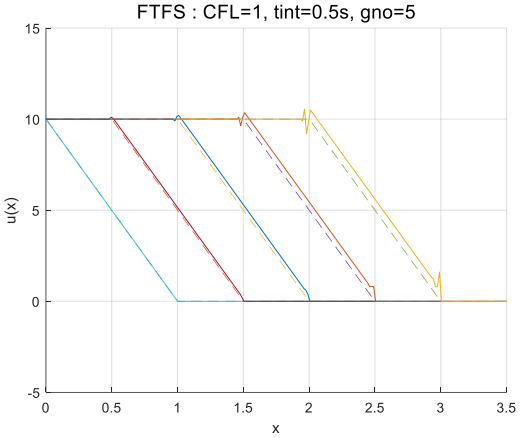
다음 Fig. 1 ~ Fig. 6은 그래프의 시간 간격을 $tint = 0.5s$, 격자의 개수 $gridn = 200$으로 설정한 후 CFL 값을 $0.5, 1.0, 1.5$로, 그래프의 개수 gno를 $5, 10$으로 변화시켜 Euler's FTFS Method로 해석한 결과이다.



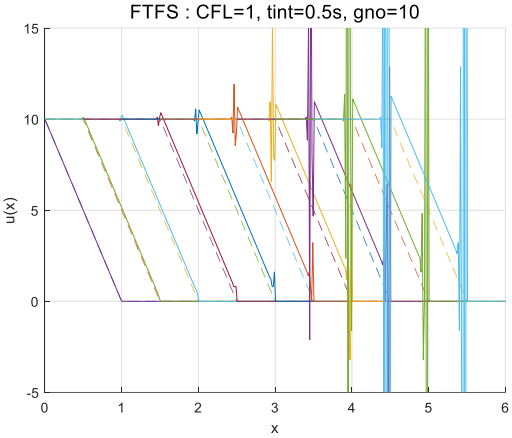
〈 Fig. 1 - FTFS CFL=0.5, gno=5 〉



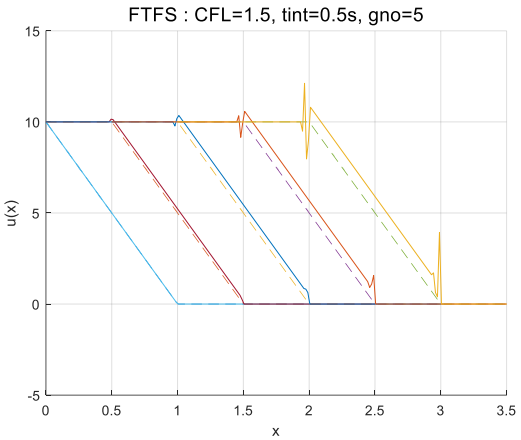
〈 Fig. 2 - FTFS CFL=0.5, gno=10 〉



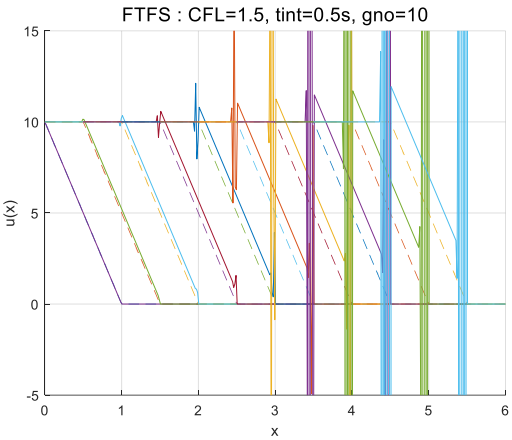
〈 Fig. 3 - FTFS CFL=1.0, gno=5 〉



〈 Fig. 4 - FTFS CFL=1.0, gno=10 〉



〈 Fig. 5 - FTFS CFL=1.5, gno=5 〉



〈 Fig. 6 - FTFS CFL=1.5, gno=10 〉

Fig. 1, Fig. 3, Fig. 5는 5개의 그래프까지 나타낸 것이며, Fig. 2, Fig. 4, Fig. 6은 10개의 그래프를 나타낸 것이다. 이 두 그룹을 살펴보면, iteration이 거듭될수록 unstable한 결과를 얻는 것을 알 수 있다. 또한 높은 CFL 값에서 iteration을 거듭할수록 낮은 CFL 에서의 값보다 더 unstable함을 알 수 있다. 그래프의 진동은 step function이 하강할 때와 하강이 멈추었을 때 크게 나타나며 step function이 일정한 기울기를 가지는 구간에서는 진동이 거의 발생하지 않았다. n이 증가할수록 ideal solution과의 오차가 점점 증가하는 것도 확인할 수 있으며 이 또한 CFL 값이 증가할수록 오차가 증가하는 폭이 커지며 발산하는 것을 확인할 수 있다. 이는 오차가 누적되어 iteration이 거듭될수록 오차가 증가하는 것으로 해석할 수 있다. 따라서 Euler's FTFS Method를 사용하여 구한 해는 n이 증가하면 CFL 값과 관계없이 발산하는 Unconditionally Unstable임을 알 수 있다.

2.2. Euler's FTCS Method

Euler's FTCS Method 또한 대표적인 양해법으로, 시간에 대해 전방 차분법(Forward Time)을 사용하고, 공간에 대해서는 중심 차분법(Central Space)을 사용하는 방법으로 공간에 대해서는 2차 정확도를 가진다. 하지만 시간에 대해서 1차 정확도를 가지므로 전체적으로는 1차 정확도를 가진다. Euler's FTCS Method로 풀이한 Wave Equation은 다음과 같다.

$$\frac{u_i^{n+1}-u_i^n}{\Delta t}+a\frac{u_{i+1}^n-u_{i-1}^n}{2\Delta x}=0$$

위 식을 미지수 u_i^{n+1}에 대해 표현하면 다음과 같다.

$$u_i^{n+1}=u_i^n-\frac{a\Delta t}{2\Delta x}(u_{i+1}^n-u_{i-1}^n)$$

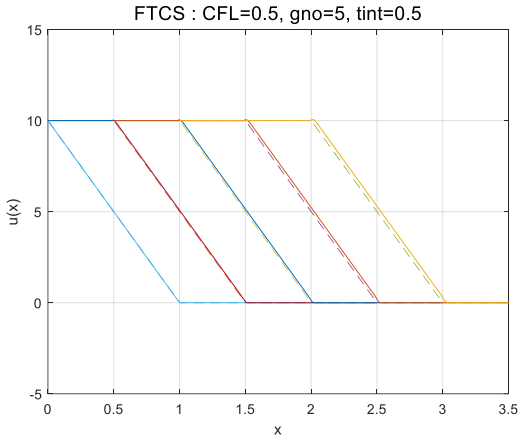
작성한 MATLAB CODE는 다음과 같다.

```
%% Euler's FTCS Method
clear; close all;

%% data input
clear; close all;
prompt = {'gridn', 'gno', 'tint', 'CFL'};
dlgtitle = 'Data'; dims = [1 35];
definput = {'200','5', '0.5', '0.8'}; % set default value
inputdata = inputdlg(prompt,dlgtitle,dims,definput); % input data
dialogue
gridn = str2double(inputdata{1}); % no. of grid
gno = str2double(inputdata{2}); % no. of graph
tint = str2double(inputdata{3}); % time interval between graphs
CFL = str2double(inputdata{4}); % CFL number

a=1; % speed of wave
width=3; % width of wave
x1=linspace(-1,2,gridn);
x(:,1)=x1;

%% exact solution
for i=2:1:gno
    x(:,i)=x(:,i-1)+a*tint;
end
for i=1:1:length(x1)
```

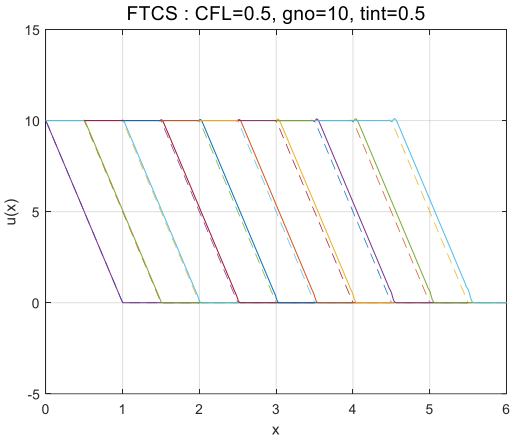
```
    if 0>x(i)
        u(1,i)=10;
    elseif 0<x(i) && x(i)<=1
        u(1,i)=-10*x(i)+10;
    else
        u(1,i)=0;
    end
end

%% FTCS method
u1(1,:)=u(1,:);
for n=1:1:gno-1
    for i=2:1:length(x1)-1
        u1(n+1,i)=u1(n,i)-0.5*CFL*(u1(n,i+1)-u1(n,i-1));
    end
    u1(n+1,1)=u1(n,1);
end

%% plot
plot(x,u,'--') % plot exact solution
hold on; grid on;
for i=1:1:gno % plot FTCS
    plot(x(:,i),u1(i,:))
end
axis([0 1+gno*tint -5 15])

title(['FTCS : CFL=',num2str(CFL),', gno=',num2str(gno),',
tint=',num2str(tint)],'Fontsize', 14)
xlabel('x')
ylabel('u(x)')
```

〈 Table. 2 - Euler's FTCS Method MATLAB CODE 〉

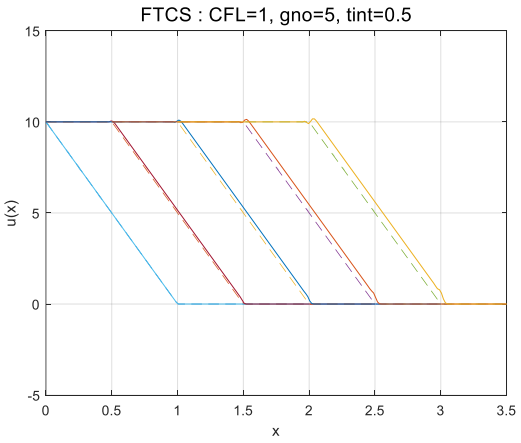
다음 Fig. 1 ~ Fig. 6은 그래프의 시간 간격을 $tint = 0.5s$, 격자의 개수 $gridn = 200$으로 설정한 후 CFL 값을 $0.5, 1.0, 1.5$로, 그래프의 개수 gno를 $5, 10$으로 변화시켜 Euler's FTCS Method로 해석한 결과이다.



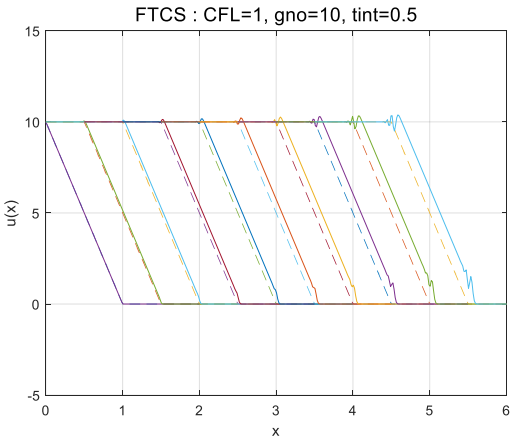
〈 Fig. 7 - FTCS CFL=0.5, gno=5 〉



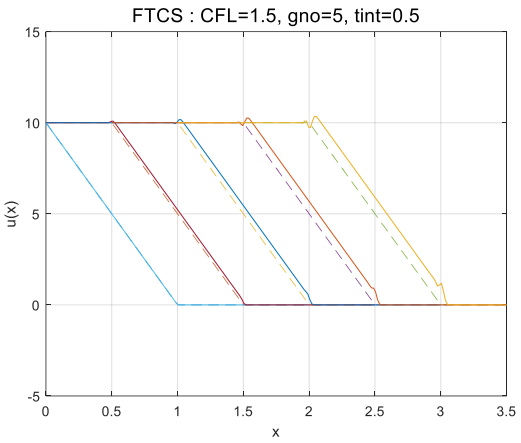
〈 Fig. 8 - FTCS CFL=0.5, gno=10 〉



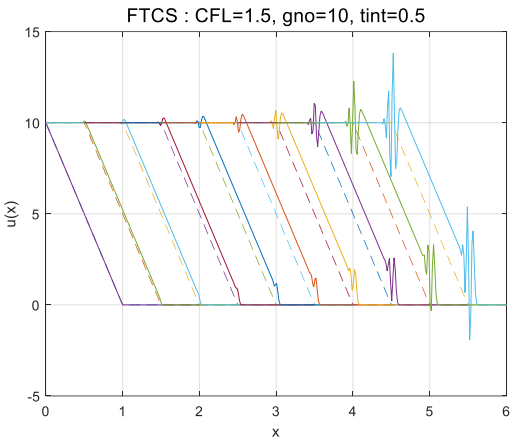
〈 Fig. 9 - FTCS CFL=1.0, gno=5 〉



〈 Fig. 10 - FTCS CFL=1.0, gno=10 〉



〈 Fig. 11 - FTCS CFL=1.5, gno=5 〉



〈 Fig. 12 - FTCS CFL=1.5, gno=10 〉

Fig. 7, Fig. 9, Fig. 11은 5개의 그래프까지 나타낸 것이며, Fig. 8, Fig. 10, Fig. 12는 10개의 그래프를 나타낸 것이다. FTCS Method에서는 초기에는 FTFS Method보다 더 안정적인 결과를 확인할 수 있었지만, FTFS의 결과와 마찬가지로 iteration이 거듭될수록 unstable하며 높은 CFL 값에서 더 unstable함을 알 수 있었고, 일정한 기울기를 가지는 구간에서는 진동이 거의 발생하지 않았다. n이 증가할수록 u의 값이 ideal solution보다 점점 증가하는 것도 확인할 수 있으며 이 또한 CFL 값이 증가할수록 오차가 증가하는 폭이 커지며 발산하는 것을 확인할 수 있다따라서 Euler's FTCS Method 또한 CFL 값과 관계없이 발산하는 Unconditionally Unstable임을 알 수 있다.

2.3. First-Order Upwind Differencing Method

First-Order Upwind Differencing Method는 공간에 대해 후방 차분법을 사용한 식이다.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0$$

위 식을 미지수 u_i^{n+1}에 대해 표현하면 다음과 같다.

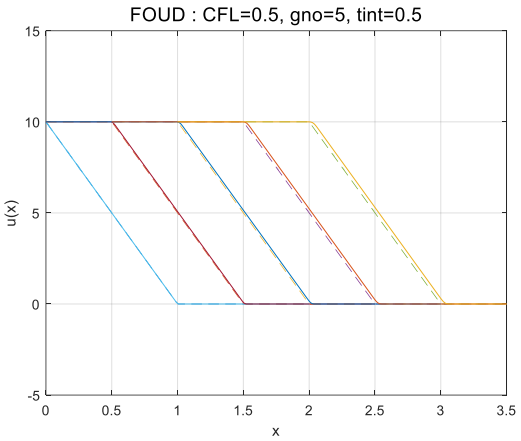
$$u_i^{n+1} = u_i^n - \frac{a\Delta t}{\Delta x}(u_i^n - u_{i-1}^n)$$

작성한 MATLAB CODE는 다음과 같다.

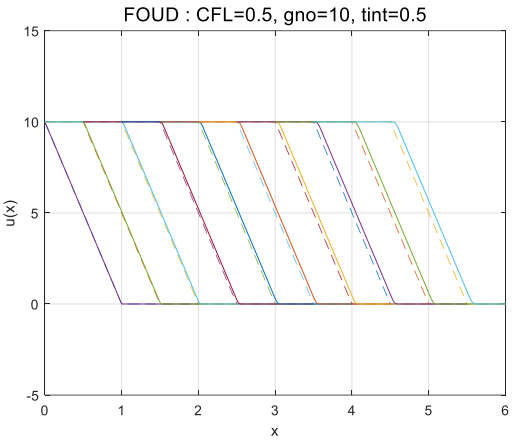
```
%% First-Order Upwind Differencing Method
clear; close all;

%% data input
clear; close all;
prompt = {'gridn', 'gno', 'tint', 'CFL'};
dlgtitle = 'Data'; dims = [1 35];
definput = {'50','5', '0.5', '0.8'}; % set default value
inputdata = inputdlg(prompt,dlgtitle,dims,definput); % input data
dialogue
gridn = str2double(inputdata{1}); % no. of grid
gno = str2double(inputdata{2}); % no. of graph
tint = str2double(inputdata{3}); % time interval between graphs
CFL = str2double(inputdata{4}); % CFL number

a=1; % speed of wave
width=3; % width of wave
x1=linspace(-1,2,gridn);
x(:,1)=x1;

%% exact solution
for i=2:1:gno
    x(:,i)=x(:,i-1)+a*tint;
end
for i=1:1:length(x1)
    if 0>x(i)
        u(1,i)=10;
    elseif 0<x(i) && x(i)<=1
        u(1,i)=-10*x(i)+10;
    else
        u(1,i)=0;
    end
end

%% FOUD method
u1(1,:)=u(1,:);
for n=1:1:gno-1
    for i=2:1:length(x1)-1
        u1(n+1,i)=u1(n,i)-CFL*(u1(n,i)-u1(n,i-1));
    end
    u1(n+1,1)=u1(n,1);
end

%% plot
plot(x,u,'--') % plot exact solution
hold on; grid on;
for i=1:1:gno % plot FOUD
    plot(x(:,i),u1(i,:))
end
axis([0 1+gno*tint -5 15])
title(['FOUD : CFL=',num2str(CFL),', gno=',num2str(gno),',
tint=',num2str(tint)],'Fontsize', 14)
xlabel('x')
ylabel('u(x)')
```

〈 Table. 3 - First-Order Upwind Differencing Method MATLAB CODE 〉

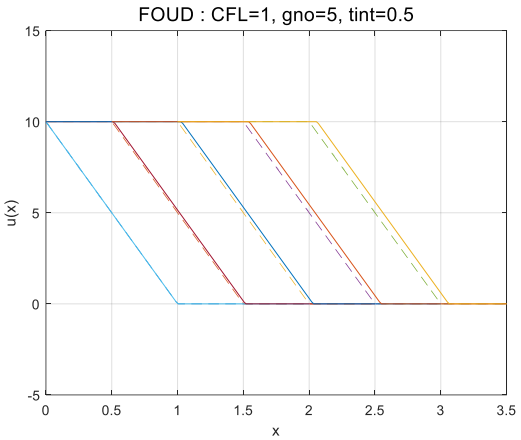
다음 Fig. 13 ~ Fig. 18은 그래프의 시간 간격을 $tint = 0.5s$, 격자의 개수 $gridn = 200$으로 설정한 후 CFL 값을 $0.5, 1.0, 1.5$로, 그래프의 개수 gno를 $5, 10$으로 변화시켜 First-Order Upwind Differencing Method로 해석한 결과이다.



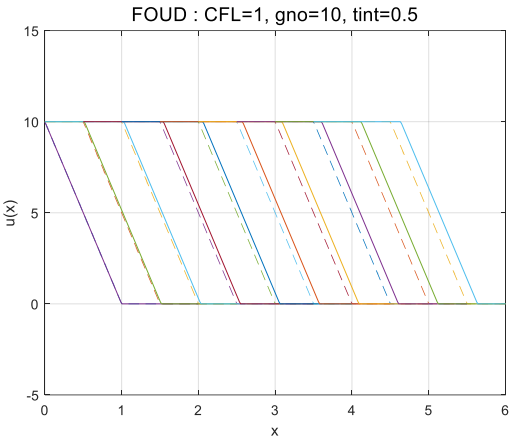
〈 Fig. 13 - FOUD CFL=0.5, gno=5 〉



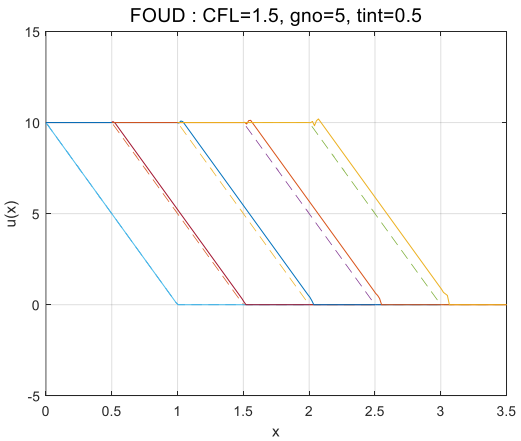
〈 Fig. 14 - FOUD CFL=0.5, gno=10 〉



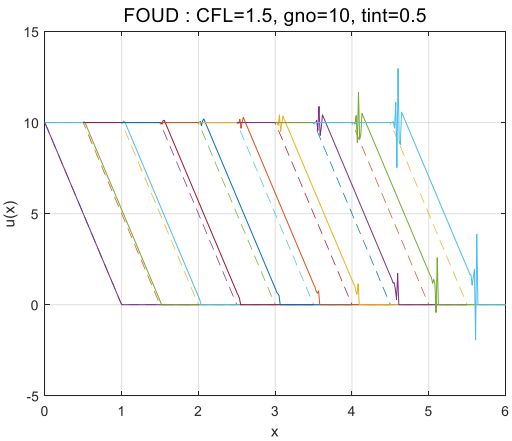
〈 Fig. 15 - FOUD CFL=1.0, gno=5 〉



〈 Fig. 16 - FOUD CFL=1.0, gno=10 〉



〈 Fig. 17 - FOUD CFL=1.5, gno=5 〉



〈 Fig. 18 - FOUD CFL=1.5, gno=10 〉

Fig. 13 ~ Fig. 16을 보면 CFL 값이 1이하일 때에는 FTFS나 FTCS와 달리 안정적인 상태인 것을 확인할 수 있다. 하지만 Fig. 17과 Fig. 18을 보면 iteration을 거듭할수록 점점 결과값이 불안정해지는 것을 확인할 수 있다. 따라서 First-Order Upwind Differencing Method는 $CFL \leq 1$일 때 Conditionally Stable한 것을 알 수 있다.

3. Implicit Method

3.1. Euler's BTCS Method

Euler's BTCS Method는 대표적인 음해법(Implicit method)으로 항상 수렴된 해(Unconditionally Stable)를 가지는 것이 특징이다. 시간에 대해서 후방 차분법(Backward Time)을 사용하고 공간에 대해서는 중심 차분법(Central Space)를 사용한다.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{\Delta x} = 0$$

위 식에서 미지수만 좌변 남기도록 이항하면 다음과 같다.

$$\frac{1}{2}\left(\frac{a\Delta t}{\Delta x}\right)u_{i-1}^{n+1} - u_i^{n+1} - \frac{1}{2}\left(\frac{a\Delta t}{\Delta x}\right)u_{i+1}^{n+1} = -u_i^n$$

위 식에서 계수를 간단하게 나타내면 다음과 같다.

$$b_i^n u_{i-1}^{n+1} + d_i^n u_i^{n+1} + a_i^n u_{i+1}^{n+1} = c_i^n$$

미지수가 여러 개 존재하는 식이므로 행렬로 나타내어 구하면 다음과 같다.

$$\begin{bmatrix} d_2^n & a_2^n & 0 & 0 & 0 & 0 & 0 & 0 \\ b_3^n & d_3^n & a_3^n & 0 & 0 & 0 & 0 & 0 \\ 0 & b_4^n & d_4^n & a_4^n & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & b_{imax-3}^n & d_{imax-3}^n & a_{imax-3}^n & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{imax-2}^n & d_{imax-2}^n & a_{imax-2}^n \\ 0 & 0 & 0 & 0 & 0 & 0 & b_{imax-1}^n & d_{imax-1}^n \end{bmatrix} \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \\ \vdots \\ \vdots \\ u_{imax-3}^{n+1} \\ u_{imax-2}^{n+1} \\ u_{imax-1}^{n+1} \end{bmatrix} = \begin{bmatrix} c_2^n - b_2^n u_1^{n+1} \\ c_3^n \\ c_4^n \\ \vdots \\ \vdots \\ c_{imax-3}^n \\ c_{imax-2}^n \\ c_{imax-1}^n - a_{imax-1}^n u_{imax-1}^{n+1} \end{bmatrix}$$

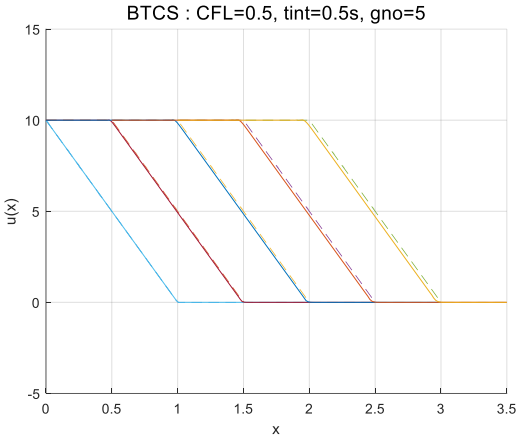
작성한 MATLAB CODE는 다음과 같다.

```
%% Euler's BTCS Method
clear; close all;

%% data input
clear; close all;
prompt = {'gridn', 'gno', 'tint', 'CFL'};
dlgtitle = 'Data'; dims = [1 35];
definput = {'200','5', '0.5', '1'}; % set default value
inputdata = inputdlg(prompt,dlgtitle,dims,definput); % input data
dialogue
gridn = str2double(inputdata{1}); % no. of grid
```

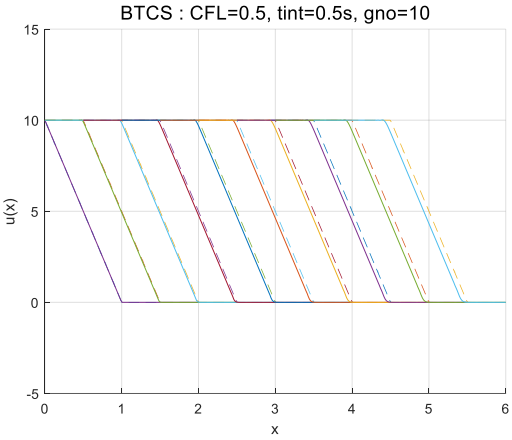
```
gno = str2double(inputdata{2}); % no. of graph
tint = str2double(inputdata{3}); % time interval between graphs
CFL = str2double(inputdata{4}); % CFL number

a=1; % speed of wave
width=3; % width of wave
x1=linspace(-1,2,gridn);
x(:,1)=x1;

%% exact solution
for i=2:1:gno
    x(:,i)=x(:,i-1)+a*tint;
end
for i=1:1:length(x1)
    if 0>x(i)
        u(1,i)=10;
    elseif 0<x(i) && x(i)<=1
        u(1,i)=-10*x(i)+10;
    else
        u(1,i)=0;
    end
end

%% BTCS method
b=0.5*CFL;
d=-1;
a=-0.5*CFL;
c=-u;
u1=u;
u1(2,:)=u(1,:);
for n=1:1:gno-1
    AA(1,1,n)=d; AA(1,2,n)=a; AA(gno-2,gno-2,n)=d; AA(gno-2,gno-3,n)=b;
    u1(n+1,1)=u1(n,1); u1(n+1,gno)=u1(n,gno);
    BB(1,n)=c(n,2)-b*u1(n+1,1);
    BB(gno-2,n)=c(n,gno-1)-a*u1(n+1,gno);
    for i=2:1:length(x1)-1
        AA(i,i-1,n)=b;
        AA(i,i,n)=d;
        AA(i,i+1,n)=a;
        BB(i,n)=c(n,i+1);
    end
    CC=AA(:,:,n)\BB(:,n);
    u1(n+1,:)=CC';
    c(n+1,:)=-u1(n+1,:);
end

%% plot
figure(1); hold on; grid on;
plot(x,u','--') % plot exact solution
for i=1:1:gno % plot BTCS
    plot(x(:,i),u1(i,:))
end
axis([0 1+gno*tint -5 15])
title(['BTCS : CFL=',num2str(CFL),', tint=',num2str(tint),'s,
gno=',num2str(gno)], 'Fontsize', 14)
xlabel('x')
ylabel('u(x)')
```

〈 Table. 4 - Euler's BTCS Method MATLAB CODE 〉

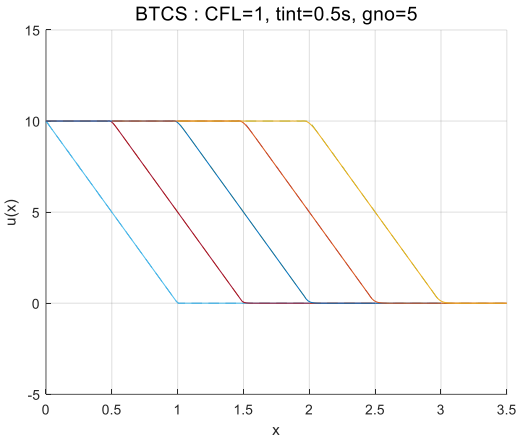
다음 Fig. 19 ~ Fig. 24는 그래프의 시간 간격을 $tint = 0.5s$, 격자의 개수 $gridn = 200$으로 설정한 후 CFL 값을 $0.5, 1.0, 1.5$로, 그래프의 개수 gno를 $5, 10$으로 변화시켜 Euler's BTCS Method로 해석한 결과이다.



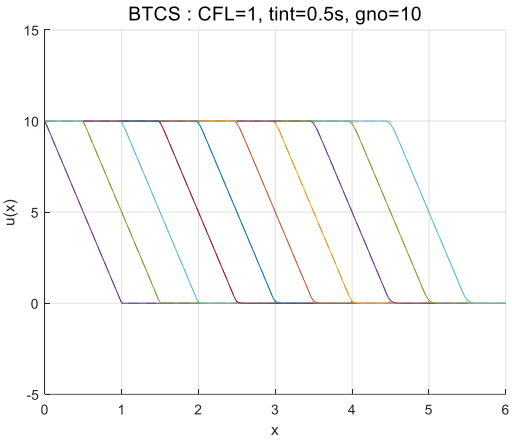
〈 Fig. 19 - BTCS CFL=0.5, gno=5 〉



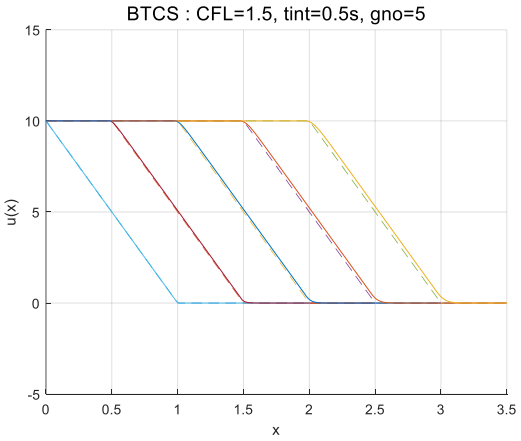
〈 Fig. 20 - BTCS CFL=0.5, gno=10 〉



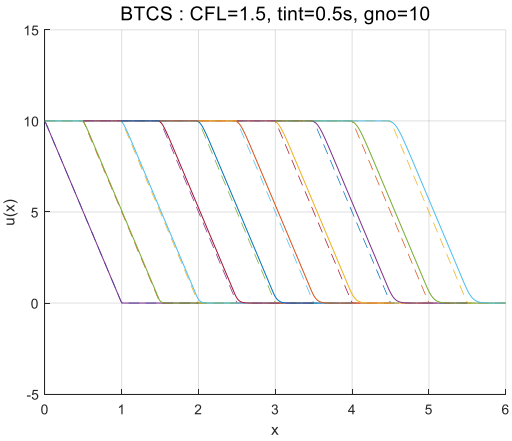
〈 Fig. 21 - BTCS CFL=1.0, gno=5 〉



〈 Fig. 22 - BTCS CFL=1.0, gno=10 〉



〈 Fig. 23 - BTCS CFL=1.5, gno=5 〉



〈 Fig. 24 - BTCS CFL=1.5, gno=10 〉

Fig. 19 ~ Fig. 24을 보면 $CFL = 1.0$일 때 오차가 거의 없었고 CFL의 변화에 따라 오차가 발생하는 것을 확인할 수 있다. 하지만 Explicit Method를 사용하였을 때와는 달리 Implicit Method인 Euler's BTCS Method를 사용하였을 때 CFL 값과 iteration에 상관없이 발산하지 않는 안정적인 해를 구할 수 있었다. 따라서 Euler's BTCS Method를 사용하면 Unconditionally Stable한 해를 찾을 수 있다.

3.2. Crank-Nicolson Method

Crank-Nicolson Method는 가장 많이 사용하는 음해법(Implicit method)으로 항상 수렴된 해(Unconditionally Stable)를 가지는 것이 특징이다. 2차 정확도를 가지며 음해법과 양해법을 모두 이용하는 방법이다. Time step의 절반은 양해법을 사용하고 절반은 음해법을 사용하는 것이 특징이다. 두 식은 다음과 같다.

$$\frac{u_i^{n+\frac{1}{2}} - u_i^n}{\frac{\Delta t}{2}} + a\frac{u_{i+1}^{n+1} - u_{i-1}^n}{2\Delta x} = 0$$

$$\frac{u_i^{n+1} - u_i^{n+\frac{1}{2}}}{\frac{\Delta t}{2}} + a\frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} = 0$$

두 식을 더한 후 미지수만 좌변으로 정리하면 다음과 같다.

$$-\frac{1}{4}\left(\frac{a\Delta t}{\Delta x}\right)u_{i-1}^{n+1} + u_i^{n+1} + \frac{1}{4}\left(\frac{a\Delta t}{\Delta x}\right)u_{i+1}^{n+1} = u_i^n - \frac{1}{4}\frac{a\Delta t}{\Delta x}(u_{i+1}^n - u_{i-1}^n)$$

3.1절과 같이 위 식에서 계수를 간단하게 정리하고 행렬로 나타내면 다음과 같다.

$$b_i^n u_{i-1}^{n+1} + d_i^n u_i^{n+1} + a_i^n u_{i+1}^{n+1} = c_i^n$$

미지수가 여러 개 존재하는 식이므로 행렬로 나타내어 구하면 다음과 같다.

$$\begin{bmatrix} d_2^n & a_2^n & 0 & 0 & 0 & 0 & 0 & 0 \\ b_3^n & d_3^n & a_3^n & 0 & 0 & 0 & 0 & 0 \\ 0 & b_4^n & d_4^n & a_4^n & 0 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & b_{imax-3}^n & d_{imax-3}^n & a_{imax-3}^n & 0 \\ 0 & 0 & 0 & 0 & 0 & b_{imax-2}^n & d_{imax-2}^n & a_{imax-2}^n \\ 0 & 0 & 0 & 0 & 0 & 0 & b_{imax-1}^n & d_{imax-1}^n \end{bmatrix} \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ u_4^{n+1} \\ \vdots \\ \vdots \\ u_{imax-3}^{n+1} \\ u_{imax-2}^{n+1} \\ u_{imax-1}^{n+1} \end{bmatrix} = \begin{bmatrix} c_2^n - b_2^n u_1^{n+1} \\ c_3^n \\ c_4^n \\ \vdots \\ \vdots \\ c_{imax-3}^n \\ c_{imax-2}^n \\ c_{imax-1}^n - a_{imax-1}^n u_{imax-1}^{n+1} \end{bmatrix}$$

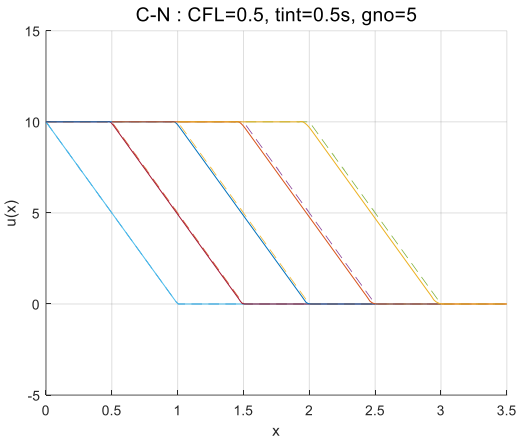
작성한 MATLAB CODE는 다음과 같다.

```
%% Crank-Nicolson Method
clear; close all;

%% data input
clear; close all;
prompt = {'gridn', 'gno', 'tint', 'CFL'};
dlgtitle = 'Data'; dims = [1 35];
definput = {'200','5', '0.5', '1'}; % set default value
```

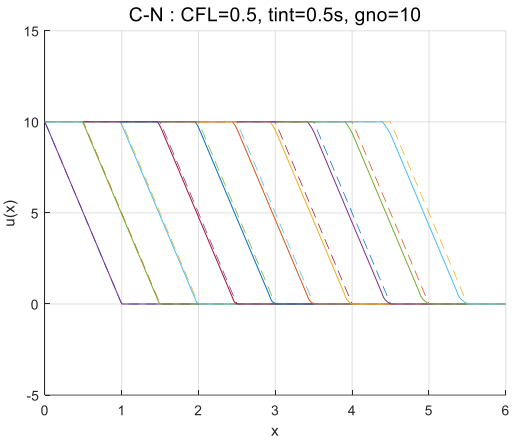
```
inputdata = inputdlg(prompt,dlgtitle,dims,definput); % input data
dialogue
gridn = str2double(inputdata{1}); % no. of grid
gno = str2double(inputdata{2}); % no. of graph
tint = str2double(inputdata{3}); % time interval between graphs
CFL = str2double(inputdata{4}); % CFL number

a=1; % speed of wave
width=3; % width of wave
x1=linspace(-1,2,gridn);
x(:,1)=x1;

%% exact solution
for i=2:1:gno
    x(:,i)=x(:,i-1)+a*tint;
end
for i=1:1:length(x1)
    if 0>x(i)
        u(1,i)=10;
    elseif 0<x(i) && x(i)<=1
        u(1,i)=-10*x(i)+10;
    else
        u(1,i)=0;
    end
end

%% C-N method
b=-0.25*CFL; d=1; a=0.25*CFL;
u1=u;
u1(2,:)=u(1,:);
for n=1:1:gno-1
    c(n,1)=u1(n,1); c(n,length(x1))=u1(n,length(x1));
    for i=2:1:length(x1)-1
        c(n,i)=u1(n,i)-0.25*CFL*(u1(1,i+1)-u1(1,i-1));
    end
    AA(1,1,n)=d; AA(1,2,n)=a; AA(gno-2,gno-2,n)=d; AA(gno-2,gno-3,n)=b;
    u1(n+1,1)=u1(n,1); u1(n+1,gno)=u1(n,gno);
    BB(1,n)=c(n,2)-b*u1(n+1,1);
    BB(gno-2,n)=c(n,gno-1)-a*u1(n+1,gno);
    for i=2:1:length(x1)-1
        AA(i,i-1,n)=b; AA(i,i,n)=d; AA(i,i+1,n)=a; BB(i,n)=c(n,i+1);
    end
    CC=AA(:,:,n)\BB(:,n);
    u1(n+1,:)=CC';
    c(n+1,:)=-u1(n+1,:);
end

%% plot
figure(1); hold on; grid on;
plot(x,u','--') % plot exact solution
for i=1:1:gno % plot C-N
    plot(x(:,i),u1(i,:))
end
axis([0 1+gno*tint -5 15])
title(['C-N : CFL=',num2str(CFL),', tint=',num2str(tint),'s,
gno=',num2str(gno)], 'Fontsize', 14)
xlabel('x')
ylabel('u(x)')
```

〈 Table. 5 - Crank-Nicolson Method MATLAB CODE 〉

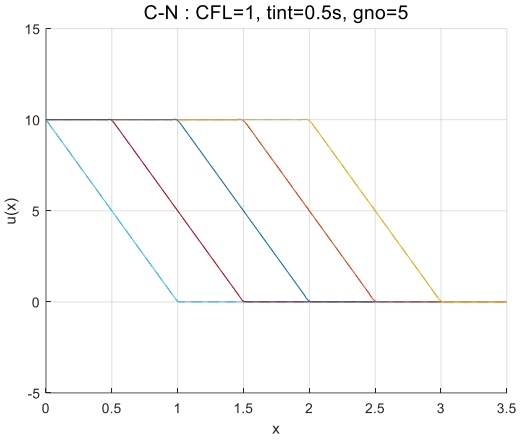
다음 Fig. 25 ~ Fig. 30는 그래프의 시간 간격을 $tint = 0.5s$, 격자의 개수 $gridn = 200$으로 설정한 후 CFL 값을 $0.5, 1.0, 1.5$로, 그래프의 개수 gno를 $5, 10$으로 변화시켜 Crank-Nicolson Method로 해석한 결과이다.



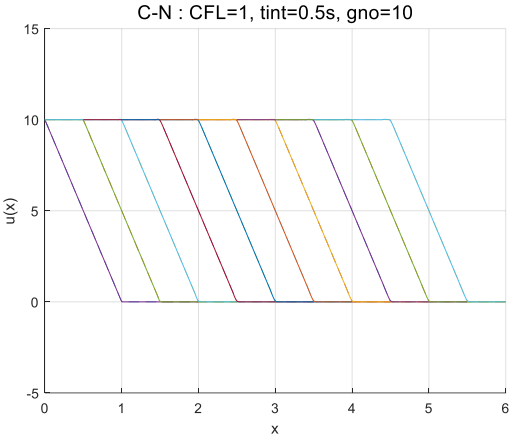
〈 Fig. 25 - C-N CFL=0.5, gno=5 〉



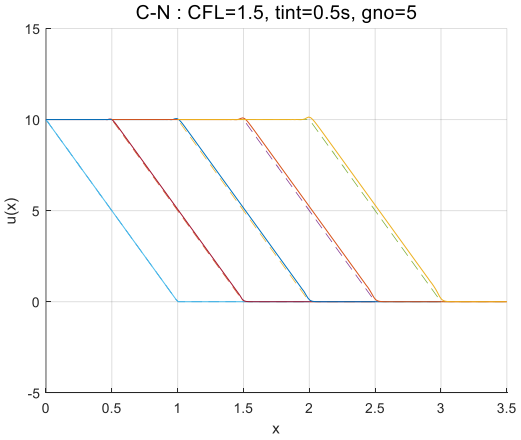
〈 Fig. 26 - C-N CFL=0.5, gno=10 〉



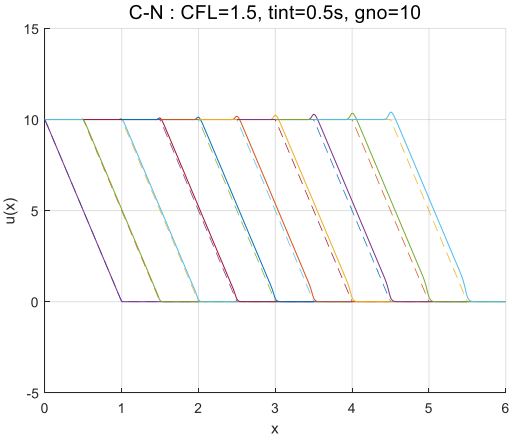
〈 Fig. 27 - C-N CFL=1.0, gno=5 〉



〈 Fig. 28 - C-N CFL=1.0, gno=10 〉



〈 Fig. 29 - C-N CFL=1.5, gno=5 〉



〈 Fig. 30 - C-N CFL=1.5, gno=10 〉

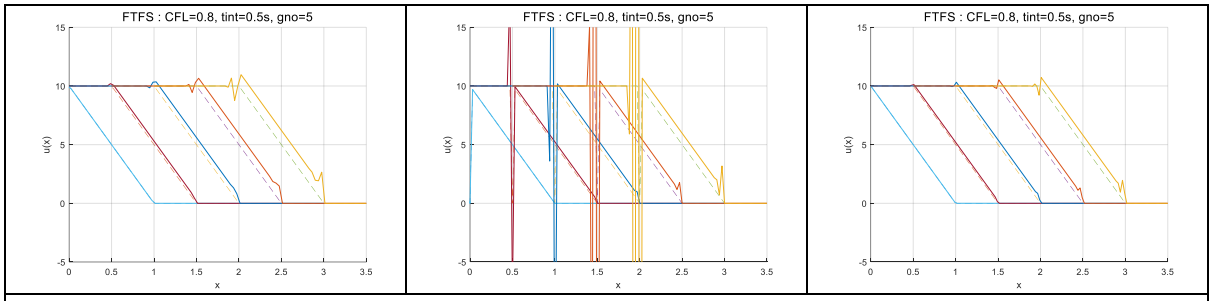
Crank-Nicolson Method의 경우에도 Euler's BTCS Method와 마찬가지로 $CFL = 1.0$일 때 오차가 거의 없었고 CFL의 변화에 따라 오차가 발생하였다. 안정성 또한 CFL 값과 iteration에 상관없이 발산하지 않는 안정적인 해를 구할 수 있었다. Euler's BTCS Method와 마찬가지로 Crank-Nicolson Method를 사용하면 Unconditionally Stable한 해를 찾을 수 있다

4. 결론

Explicit Method인 Euler's FTFS Method와 FTCS Method를 사용할 때에는 time step을 줄여 iteration을 많이 하면 Unconditionally Untable하여 CFL 값과 관계없이 해가 발산하는 것을 확인할 수 있었고, 높은 CFL 값에서 적은 iteration에서도 해가 발산하기 시작하는 것을 확인할 수 있었다. First-Order Upwind Differencing Method를 사용할 때에도 Conditionally Stable하여 $CFL > 1$ 일 때 해가 발산하는 것을 확인할 수 있었다.

하지만 Implicit Method를 사용할 때에는 Euler's BTCS Method와 Crank-Nicolson Method 모두 Unconditionally Stable하여 CFL과 iteration에 상관없이 안정적으로 수렴하는 것을 확인할 수 있었다. 두 방법 모두 $CFL = 1$ 일 때 정확한 해를 나타내었다.

두 방법 모두 격자의 개수를 늘렸을 때 더 정확한 해를 나타내는 것을 확인할 수 있었다. 하지만 일부 Explicit Method의 경우 특정 격자의 개수에서 해가 심하게 진동하는 것을 확인할 수 있었는데 그래프는 다음과 같다. 일정한 $CFL, tint, gno$에서 $gridn = 80, 100, 120$으로 증가시켰을 때 FTFS Method에서 확인할 수 있는 그래프이다.



〈 Fig. 31 - FTFS Method, $CFL = 0.8, tint = 0.5s, gno = 5$ / $gridn = 80, 100, 120$ 〉

위와 같은 현상은 특정한 error가 발생하여 적은 iteration에서도 발산한다고 유추해볼 수는 있지만 정확한 설명을 위해서는 추가적인 연구가 필요할 것이다.

Explicit Method를 사용하면 $CFL = a\frac{\Delta t}{\Delta x}$ 을 높게 설정하지 못하기 때문에 작은 시간 간격을 설정하여야 해가 발산하지 않게 되어 계산량이 Implicit Method보다 많아지게 되며 해석 속도가 늦어지게 된다. 하지만 time accuracy가 중요한 비정상 유동 등의 해석에는 시간 간격을 늘리게 되면 해의 정확도가 낮아지므로 Implicit Method보다 2차 공간의 정확도를 가지는 Explicit Method를 사용하는 것이 유리하다.