Haobo Zhao
Numerical Methods HW3 Answer Sheet

1 (a) The Exact Solution

The problem shown is 1-dimensional heat-transfer equation, by using separate variables method, the Exact Solution is showing below:

$$\text{Exact Soln: } u(t,x) = e^{-m^2 t}\sin mx \tag{1}$$

(The more detailed answer is showing on appendix page)

2 (b) The scheme for Forward, Backward, and C-N method

2.1 Forward Euler Scheme

The formula of Forward Euler method is shown as following:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} \tag{2}$$

Iteration formula for the Forward Euler is showing below:

$$T_i^{n+1} = \frac{\Delta t}{\Delta x^2}\left(T_{i+1}^n + T_{i-1}^n\right) + \left(1 - \frac{2\Delta t}{\Delta x^2}\right)T_i^n \tag{3}$$

2.2 Backward Euler Scheme

Formula for Backward Euler:

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{T_{i+1}^{n+1} - 2T_i^{n+1} - +T_{i-1}^{n+1}}{\Delta x^2} \tag{4}$$

For Backward Euler, the scheme could be derived as:

$$\begin{bmatrix} 1+2r & -r & & & & \\ -r & 1+2r & -r & & 0 & \\ & -r & \dots & \dots & & \\ & & \dots & \dots & -r & \\ & 0 & & \dots & 1+2r & -r \\ & & & & -r & 1+2r \end{bmatrix} \begin{bmatrix} T_1^{n+1} \\ T_2^{n+1} \\ \dots \\ \dots \\ T_{xm-2}^{n+1} \\ T_{xm-1}^{n+1} \end{bmatrix} = \begin{bmatrix} T_1^n \\ T_2^n \\ \dots \\ \dots \\ T_{xm-2}^n \\ T_{xm-1}^n \end{bmatrix} \tag{5}$$

For simpify notation, could show

$$\{A\}\left[T^{n+1}\right] = [T^n]$$

2.3 Central Euler Scheme (Crank-Nicolson method)

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{1}{2}\left(\frac{T_{i+1}^n - 2T_i^n + T_{i-1}^n}{\Delta x^2} + \frac{T_{i+1}^{n+1} - 2T_i^{n+1} - +T_{i-1}^{n+1}}{\Delta x^2}\right) \tag{6}$$

$$\begin{bmatrix} 2(1+r) & -r & & & \\ -r & 2(1+r) & -r & 0 & \\ & -r & \dots & \dots & \\ & & \dots & \dots & -r \\ & 0 & & \dots & 2(1+r) & -r \\ & & & & -r & 2(1+r) \end{bmatrix} \begin{bmatrix} T_1^{n+1} \\ T_2^{n+1} \\ \dots \\ \dots \\ T_{xm-2}^{n+1} \\ T_{xm-1}^{n+1} \end{bmatrix} = \begin{bmatrix} 2(1-r) & r & & & \\ r & 2(1-r) & r & 0 & \\ & r & \dots & \dots & \\ & & \dots & \dots & r \\ & 0 & & \dots & 2(1-r) & r \\ & & & & r & 2(1-r) \end{bmatrix} \begin{bmatrix} T_1^n \\ T_2^n \\ \dots \\ \dots \\ T_{xm-2}^n \\ T_{xm-1}^n \end{bmatrix} \tag{7}$$

Same, the scheme could be simply show as:

$$\{A\}\left[T^{n+1}\right] = \{B\}\left[T^n\right]$$

where the matrix A and B is showing above.

2.4 TDMA Solver

We discussed Forward Euler, Backward Euler, and Crank-Nicolson methods above, and get their scheme. However, Backward Euler and C-N methods are implicit scheme, where matrix A on the left part cannot be easily solve by using its Inverse (epically for big matrix with millions points). The only choice that is obvious is Gauss Elimination.

Using Gauss Elimination to solve matrix equation can be simplify to certain steps. However, for the three-tridiagonal line matrix like this, Gauss Elimination can be simplify to using only three lines to operate.

The TDMA solver algorithm is showing below:

```
def TDMA(a, b, c, D):  # TDMA solver

    import copy
    Do = copy.deepcopy(D)
    ao = copy.deepcopy(a)
    bo = copy.deepcopy(b)
    co = copy.deepcopy(c)

    Lx = len(D)
    co[1] = co[1] / bo[1]
    Do[1] = Do[1] / bo[1]
    for nx in range(2, Lx-1):
      bo[nx] -= co[nx - 1] * ao[nx]
      Do[nx] -= Do[nx - 1] * ao[nx]
      co[nx] = co[nx] / bo[nx]
      Do[nx] = Do[nx] / bo[nx]
    for nx in reversed(range(1, Lx-1)):
      Do[nx] -= co[nx] * Do[nx + 1]
    return Do
```

Figure 1: TDMA Solver code

For backward Euler method and C-N method, we can notice their matrix A are different. Also, it is obvious that the C-N method also need us to first deal with matrix B. Thus, the Solver for Backward Euler and C-N are different. However, they can use same TDMA solver by input different a, b, c, d.

(The more detailed Algorithm is showing on Appendix page)

2.5 Backward Euler Solver

The Backward Euler Solver is showing below:

```
40
41 def solver_BE(Lx, Lt, r, Tbe):
42     a_be = [-r        for _ in range(0, Lx )]
43     b_be = [1 + 2 * r for _ in range(0, Lx )]
44     c_be = [-r        for _ in range(0, Lx )]
45     for nt in range(1, Lt+1):
46         Dbe = Tbe[nt - 1].copy()
47         Tbe[nt] = TDMA(a_be, b_be, c_be, Dbe)
48     return Tbe
49
```

Figure 2: Backward Euler Solver code

2.6 Crank-Nicolson Solver

```
50
51 def solver_CN(Lx, Lt, r, Tcn):
52     a_cn = [-r            for _ in range(0, Lx )]
53     b_cn = [(2 * (1 + r)) for _ in range(0, Lx )]
54     c_cn = [-r            for _ in range(0, Lx )]
55     a1 = [r             for _ in range(0, Lx )]
56     b1 = [(2 * (1 - r)) for _ in range(0, Lx )]
57     c1 = [r             for _ in range(0, Lx )]
58
59     for nt in range(1, Lt+1):
60         Dcn = Tcn[nt -1].copy()
61
62         Dcn[1] = b1[1] * Tcn[nt -1][1] + c1[1] * Tcn[nt -1][2]
63         for nx in range(2, Lx - 1):
64             Dcn[nx] = a1[nx] * Tcn[nt -1][nx - 1] + b1[nx] * Tcn[nt -1][nx] + c1[nx] * Tcn[nt -1][nx + 1]
65         Dcn[Lx-1] = a1[Lx-1]*Tcn[nt -1][Lx-2] + b1[Lx-1]*Tcn[nt -1][Lx-1]
66         Tcn[nt] = TDMA(a_cn, b_cn, c_cn, Dcn)
67     return Tcn
```

Figure 3: C-N method Solver code

3 (c) m=2, $\triangle x$=2π/20 and r=1/3

3.1 Plot Compare Numerical and Exact Solutions

Apply Solvers for Forward Euler, Backward Euler, and Crank-Nicolson, and Exact Solution, the figures is showing below:

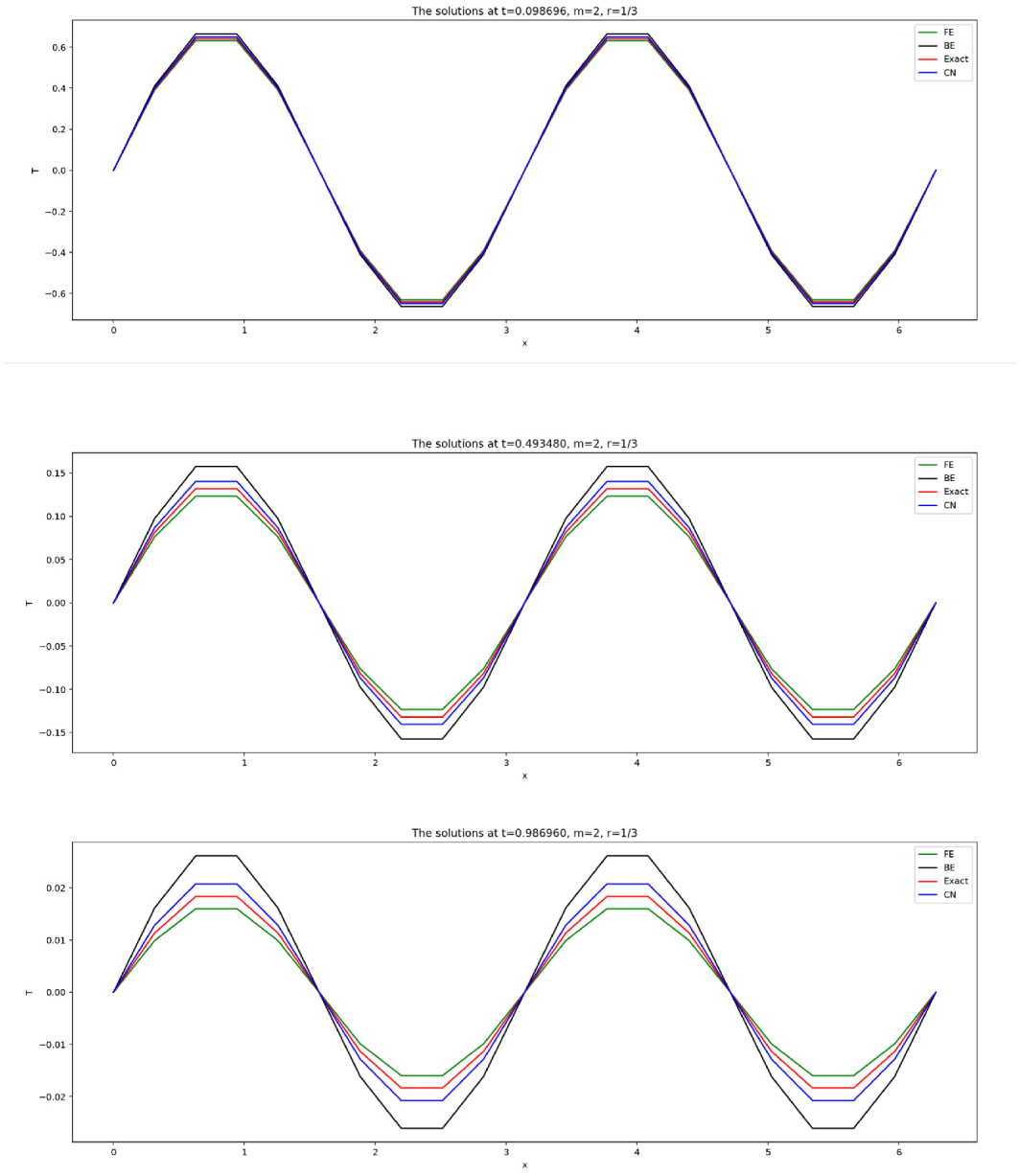


Figure 4: figures for different methods

It shows during time, the error from numerical solutions to exact solution is getting bigger, while the Crank-Nicolson method and Forward Euler method is more closer to the exact solution than others.

3.2 Errors and Characteristics

3.2.1 Error Scale

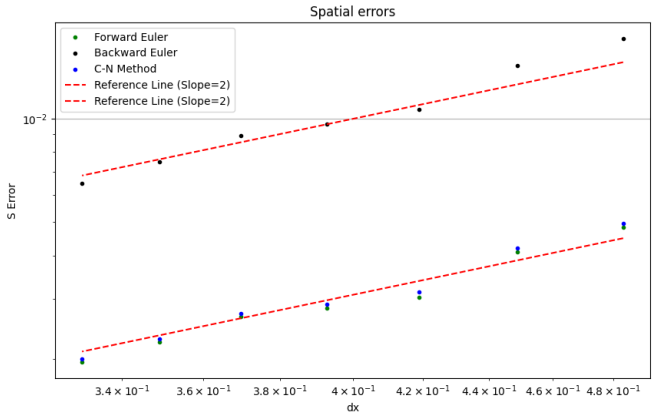


Figure 5: log-log plot for error increasing among dx

The figure could show error growing is $O\left(\triangle x^2\right)$, as its plot have slop 2

3.2.2 VN stability analysis

Among Von-Neumann stability analysis, the error could be express as e^{ikx}, while the exact solution is the numerical solution minus error, and the numerical solution fulfill the equation requirement, it turned out error fulfill our differential equation. Thus, the error of different methods could be shown, as the absolute value of error amplification should never greater than 1, the VN stability analysis shows its registration for different methods:

For the Forward Euler method, is:

$$|1 + 2r\left(cos\left(\beta_k\right) - 1\right)| < 1$$

For Backward Euler method, is:

$$\left|\left[1 + 2r\left(1 - cos\left(\beta_k\right)\right)\right]^{-1}\right| < 1$$

For Crank-Nicolson method, is:

$$\left|\frac{1 - r\left(1 - cos\left(\beta_k\right)\right)}{1 + r\left(1 - cos\left(\beta_k\right)\right)}\right| < 1$$

It could be seen for both Backward Euler and Crank-Nicolson method, the Von-Neumann Stability limit is always fulfilled, that its implicit method's unconditionally stable characterization.

For Forward Euler method, it turns out to full fill Von-Neumann Stability limit , $r \leqslant \frac{1}{2}$ is its requirement.

3.2.3 If r > 1/2

If r is larger than 1/2, we known it won't influence implicit scheme, only different is Forward Euler method.

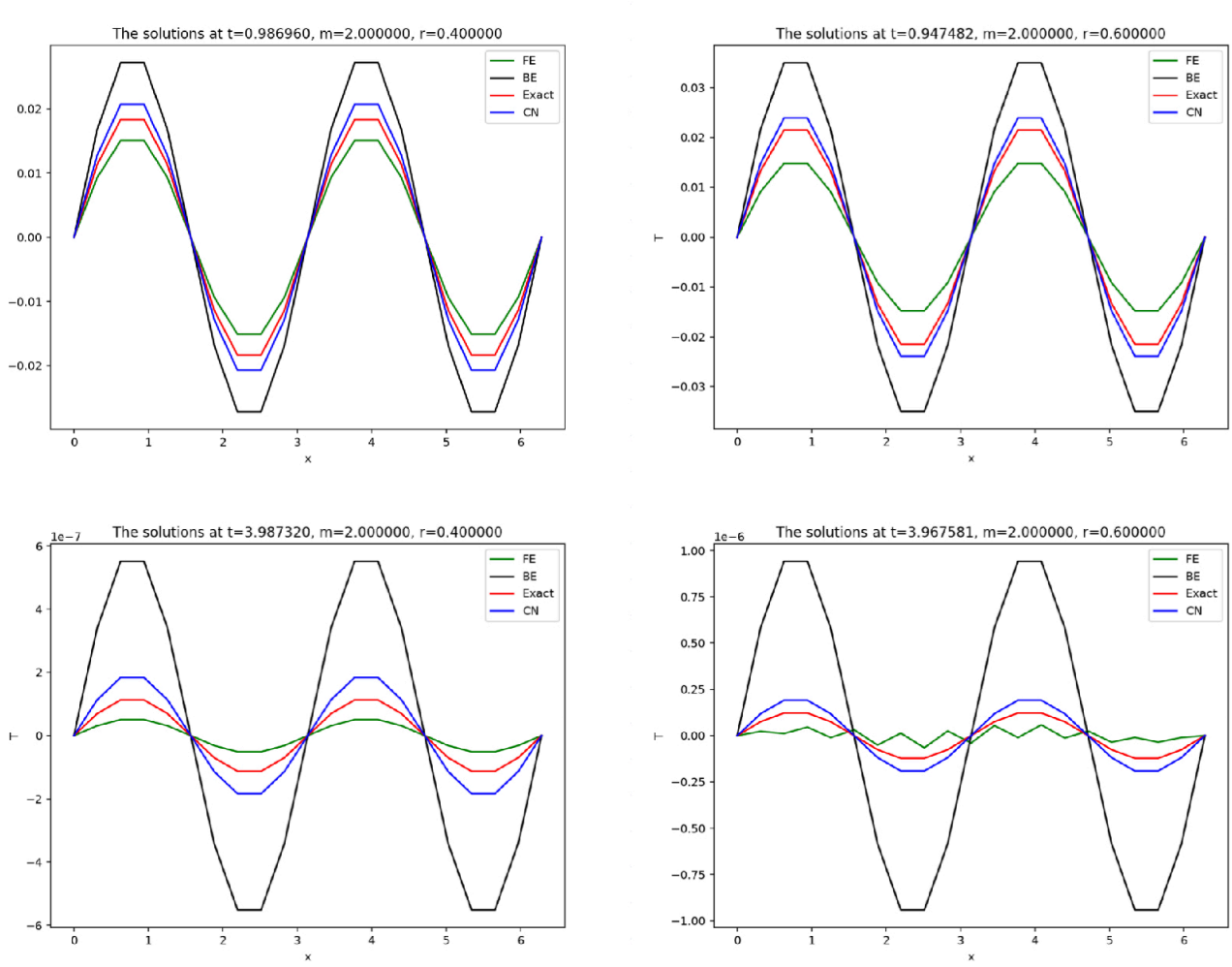


Figure 6: left part is r=0.4, right part is r=0.6

For the lower right figure, it is obvious that the Forward Euler method, the solution becomes unstable.

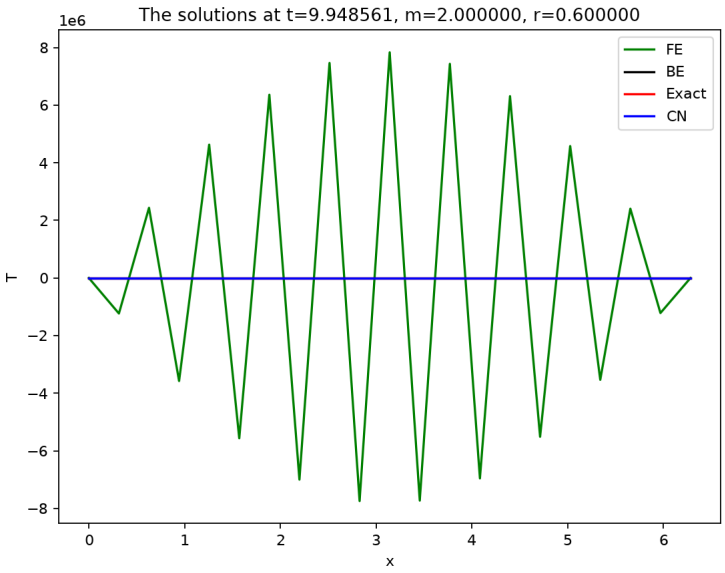


Figure 7: t=10,r=0.6, clearly unstable for Forward Euler

4 (d) $\triangle x$=2π/20 and r=0.5, and for m=3, 5, and 7.

4.1 The Plot

4.1.1 The Solutions at t=0.5s

For time = 0.5s, the solutions are showing below:

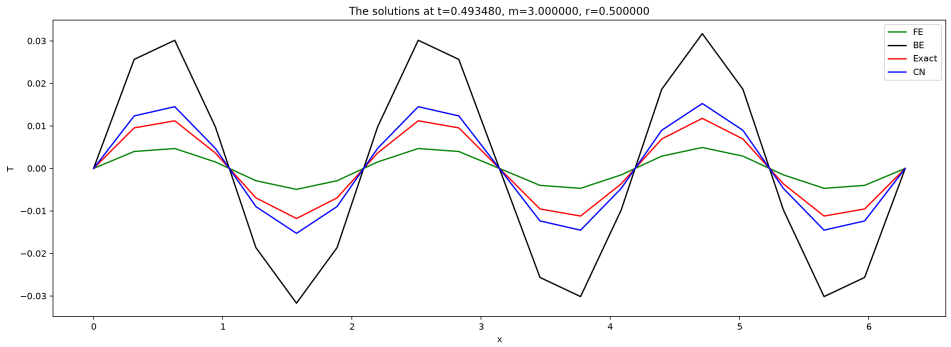


Figure 8: t = 0.5, m =3, r = 0.5

It can be seen on this situation, Backward Euler and Forward Euler have the low accuracy, while C-N method is most close to exact solution.

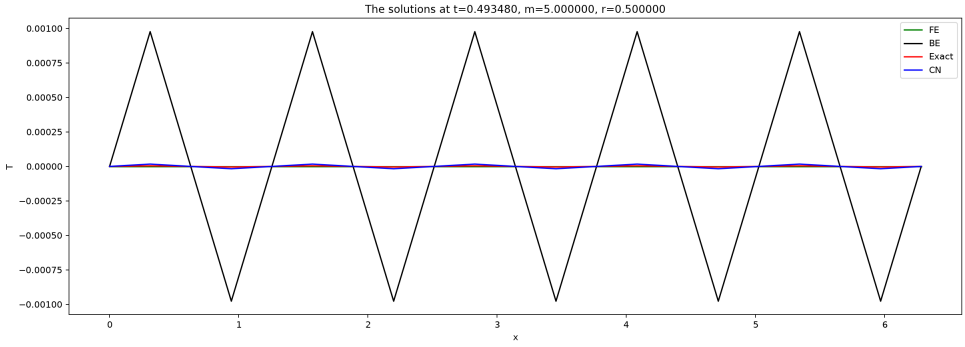


Figure 9: t = 0.5, m =5, r = 0.5

It can be seen on this situation, Backward Euler have the lowest accuracy.

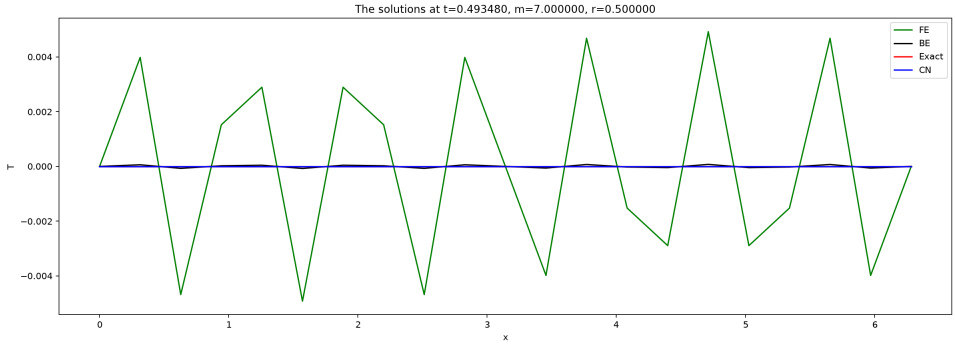


Figure 10: t = 0.5, m =7, r = 0.5

It can be seen on this situation, Forward Euler turns out have the lowest accuracy, while C-N method is still most accurate to exact solution.

4.1.2 The numerical solutions at t=0.5s compare to t=0.1s

Compare t=0.5 to t=0.1, the plots is showing below:

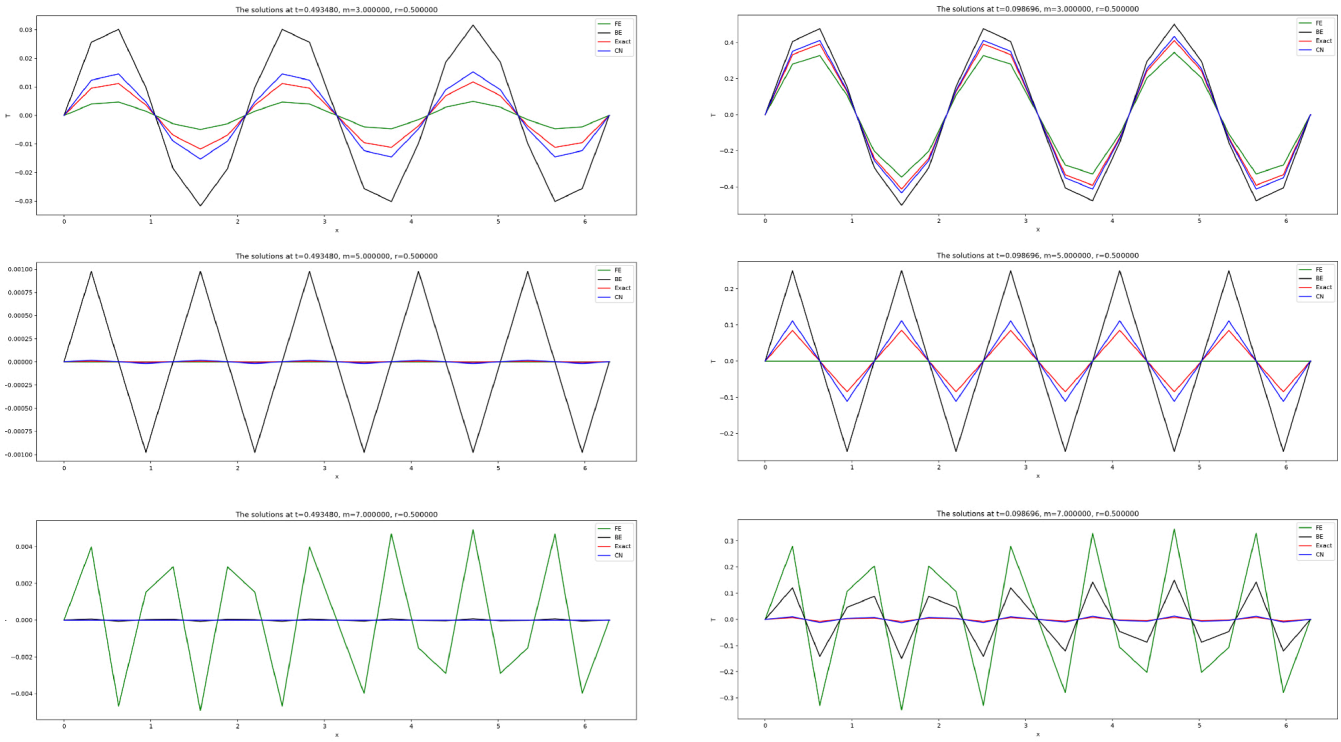


Figure 11: Comparison of t=0.5 and t=0.1

It is clear that the C-N methods is most close to the exact solution, Backward Euler is the most inaccurate numerical solution for m=3 and m=5, and Forward Euler solution almost blow up at m=7, r = 0.5.

The comparison of t=0.5 and t=0.1 is showing much more clear that the solution we explained above.

4.2 Amplification factor Analysis

For exact solution, the Amplification factor G is showing below:

$$G_k^e = e^{-r\beta_k^2}$$

For the Forward Euler method, G is:

$$G_k = 1 + 2r\left(cos\left(\beta_k\right) - 1\right)$$

For Backward Euler method, G is:

$$G_k = \left[1 + 2r\left(1 - cos\left(\beta_k\right)\right)\right]^{-1}$$

For Crank-Nicolson method, G is:

$$G_k = \frac{1 - r\left(1 - cos\left(\beta_k\right)\right)}{1 + r\left(1 - cos\left(\beta_k\right)\right)}$$

While r=0.5, m=3/5/7, the Amp number G for each situation can be showing below:

m	3	5	7
β_k	0.94	1.57	2.19
G(EX)	0.64138	0.29121	0.08909
G(FE)	0.58778	1.11022e-16	-0.58778
G(BE)	0.70811	0.50000	0.38643
G(CN)	0.65822	0.33333	0.11489

Could be seen at m=3, Bk=0.94, showing in the table and the first line of the plot, Amplification Factor from Max to Min is G(BE) >G(CN) >G(EX) >G(FE), where CN is most close to the Exact solution.

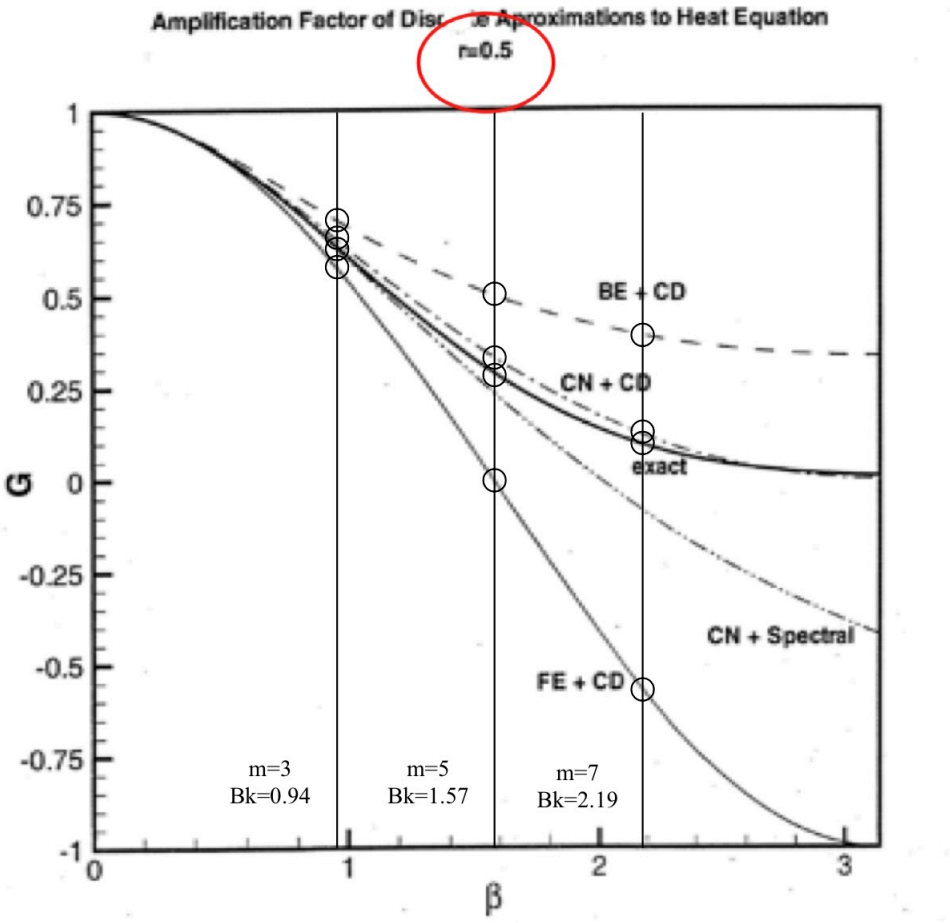


Figure 12: Amp Figure for EX, FE, BE, CN

We also can notice at m=5, Bk=1.57, showing in the table and the second line of the plot, Amplification Factor from Max to Min is still BE,CN,EX,FE, where CN is most close to the Exact solution,

though the difference between numerical solutions and Exact solutions of Amplification Number is much larger than m=3. Also need to notice that in this case, Amplification Number of Forward Euler is pretty close to 0, where is also shown in the solution pictures before.

At m=7, Bk=2.19, showing in the table and the third line of the plot, Amplification Factor from Max to Min is still BE,CN,EX,FE, where CN is most close to the Exact solution, also need to notice at this case, FE Forward Euler method's Amp Number is smaller than 0.

5 Appendix

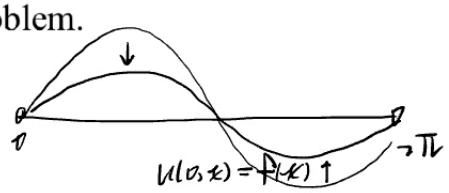
5.1 Exact Solution

Consider the unsteady conduction problem

$$u_t = u_{xx} \text{ for } 0 \le x \le 2\pi \text{ given } u(0,t) = u(2\pi,t) = 0 \text{ and } u(x,0) = \sin(mx).$$

(a) Derive the exact solution to this problem.

$\partial_t u = \partial_{xx} u$



Seperation variables:

$u = u(t,x) = A(t)\,B(x)$

$B(x)\frac{dA(t)}{dt} = A(t)\frac{d^2B(x)}{dx^2}$

$\frac{1}{A(t)}\frac{dA(t)}{dt} = \frac{1}{B(x)}\frac{d^2B(x)}{dx^2} = F(t) = G(x) = k$

$\begin{cases} \frac{dA(t)}{A(t)} = k\,dt \rightarrow A(t) = Ce^{kt} \\ \frac{d^2B(x)}{B(x)dx^2} = k \end{cases}$

Try $B(x) = \sum_n C_n e^{nx}$

for each n: $(n)^2 = k$, $k = n^2$

$n = \sqrt{k}$ or $n = -\sqrt{k}$

$B(x) = C'_k e^{\sqrt{k}x} + C'_{-k} e^{-\sqrt{k}x}$

$u(t,x) = e^{kt}(C_k e^{\sqrt{k}x} + C_{-k} e^{-\sqrt{k}x}) = e^{kt}(\alpha\cos(\frac{\sqrt{k}}{1})x + \beta\sin(\frac{\sqrt{k}}{1})x)$

IC: $u(0,x) = \sin mx \rightarrow \alpha = 0,\ \frac{\sqrt{k}}{1} = m,\ \beta = 1$

$k = (1m)^2 = -m^2$

$u(t,x) = e^{-m^2t}\sin mx$

BC: $u(t,0) = u(t,2\pi) = 0\ \forall$

Exact Soln: $u(t,x) = u(x,t) = e^{-m^2t}\sin mx$

5.2 Backward Euler and C-N Scheme

(1) Forward Euler: $\frac{T_i^{n+1}-T_i^n}{\Delta t}=\frac{T_{i+1}^n-2T_i^n+T_{i-1}^n}{\Delta x^2}$

Iteration formula: $T_i^{n+1}=\frac{\Delta t}{\Delta x^2}(T_{i+1}^n+T_{i-1}^n)+(1-\frac{2\Delta t}{\Delta x^2})T_i^n$

Backward Euler: $\frac{T_i^{n+1}-T_i^n}{\Delta t}=\frac{T_{i+1}^{n+1}-2T_i^{n+1}+T_{i-1}^{n+1}}{\Delta x^2}$

Scheme: $(1+\frac{2\Delta t}{\Delta x^2})T_i^{n+1}-\frac{\Delta t}{\Delta x^2}(T_{i-1}^{n+1}+T_{i+1}^{n+1})=T_i^{(n)}$

let $\frac{\Delta t}{\Delta x^2}=r$, then $(1+2r)T_i^{n+1}-r(T_{i-1}^{n+1}+T_{i+1}^{n+1})=T_i^n$

Scheme BC:

- for $n=0$ $(1+2r)T_i^1-r(T_{i-1}^1+T_{i+1}^1)=T_i^0$
- for $i=0$: $T_0=0=T(t,0)$
- for $i=i_{max}$: $T_{imax}=T(t,2\pi)=0$
- for $i=1$: $(1+2r)T_1^{n+1}-r(\cancel{T_0^{n+1}}+T_2^{n+1})=T_1^n$
- for $i=i_{max}-1=im-1$: $(1+2r)T_{im-1}^{n+1}-r(T_{im-2}^{n+1}+\cancel{T_{im}^{n+1}})=T_{im-1}^n$

Scheme:

$$\begin{bmatrix}1+2r & -r & 0 & & \\ -r & \ddots & \ddots & & \\ & \ddots & \ddots & -r & \\ & & -r & 1+2r & \\ 0 & & & & \end{bmatrix}\begin{bmatrix}T_1\\ \uparrow\\ T_i\\ \downarrow\\ T_{im-1}\end{bmatrix}^{n+1}=\begin{bmatrix}T_1\\ \uparrow\\ T_i\\ \downarrow\\ T_{im-1}\end{bmatrix}^n$$

$$[A]\{T^{n+1}\}=\{T^n\}$$

$$\{T^{n+1}\}=[A]^{-1}\{T^n\}$$

C-N Method: $\frac{T_i^{n+1}-T_i^n}{\Delta t}=\frac{1}{2}(\frac{T_{i+1}^{n+1}-2T_i^{n+1}+T_{i-1}^{n+1}}{\Delta x^2}+\frac{T_{i+1}^n-2T_i^n+T_{i-1}^n}{\Delta x^2})$

$r=\frac{\Delta t}{\Delta x^2}$ $\quad 2(1+r)T_i^{n+1}-r(T_{i+1}^{n+1}+T_{i-1}^{n+1})=2(1-r)T_i^n+r(T_{i+1}^n+T_{i-1}^n)$

Scheme BC:

- $i=0$: $T_0=0$
- $i=1$: $2(1+r)T_1^{n+1}-r(\cancel{T_0^{n+1}}+T_2^{n+1})=2(1-r)T_1^n+r(\cancel{T_0^n}+T_2^n)$
- $i=i_{max}=im$: $T_{im}=0$
- $i=i_{max}-1=im-1$: $2(1+r)T_{im-1}^{n+1}-r(T_{im}^{n+1}+T_{im-2}^{n+1})=2(1-r)T_{im-1}^n+r(\cancel{T_{im}^n}+T_{im-2}^n)$

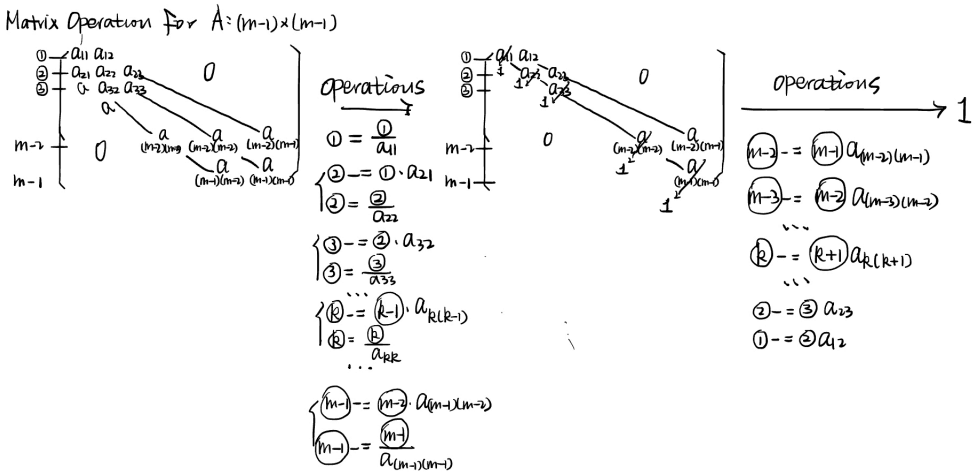
Scheme:

$$\begin{bmatrix}2(1+r) & -r & 0 & & \\ -r & \ddots & \ddots & & \\ & \ddots & \ddots & -r & \\ & & -r & 2(1+r) & \\ 0 & & & & \end{bmatrix}\begin{bmatrix}T_1\\ \uparrow\\ T_i\\ \downarrow\\ T_{im-1}\end{bmatrix}^{n+1}=\begin{bmatrix}2(1-r) & r & 0 & & \\ r & \ddots & \ddots & & \\ & \ddots & \ddots & r & \\ & & r & 2(1-r) & \\ 0 & & & & \end{bmatrix}\begin{bmatrix}T_1\\ \uparrow\\ T_i\\ \downarrow\\ T_{im-1}\end{bmatrix}^n$$

$$[B]\{T^{n+1}\}=[C]\{T^n\}$$

$$\{T^{n+1}\}=[B]^{-1}[C]\{T^n\}$$

5.3 Gauss Elimination



5.4 Solver Algorithm