Time Marching Problems Documentation

Release 0.0.1

Sayop Kim

March 01, 2016

Contents

This document is online available. The *pdf* format of this documentation may have some inappropriate image size and not-well organized order of image and its description. You are recommended to see this documentation visiting http://cfm02-gatech.readthedocs.org.

Contents:

CHAPTER 1

Code Instruction

The present project is aimed to develop a computer program for solving an unsteady solution with different numerical method. The code being used for answering all the question here is written with Python language. This program is to run with simple command:

```
$ python main.py
```

1.1 Quick instruction for running the simulation

The Python code used for this project can be cloned from *github.com* repository:

```
$ git clone https://github.com/sayop/CFM02.git
```

Once you clone the code, you will see the following set of files and directories:

```
$ sayop@reynolds:~$ ls CFM02/
docs  README.md  src
```

docs contains the document files set for the current project using *Sphinx* software. This *pdf* document is online available at: http://cfm02-gatech.readthedocs.org. The Python script for this simulation is stored in *src* folder.

Before running the simulation, you need to open the file named *input.in* using editor for example, VI on unix system:

```
$ vi input.in
```

Then, you should be able to see the following set of simulation parameters:

```
#grid dimension
iDim            6001
xmin            5
xmax            45
#flow properties
U               1
gamma           0.01
#boundary condition
phiL            0.0
phiR            0.0
#simulation setup
tStart          10.0
tEnd            40.0
maxIter         999999
Courant         0.75
implicit        0.0
```

```
#Post-Process
nIterWrite      200
xMeas1          15.0
xMeas2          25.0
```

The parameter's name above will literally tell you what every single variables indicates in the simulation. For the post-processing as requested in this project, two measurement point are specified with *xMeas1* and *xMeas2*. *nIterWrite* will write a solution plot and CSV file at speicifed interval of time integration number.

Most importantly the current project code is constructed with β method as described in the following section. The parameter *implicit* will specified β value. *Courant* number will change the time integration interval, dt.

CHAPTER 2

Results

2.1 Test Matrices

Given equation to solve is following Burger's equation with having both convection and diffusion terms:

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

One dimensional domain is set to 40 as size with $5x45$. Boundary conditions at both left and right node points are set to 0.

Consider two cases:

- CASE1: Euler equation without diffusion term: $\Gamma = 0$
- CASE2: Burger's equation with diffusion term: $\Gamma = 001$

2.1.1 Euler equation (Pure convection problem)

- Grid setups: N = 1001, 2001, 4001, 6001
- Different Couran't number conditions: Cr = 0.25, 0.5, 0.75, 0.1 for N = 4001 only
- Test methods: Euler Explicit (EE), Euler Implicit (EI), and Crank-Nicolson (CR)
- No diffusivity: Γ = 0

Stability check

1. Test on different time step

- Stability was checked on grid setup with N = 4001 with different time step conditions.
- If unstable solution appears with any inappropriate values, stability is NOT considered to be achieved.

Courant no. (dt)	EE	EI	CR
0.25 (0.0025)	X	O	O
0.5 (0.005)	X	O	O
0.75 (0.0075)	X	O	O
0.1 (0.01)	X	O	O

(O: stable, X: unstable)

2. Test on different grid resolution

- Stability was checked on different grid setup with fixed time step of dt = 0.005 [sec]
- Note that Courant number must vary for different grid size to fix the time step: Following Courant numbers in table were chosed to set the 'dt' constant.

Number of nodes (Cr)	EE	EI	CR
1001 (0.125)	X	O	O
2001 (0.25)	X	O	O
4001 (0.5)	X	O	O
6001 (0.75)	X	O	O

(O: stable, X: unstable)

2.1.2 Burger's equation (Convection + Diffusion problem)

- Grid setups: N = 1001, 2001, 4001, 6001
- Different Couran't number conditions: Cr = 0.25, 0.5, 0.75, 0.1 for N = 4001 only
- Test methods: Euler Explicit, Euler Implicit, and Crank-Nicolson
- Diffusivity: $\Gamma = 0.01$

Stability check

1. Test on different time step

- Stability was checked on grid setup with N = 4001 with different time step conditions.
- If unstable solution appears with any inappropriate values, stability is NOT considered to be achieved.

Courant no. (dt)	EE	EI	CR
0.25 (0.0025)	O	O	O
0.5 (0.005)	O	O	O
0.75 (0.0075)	X	O	O
0.1 (0.01)	X	O	O

(O: stable, X: unstable)

2. Test on different grid resolution

- Stability was checked on different grid setup with fixed time step of dt = 0.005 [sec]
- Note that Courant number must vary for different grid size to fix the time step: Following Courant numbers in table were chosed to set the 'dt' constant.

- *Peclet* number are specifed in the following table. (Note: *Peclet* number can only be defined in diffusion term contained solution)

Number of nodes (Cr / Pe)	EE	EI	CR
1001 (0.125 / 4.0)	O	O	O
2001 (0.25 / 2.0)	O	O	O
4001 (0.5 / 1.0)	O	O	O
6001 (0.75 / 0.667)	X	O	O

(O: stable, X: unstable)

2.2 Problem1 - a

Develop a finite difference algorithm for this transportation equation. Use Euler explicit, Euler implicit and Crank-Nicolsol schemes. Employ TDMA for solving implicit terms.

2.2.1 Generalized form of numerical algorithm

In this project, three different methods are supposed to be employed to solve the give transport equation. Many terms of three methods are having very similar form in common, so it is better to construct the algorithm with a parameters that switch the solution methods in specified condition. This is called β-method. In this method of solution, a user-specified parameter β is employed amd used for swtiching the solution method with proper value varying from 0 to 1.

The given equation is manipulated by employing a new function $f(\phi)$ that represents the time derivative quantity as a function of dependent variable ϕ.

$$\frac{\partial \phi}{\partial t} = f(\phi)$$

where,

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

This form of equation can then be recast with forward finite difference in time as a explicit form:

$$\phi^{n+1} = \phi^n + \Delta t f(\phi^n)$$

An implicit form of this finite difference can also be expressed as:

$$\phi^{n+1} = \phi^n + \Delta t f(\phi^{n+1})$$

Here, a new parameter β applies to give finite difference above and creates a new from of finite difference as:

$$\phi^{n+1} = \phi^n + \Delta t \left[(1-\beta) f(\phi^n) + \beta f(\phi^{n+1})\right]$$

Applying central finite difference with second order accuracy in space to $f(\phi)$ gives following:

$$f(\phi^n) = -U\frac{\phi^n_{i+1} - \phi^n_{i-1}}{2\Delta x} + \Gamma \frac{\phi^n_{i+1} - 2\phi^n_i + \phi^n_{i-1}}{\Delta x^2}$$

Manipulating the above form of solution method gives the generalized form of solution method. This can be applied to any implicit method as well as explicit method. $\beta = 0$ indicates the Euler explicit method and $\beta = 1$ reformulates the method to Euler implicit. $\beta = 0.5$ will form a Crank-Nicolson method.

In case of Euler explicit method, only one unkown appears in the left hand side at i position, so it can be solved exclusived for the new time levet at the corresponding position with known values of neighbor node information. Otherwise, three unknowns including left and right neighbors will appear so it should be resolved implicitly. For the implicit solution method, block-iterative method, TDMA for example, should be employed for resolving all the unkown values for the next time level.

With give generalized method of formulation, the code will construct A matrix for any case of β value. However, if β is specified with 0, the coefficient matrix A will be formed as a unity diagonal matrix with all zero values of off-diagonal elements. Thus, the explicit case with $\beta = 0$ does not necessarily have to perform TDMA method, which is computationally expensive. Rather, it can be exclusively solved as the present Python code is constructed for this purpose.

2.3 Problem1 - b

Plot ϕ vs. x at t = 20, 30, 40 and compare the numerical solution using the three methods with the analytical solution.

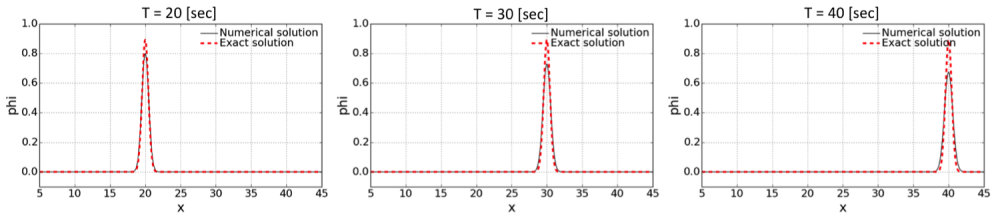
All the test case shown here were done with single set of grid size, N = 4001, and time step, dt = 0.005.

2.3.1 Euler equation (Pure convection problem)

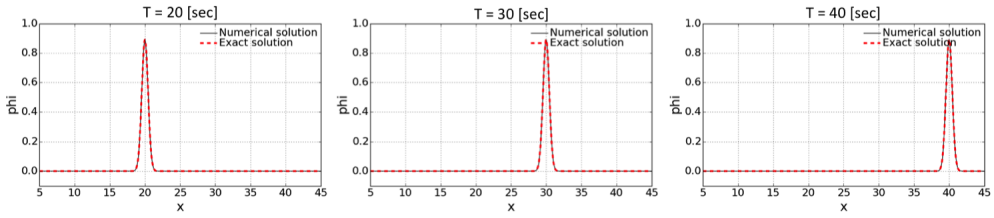
- CASE1: Euler-Explicit (EE) method

  This case was NOT successfully done with stability. Any change of time step and grid resolution didn't give stable solution because the Euler explicit for pure convection problem is unconditionally unstable.

- CASE2: Euler-Implicit (EI) method

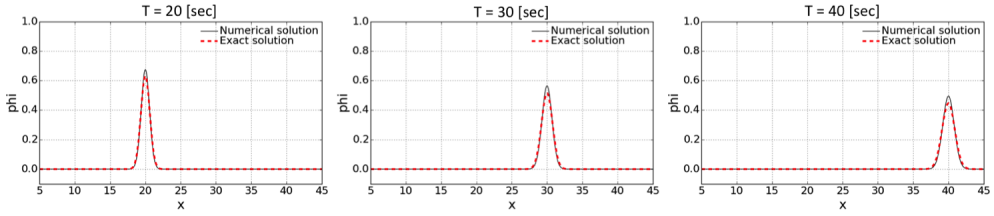


- CASE3: Crank-Nicolson (CR) method

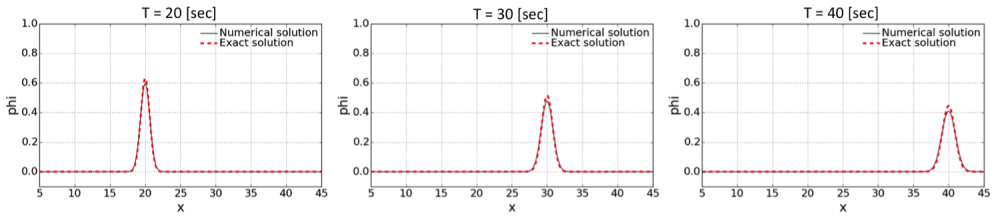


2.3.2 Burger's equation (Convection + Diffusion)

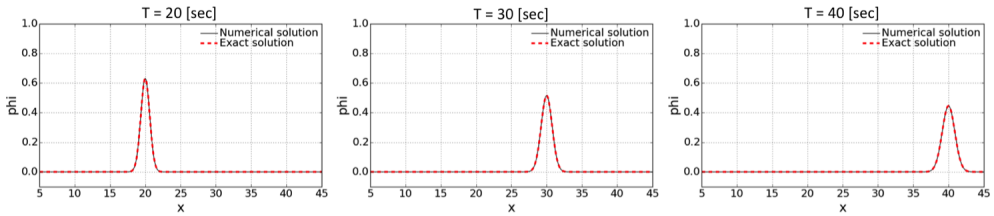
- CASE1: Euler-Explicit (EE) method



- CASE2: Euler-Implicit (EI) method



- CASE3: Crank-Nicolson (CR) method



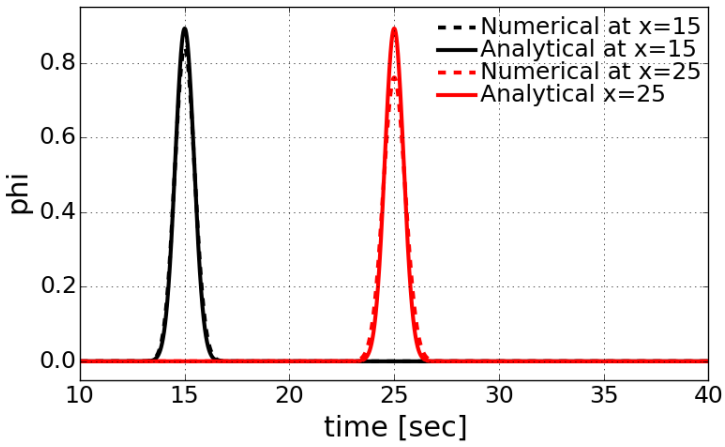
2.4 Problem1 - c

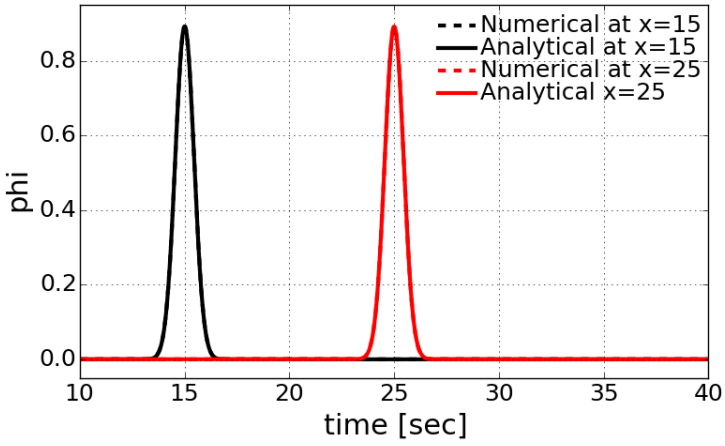
2.4.1 Euler equation (Pure convection problem)

- CASE1: Euler-Explicit (EE) method

  This case was NOT successfully done with stability.

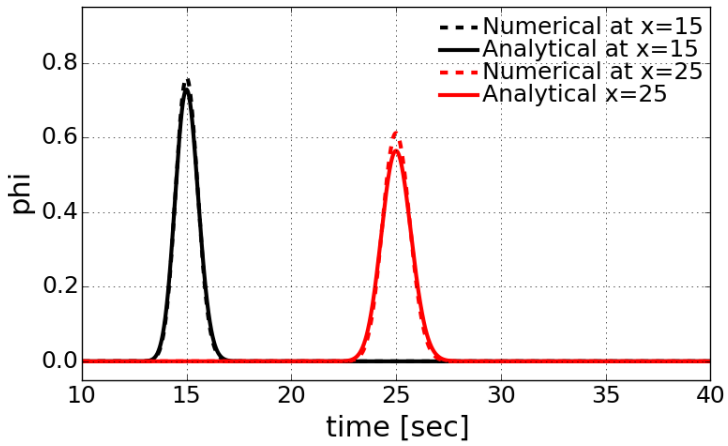
- CASE2: Euler-Implicit (EI) method
- CASE3: Crank-Nicolson (CR) method


Numerical at x=15
Analytical at x=15
Numerical at x=25
Analytical x=25
phi
0.0
0.2
0.4
0.6
0.8
10
15
20
25
30
35
40
time [sec]

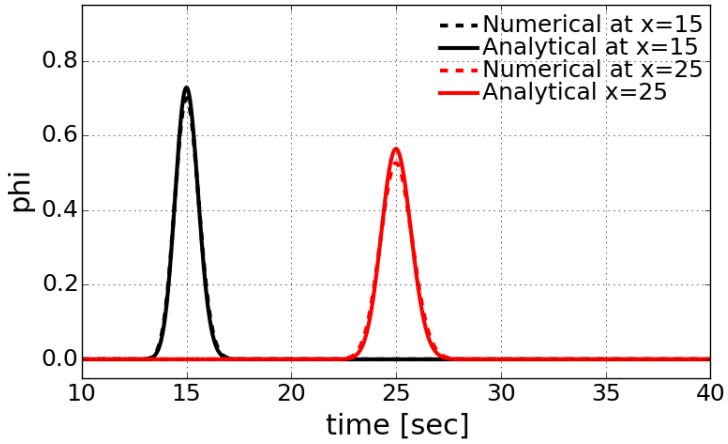

Numerical at x=15
Analytical at x=15
Numerical at x=25
Analytical x=25
phi
0.0
0.2
0.4
0.6
0.8
10
15
20
25
30
35
40
time [sec]

2.4.2 Burger's equation (Convection + Diffusion)

- CASE1: Euler-Explicit (EE) method



- CASE2: Euler-Implicit (EI) method



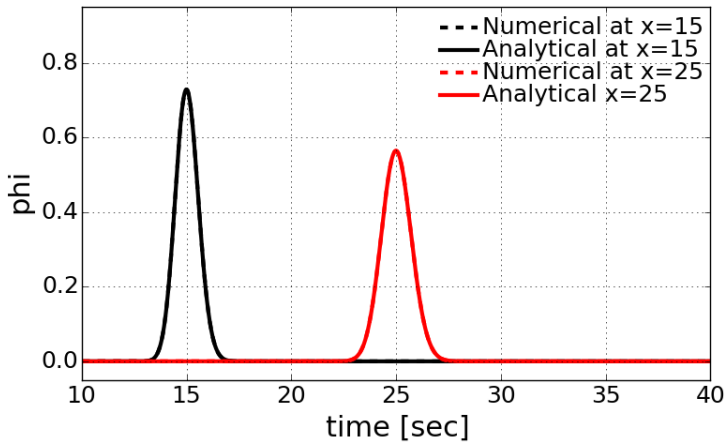
- CASE3: Crank-Nicolson (CR) method

2.5 Problem1 - d

Examine the numerical errors for different schemes as a function of the grid spacing and the integration time step. Evaluate the spatial and temporal accuracy of the numerical schemes.

Here, temporal and spatial accuracies were evaluated using following parameter:

$$P_{temporal} = \frac{log(\frac{\phi_{2t} - \phi_{4t}}{\phi_t - \phi_{2t}})}{log(2)}$$

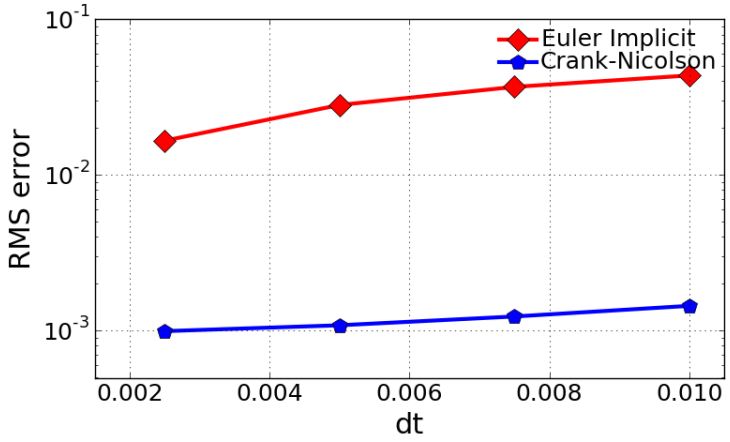


$$P_{spatial} = \frac{log(\frac{\phi_{2h} - \phi_{4h}}{\phi_h - \phi_{2h}})}{log(2)}$$

2.5.1 Euler equation (Pure convection problem)

In this test, the Euler-Explicit scheme was not employed because it is unconditionally unstable method for pure convection problem.

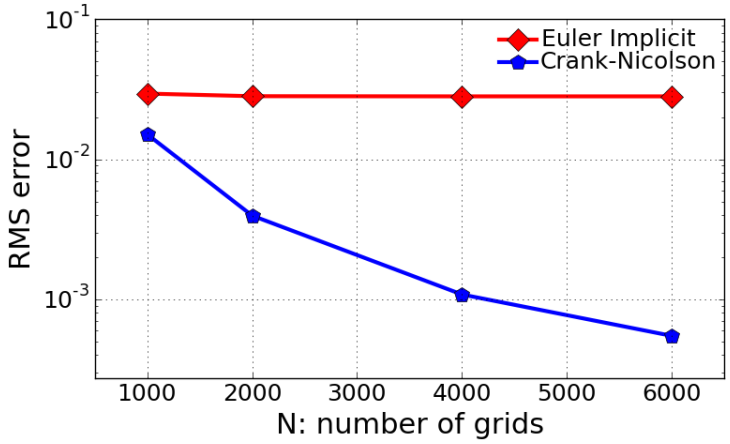
1. Errors as a function of time step, dt
    - The evaluation of time step change effect was done for single case of grid size, N = 4001.
    - Crank-Nicolson scheme gives better accuracy over the entire set of test cases than Euler implicit, because the CR scheme is a second order accuracy in both space and time.
    - In general the error drops monotonically with decrease of time step.



Courant no. (dt)	EI	CR
0.25 (0.0025)	0.0166148385052	0.00099101659914
0.5 (0.005)	0.0281716938047	0.00108091830901
0.75 (0.0075)	0.0368227088826	0.0012308579817
0.1 (0.01)	0.0436049875194	0.00144069783592

- Evaluation of temporal accuracy $P_{temporal}$ defined below with different methods:
  - $P_{temporal}$ for EI = 0.4173
  - $P_{temporal}$ for CR = 2.0007

2. Errors as a function of grid spacing
   - The evaluation of grid resolution effect on error was done for single set of time step, dt = 0.005, so different Courant number.
   - Increase of grid point number drops the error. But change of error is not much clear with Euler implicit.
   - Flat profile of errors in Euler implicit case may suggest to run the simulation with much smaller number of grid points in order to see increasing errors with less grid points.



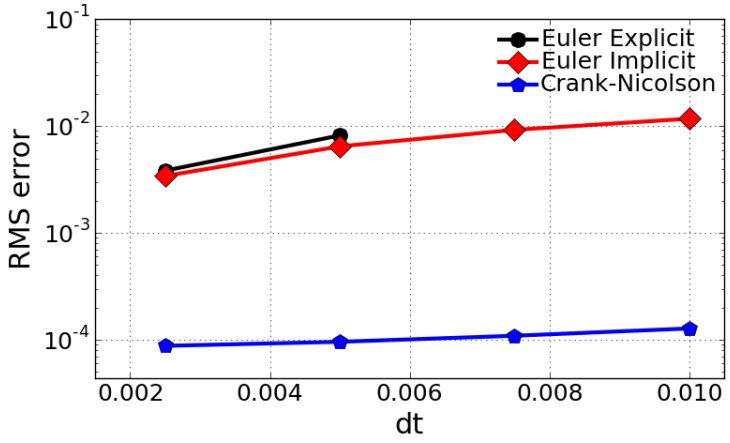
Number of nodes (Cr)	dx	EI	CR
1001 (0.125)	0.04	0.0294768440151	0.0150567999236
2001 (0.25)	0.02	0.0282487385682	0.00395484817586
4001 (0.5)	0.01	0.0281716938047	0.00108091830901
6001 (0.75)	0.00667	0.0281685389275	0.000547240822165

- Evaluation of spatial accuracy $P_{spatial}$ defined below with different methods:
  - $P_{spatial}$ for EI = 3.9946
  - $P_{spatial}$ for CR = 1.9497

2.5.2 Burger's equation (Convection + Diffusion)

In this test, only a part of Euler-Explicit cases was employed for analysis because the method was unstable beyond certain value of Courant number.

1. Errors as a function of time step, dt
    - The evaluation of time step change effect was done for single case of grid size, N = 4001.
    - As confirmed above, Crank-Nicolson scheme is much accurate compared to other two methods.
    - Euler explicit and Euler implicit are showing very comparable error values each other, almost same order of magnitude.
    - Getting error values over dt = 0.005 was not possible for Euler explicit.
    - This suggest that Euler explicit should be run with Courant number less than 0.5.

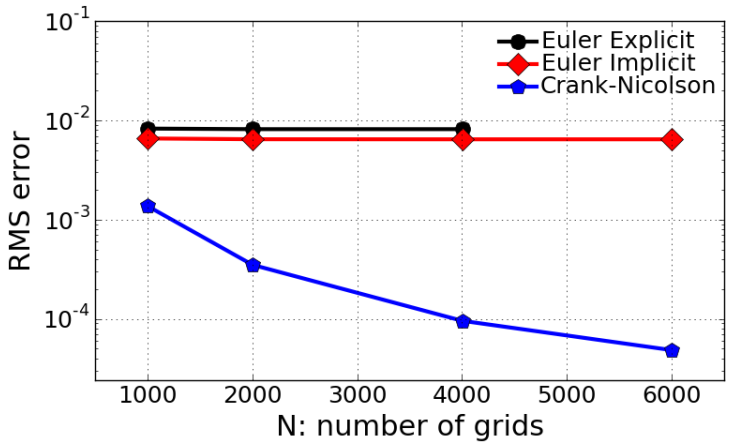


Courant no. (dt)	EE	EI	CR
0.25 (0.0025)	0.00383043947008	0.00340998441819	8.76084904269e-05
0.5 (0.005)	0.00818535015496	0.00646574830074	9.55709046347e-05
0.75 (0.0075)	inf	0.00922821563363	0.000108841001593
0.1 (0.01)	inf	0.0117416771678	0.000127423869135

- Evaluation of temporal accuracy $P_{temporal}$ defined below with different methods:
    - $P_{temporal}$ for EI = 0.7879
    - $P_{temporal}$ for CR = 2.0001

2. Errors as a function of grid spacing
    - Crank-Nicolson scheme shows dramatic change of RMS error value with different grid size as confirmed in Euler equation problem.
    - Euler explicit and Euler implicit have same order of magnitude for their accuracy. This is because these two methods have same order of accuracy in both time and space.

- N = 6001 case was failed to get stable solution.



Number of nodes (Cr)	dx	EE	EI	CR
1001 (0.125)	0.04	0.00829334418324	0.00659287238512	0.00136897946621
2001 (0.25)	0.02	0.00818432743363	0.00647642954911	0.000350398456828
4001 (0.5)	0.01	0.00818535015496	0.00646574830074	9.55709046347e-05
6001 (0.75)	0.00667	inf	0.00646469348398	4.83756513792e-05

- Evaluation of spatial accuracy $P_{spatial}$ defined below with different methods:
  - $P_{spatial}$ for EI = 3.4465
  - $P_{spatial}$ for CR = 1.9990

2.6 Problem1 - e

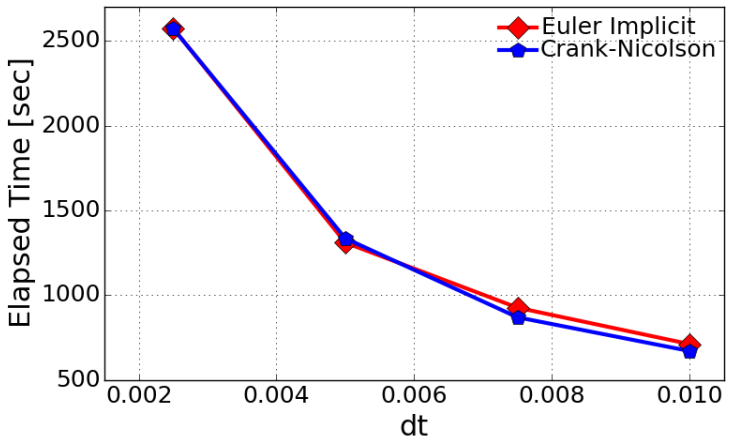
Discuss the computational time for different methods. Examine how the computational time depends on the grid resolution and evaluate the temporal complexity of the methods.

2.6.1 Euler equation (Pure convection problem)

In this test, only a part of Euler-Explicit cases was employed for analysis because the method was unstable beyond certain value of Courant number.

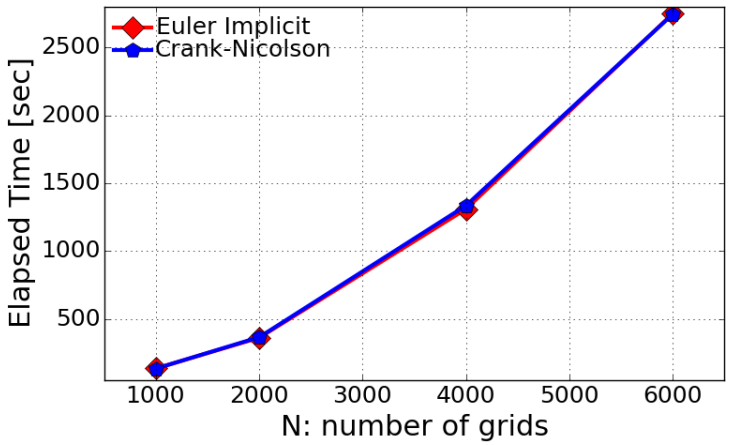
1. Computational time with different time step, dt
   - The evaluation of computational time was done for single case of grid size, N = 4001.
   - Both Euler implicit and Crank-Nicolson methods show almost equivalent levels of computational time.

- The computational time is dramatically dropping with increase of time step. This is quite natural phenomena because the bigger time increment will jump up to the targeted solution in time.



2. Computational time with different grid resolution

   - The evaluation of computational time was done for single set of time step, dt = 0.005, so different Courant number for variable grid size.
   - Using more number of grid points will consume more computational resource so longer computational time.

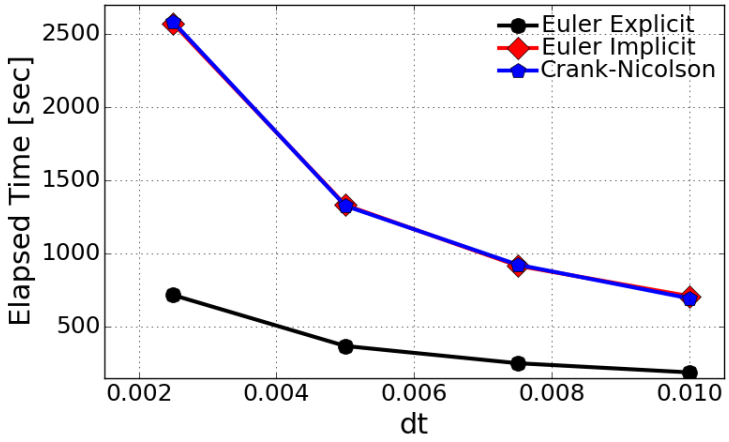


2.6.2 Burger's equation (Convection + Diffusion)

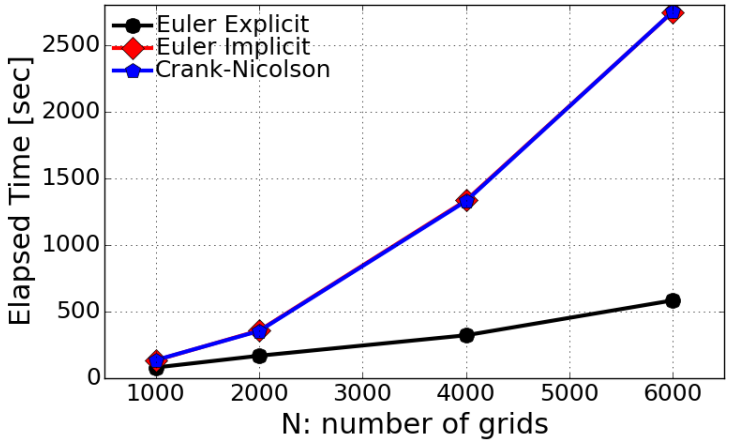
In this test, all the conditions of Euler-Explicit cases were employed for time consumption analysis regardless of instability of numerical solution. Even if instability happens in the solution, the computational time can be evaluated because the simulation was running without stopping. In that case, accuracy of simulation cannot be assessed as observed in the previous section.

1. Computational time with different time step, dt

- The evaluation of computational time was done for single case of grid size, N = 4001.
- Most interestingly, Euler explicit is quite efficient in computational time consumption.
- In contrary to both implicit schemes, explicit scheme will not need to construct matrix.
- Also it does NOT necessarily conduct any vector calculus which would basically require higher consumption of CPU time and memory storage.



2. Computational time with different grid resolution
   - The evaluation of computational time was done for single set of time step, dt = 0.005, so different Courant number for variable grid size.
   - The pattern is same as the discussions above.



2.7 Problem1 - f

Discuss stability for different methods.

- Euler explicit method:
  - Euler explicit is conditionally stable for Burger's equation, which is having diffusion term.

  - This diffusion term tends to smooth the numerical solution out such that some possibility of instailiby appearance is reduced.
  - However, the stability can be acquired with proper *Peclet* number criteria, $Pe \leq 2$
  - Euler explicit is necessarily unstable for Euler equation which is NOT having a diffusion term.
  - The central finite difference in Euler explicit does NOT guarantee the stability because numerical domain of influence of this scheme covers the redundant neighbor in pure convection problem.
  - This type of central finite difference creates a truncation error which makes numerical solution unstable.
- Euler implicit method:
  - This scheme is unconditionallyl stable.
  - This means any choice of dt and grid space will give stable solution set with some possibility of inaccuracy.
- Crank-Nicolson method:
  - This method of solution tends to be more stable than the Euler explicit even though it slows down the simulation.

Examine the maximum time step that leads to a stable solution.

- The maximum time step will depend on what type of solution method you use.
- Theoretically, Euler implicit will ensure you have stable solution with any time step choice if the equation is linear.
- The maximum time step you may choose for Euler explicit should be determined with consideration of *Peclet* number for Burger's equation only. Euler equation will be 100% unstable.

Which method provides the fastest solution for a given value of the numericall error?

- Euler explicit method will be fastest way of solving the problem if given equation contains the diffusion terms.
- Otherwise, for Euler equation with pure convection term, Crank-Nicolson scheme is the best solution for the faster solution rather than the Euler implicit.