# Homework 1

| | |
|---|---|
| 소유자 | Hyejun Park |
| 생성 일시 | @2025년 10월 22일 오후 12:57 |

## Exercise 1.1 1D Poisson equation

### (a) compute the exact solution of (1). Add these to the code

$$-u''(x) = f(x)$$

for $x \in (0,1), u(0) = a, u(1) = b$

(a) Given the boundary conditions a= b= 0, and the right hand side functions f(x) = 1 and f(x) = sin(pix), compute the exact solution of (1). Add these to the code

#### CASE f(x) =1

If f(x) = 1,

$$u''(x) = -1$$

Integrate :

$$u(x)' = -x + C_1$$

$$u(x) = -\frac{1}{2}x^2 + C_1 x + C_2$$

Boundary condition:

- x = 0 , u(0) = 0 → C2
- x = 1, u(1) = 1/2 → C1

Thus,

$$u(x) = -\frac{1}{2}x^2 + \frac{1}{2}x = -\frac{1}{2}x(x-1)$$

CASE **f(x) = sin(πx)**

$$-u''(x) = sin(\pi x)$$

$$u''(x) = -sin(\pi x)$$

$$u'(x) = \frac{1}{\pi}cos(\pi x) + C_1$$

$$u(x) = \frac{1}{\pi^2}(sin(\pi x)) + C_1 x + C_2$$

Boundary Conditions

- x = 0 → u(0)= 0 →C2 =0
- x =1 → $sin(\pi)/\pi^2 + C_1$
    - $sin(\pi) = 0$ → C1 = 0

Thus,

$$u(x) = \frac{sin(\pi x)}{\pi^2}$$

In code, this is represented as below :

```
    # right-hand-side function and exact solution
    if (testfunction == "const"):
        testproblem["f"] = lambda x: x * 0 + 1 # f(x) = 1
        testproblem["uexact"] = lambda x: 0.5 * x * (1 - x)  # u(x) = 1/2x(1 -
x)
    elif (testfunction == "sin"):
        testproblem["f"] = lambda x: np.sin(np.pi * x) # f(x) = sin pi x
        testproblem["uexact"] = lambda x : np.sin(np.pi * x) / (np.pi ** 2)  #
u(x) = sin pi x / pi^2
```

---

# ▼ (b) solve type == full

We want to code up the reduced system (1.21) from the lecture notes.

First, we do it the ”naive” way using a `dense matrix` representation. In the given code (solver type== "full"), the diagonal and the off-diagonals are added to a numpy (full) matrix and the system is solved with np.linalg.solve()

# Reduced System(1.21)

PDE → Au = b

Divide [0,1] to N+1 sections

→ We get N+2 dots

0, x1, x2,...., xN, xN+1 = 1 → NxN matrix

$D_c = \frac{u_{i+1} - 2ui + u_{i-1}}{dx^2}, h = \frac{1-0}{N+1}$

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{dx^2} = f(x_i)$$

$$-u_{i-1} + 2u_i - u_{i+1} = dx^2 f(x_i)$$

We can represent above in python like below :

```
rhs = dx**2 * f(x[1:-1])
# x[1:-1] : x1 ~ tail
# f(x[1:-1]) : calc f(x) at x
```

---

$$\begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & 0 \\ 0 & -1 & 2 & \dots & 0 \\ \\ 0 & \dots & - & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \dots \\ u_N \end{bmatrix} = h^2 \begin{bmatrix} f(x_1) \\ f(x_2) \\ \dots \\ f(x_N) \end{bmatrix}$$

---

```
l = -np.ones(N-1)  # left diag. (-1)
d = 2*np.ones(N)   # center diag. (2)
r = -np.ones(N-1)  # right diag. (-1)
```

so the full get_rhs_diag should look like :

```
def get_rhs_diag(f,N,x,dx):
    # create right-hand-side vector
        # x[0] = xL, x[N+1] = xR, 내부점 x[1:N]
    rhs = dx**2 * f(x[1:-1])
```

```
    # compute diagonals of matrix and store each of them in a vector,
    # which we will use later on to set up the matrix
    l = -np.ones(N-1)
    d = 2*np.ones(N)
    r = -np.ones(N-1)

    return rhs, l, d, r
```

---

Full matrix solver code
"full" : solve with dense matrix

```
    # solve with full matrix
    if solver_type == "full":


        # create full matrix
        matrix = np.zeros((N, N)) # init with 0 NxN array
        # zip : pairs elements from two lists together
        # [l,d,r] : left, center, right diagonal values
        # [-1,0,1] : offsets(position) for diagonals → 0: main diagonal, -
1: lower diagonal, 1: upper diagonal
        # np.diag(etries,offset) : add entries vectors to offset diagonals
of matrix
        for entries, offset in zip([l, d, r], [-1, 0, 1]):
            matrix += np.diag(entries, offset) # add each diagonal to the
matrix → tridiagonal matrix

        # note: the result of the last three lines can also be accomplish
ed
        # matrix = functools.reduce(
        #     lambda a, b: a + np.diag(b[0], b[1]), zip([l, d, r], [-1, 0, 1]), n
p.zeros((N, N)))

        # solve
        solution = np.linalg.solve(matrix, rhs)
```

# Visualize

- l = [-1,-1,-1,-1] ⇒ left diag.

- d = [2,2,2,2] ⇒ center diag.
- r = [-1,-,1-,1-1] ⇒ right diag.

How this code works :

1. NxN Matrix
2. matrix += np.diag(l,-1) ⇒ fill out the left diagnoals with -1
3. matrix += np.diag(l,2) ⇒ fill out the center diagonals with 2
4. matrix += np.diag(r,-1) ⇒ fill out the right diag. with -1

so it will look like this :

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & 2 & -1 \end{bmatrix}$$

---

## get_rhs_diag(f,N,x,dx)

---

# (c) Solve for the mesh res.

Run the code implemented in (b) to solve the two test problems defined in (a).

Solve for the mesh resolutions N∈[20,40,80,160,320,640,1280] and compute the maximum norm of the error maxi |ui−u(xi)|. What convergence order do you get?

## Code

### Tester

```
import numpy as np
import Poisson_template as program

testproblems = ["const", "sin"]  # choose
parameters = program.define_default_parameters()

#  2 . N values given
```

```
N_values = [20, 40, 80, 160, 320, 640, 1280]

for problem_name in testproblems:
    print(f"\nSolving test problem: {problem_name}")
    testproblem = program.get_testproblem(problem_name)

    err_max_list = []

    for N in N_values:
        # solver exec.
        err_max, _ = program.my_driver("full", testproblem, parameters,
N)
        # err_max, _ = program.my_driver("sparse", testproblem, param
eters, N)
        err_max_list.append(err_max)

    # max err
    print("N\tMax error")
    for N, err in zip(N_values, err_max_list):
        print(f"{N}\t{err:.2e}")

    # (log-log)
    err_max_array = np.array(err_max_list)
    rate = np.log(err_max_array[:-1]/err_max_array[1:])/np.log(2)
    print("Convergence rates:", rate)
```

## Result

```
Solving test problem: sin
Error in Max norm:      1.89e-04
Solved in:              4.82e-05 seconds
Error in Max norm:      4.96e-05
Solved in:              4.51e-05 seconds
Error in Max norm:      1.27e-05
Solved in:              1.05e-04 seconds
Error in Max norm:      3.21e-06
Solved in:              2.06e-04 seconds
Error in Max norm:      8.09e-07
```

```
Solved in:            6.06e-04 seconds
Error in Max norm:      2.03e-07
Solved in:            1.43e-02 seconds
Error in Max norm:      5.08e-08
Solved in:            2.56e-02 seconds

N       Max error
20      1.89e-04
40      4.96e-05
80      1.27e-05
160     3.21e-06
320     8.09e-07
640     2.03e-07
1280    5.08e-08
Convergence rates: [1.92867986 1.96412327 1.98201217 1.99099437 1.9
9549438 1.99774683]
```

→ as N increase, max error decrease

→ convergence order =2 ( $p = \frac{log(error_{i+1}/error_i)}{log(h_{i+1}/h_i)}$)

```
Solving test problem: const
Error in Max norm:      7.63e-17
Solved in:            3.04e-04 seconds
Error in Max norm:      6.80e-16
Solved in:            5.98e-05 seconds
Error in Max norm:      1.12e-15
Solved in:            9.51e-05 seconds
Error in Max norm:      1.79e-15
Solved in:            1.86e-03 seconds
Error in Max norm:      7.04e-15
Solved in:            1.07e-03 seconds
Error in Max norm:      1.38e-14
Solved in:            1.04e-02 seconds
Error in Max norm:      3.43e-14
Solved in:            3.64e-02 seconds

N       Max error
20      7.63e-17
```

```
40      6.80e-16
80      1.12e-15
160     1.79e-15
320     7.04e-15
640     1.38e-14
1280    3.43e-14
Convergence rates: [-3.15527823 -0.72514016 -0.67137725 -1.9746146
8 -0.96835437 -1.31609902]
```

→ for const : the numerical solution is essentially exact (error ≈ 0), so the computed convergence rates are meaningless or dominated by floating-point noise

## ▼ (d) add timer

Add timers around the solution step. What do you observe regarding runtime? The Gaussian elimination for a N×N matrix is in O(N3). Using the runtime, compare the cost of solving the reduced system to a Gaussian elimination: is it the same, better,
or worse? Hint: For timing, you can use, e.g., the functions t = time.time()
and
t solution = time.time() - t.

I have already added timer during solving for(c), and the result is posted on (c).

- Time increase as N increase

Comparison Part

- Gaussian elimination for a N×N matrix is in O(N3)
- In this code, because we use `solution = np.linalg.solve(matrix, rhs)` , the run time is still O(N3). Therefore, it is the same. However, if we change this to Thomas algorithm we roughly get O(N) code.

## ▼ (e) solve sparse

We want to examine the question: Does the usage of sparse matrices reduce the time to solution? For this task, you need to implement the case solver type == "sparse".

A useful package for sparse linear algebra in Python is scipy. From here, you could use, e.g., the functions scipy.sparse.csr matrix(), scipy.sparse.diags(), and
scipy.sparse.linalg.spsolve().

How do the timings compare with the approach
using the dense matrix?

### ▼ Code

```
elif solver_type == "sparse":
    # create sparse tridiagonal matrix
    offsets = [-1, 0, 1]
    data = [l, d, r]
    sparse_matrix = diags(data, offsets, format='csr')

    # start timer
    t0 = time.time()

    # solve sparse system
    solution = spsolve(sparse_matrix, rhs)

    # end timer
    t_solution = time.time() - t0
```

### ▼ Output

- compared to "full"
- For small to moderate N, the runtime of the sparse solver is comparable to the dense solver.
- However, as N grows larger, the sparse solver is expected to outperform the dense one due to reduced computational complexity and memory usage

```
Solving test problem: sin
Error in Max norm:      1.89e-04
Solved in:              3.39e-05 seconds
Error in Max norm:      4.96e-05
Solved in:              2.57e-05 seconds
```

```
Error in Max norm:      1.27e-05
Solved in:              3.08e-05 seconds
Error in Max norm:      3.21e-06
Solved in:              4.08e-05 seconds
Error in Max norm:      8.09e-07
Solved in:              7.87e-05 seconds
Error in Max norm:      2.03e-07
Solved in:              1.21e-04 seconds
Error in Max norm:      5.08e-08
Solved in:              2.18e-04 seconds

N       Max error
20      1.89e-04
40      4.96e-05
80      1.27e-05
160     3.21e-06
320     8.09e-07
640     2.03e-07
1280    5.08e-08
Convergence rates: [1.92867986 1.96412327 1.98201217 1.99099437
1.99549438 1.99774683]
(base) ➜  homework1 git:(main) ×
```

```
Solving test problem: const
Error in Max norm:      6.25e-17
Solved in:              8.85e-04 seconds
Error in Max norm:      7.22e-16
Solved in:              3.70e-05 seconds
Error in Max norm:      1.08e-15
Solved in:              3.79e-05 seconds
Error in Max norm:      1.64e-15
Solved in:              4.91e-05 seconds
Error in Max norm:      6.80e-15
Solved in:              7.99e-05 seconds
Error in Max norm:      1.44e-14
Solved in:              1.23e-04 seconds
Error in Max norm:      3.37e-14
Solved in:              2.36e-04 seconds
```

```
N       Max error
20      6.25e-17
40      7.22e-16
80      1.08e-15
160     1.64e-15
320     6.80e-15
640     1.44e-14
1280    3.37e-14
Convergence rates: [-3.53051472 -0.5849625  -0.59724083 -2.053
99489 -1.08572987 -1.22377896]
```

---

## Exercise 1.2 Different Quotient

▼ Definitions

### Definitions

**Forward difference:**

$$D^+ f(x) = \frac{f(x+h) - f(x)}{h}$$

**Backward difference:**

$$D^- f(x) = \frac{f(x) - f(x-h)}{h}$$

**Central difference (first derivative):**

$$D_c^1 f(x) = \frac{f(x+h) - f(x-h)}{2h}$$

**Central difference (second derivative):**

$$D_c^2 f(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

## ▼ (a) Show that $D^+D^- = D_2^c$

Compute $D^+D^-f(x)$:

Apply D-:

$$g(x) := D^-f(x) = \frac{f(x) - f(x-h)}{h}$$

Apply D+:

$$D^+(g(x)) = \frac{g(x+h) - g(x)}{h}$$

- $g(x+h) = D^-f(x+h) = \frac{f(x+h)-f(x)}{h}$
- $g(x) = D^-f(x) = \frac{f(x)-f(x-h)}{h}$

Then,

$$D^+D^-f(x) = \frac{\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}}{h}$$

$$= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

which is same the central difference

---

## ▼ (b) Show that $D_c^1 D_c^1 = D_c^2$

Compute $D_c^1 D_c^1 f(x)$:

$$g(x) := D_c^1 f(x) = \frac{f(x+h) - f(x-h)}{2h}$$

Apply $D_c^1$ :

$$D_c^1 g(x) = \frac{g(x+h) - g(x-h)}{2h}$$

- $g(x+h) = \frac{f(x+2h)-f(x)}{2h}$
- $g(x-h) = \frac{f(x)-f(x-2h)}{2h}$

Then,

$$D_c^1 D_c^1 f(x) = \frac{\frac{f(x+2h)-f(x)}{2h} - \frac{f(x)-f(x-2h)}{2h}}{2h}$$

$$= \frac{f(x+2h) - 2f(x) + f(x-2h)}{(2h)^2}$$

## (c) How can you modify $D_c^1$ to obtain $D_c^2$ in part (b)?

Use $D_c^1$ with step size 2h

$$D_c^1|_{h \to 2h} = \frac{f(x+2h) - f(x-2h)}{4h}$$

# Exercise 1.3 1D linear advection equation

## (a) Complete compute_dt

CFL Condition:

$$v = \frac{a\Delta t}{\Delta x} \leq CFL$$

Thus, dt (time gap) should be :

$$\Delta t = CFL * \frac{\Delta x}{a}$$

which will be written as :

```
def compute_dt(CFL, a, dx):
    dt = CFL * dx / a
    return dt
```

## (b) Update Codes

### update ftcs(U,a,dt,dx)

FTCS formula :

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

- index out of Bounds : U[0], U[-1]
  - we need i+1, i-1 with the formula
  - i=1 → i-1 = 0 OK
  - i=N → i+1 = N+1  OK
  - so update with U[1 :-1]

```
def update_ftcs(U, a, dt, dx):
    # 저장용 복사
    U_temp = U.copy()
    U_temp[1:-1] = U[1:-1] - (a * dt / (2*dx)) * (U[2:] - U[:-2])
    U[:] = U_temp
```

### update upwind

Upwind formula when a > 0 :

$$u_i^{n_1} = u_i^n - \frac{a\Delta t}{\Delta x}(u_i^n - u_{i-1}^n)$$

```
def update_upwind(U, a, dt, dx):
    U_temp = U.copy()
    U_temp[1:-1] = U[1:-1] - (a * dt / dx) * (U[1:-1] - U[:-2])
    U[:] = U_temp
```

## (c) Compute error

$$err_{max} = max_i|U_i - u_{exact}(x_i, t)|$$

```
def compute_error(x, time, uexact, U):
    U_exact = uexact(x, time)
    err_max = np.max(np.abs(U - U_exact))
    return err_max
```

## ▼ (d) Test

$$u_0(x) = sin(2\pi x)$$

What do you observe for FTCS and for upwind? Estimate the convergence order of the upwind scheme using a convergence table.

▼ Test Code

```
"""
Run tests for 1D linear advection:
- Compare FTCS and Upwind schemes
- Check convergence for Upwind
- Investigate CFL effects
"""

import os
import numpy as np
import advection_template as adv

def main():
    # Ensure results folder exists
    if not os.path.exists("results"):
        os.makedirs("results")

    testproblem = adv.get_testproblem()

    # --- Part D: FTCS and Upwind comparison ---
    print("=== FTCS Scheme ===")
    parameters = adv.define_default_parameters()
    parameters["Nrefine"] = 2  # N = 40, 80, 160
    adv.my_driver("FTCS", testproblem, parameters, parameters["N"])

    print("=== Upwind Scheme ===")
    adv.my_driver("upwind", testproblem, parameters, parameters
["N"])

    # --- Part E: CFL influence for Upwind ---
    CFL_values = [0.8, 1.0, 1.2]
```

```
    N = 40

    for CFL in CFL_values:
        print(f"\n--- Upwind Scheme with CFL = {CFL} ---")
        parameters = adv.define_default_parameters()
        parameters["CFL"] = CFL
        parameters["Nrefine"] = 0  # single run
        adv.my_driver("upwind", testproblem, parameters, N)

    # --- Part 3: Show convergence table ---
    print("\nConvergence table stored in 'results/error.txt':")
    if os.path.exists("results/error.txt"):
        with open("results/error.txt") as f:
            print(f.read())
    else:
        print("No convergence table found. Run with Nrefine > 0.")

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

▼ result

```
base) ➜  homework1 git:(main) ✗ python advection_de_test.py
=== FTCS Scheme ===

# START PROGRAM
Taking time step 1:     update from 0.000000    to 0.020000
Taking time step 2:     update from 0.020000    to 0.040000
Taking time step 3:     update from 0.040000    to 0.060000
Taking time step 4:     update from 0.060000    to 0.080000
Taking time step 5:     update from 0.080000    to 0.100000
Taking time step 6:     update from 0.100000    to 0.120000
Taking time step 7:     update from 0.120000    to 0.140000
Taking time step 8:     update from 0.140000    to 0.160000
Taking time step 9:     update from 0.160000    to 0.180000
Taking time step 10:     update from 0.180000    to 0.200000
Taking time step 11:     update from 0.200000    to 0.220000
Taking time step 12:     update from 0.220000    to 0.240000
```

```
Taking time step 13:   update from 0.240000   to 0.260000
Taking time step 14:   update from 0.260000   to 0.280000
Taking time step 15:   update from 0.280000   to 0.300000
Taking time step 16:   update from 0.300000   to 0.320000
Taking time step 17:   update from 0.320000   to 0.340000
Taking time step 18:   update from 0.340000   to 0.360000
Taking time step 19:   update from 0.360000   to 0.380000
Taking time step 20:   update from 0.380000   to 0.400000
Taking time step 21:   update from 0.400000   to 0.420000
Taking time step 22:   update from 0.420000   to 0.440000
Taking time step 23:   update from 0.440000   to 0.460000
Taking time step 24:   update from 0.460000   to 0.480000
Taking time step 25:   update from 0.480000   to 0.500000
Taking time step 26:   update from 0.500000   to 0.520000
Taking time step 27:   update from 0.520000   to 0.540000
Taking time step 28:   update from 0.540000   to 0.560000
Taking time step 29:   update from 0.560000   to 0.580000
Taking time step 30:   update from 0.580000   to 0.600000
Taking time step 31:   update from 0.600000   to 0.620000
Taking time step 32:   update from 0.620000   to 0.640000
Taking time step 33:   update from 0.640000   to 0.660000
Taking time step 34:   update from 0.660000   to 0.680000
Taking time step 35:   update from 0.680000   to 0.700000
Taking time step 36:   update from 0.700000   to 0.720000
Taking time step 37:   update from 0.720000   to 0.740000
Taking time step 38:   update from 0.740000   to 0.760000
Taking time step 39:   update from 0.760000   to 0.780000
Taking time step 40:   update from 0.780000   to 0.800000
Taking time step 41:   update from 0.800000   to 0.820000
Taking time step 42:   update from 0.820000   to 0.840000
Taking time step 43:   update from 0.840000   to 0.860000
Taking time step 44:   update from 0.860000   to 0.880000
Taking time step 45:   update from 0.880000   to 0.900000
Taking time step 46:   update from 0.900000   to 0.920000
Taking time step 47:   update from 0.920000   to 0.940000
Taking time step 48:   update from 0.940000   to 0.960000
Taking time step 49:   update from 0.960000   to 0.980000
Taking time step 50:   update from 0.980000   to 1.000000
```

```
Have reached time tend; stop now
The PostScript backend does not support transparency; partially tra
nsparent artists will be rendered opaque.
Error in maximum norm:   4.80e-01

=== Upwind Scheme ===

# START PROGRAM
Taking time step 1:     update from 0.000000    to 0.020000
Taking time step 2:     update from 0.020000    to 0.040000
Taking time step 3:     update from 0.040000    to 0.060000
Taking time step 4:     update from 0.060000    to 0.080000
Taking time step 5:     update from 0.080000    to 0.100000
Taking time step 6:     update from 0.100000    to 0.120000
Taking time step 7:     update from 0.120000    to 0.140000
Taking time step 8:     update from 0.140000    to 0.160000
Taking time step 9:     update from 0.160000    to 0.180000
Taking time step 10:    update from 0.180000    to 0.200000
Taking time step 11:    update from 0.200000    to 0.220000
Taking time step 12:    update from 0.220000    to 0.240000
Taking time step 13:    update from 0.240000    to 0.260000
Taking time step 14:    update from 0.260000    to 0.280000
Taking time step 15:    update from 0.280000    to 0.300000
Taking time step 16:    update from 0.300000    to 0.320000
Taking time step 17:    update from 0.320000    to 0.340000
Taking time step 18:    update from 0.340000    to 0.360000
Taking time step 19:    update from 0.360000    to 0.380000
Taking time step 20:    update from 0.380000    to 0.400000
Taking time step 21:    update from 0.400000    to 0.420000
Taking time step 22:    update from 0.420000    to 0.440000
Taking time step 23:    update from 0.440000    to 0.460000
Taking time step 24:    update from 0.460000    to 0.480000
Taking time step 25:    update from 0.480000    to 0.500000
Taking time step 26:    update from 0.500000    to 0.520000
Taking time step 27:    update from 0.520000    to 0.540000
Taking time step 28:    update from 0.540000    to 0.560000
Taking time step 29:    update from 0.560000    to 0.580000
Taking time step 30:    update from 0.580000    to 0.600000
```

```
Taking time step 31:    update from 0.600000    to 0.620000
Taking time step 32:    update from 0.620000    to 0.640000
Taking time step 33:    update from 0.640000    to 0.660000
Taking time step 34:    update from 0.660000    to 0.680000
Taking time step 35:    update from 0.680000    to 0.700000
Taking time step 36:    update from 0.700000    to 0.720000
Taking time step 37:    update from 0.720000    to 0.740000
Taking time step 38:    update from 0.740000    to 0.760000
Taking time step 39:    update from 0.760000    to 0.780000
Taking time step 40:    update from 0.780000    to 0.800000
Taking time step 41:    update from 0.800000    to 0.820000
Taking time step 42:    update from 0.820000    to 0.840000
Taking time step 43:    update from 0.840000    to 0.860000
Taking time step 44:    update from 0.860000    to 0.880000
Taking time step 45:    update from 0.880000    to 0.900000
Taking time step 46:    update from 0.900000    to 0.920000
Taking time step 47:    update from 0.920000    to 0.940000
Taking time step 48:    update from 0.940000    to 0.960000
Taking time step 49:    update from 0.960000    to 0.980000
Taking time step 50:    update from 0.980000    to 1.000000
Have reached time tend; stop now
The PostScript backend does not support transparency; partially tra
nsparent artists will be rendered opaque.
Error in maximum norm:   9.40e-02
```

```
Error in maximum norm:
FTCS: 4.80e-01
Upwind: 9.40e-02
```

FTCS vs Upwind Observation

- The `FTCS scheme` shows a much **larger error** than the upwind scheme for this advection problem.
- The `Upwind scheme` is more stable and accurate for this test case.

---

# ▼ (e) CFL Observation

Solve the equation using the upwind scheme with the same initial data for N = 40 and v ∈ {0.8,1.,1.2}. What do you observe? Which behavior was expected from the lecture?

Using the same test source code from (d),

- Upwind CFL = 0.8 : max error ≈ 0.094
- Upwind, CFL = 1.0 : max error ≈ **3.77e-15**
- Upwind, CFL = 1.2 :max error ≈ 0.0988 (gets larger again)

Thus,

- CFL ≤ 1.0 → stable, small error
- CFL > 1.0 → unstable, error increases

▼ full result

```
--- Upwind Scheme with CFL = 0.8 ---

# START PROGRAM
Taking time step 1:     update from 0.000000    to 0.020000
Taking time step 2:     update from 0.020000    to 0.040000
Taking time step 3:     update from 0.040000    to 0.060000
Taking time step 4:     update from 0.060000    to 0.080000
Taking time step 5:     update from 0.080000    to 0.100000
Taking time step 6:     update from 0.100000    to 0.120000
Taking time step 7:     update from 0.120000    to 0.140000
Taking time step 8:     update from 0.140000    to 0.160000
Taking time step 9:     update from 0.160000    to 0.180000
Taking time step 10:    update from 0.180000    to 0.200000
Taking time step 11:    update from 0.200000    to 0.220000
Taking time step 12:    update from 0.220000    to 0.240000
Taking time step 13:    update from 0.240000    to 0.260000
Taking time step 14:    update from 0.260000    to 0.280000
Taking time step 15:    update from 0.280000    to 0.300000
Taking time step 16:    update from 0.300000    to 0.320000
Taking time step 17:    update from 0.320000    to 0.340000
Taking time step 18:    update from 0.340000    to 0.360000
Taking time step 19:    update from 0.360000    to 0.380000
Taking time step 20:    update from 0.380000    to 0.400000
```

```
Taking time step 21:   update from 0.400000   to 0.420000
Taking time step 22:   update from 0.420000   to 0.440000
Taking time step 23:   update from 0.440000   to 0.460000
Taking time step 24:   update from 0.460000   to 0.480000
Taking time step 25:   update from 0.480000   to 0.500000
Taking time step 26:   update from 0.500000   to 0.520000
Taking time step 27:   update from 0.520000   to 0.540000
Taking time step 28:   update from 0.540000   to 0.560000
Taking time step 29:   update from 0.560000   to 0.580000
Taking time step 30:   update from 0.580000   to 0.600000
Taking time step 31:   update from 0.600000   to 0.620000
Taking time step 32:   update from 0.620000   to 0.640000
Taking time step 33:   update from 0.640000   to 0.660000
Taking time step 34:   update from 0.660000   to 0.680000
Taking time step 35:   update from 0.680000   to 0.700000
Taking time step 36:   update from 0.700000   to 0.720000
Taking time step 37:   update from 0.720000   to 0.740000
Taking time step 38:   update from 0.740000   to 0.760000
Taking time step 39:   update from 0.760000   to 0.780000
Taking time step 40:   update from 0.780000   to 0.800000
Taking time step 41:   update from 0.800000   to 0.820000
Taking time step 42:   update from 0.820000   to 0.840000
Taking time step 43:   update from 0.840000   to 0.860000
Taking time step 44:   update from 0.860000   to 0.880000
Taking time step 45:   update from 0.880000   to 0.900000
Taking time step 46:   update from 0.900000   to 0.920000
Taking time step 47:   update from 0.920000   to 0.940000
Taking time step 48:   update from 0.940000   to 0.960000
Taking time step 49:   update from 0.960000   to 0.980000
Taking time step 50:   update from 0.980000   to 1.000000
Have reached time tend; stop now
The PostScript backend does not support transparency; partially tra
nsparent artists will be rendered opaque.
Error in maximum norm:   9.40e-02


--- Upwind Scheme with CFL = 1.0 ---
```

```
# START PROGRAM
Taking time step 1:     update from 0.000000    to 0.025000
Taking time step 2:     update from 0.025000    to 0.050000
Taking time step 3:     update from 0.050000    to 0.075000
Taking time step 4:     update from 0.075000    to 0.100000
Taking time step 5:     update from 0.100000    to 0.125000
Taking time step 6:     update from 0.125000    to 0.150000
Taking time step 7:     update from 0.150000    to 0.175000
Taking time step 8:     update from 0.175000    to 0.200000
Taking time step 9:     update from 0.200000    to 0.225000
Taking time step 10:    update from 0.225000    to 0.250000
Taking time step 11:    update from 0.250000    to 0.275000
Taking time step 12:    update from 0.275000    to 0.300000
Taking time step 13:    update from 0.300000    to 0.325000
Taking time step 14:    update from 0.325000    to 0.350000
Taking time step 15:    update from 0.350000    to 0.375000
Taking time step 16:    update from 0.375000    to 0.400000
Taking time step 17:    update from 0.400000    to 0.425000
Taking time step 18:    update from 0.425000    to 0.450000
Taking time step 19:    update from 0.450000    to 0.475000
Taking time step 20:    update from 0.475000    to 0.500000
Taking time step 21:    update from 0.500000    to 0.525000
Taking time step 22:    update from 0.525000    to 0.550000
Taking time step 23:    update from 0.550000    to 0.575000
Taking time step 24:    update from 0.575000    to 0.600000
Taking time step 25:    update from 0.600000    to 0.625000
Taking time step 26:    update from 0.625000    to 0.650000
Taking time step 27:    update from 0.650000    to 0.675000
Taking time step 28:    update from 0.675000    to 0.700000
Taking time step 29:    update from 0.700000    to 0.725000
Taking time step 30:    update from 0.725000    to 0.750000
Taking time step 31:    update from 0.750000    to 0.775000
Taking time step 32:    update from 0.775000    to 0.800000
Taking time step 33:    update from 0.800000    to 0.825000
Taking time step 34:    update from 0.825000    to 0.850000
Taking time step 35:    update from 0.850000    to 0.875000
Taking time step 36:    update from 0.875000    to 0.900000
Taking time step 37:    update from 0.900000    to 0.925000
```

```
Taking time step 38:     update from 0.925000    to 0.950000
Taking time step 39:     update from 0.950000    to 0.975000
Taking time step 40:     update from 0.975000    to 1.000000
Have reached time tend; stop now
The PostScript backend does not support transparency; partially tra
nsparent artists will be rendered opaque.
Error in maximum norm:   3.77e-15


--- Upwind Scheme with CFL = 1.2 ---

# START PROGRAM
Taking time step 1:      update from 0.000000    to 0.030000
Taking time step 2:      update from 0.030000    to 0.060000
Taking time step 3:      update from 0.060000    to 0.090000
Taking time step 4:      update from 0.090000    to 0.120000
Taking time step 5:      update from 0.120000    to 0.150000
Taking time step 6:      update from 0.150000    to 0.180000
Taking time step 7:      update from 0.180000    to 0.210000
Taking time step 8:      update from 0.210000    to 0.240000
Taking time step 9:      update from 0.240000    to 0.270000
Taking time step 10:     update from 0.270000    to 0.300000
Taking time step 11:     update from 0.300000    to 0.330000
Taking time step 12:     update from 0.330000    to 0.360000
Taking time step 13:     update from 0.360000    to 0.390000
Taking time step 14:     update from 0.390000    to 0.420000
Taking time step 15:     update from 0.420000    to 0.450000
Taking time step 16:     update from 0.450000    to 0.480000
Taking time step 17:     update from 0.480000    to 0.510000
Taking time step 18:     update from 0.510000    to 0.540000
Taking time step 19:     update from 0.540000    to 0.570000
Taking time step 20:     update from 0.570000    to 0.600000
Taking time step 21:     update from 0.600000    to 0.630000
Taking time step 22:     update from 0.630000    to 0.660000
Taking time step 23:     update from 0.660000    to 0.690000
Taking time step 24:     update from 0.690000    to 0.720000
Taking time step 25:     update from 0.720000    to 0.750000
Taking time step 26:     update from 0.750000    to 0.780000
```

```
Taking time step 27:     update from 0.780000    to 0.810000
Taking time step 28:     update from 0.810000    to 0.840000
Taking time step 29:     update from 0.840000    to 0.870000
Taking time step 30:     update from 0.870000    to 0.900000
Taking time step 31:     update from 0.900000    to 0.930000
Taking time step 32:     update from 0.930000    to 0.960000
Taking time step 33:     update from 0.960000    to 0.990000
Taking time step 34:     update from 0.990000    to 1.000000
Have reached time tend; stop now
The PostScript backend does not support transparency; partially tra
nsparent artists will be rendered opaque.
Error in maximum norm:   9.88e-02


Convergence table stored in 'results/error.txt':
No convergence table found. Run with Nrefine > 0.
```