# Solution of the 1D steady transport equation with the finite difference method

September $16^{th}$, 2024

Ben Wilfong

## Files

```
HW1_root
├── CMakeLists.txt
├── README.md
├── homework.pdf
├── main.inp
├── make.sh
├── src
│   ├── m_global_parameters.f90
│   ├── m_helpers.f90
│   ├── m_linear_algebra.f90
│   └── p_main.f90
└── plots
    ├── outputB10.csv
    ├── outputB20.csv
    ├── outputB30.csv
    ├── outputB40.csv
    ├── outputB50.csv
    ├── outputC1.csv
    ├── outputC2.csv
    ├── outputC3.csv
    └── plots.m
```

## Compiling

This code can be compiled in a terminal by running `chmod u+x ./mfc.sh` followed by `./mfc.sh` in the `HW1_root/` directory. This will compile and link the source code in the `src/` directory and create an executable called `main` in the `HW1_root/` directory.

## Running

The problem parameters are defined in the namelist file `main.inp` in `HW1_root/`. The input parameters are:

- gamma0: The coefficient for the zero degree term in the polynomial defining gamma
- gamma1: The coefficient for the first degree term in the polynomial defining gamma

- L: The length of the domain
- N: The number of cells to use
- Q0: The coefficient for the zero degree term in the polynomial defining Q
- Q1: The coefficient for the first degree term in the polynomial defining Q
- U: The convective velocity
- tol: Convergence tolerance for nonlinear problems
- bench: Toggles benchmarking mode. When true, the code is ran $1 \times 10^6$ times and the average runtime is output.

Once the input parameters are set, the code is ran with `./main`. After execution, the solution is writen to `HW1_root/output.csv` with the first column holding the x coordinates and the second column holding the solution. Additionally, the following will be written to the terminal:

```
Solving the 1D Steady Transport Equation with:
gamma0:   0.10000000000000001
gamma1:   0.10000000000000001
U:    0.0000000000000000
Q0:    0.0000000000000000
Q1:    0.0000000000000000
N:           20
Execution time:    4.99999987E-05  s
Iterations:           12
```

This serves as a sanity check for the problem parameters and provides the runtime and number of iterations to convergence. Plots are created by running `plots.m` from the `root/plots/` directory.

# Part A - Numerical Method

The one dimensional steady state transport equation

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

can be discretized with central differences as

$$U\frac{\phi_{i+1}-\phi_{i-1}}{2\Delta x} - \underbrace{\frac{\left(\Gamma\frac{\partial\phi}{\partial x}\right)_{i+1/2} - \left(\Gamma\frac{\partial\phi}{\partial x}\right)_{i-1/2}}{\Delta x}}_{b} = Q_i. \tag{1}$$

The first derivatives in the numerator of $b$ can be written as

$$\left(\Gamma\frac{\partial\phi}{\partial x}\right)_{i+1/2} = \Gamma_{i+1/2}\frac{\phi_{i+1}-\phi_i}{\Delta x} \quad \text{and} \quad \left(\Gamma\frac{\partial\phi}{\partial x}\right)_{i-1/2} = \Gamma_{i-1/2}\frac{\phi_i-\phi_{i-1}}{\Delta x}. \tag{2}$$

Substituting eq. (2) into eq. (1) and rearanging gives

$$\begin{aligned}
U\frac{\phi_{i+1}-\phi_{i-1}}{2\Delta x} - \frac{\Gamma_{i+1/2}\frac{\phi_{i+1}-\phi_i}{\Delta x} - \Gamma_{i-1/2}\frac{\phi_i-\phi_{i-1}}{\Delta x}}{\Delta x} &= Q_i \\
U\frac{\phi_{i+1}-\phi_{i-1}}{2\Delta x} - \frac{\Gamma_{i+1/2}\phi_{i+1} - (\Gamma_{i+1/2}+\Gamma_{i-1/2})\phi_i + \Gamma_{i-1/2}\phi_{i-1}}{(\Delta x)^2} &= Q_i \\
\left(\frac{U}{2\Delta x} - \frac{\Gamma_{i+1/2}}{(\Delta x)^2}\right)\phi_{i+1} + \left(\frac{\Gamma_{i+1/2}+\Gamma_{i-1/2}}{(\Delta x)^2}\right)\phi_i - \left(\frac{U}{2\Delta x} + \frac{\Gamma_{i-1/2}}{(\Delta x)^2}\right)\phi_{i-1} &= Q_i.
\end{aligned} \tag{3}$$

Equation (3) can be cast as the system of linear equations

$$\underbrace{\begin{bmatrix}
1 & & & \\
-\left(\frac{U}{2\Delta x}+\frac{\Gamma_{1/2}}{(\Delta x)^2}\right) & \frac{\Gamma_{3/2}+\Gamma_{1/2}}{(\Delta x)^2} & \frac{U}{2\Delta x}-\frac{\Gamma_{3/2}}{(\Delta x)^2} & \\
\ddots & \ddots & \ddots & \\
 & -\left(\frac{U}{2\Delta x}+\frac{\Gamma_{N-3/2}}{(\Delta x)^2}\right) & \frac{\Gamma_{N-1/2}+\Gamma_{N-3/2}}{(\Delta x)^2} & \frac{U}{2\Delta x}-\frac{\Gamma_{i-1/2}}{(\Delta x)^2} \\
 & & & 1
\end{bmatrix}}_{\text{A}}
\underbrace{\begin{pmatrix}\phi_0\\ \phi_1\\ \vdots\\ \phi_{N-1}\\ \phi_N\end{pmatrix}}_{\Phi}
= \underbrace{\begin{pmatrix}1\\ Q_1\\ \vdots\\ Q_{N-1}\\ 0\end{pmatrix}}_{b}$$

for a discretization at $\{x_0,\ x_1,\ \ldots\ x_N\} = \{0,\ \Delta x,\ 2\Delta x, \ldots,\ N\Delta x\}$. The boundary conditions are given by $\phi_{-1/2} = 1$ and $\phi_{N+1/2} = 0$.

For the cases in which $U$, $\Gamma$, and $Q$ are constant, the solution is found via the following steps:

1. Populate the matrix A
2. Populate the vector $b$
3. Solve $A\Phi = b$ for the vector $\Phi$ using TDMA

When $U$, $\Gamma$, or $Q$ depend on the value of $\phi$ the problem becomes nonlinear and an iterative approach must be taken. For nonlinear problems, the solution is found via the following steps:

1. Make an initial guess for the vector $\Phi^{(0)}$
2. Calculate $U(\Phi^{(n)})$, $\Gamma(\Phi^{(n)})$, and $Q(\Phi^{(n)})$
3. Populate the matrix $A$
4. Populate the vector $b$
5. Solve $A\Phi^{(n+1)} = b$ for the vector $\Phi^{(n+1)}$ using TDMA
6. Repeat steps 2 though 5 until convergence

For this problem I will use a discretization of $\phi(x) = 1 - x/L$ as an initial guess.

# Part B - Linear Problems

Figure 2 shows the solution, average error, and runtime for the case $U = 1$, $\Gamma = 0.1$, and $Q = 0$ with $N = 10,\ 20,\ 30,\ 40,\ 50$. The error decreases with increasing $N$ as expected. The plot of average error versus $N$ shows that the method is second-order-accurate. This is because the error decreases with a slope of two in loglog space. This order of accuracy is expected given that central differences were used to discretize the derivatives. The runtime on a single Apple M2 core, calculated as an average of $1 \times 10^6$ samples, increases with $N$ with a slope of approximately one in loglog space. This indicates that the method is $\mathcal{O}(n)$, which agrees with the theoretical time complexity of the TDMA. The small difference is slopes is likely a result of the small problem size, meaning that the asymptotic assumption is not yet met.

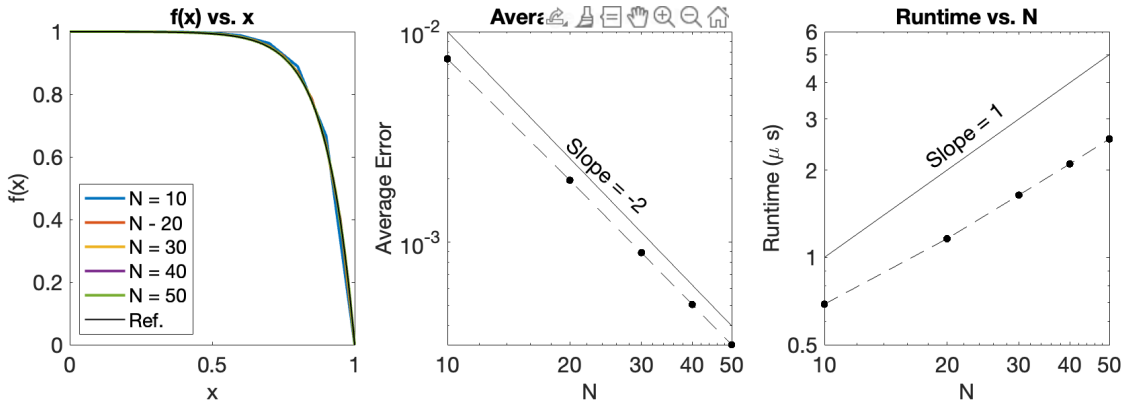


Figure 1: Solution, average error, and runtime for $N = 10, 20, 30, 40, 50$

# Part C - Non-linear Problems

**??** shows the solution and pointwise error for the case $U = 0,\ \Gamma = 0.1 + 0.1\varphi$ and $N = 20$ for $Q = 0,\ 0.1$, and $0.1x$. The solution for $Q = 0.1x$ lies between the solutions for $Q = 0$ and $Q = 0.1$. This is because the total source flux, given by $Q = \int_0^L Q\mathrm{d}l$, satisfies $Q_0 < Q_{0.1x} < Q_{0.1}$ and a larger total source flux results in a higher curve. The pointwise error is between the discretized solution and the exact solution is less than the specified tolerance of $1 \times 10^{-3}$. The iterative solve is what differentiates the approach for solving this nonlinear problem from the approach to solving the linear equation. Since the coefficients depend on the solution, an initial guess for the solution must be made and iterated on until a specified convergence tolerance has been met. The runtime on a single Apple M2 core for each $Q$, averaged over $1 \times 10^6$ runs is given in table 1. This runtime is 5.3 times longer than the linear solve for the $N = 20$ case. The increase in runtime is because 5 iterations are required to reach convergence and each iteration requires the additional calculation of an error estimate.

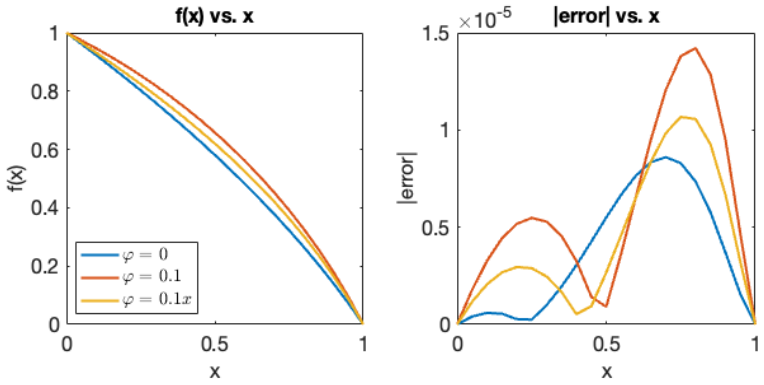


Figure 2: Solution and pointwise error for $Q = 0, 0.1, 0.1x$

Table 1: Run time for $Q = 0, 0.1, 0.1x$

| $Q$ | 0 | 0.1 | 0.1x |
|---|---|---|---|
| Runtime ($\mu$s) | 6.3 | 6.3 | 6.2 |