Super Time Stepping for Advection-Diffusion

James MacPherson
Advisor: Dr. V. Alexiades

July 2020

Outline

Introduction

Explicit time-stepping schemes for PDEs are very simple and easy to implement (and to parallelize on clusters). However, they are subject to CFL stability condition that limits the size of the time-step requiring many time-steps to reach a desired final time. Instead of requiring stability on each time-step, Super-Time-Stepping (STS) imposes stability only over a "super-step" consisting of N sub-steps, while maximizing the duration of the super-step. STS maintains the simplicity of the explicit scheme and can be up to N times faster than the explicit scheme on diffusion problems. Theoretically, STS is developed for pure diffusion problems only. The primary goal of this work is to explore the effectiveness of STS on increasingly advection-dominated advection-diffusion processes.

Conservation Law

Conservation of a space-and-time-varying quantity $u(\vec{x}, t)$ with flux $\vec{F}(\vec{x}, t)$ and internal source $S(\vec{x}, t)$ in a region Ω is expressed by the partial differential equation (PDE)

$$\frac{\partial u}{\partial t} + \nabla \cdot \vec{F} = S \quad \text{at each point of } \Omega \tag{2.1}$$

Here $u(\vec{x}, t)$ denotes the amount per unit volume of the conserved quantity at location $\vec{x}$ at time t, the flux $\vec{F}(\vec{x}, t)$ is the amount crossing a unit area per unit time, and the source $S(x, t)$ is the amount of u generated (or lost) per unit volume per unit time internally in Ω at location $\vec{x} \in \Omega$ at time t.

Flux

The Flux $\vec{F}$ can always be expressed as the sum of advective + non-advective components: $\vec{F} = \vec{F}^{adv} + \vec{F}^{non-adv}$, with advective flux $\vec{F}^{adv} = \vec{V}u$ for a given advective velocity $\vec{V}$.

The non-advective flux is specified by a **constitutive law** describing the specific physical process(es) one wants to consider.

The most fundamental natural process is **diffusion**, with constitutive law given by **Fick's law**: $\vec{F}^{dif} = -D\nabla u$, with diffusivity D.

1-dimensional PDE

We will consider only the no-source case $S \equiv 0$ and in one space dimension, in an interval $\Omega = (a, b)$, so the PDE is

$$u_t + F_x = 0. \tag{2.2}$$

For 1-D advection-diffusion, the flux is $F = F^{adv} + F^{dif} = Vu - Du_x$. Thus the PDE for constant velocity V and diffusivity D, is

$$u_t + Vu_x = Du_{xx}\,, \tag{2.3}$$

involving the two physical parameters, velocity V and diffusivity D.

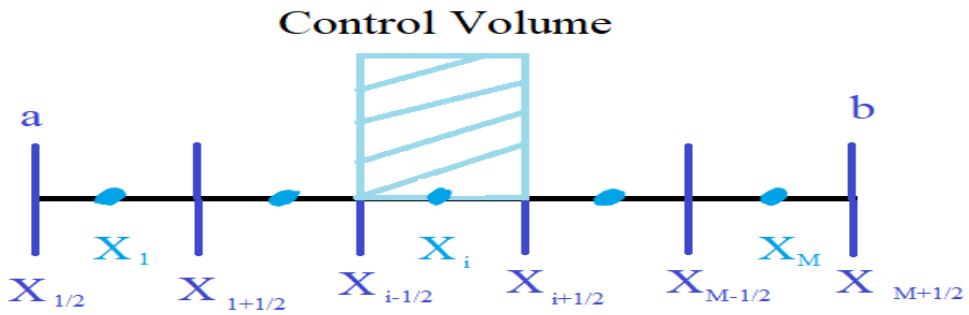
Dimensionless Form

It can be written in dimensionless form involving a single parameter, the **Peclet Number,** Pe, in terms of dimensionless space $\xi = x/\hat{x}$, dimensionless time $\tau = tD/\hat{x}^2$, for some length scale $\hat{x}$, and dimensionless $w(\xi, \tau) = u(x, t)$

$$w_\tau + Pe\ w_\xi = w_{\xi\xi}, \qquad \text{with } Pe = V\hat{x}/D. \tag{2.4}$$

For pure diffusion $Pe = 0$, and for pure advection $Pe = \infty$.
Small Pe signifies the process is diffusion-dominated while large Pe signifies advection-dominated process.

Mesh



To discretize the PDE (2.2), we discretize the interval $[a, b]$ into M control volumes V_i, each of width $\Delta x = (b - a)/M$, with nodes $x_0 = a$, $x_1 = a + \Delta x/2$, $x_i = a + (i - 1)\Delta x/2$ for $i = 2 : M$, $x_{M+1} = b$. The left and right faces of control volume V_i will be $x_{i-1/2} = x_i - \Delta x/2$ and $x_{i+1/2} = x_i + \Delta x/2$. This creates a mesh array $x(0 : M + 1)$.
The time interval $[t_0, t_{end}]$ is discretized into N_{steps} time-steps, each of duration $\Delta t = \frac{t_{end} - t_0}{\text{Nsteps}}$.

Finite Volume Discretization 1

We discretize the conservation law (2.2) for any flux, and then the flux separately.
The Finite Volume discretization of the PDE (2.2) is obtained as follows. We integrate over each control volume: $V_i = [x_{i-1/2}, x_{i+1/2}]$ and over each time-step $[t_n, t_{n+1}]$.

$$\int_{V_i} u_t dx + \int_{x_{i-1/2}}^{x_{i+1/2}} F_x dx = 0$$

$$\frac{d}{dt} \int_{V_i} u(x,t) dx + F|_{x_{i-1/2}}^{x_{i+1/2}} = 0$$

$$\int_{t_n}^{t_{n+1}} \frac{d}{dt} \int_{V_i} u(x,t) dx dt + \int_{t_n}^{t_{n+1}} F|_{x_{i-1/2}}^{x_{i+1/2}} dt = 0.$$

Finite Volume Discretization 2

Setting $U_i^n := \frac{1}{\Delta x}\int_{V_i} u(x,t_n)dx =$ mean value of u over V_i at time t_n, and $\bar{F}^n_{i-1/2} := \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} F(x_{i-1/2},t)dt =$ mean flux over (t_n, t_{n+1}) across face at $x_{i-1/2}$, we obtain

$$\Delta x\,[U_i^{n+1} - U_i^n] + \Delta t\,[\bar{F}^n_{i+1/2} - \bar{F}^n_{i-1/2}] = 0,$$

which expresses **exact discrete conservation** over V_i during (t_n, t_{n+1}) (no approximation has been made).

Finite Volume Discretization 3

For small Δt, taking the mean flux $\bar{F}^n_{i-1/2}$ to be the flux at t_n, $F^n_{i-1/2}$ (which amounts to Forward Euler time-stepping), we obtain the **explicit Finite Volume discretization** of the conservation law (2.2):

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}[F^n_{i+1/2} - F^n_{i-1/2}] \tag{2.5}$$

which expresses discrete conservation of U for any flux F.

Diffusive Flux and CFL

The fluxes are discretized separately as follows.
The diffusive flux $F^{dif} = -Du_x$ is discretized by finite difference as

$$F^{dif}_{i-1/2} = -D\frac{U_i - U_{i-1}}{x_i - x_{i-1}} = -D\frac{U_i - U_{i-1}}{\Delta x}. \tag{2.6}$$

The Courant-Friedrichs-Lewy (CFL) stability condition for the explicit scheme for diffusion is

$$\Delta t \leq \frac{(\Delta x)^2}{2D} = \Delta t_{expl}\,. \tag{2.7}$$

Advective Flux and CFL

The advective flux $F^{adv} = Vu$ is discretized by the **upwind scheme** as

$$F^{adv}_{i-1/2} = \begin{cases} VU_{i-1}, & \text{if } V > 0 \\ VU_i, & \text{if } V < 0 \end{cases} \tag{2.8}$$

Thus, for $V > 0$, the advection-diffusion flux is

$$F_{i-1/2} = VU_{i-1} - D\frac{U_i - U_{i-1}}{\Delta x} \tag{2.9}$$

and the CFL stability condition becomes

$$\Delta t \leq \frac{1}{\frac{V}{\Delta x} + \frac{2D}{(\Delta x)^2}} = \Delta t_{expl}\,. \tag{2.10}$$

Note that when $V = 0$ (2.10) reduces to (2.7), and when $D = 0$ it reduces to the CFL for pure advection: $\Delta t \leq \Delta x / V$.

Initial and Boundary Conditions

The PDE requires initial condition: $u(x,0) = u_{init(x)}$, and some Boundary Condition at $x = a$ and $x = b$. In our experiments we will only use Dirichlet type BCs, i.e. the boundary values of u: $u(a,t) = u_a(t)$, $u(b,t) = u_b(t)$ are specified at $x = a$ and $x = b$. Then in the Finite Volume scheme, at $x = a$, the boundary value $U_0^n = u_a(t_n)$ is known and the boundary flux is approximated by

$$F_{1/2}^n = VU_0^n - D\frac{U_1^n - U_0^n}{\Delta x/2}. \tag{2.11}$$

Similarly, at $x = b$, the boundary value $U_{M+1}^n = u_b(t_n)$ is known and the boundary flux is approximated by (for $V > 0$)

$$F_{M+1/2}^n = VU_M^n - D\frac{U_{M+1}^n - U_M^n}{\Delta x/2}. \tag{2.12}$$

Lax-Wendroff I

The upwind scheme, which is of 1st order, was not providing sufficient accuracy in our experiments, so we turned to higher order methods. The most basic 2nd order method is the Lax-Wendroff scheme [3]

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{2\Delta x} V(U_{i+1}^n - U_{i-1}^n) + \frac{1}{2}(\frac{\Delta t}{\Delta x})^2 V^2(U_{i-1}^n - 2U_i^n + U_{i+1}^n). \tag{2.13}$$

This can be rewritten to fit the Finite Volume scheme with flux function

$$F_{i-1/2}^n = \frac{1}{2} V(U_{i-1}^n + U_i^n) - \frac{1}{2}\frac{\Delta t}{\Delta x} V^2(U_i^n - U_{i-1}^n) \tag{2.14}$$

Lax-Wendroff II

The Lax-Wendroff scheme is of order two, but it does introduce unphysical oscillations (at discontinuities, such as shocks). To suppress oscillations one uses the technology of **flux-limiters**. The Flux Function (2.14) can be rewritten as:

$$F^n_{i-1/2} = VU^n_{i-1} + \frac{1}{2}V(1 - \frac{\Delta t}{\Delta x}V)(U^n_i - U^n_{i-1}) \qquad (2.15)$$

The first term is the upwind flux, and the following term is the corrective term, which causes the oscillations in Lax-Wendroff.

Flux Limiters I

To suppress oscillations, the corrective term is multiplied by a limiter function $\phi(\theta)$:

$$F^n_{i-1/2} = VU^n_{i-1} + \frac{1}{2}V(1 - \frac{\Delta t}{\Delta x}V)(U^n_i - U^n_{i-1})\phi(\theta). \tag{2.16}$$

Note that $\phi = 0$ gives the upwind method, and $\phi = 1$ the Lax-Wendroff method. θ is considered the smoothness factor and is calculated as

$$\theta = \frac{U_I - U_{I-1}}{U_i - U_{i-1}} \text{ with}$$

$$I = \begin{cases} i-1, \text{ if } V > 0 \\ i, \text{ if } V \leq 0 \end{cases}$$

Flux Limiters II

Some of the popular limiter functions $\phi(\theta)$ are:

- Minmod: $\phi(\theta) = max(0, min(1, \theta))$
- Superbee: $\phi(\theta) = max(0, min(1, 2\theta), min(2, \theta))$
- Van Leer: $\phi(\theta) = (\frac{\theta+|\theta|}{1+|\theta|})$
- Monotonized Centered (MC): $\phi(\theta) = max(0, min((1+\theta)/2, 2, 2\theta))$

We use MC in the experiments. More information regarding conditions and comparisons of limiter methods can be found in [3] and [4].

STS - Explicit Scheme

We briefly outline the idea and implementation of the Super-Time-Stepping scheme referring to [2] for details. Consider the following time dependent problem

$$\frac{dU}{dt}(t) + AU(t) = 0, \quad t > 0, \quad U(0) = U_0\,, \tag{3.1}$$

where A is $M \times M$ matrix resulting from space discretization of a parabolic PDE, such as $u_t - Du_{xx} = 0$ on a grid with M nodes. The standard explicit (forward Euler) scheme for the ODE (3.1) is

$$U^{n+1} = U^n - \Delta t A U^n = (I - \Delta t A)U^n, \quad n = 0, 1, \ldots, \quad U^0 = U_0. \tag{3.2}$$

STS - Stability Condition

For numerical stability, the time-step Δt must be restricted to satisfy

$$\rho(I - \Delta t A) < 1, \tag{3.3}$$

where ρ denotes the spectral radius of the matrix. If λ_{max} is the largest eigenvalue of $I - \Delta t A$, then

$$\Delta t < \Delta t_{expl} := \frac{2}{\lambda_{max}}. \tag{3.4}$$

This is the CFL condition. For the one dimensional diffusion equation $u_t = D u_{xx}$, $\lambda_{max} = 4D/(\Delta x)^2$, so (3.4) yields (2.7).

Super-Time-Stepping

In order to relax the condition, we do not require stability at the end of each time step Δt. We instead require stability at the end of a cycle of N time-steps, leading to a Runge-Kutta type method with N stages. We introduce a **super-step** ΔT consisting of N sub-steps $\tau_1, \tau_2, \ldots, \tau_N$. The idea is to maintain stability for the super-step while maximizing its duration $\Delta T = \sum_{j=1}^{N} \tau_j$.
The new algorithm can now be written as

$$U^{n+1} = (\Pi_{j=1}^{N}(I - \tau_j A))U^n, n = 0, 1, \ldots \tag{3.5}$$

Stability for Super-Time-Stepping

The corresponding stability condition is

$$\rho(\Pi_{j=1}^{N}(I - \tau_j A)) < 1 \tag{3.6}$$

The relation is satisfied if

$$|\Pi_{j=1}^{N}(1 - \tau_j \lambda)| < 1, \qquad \forall \lambda \in [\lambda_{min}, \lambda_{max}], \tag{3.7}$$

where $0 < \lambda_{min}$ denotes the smallest eigenvalue of A. We are looking for time-steps $\tau_1, \ldots, \tau_N$ that satisfy (3.7) and maximize the duration ΔT.

Choosing the substeps τ_j

It turns out that $\tau_1, \ldots, \tau_N$ can be found explicitly [2], thanks to the optimality properties of the Chebyshev polynomials. They are given by

$$\tau_j = \Delta t_{expl} \left((-1+\nu) cos(\frac{2j-1}{N}\frac{\pi}{2}) + 1 + \nu \right)^{-1}, \quad j = 1, \ldots, N, \quad (3.8)$$

where Δt_{expl} is found from the CFL condition as in (2.10), and $\nu > 0$ is a paremeter to be chosen, along with N.
Note that **the choice $N = 1$, $\nu = 0$ gives the explicit scheme itself**, which is very convenient.

Super-Time-Stepping Duration

One can show the relation

$$\Delta T = \sum_{j=1}^{N} \tau_j = \Delta t_{expl} \frac{N}{2\sqrt{\nu}} \left(\frac{(1+\sqrt{\nu})^{2N} - (1-\sqrt{\nu})^{2N}}{(1+\sqrt{\nu})^{2N} + (1-\sqrt{\nu})^{2N}} \right) \tag{3.9}$$

$$\text{and that } \Delta T \to N^2 \Delta t_{expl} \text{ as } \nu \to 0\,. \tag{3.10}$$

So when $\nu \sim 0$, STS will be N times faster than the standard explicit scheme, at essentially the same cost.
This is where the speedup is coming from!
We note that $\nu = 0$ corresponds to the stability limit, therefore choosing ν too small can lead to instability. On the other hand, increasing ν produces smaller τ_j and shorter super-step duration, thus increasing accuracy at the expense of efficiency.

Implementation of Super-Time-Stepping

Implementing the STS scheme is quite simple and can easily be inserted into the code for an explicit scheme.
First we choose N and ν, where N is once again the number of sub-steps in a super-step, and ν can be thought of as a "knob" trading stability and efficiency for accuracy.
After choosing N and ν, calculate each τ_j from (3.8).
Then execute N explicit (Euler) updates with $\Delta t = \tau_j$, j=1:N, without outputting until the completion of the N sub-steps of the super-step.
Repeat taking super-steps till the desired final time is reached.

Summary of Super-Time-Stepping

Although the theory requires the matrix A in (3.1) to be symmetric positive definite (which is the case for pure diffusion but not for advection-diffusion), this might not be required in practice.
The purpose of this work is to test how STS performs on a range of advection-diffusion problems from diffusion dominated (small Peclet number) to advection dominated (large Peclet number).

Pure Diffusion problem with similarity solution

We apply the STS scheme on a pure diffusion problem whose explicit solution is known, so that we can evaluate the error.
Such a problem is diffusion on a semi-infinite interval $[0, \infty)$, where $u = 0$ initially, and $u = 1$ is imposed at $x = 0$:

$$\begin{cases} u_t = Du_{xx}, \quad 0 < x < \infty, \quad t > 0 \\ u(x, 0) = 0 \\ u(0, t) = 1 \end{cases} \tag{4.1}$$

To obtain the exact solution, we seek a similarity solution by setting $\xi = x/\sqrt{Dt}$ and $u(x, t) = y(\xi)$ resulting in the ODE $y'' + (\xi/2)y' = 0$.
Solving for y explicitly gives $y(\xi) = 1 - \text{erf}(\xi/2)$ with $\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-s^2} ds$ the *error function*. Thus the exact solution of (4.1) is

$$u_{exact}(x, t) = 1 - \text{erf}(\frac{x}{2\sqrt{Dt}}). \tag{4.2}$$

Numerical Scheme

To solve the problem numerically, we apply the Finite Volume scheme (§2.2) for pure diffusion (with $V \equiv 0$), on the finite interval $[a, b] = [0, 1]$ with initial condition $u(x, 0) = 0$ (so $U_i^0 = 0$), boundary conditions $u(a, t) = 1$ (so $U_0 = 1$) and $u(b, t) = u_{exact}(b, t) =$ exact solution at $x = b = 1$ (so $U_{M+1}^n = u_{exact}(b, t_n)$), over the time interval from $t_0 = 0$ to $t_{end} = 100$.
At runtime, we specify MM = number of nodes per unit length, and set Δx=1/MM, $M = (b - a)/\Delta x = (b - a)*$MM = number of nodes in $[a, b]$, and $\Delta t_{expl} = \Delta x^2/(2D)$ for diffusion according to (2.7).

Measuring Efficiency

We measure efficiency of a run by the total number of time-steps executed. In each run, we record **nSupSteps** = number of super-steps of STS, and **nsteps** = total number of time-steps executed. By running Euler (using $N = 1$, $\nu = 0$) we find **nstepsEuler** = number of steps taken by the Euler scheme, and we calculate the **Speedup** by

$$\text{Speedup} = \text{nstepsEuler} \;/\; \text{nsteps}. \tag{4.3}$$

The higher the Speedup, the more efficient STS is, and Speedup tell us how many times faster STS is than the basic explicit scheme.
We also compute the Error = $\max(u_{exact} - U)$ over all nodes at the last time-step.
We impose an error tolerance TOL=1.0e-3.
If the error is larger than TOL, we record the Speedup as zero.

Results I

Table: Pure Diffusion on $[a, b]$=[0,1] with MM=1024

N	ν	nSupSteps	nsteps	Error	Speedup
1	0.0	20971520	20971520	4.88e-04	1.00
1	1.e-5	20971730	20971730	3.11e-09	1.00
1	1.e-1	23068672	23068672	2.73e-09	0.91
5	0.0	838861	4194305	5.6615e-04	5.00
5	1.e-5	839138	4195690	2.1876e-06	5.00

For $N = 1$, even using a ν as small as 1.e-5 reduces the error from 4.88e-4 down to 3.11e-9. At $\nu = 0.1$, the error reduces slightly to 2.73e-9 but the speedup drops to 0.91.
For $N = 5$, the scheme is 5 times faster at ν=0, 1.e-5.

Results II

Table: Pure Diffusion on $[a, b]$=[0,1] with MM=1024

N	ν	nSupSteps	nsteps	Error	Speedup
10	0.0	209716	2097160	1.35e-03	0.00
10	1.e-5	209995	2099950	4.88e-06	9.99
10	1.e-3	236898	2368980	5.04e-07	8.85
10	1.e-1	1326361	13263610	1.06e-08	1.58
20	0.0	52429	1048580	2.75e-03	0.00
20	1.e-5	52708	1054160	1.02e-05	19.89
20	1.e-1	663178	13263560	1.05e-08	1.58

When $N = 10$ and ν =0.0, the loss of accuracy causes the scheme to surpass the error tolerance. But ν=1.e-5 improves the accuracy and we can see the speedup is 9.99 times faster. By increasing ν, error improves but the speedup drops.

At $N = 20$, the results are very similar. The value ν=1.e-5 gives speedup of 19.89 (giving the best speedup). As ν increases, the speedup slows down just as in the case of $N = 10$.

Results III

Table: Pure Diffusion on $[a, b]$=[0,1] with MM=1024

N	ν	nSupSteps	nsteps	Error	Speedup
50	1.e-3	26623	1331150	7.82e+00	0.00
50	1.e-2	83887	4194350	2.91e-04	5.00
50	1.e-1	265272	13263600	1.04e-08	1.58

For $N = 50$, a large ν is needed for acceptable error. At $\nu = 0.1$, the scheme only has a speedup of 1.58.

Summary of STS for Pure Diffusion

We see that the speedup can be up to almost N, as theory predicts, obtained for very small ν, but $\nu = 0$ may lead to instability producing Error $>$ TOL. By increasing ν, the Error improves at the expense of efficiency. However, there is a limit to speeding up the scheme by increasing N, as seen when $N = 50$. The scheme requires a larger ν that sacrifices efficiency just to maintain accuracy.

Advection-Diffusion PDE

Consider the advection-diffusion equation with constant $V > 0$ and $D > 0$

$$u_t + Vu_x = Du_{xx}. \tag{5.1}$$

To construct explicitly solvable problems for it, we transform it to a pure diffusion PDE, which can be solved explicitly for some simple initial profiles as we show below.
Indeed, if $u(x, t)$ solves (5.1), setting $\xi = x - (x_0 + Vt)$, for some x_0, the function $w(\xi, t) = u(x, t)$ solves the diffusion equation

$$w_t = Dw_{\xi\xi}. \tag{5.2}$$

Initial Conditions

We will be using this fact to solve (5.2) with square bump initial profile

$$w(\xi, 0) = w_0(\xi) := \begin{cases} A, & \xi_1 \leq \xi \leq \xi_2 \\ 0, & \text{otherwise,} \end{cases} \tag{5.3}$$

where A is the height of the square bump, ξ_1 is the left-most or 'starting' point for the bump, and ξ_2 is the right-most or 'ending' point for the bump.

Exact Solution I

The exact solution is given by the *Fourier Poisson Integral*

$$w(\xi, t) = \int_{-\infty}^{\infty} H(\xi - y, t) w_0(y) dy \tag{5.4}$$

with H the heat kernel

$$H(x, t) = \frac{1}{\sqrt{4\pi D t}} e^{-\frac{x^2}{4Dt}}. \tag{5.5}$$

Thanks to the simplicity of $w_0(\xi)$ in (5.3) the integral in (5.4) can be evaluated exactly, resulting in

$$w(\xi, t) = A \int_{\xi_1}^{\xi_2} \frac{1}{\sqrt{4\pi D t}} e^{-\frac{(\xi - y)^2}{4Dt}} dy. \tag{5.6}$$

Exact Solution II

Substituting $s = \frac{\xi - y}{\sqrt{4Dt}}$, $ds = -\frac{dy}{\sqrt{4Dt}}$, and $s_i = \frac{\xi - \xi_i}{\sqrt{4Dt}}$, $i = 1, 2$, equation (5.6) results in

$$\frac{A}{\sqrt{4\pi Dt}} \int_{s_1}^{s_2} e^{-s^2}(-\sqrt{4\pi Dt})ds = \frac{A}{2}[\text{erf}(s_1) - \text{erf}(s_2)] \tag{5.7}$$

and therefore the exact solution of the diffusion PDE (5.2) with initial profile (5.3) is

$$w(\xi, t) = \frac{A}{2}[\text{erf}(\frac{\xi - \xi_1}{\sqrt{4Dt}}) - \text{erf}(\frac{\xi - \xi_2}{\sqrt{4Dt}})]. \tag{5.8}$$

Exact Solution III

Setting $\xi = x - x_i - Vt$, with $x_i = \xi_i$, $i = 1, 2$, we obtain a solution of the advection-diffusion equation for $u(x, t) = w(\xi, t)$

$$u(x,t) = \frac{A}{2}[\text{erf}(\frac{x - Vt - x_1}{\sqrt{4Dt}}) - \text{erf}(\frac{x - Vt - x_2}{\sqrt{4Dt}})]. \tag{5.9}$$

which solves (5.1) with initial profile the square bump

$$u(x,0) = u_0(x) := \begin{cases} A, & x_1 \leq x \leq x_2 \\ 0, & \text{otherwise}, \end{cases} \tag{5.10}$$

Boundary Conditions

Evaluating this exact solution $u(x,t)$ at $x = a$ and $x = b$ (chosen appropriately for $[a,b]$ to contain the propagating bump) gives the Dirichlet boundary conditions for numerical solution

$$\begin{cases} U_0^n = \frac{A}{2}[\text{erf}(\frac{a-Vt_{n-1}}{\sqrt{4Dt_n}}) - \text{erf}(\frac{a-Vt_{n-2}}{\sqrt{4Dt_n}})] \\ U_{M+1}^n = \frac{A}{2}[\text{erf}(\frac{b-Vt_{n-1}}{\sqrt{4Dt_n}}) - \text{erf}(\frac{b-Vt_{n-2}}{\sqrt{4Dt_n}})]. \end{cases} \tag{5.11}$$

Paremeters

For the advection-diffusion problem of a square bump, whose explicit solution was presented in §5.1, we choose the following parameter values: $D = 0.01$, $A = 5.0$, $\xi_1 = 1.0$, $\xi_2 = 2.0$, $t_0 = 0$, and $t_{end} = 100$.
The interval $[a, b]$ is chosen to contain the propagating bump, as can be seen in Figures 1-5, and the values of a, b are shown in the caption of each table.

Cases

We present results for various values of the Peclet Number Pe (Pe = 0, 0.1, 1, 10, 100). The corresponding velocity V can be calculate from $V = \text{Pe} \cdot D$.
The following tables exhibit the effect of STS on various Peclet numbers, ranging from pure diffusion (Pe=0) to a highly advective one (Pe = 100).

Figure 1: Pe=0: Diffusion of Initial Square Bump Profile

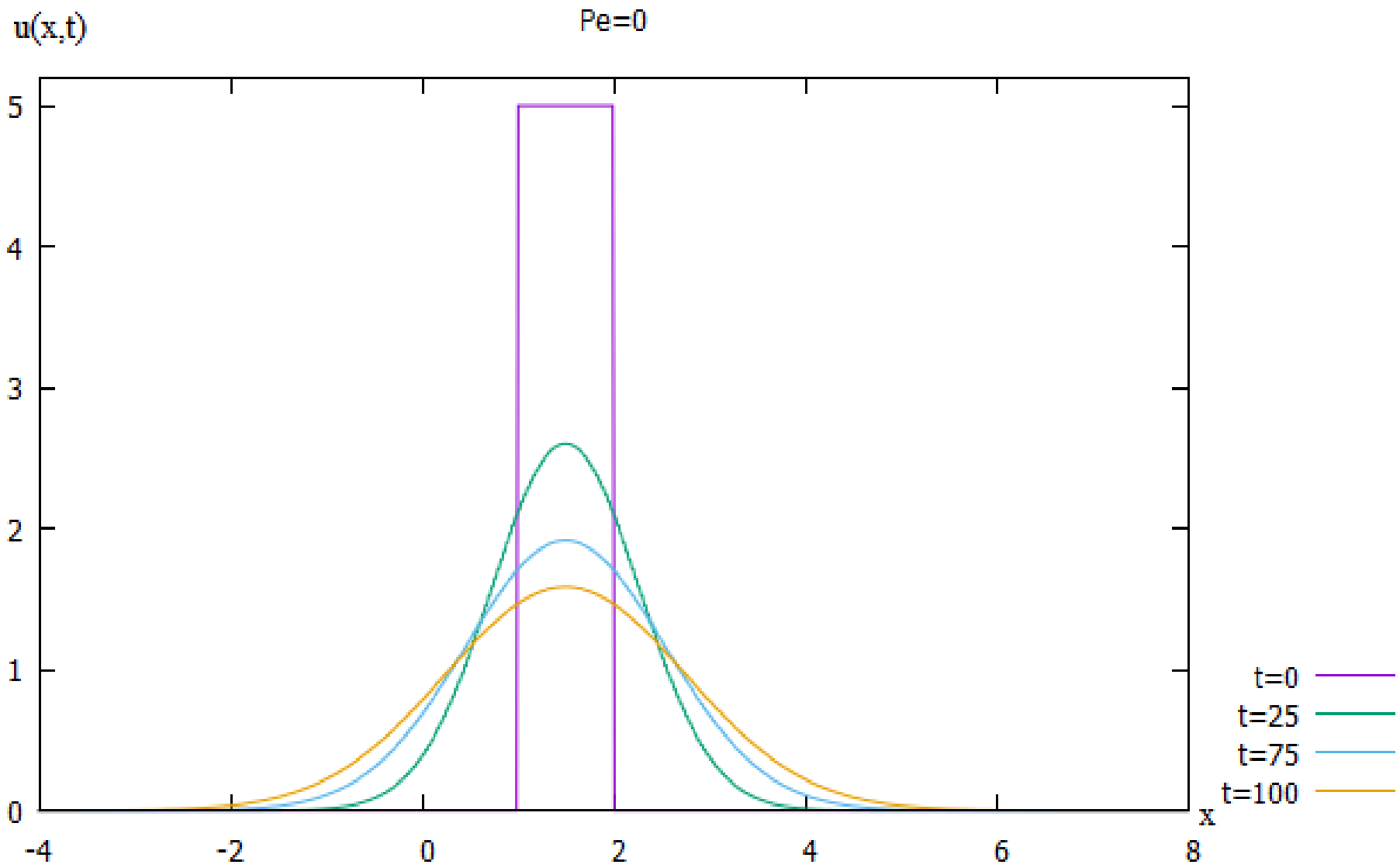


Table 2 Results I

Table: Pe=0.0, a=-4, b=8, MM=2048

N	ν	nSupSteps	nSteps	Error	Speedup
1	0.0	8388609	8388609	1.42e-04	1.00
1	1.e-05	8388693	8388693	3.13e-08	1.00
1	1.e-01	9227470	9227470	2.61e-08	0.91
5	0.0	335545	1677725	5.18e-03	0.00
5	1.e-05	335656	1678280	9.39e-07	5.00
5	1.e-01	1064128	5320640	1.28e-07	1.58

Results for Pe=0.0, hence V=0, the advection-diffusion equation (5.1) reduces to the pure diffusion equation and we expect results similar to §4, as indeed we see in Table 2. Note that for $N = 1$, changing $\nu = 0$ to ν=1.e-5 dramatically cuts the error down to 3.13e-8, which is 4 orders better than Euler! Further increase of ν does not improve the error.
At $N = 5$, ν=1.e-5 achieves full Speedup of 5. Again, further increase of ν does not improve the error.

Table 2 Results 2

Table: Pe=0.0, a=-4, b=8, MM=2048

N	ν	nSupSteps	nSteps	Error	Speedup
20	1.e-05	21084	421680	1.50e-05	19.89
20	1.e-03	31117	622340	8.43e-06	13.48
30	1.e-5	9433	282990	2.67e-05	29.643
50	1.e-02	33555	1677750	5.55e+00	0.00
50	1.e-01	106109	5305450	1.30e-07	1.58

When $N = 20$ and ν=1.e-5, almost maximum Speedup of 19.89 is achieved. The Speedup starts decreasing rapidly at ν=1.e-3 .
At $N = 30$ and ν=1.e-5, we get maximal Speedup=29.64.
At $N = 50$, the scheme runs into trouble only passing the tolerance test at $\nu = 0.1$ with low Speedup of 1.58.

Figure 2: Pe=0.1: Initial square bump diffusing and advecting at low speed (movement too small to be visible)

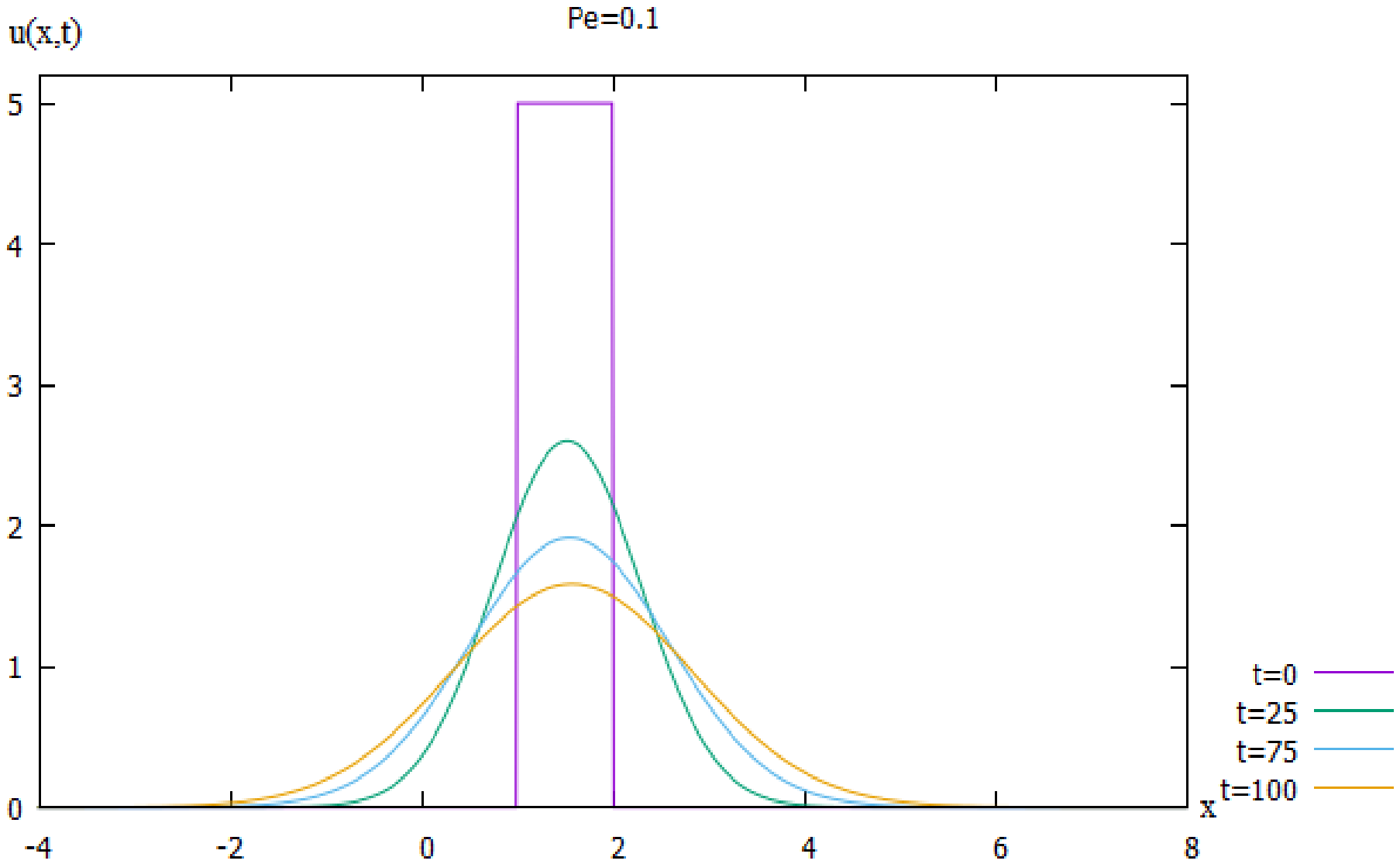


Table 3 Results I

Table: Pe=0.1, a=-4, b=8, MM=512

N	ν	nSupSteps	nSteps	Error	Speedup
1	0.0	524340	524340	3.54e-07	1.00
1	1.e-05	524345	524345	2.93e-07	1.00
1	1.e-01	576774	576774	4.16e-07	0.91
5	1.e-05	20981	104905	3.80e-04	5.00
10	1.e-05	5251	52510	3.00e-04	9.99

Table 3 shows results for Pe=0.1, our first advection-diffusion case with low velocity. For $N = 1$ larger ν values only reduce efficiency, without much improvement in error.
For $N = 10$ and ν=1.e-5, we also get full Speedup of 9.99.

Table 3 Results II

Table: Pe=0.1, a=-4, b=8, MM=512

N	ν	nSupSteps	nSteps	Error	Speedup
15	1.e-04	2401	36015	1.01e-04	14.56
20	1.e-03	1946	38920	3.52e-04	13.47
50	1.e-01	6633	331650	5.62e-06	1.58

The best Speedup is now 14.56 when $N = 15$ and ν=1.e-4.
At $N = 20$, the scheme starts to require a larger ν value to meet the error tolerance. Ideally, with $N = 20$, we hope to get a Speedup close to 20, but the best we get is 13.47 at ν=1.e-3.
For $N = 50$, the scheme fell apart entirely only remaining stable for $\nu = 0.1$ with Speedup=1.58. By introducing a small advective term with only $Pe = 0.1$, limitations on the speedup are already occurring.

Figure 3: Pe=1: Initial square bump diffusing and advecting at moderate speed (now movement is visible)

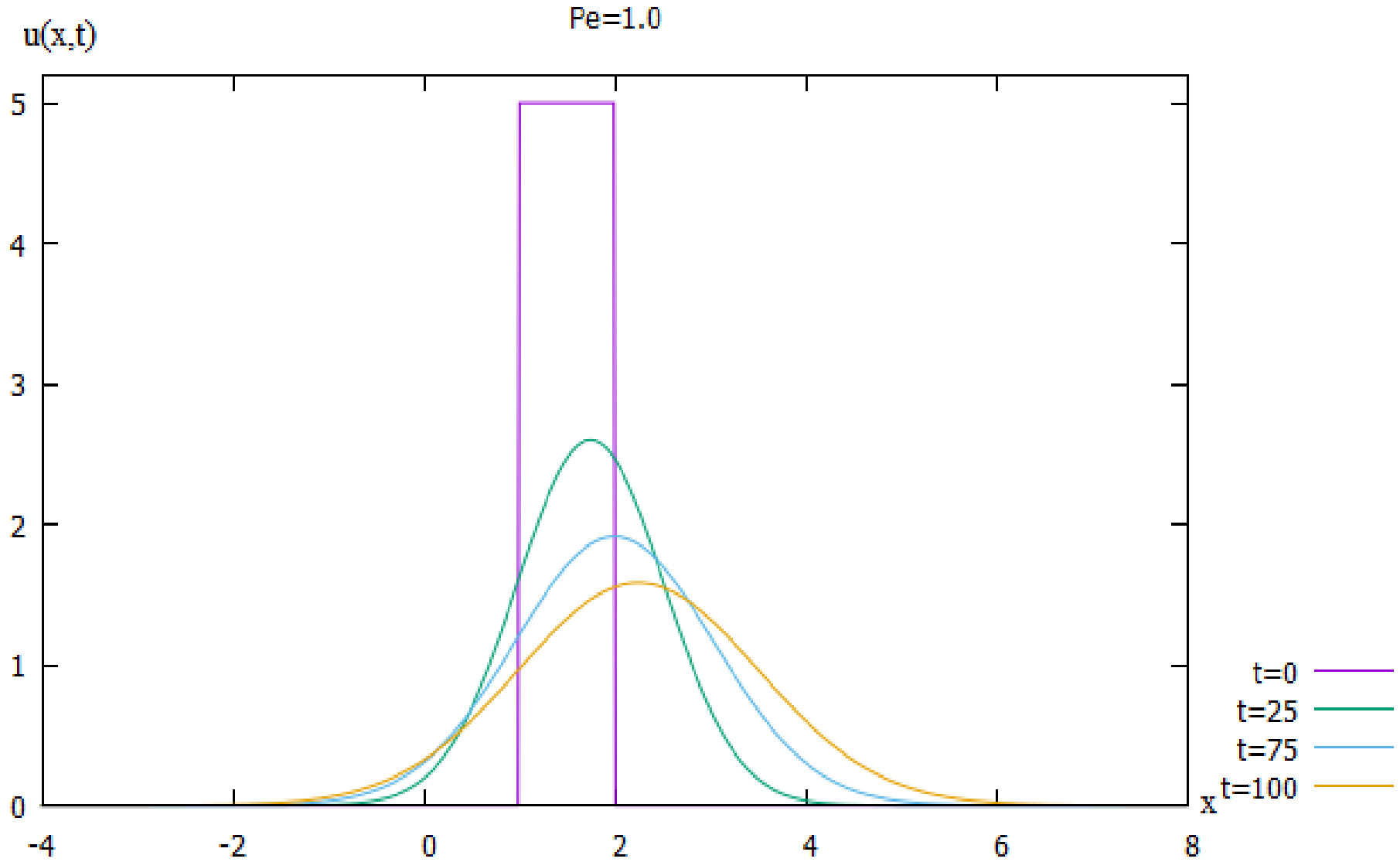


Table 4 Results

Table: Pe=1.0, a=-4, b=8, MM=512

N	ν	nSupSteps	nSteps	Error	Speedup
1	0.0	524801	524801	2.90e-06	1.00
1	1.e-05	524806	524806	4.28e-07	1.00
1	1.e-01	577281	577281	4.14e-07	0.91
5	1.e-05	21000	105000	6.84e-04	5.00
10	1.e-04	5319	53190	7.09e-04	9.87
20	1.e-03	1948	38960	5.79e-02	0.00
20	1.e-02	5252	105040	2.64e-04	5.00
50	1.e-01	6639	331950	2.09e+65	0.00

For Pe=1.0, $N = 1$ and 5 behave similar to Pe=0.1. When $N = 10$ with ν=1.e-4, Speedup=9.87, still giving approximately N times Speedup. Once $N = 20$, the scheme is unstable until ν=1.e-2 with Speedup=5. The scheme completely fails at $N = 50$, remains unstable for all ν tested.

Figure 4: Pe=10: Initial square bump diffusing and advecting at high speed

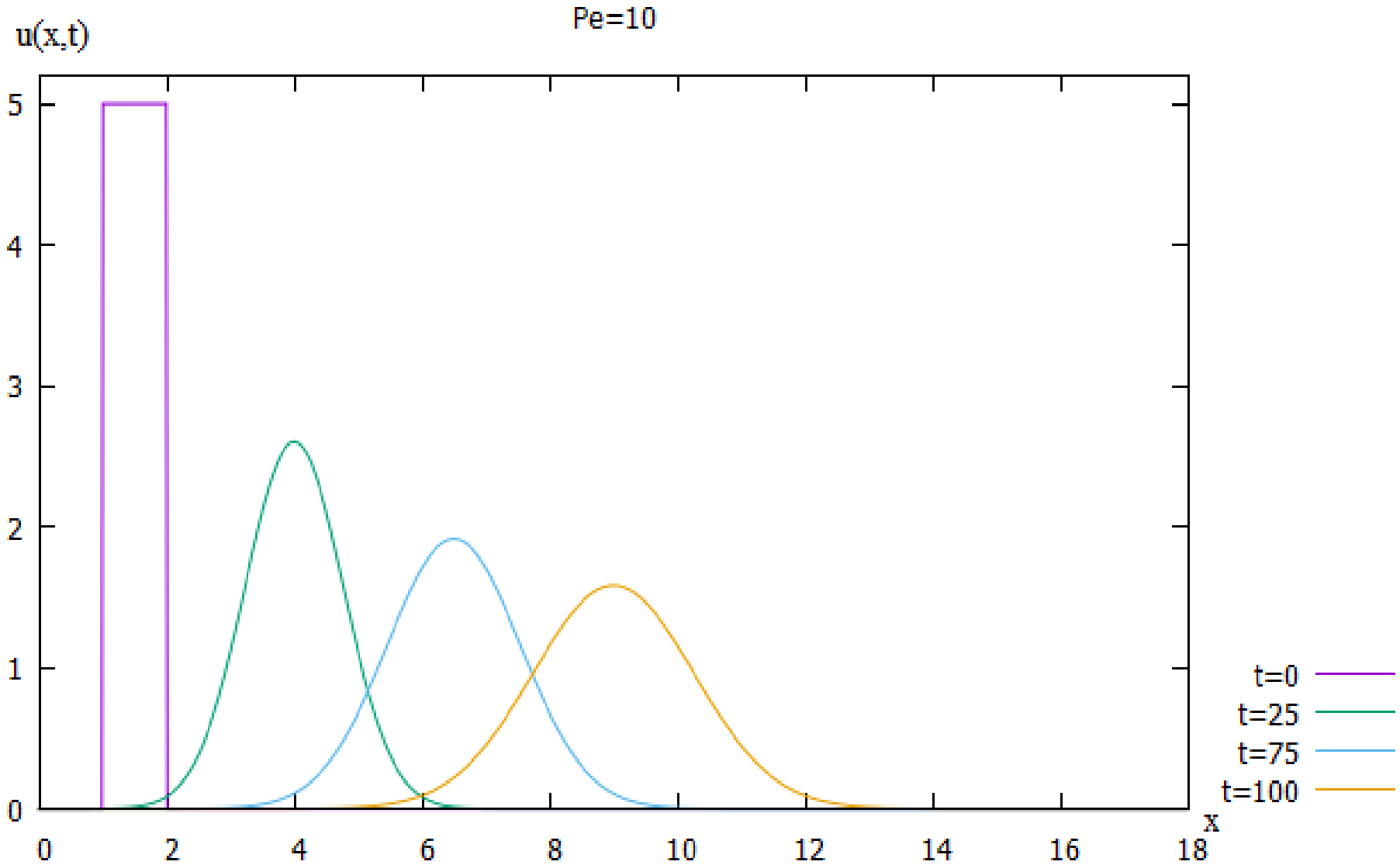


Table 5 Results

Table: Pe=10.0, a=0, b=18, MM=512

N	ν	nSupSteps	nSteps	Error	Speedup
5	1.e-03	21872	109360	6.87e-04	4.84
10	1.e-02	10979	109790	2.66e-04	4.82
10	1.e-01	33484	334840	2.37e-05	1.58
20	1.e-01	16742	334840	2.69e-05	1.58

Increasing the Peclet Number to 10, in Table 5, the best Speedup=4.84 occuring when $N = 5$ and ν=1.e-3. At $N = 10$, the scheme starts to fail with Speedup=4.82 for ν=1.e-2. At $N = 20$, the only success appears at $\nu = 0.1$ with Speedup=1.58.

Figure 5: Pe=100: Initial square bump diffusing and advecting at very high speed

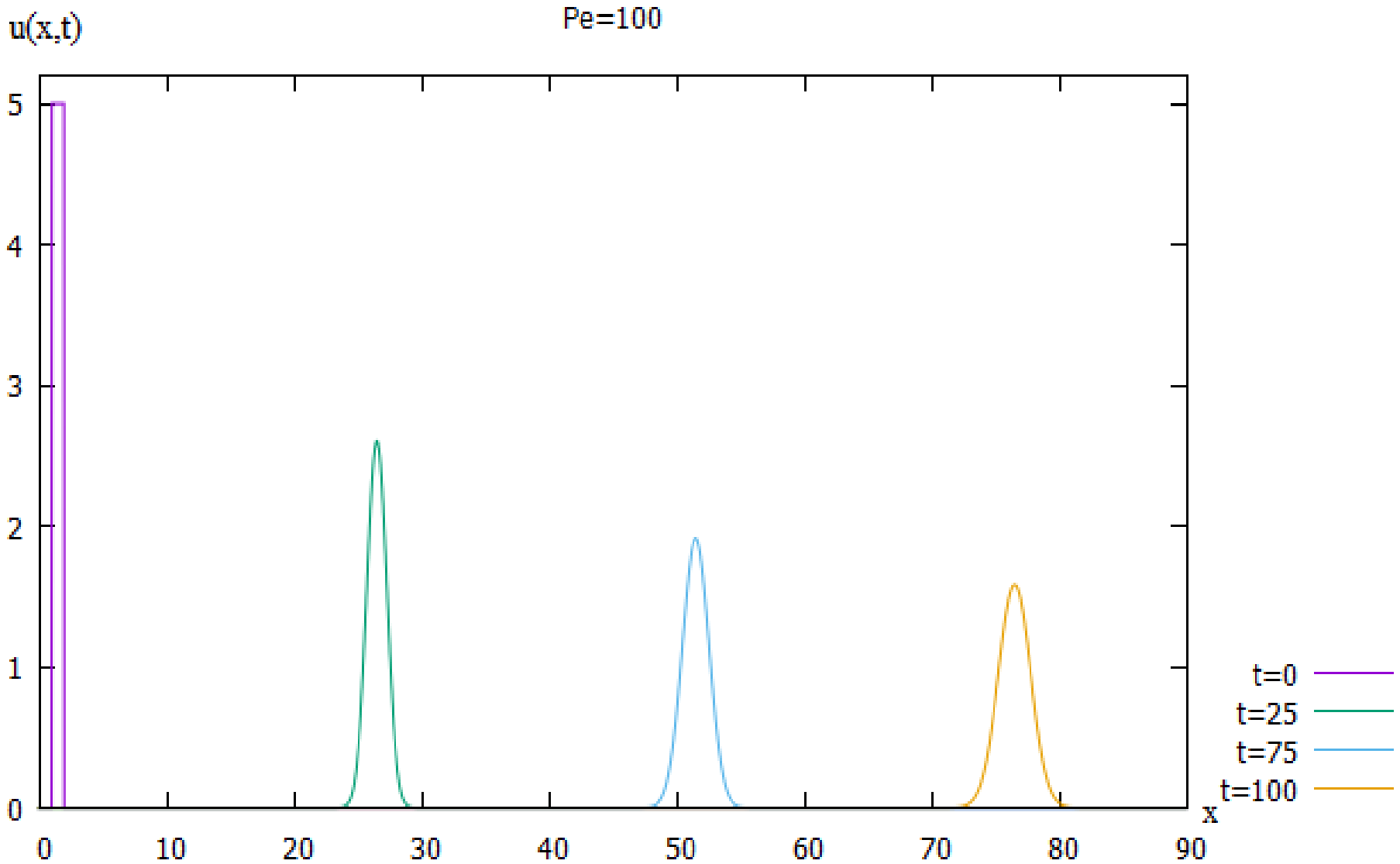


Table 6 Results I

Table: Pe=100.0, a=0, b=110, MM=512

N	ν	nSupSteps	nSteps	Error	Speedup
5	1.e-02	30171	150855	8.96e-03	0.00
5	1.e-01	73004	365020	2.16e-04	1.58
10	1.e-02	11934	119340	8.10e-02	0.00
10	1.e-01	36398	363980	2.23e-04	1.58

When $Pe = 100$, the scheme is very advective and achieves minimal *Speedup*. The scheme quickly gets into trouble with $N = 5$ for $\nu = 1.e-02, 0.1$ resulting in $error = 8.96e-3, 2.16e-4$. The scheme does achieve $Speedup = 1.58$ at $\nu = 0.1$, but it does not come close to N times faster. The same issue comes up with $N = 10$ and $\nu = 0.1$, $Speedup = 1.58$.

Table 6 Results II

Table: Pe=100.0, a=0, b=110, MM=512

N	ν	nSupSteps	nSteps	Error	Speedup
20	0.0	1440	28800	1.26e+267	0.00
20	1.e-03	2136	42720	1.01e+299	0.00
20	1.e-01	18200	364000	2.35e-04	1.58
50	0.0	231	11550	6.01e+172	0.00
50	1.e-04	303	15150	8.23e+216	0.00
50	1.e-03	732	36600	1.11e+293	0.00

The *error* quickly gets out of hand for $N = 20, 50$. The only *Speedup* occurs at $N = 20$ and $\nu = 0.1$ resulting in *Speedup* $= 1.58$.

Summary and Conclusions

The Super-Time-Stepping (STS) scheme is a very simple to implement, and the theory requires the matrix A to be symmetric positive definite. This requirement is fulfilled in the pure diffusion case (Pe=0), but not by advection-diffusion (Pe>0). By exploring the use of STS on a range of advection-diffusion problems, the theorerical requirement appears to be too strict in practice. For problems with Peclet Number up to Pe=1, STS is highly successful. It can accelerate the computation by a factor of almost N, up to $N = 10$. Speedup of 9.87 means 987% faster execution, at no cost, which is much higher than code optimazation or even parallel computing can normally achieve! Even in the highly advective case of Pe=10, we can get speedup of 4.8, meaning 480% faster execution!

References

[1] V. Alexiades and A.D. Solomon, **Mathematical Modeling of Melting and Freezing Processes**, Hemisphere Publishing (Taylor & Francis), Washington DC, 1993.
http://www.worldcatlibraries.org/wcpa/top3mset/26352070
[2] V. Alexiades, G. Amiez, P. Gremaud, Super-Time-Stepping Acceleration of Explicit Schemes For Parabolic Problems, *Communications in Numerical Methods in Engineering*, **12**: 31-42, 1996.
[3] R.J. LeVeque, **Finite Volume Methods for Hyperbolic Problems**, Cambridge University Press, 2002.
[4] P.K. Sweby, High Resolution Schemes Using Flux Limiters For Hyperbolic Conservation Laws, *SIAM Journal on Numerical Analysis* **21**(5): 995-1011, 1984.

Motivation

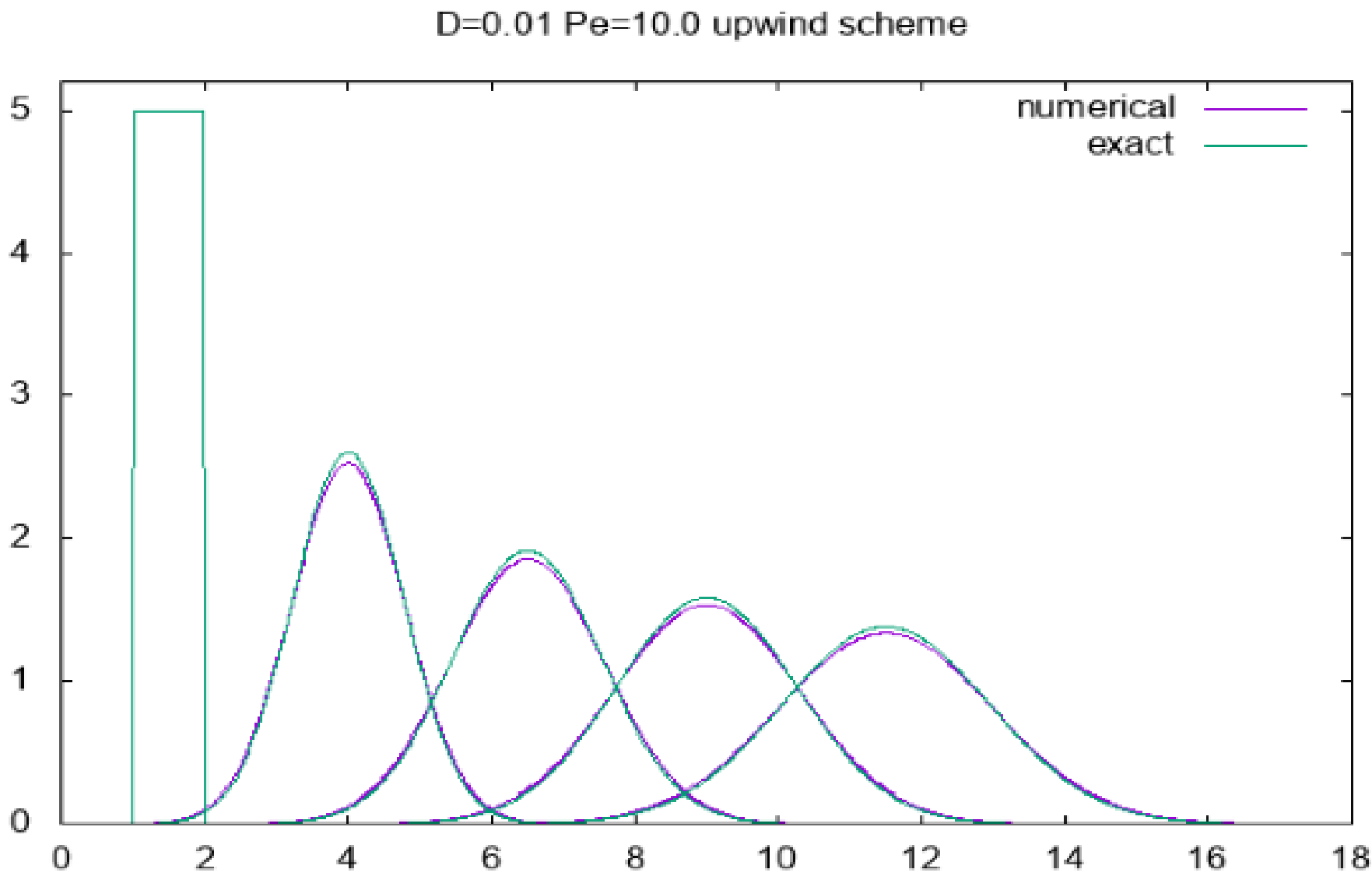
- Primarily motivated by [2]
- Speed is king: Nobody wants to wait forever on code

Experimental Need for Improvement I

Pe	Error at t=100
0.0	4.5501e-03
0.1	4.5736e-03
1.0	6.7585e-03
10.	4.5900e-02
100	2.2255e-01

Table: Upwind: D=0.01, MM=64

Experimental Need for Improvement II



Why MC method?

- Theoretically superior, see [3]
- Experimentally superior

Method	Error at t=100
Upwind	2.4362e-02
Lax-Wendroff	1.2243e-04
Minmod	1.5159e-04
Superbee	1.4859e-04
MC	3.4519e-05

Table: Pe=10, D=0.01, [a,b]=[0,18], MM=128

Why MC Method? (Error Plot Pe=10, D=0.01, t=100)