Super-Time-Stepping for Advection-Diffusion

James MacPherson
Advisor: Dr. V. Alexiades

Abstract

The goal of this paper is to explore the effectiveness of Super-Time-Stepping (STS) scheme on advection-diffusion problems. We present the explicit time-stepping Finite Volume scheme for conservation laws, and outline the idea and implementation of the STS scheme that accellerates the explicit scheme. We test its effectiveness for pure diffusion and for advection-diffusion. We conclude that STS can also accelerate advection-diffusion, but the more advection-dominated the process is, the less effective the STS scheme becomes.

1 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.

The content of this report is as follows. In section 2, we first present general conservation laws and advection-diffusion processes, and then, in §2.2, we describe the Finite Volume discretization of conservation law and of diffusion and advection fluxes. In §2.3, higher resolution methods are outlined based on flux-limiters. The STS scheme is outlined in section 3. In section 4 we present an explicitly solvable pure diffusion problem and test STS on it, verifying it performs as theory predicts. In section 5 we derive the exact solution of a square bump propagating under advection-diffusion and test STS for a range of Peclet Numbers, from purely diffusive to highly advective. Conclussions are discussed in section 6.

2 Finite Volume Discretization of Conservation Law

2.1 Conservation Law and Advection-Diffusion

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.

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.

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 V and D.

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.

2.2 Finite Volume Discretization

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 $x_{i+1/2} = x_i + \Delta x/2$.

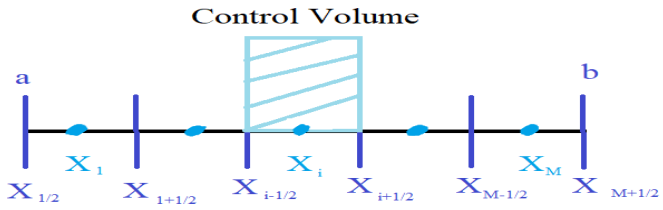

Figure 1: Mesh with M control volumes on $[a, b]$.

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 $\Delta t = \frac{t_{end}-t_0}{\text{Nsteps}}$.

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 $[t_n, t_{n+1}]$.

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

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

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

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 $\bar{F}_{i-1/2}^n := \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} F(x_{i-1/2},t)dt$ = mean flux during (t_n, t_{n+1}) across face at $x_{i-1/2}$, we obtain

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

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

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

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

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

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

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

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

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 is

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

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

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 $U^n_0 = u_a(t_n)$ is known and the boundary flux is approximated by

$$F^n_{1/2} = VU^n_0 - D\frac{U^n_1 - U^n_0}{\Delta x/2}. \tag{2.11}$$

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

$$F^n_{M+1/2} = VU^n_M - D\frac{U^n_{M+1} - U^n_M}{\Delta x/2}. \tag{2.12}$$

2.3 Higher Resolution Methods

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]

$$U^{n+1}_i = U^n_i - \frac{\Delta t}{2\Delta x}V(U^n_{i+1} - U^n_{i-1}) + \frac{1}{2}(\frac{\Delta t}{\Delta x})^2V^2(U^n_{i-1} - 2U^n_i + U^n_{i+1}). \tag{2.13}$$

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

$$F^n_{i-1/2} = \frac{1}{2}V(U^n_{i-1} + U^n_i) - \frac{1}{2}\frac{\Delta t}{\Delta x}V^2(U^n_i - U^n_{i-1}). \tag{2.14}$$

The Lax-Wendroff scheme is of order two, but it does introduces unphysical oscillations (at discontinuities, such as shocks). To suppress oscillations one uses the technology of **flux-limiters** [3],[4]. The Flux Function (2.14) can be rewritten as:

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

The first term is the upwind flux, and the next term is the correction, which causes the oscillations in Lax-Wendroff. To suppress oscillations, the correction 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

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

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

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 limiiter methods can be found in [3] and [4].

3 The Super-Time-Stepping (STS) Scheme

We briefly outline the idea and implementation of the Super-Time-Stepping scheme referring to [2] for more 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

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

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 stability condition. For the one dimensional diffusion equation $u_t = Du_{xx}$, $\lambda_{max} = 4D/(\Delta x)^2$, which is (2.7).

In order to relax the stability 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

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

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.

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

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

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.

One can show the relation

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

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

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.

Implementing the STS scheme is very simple and can easily be inserted into the code for an explicit scheme. First need to 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. And repeat till the desired final time is reached.

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).

4 Performance on Pure Diffusion

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 $[0, \infty)$, with $u = 0$ initially, and $u = 1$ is imposed at $x = 0$:

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

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

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

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 $U_0^n = 1$) 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).

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.

We present runs with N=1, 2, 10, 20, 50, and for each N we test various small ν and list results for $\nu = 0$, 1.e-5, 1.e-3, 1.e-1 in Table 1.

Table 1: 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-3	20992492	20992492	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
5	1.e-3	866368	4331840	2.3250e-07	4.84
5	1.e-1	2660317	13301585	1.0883e-08	1.58
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-3	77790	1555800	8.92e-07	13.48
20	1.e-1	663178	13263560	1.05e-08	1.58
50	0.0	8389	419450	1.78e+07	0.00
50	1.e-5	8667	433350	2.10e+03	0.00
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

The best Speedup in Table 1 occurs for $N = 20$, $\nu = 0.00001$, almost 20 times faster than Euler.

For $N = 1$, even using 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=1.e-5. 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 the speedup drops. At $N = 20$, the results are very similar. The value ν=0.00001 gives speedup of 19.89. As ν increases, the speedup slows down just as in the $N = 10$ case. For $N = 50$, a large ν is needed for acceptable error. At ν=0.1, the scheme gives speedup of only 1.58.

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.

5 Performance on Advection-Diffusion

5.1 An explicitly solvabe advection-diffusion problem

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

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

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

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

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. 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

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

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

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

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

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

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

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

which solves (5.1) with initial profile the square bump

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

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

$$\begin{cases} U_0^n = \frac{A}{2} [\text{erf}(\frac{a - x_1 - Vt_n}{\sqrt{4Dt_n}}) - \text{erf}(\frac{a - x_2 - Vt_n}{\sqrt{4Dt_n}})] \\ U_{M+1}^n = \frac{A}{2} [\text{erf}(\frac{b - x_1 - Vt_n}{\sqrt{4Dt_n}}) - \text{erf}(\frac{b - x_2 - Vt_n}{\sqrt{4Dt_n}})]. \end{cases} \tag{5.11}$$

5.2 STS on advection-diffusion problem

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.

We present results for various values of the Peclet Number: 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 shows the evolution of the square bump when Pe=0.0. Since V=0 here, there is no advective term, so the peak remains where it started. As time passes, the peak decreases and the bump spreads out by diffusion.

When 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. The best speedup occurs for $N = 30$ with ν=1.e-5 achieving Speedup=29.64. Note that for $N = 1$, changing $\nu = 0$ to ν=1.e-5 dramatically

Table 2: 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-5	8388693	8388693	3.13e-08	1.00
1	1.e-4	8389448	8389448	3.13e-08	1.00
1	1.e-3	8396998	8396998	3.12e-08	1.00
1	1.e-2	8472495	8472495	3.07e-08	0.99
1	1.e-1	9227470	9227470	2.61e-08	0.91
5	0.0	335545	1677725	5.18e-03	0.00
5	1.e-5	335656	1678280	9.39e-07	5.00
5	1.e-4	336652	1683260	9.35e-07	4.98
5	1.e-3	346548	1732740	8.97e-07	4.84
5	1.e-2	439771	2198855	6.31e-07	3.81
5	1.e-1	1064128	5320640	1.28e-07	1.58
10	0.0	83887	838870	7.94e-03	0.00
10	1.e-5	83999	839990	3.77e-06	9.99
10	1.e-4	85000	850000	3.71e-06	9.87
10	1.e-3	94760	947600	3.17e-06	8.85
10	1.e-2	173949	1739490	1.20e-06	4.82
10	1.e-1	530545	5305450	1.31e-07	1.58
20	0.0	20973	419460	2.07e-02	0.00
20	1.e-5	21084	421680	1.50e-05	19.89
20	1.e-4	22079	441580	1.41e-05	19.00
20	1.e-3	31117	622340	8.43e-06	13.48
20	1.e-2	83942	1678840	1.40e-06	5.00
20	1.e-1	265272	5305440	1.31e-07	1.58
30	0.0	9322	279660	3.36e-02	0.000
30	1.e-5	9433	282990	2.67e-05	29.643
30	1.e-4	10414	312420	2.96e-05	26.850
50	0.0	3356	167800	5.70e+07	0.00
50	1.e-5	3468	173400	7.10e+05	0.00
50	1.e-4	4407	220350	4.86e+04	0.00
50	1.e-3	10650	532500	4.11e+01	0.00
50	1.e-2	33555	1677750	5.55e+00	0.00
50	1.e-1	106109	5305450	1.30e-07	1.58

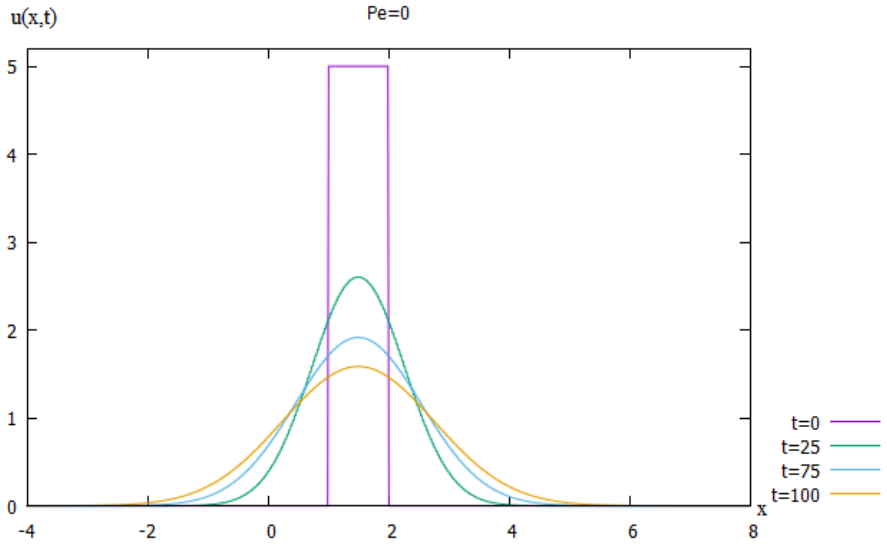

Figure 2: Pe=0: Diffusion of initial square bump.

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. At $N = 10$ and ν=1.e-5, we get maximal Speedup=9.99. 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 even better 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.

The diffusivity (D=0.01) is constant in all cases below, so the plots in the following figures will flatten over time like in Figure 1.

Table 3 shows results for Pe=0.1, our first advection-diffusion case, with low velocity (V=0.001). The best Speedup is now 14.56 with $N = 15$ and ν=1.e-4. For $N = 10$ and ν=1.e-5, we also get full Speedup of 9.99. 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. Larger ν values only reduce efficiency, without much improvement in error. With $N = 30$, could only get Speedup=1.58 at ν=1.e-1. For $N = 50$, the scheme remained stable only for $\nu = 0.1$ with Speedup=1.58.

We see that by introducing a small advective term (with $Pe = 0.1$) limitations on the speedup are already occurring.

In Figure 2, Pe=0.1 introduces a small advection term; while not visible, the bump is actually moving to the right.

In Figure 3, where Pe=1, the plot is visibly moving to the right.

In Table 4, where Pe=1, $N = 1$ and 5 behave similarly to Pe=0.1 case.

Table 3: 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-5	524345	524345	2.93e-07	1.00
1	1.e-4	524393	524393	3.38e-07	1.00
1	1.e-3	524865	524865	4.56e-07	1.00
1	1.e-2	529584	529584	4.86e-07	0.99
1	1.e-1	576774	576774	4.16e-07	0.91
5	0.0	20975	104875	2.29e-02	0.00
5	1.e-5	20981	104905	3.80e-04	5.00
5	1.e-4	21044	105220	1.30e-05	4.98
5	1.e-3	21662	108310	1.42e-05	4.84
5	1.e-2	27489	137445	1.02e-05	3.81
5	1.e-1	66515	332575	2.06e-06	1.58
10	0.0	5244	52440	6.37e-01	0.00
10	1.e-5	5251	52510	3.00e-04	9.99
10	1.e-4	5314	53140	3.82e-05	9.87
10	1.e-3	5924	59240	4.91e-05	8.85
10	1.e-2	10874	108740	1.93e-05	4.82
10	1.e-1	33163	331630	2.10e-06	1.58
15	1.e-4	2401	36015	1.01e-04	14.56
20	0.0	1312	26240	5.02e+14	0.00
20	1.e-5	1319	26380	1.26e+10	0.00
20	1.e-4	1381	27620	2.85e-02	0.00
20	1.e-3	1946	38920	3.52e-04	13.47
20	1.e-2	5248	104960	3.20e-05	5.00
20	1.e-1	16582	331640	2.08e-06	1.58
30	1.e-2	3497	104910	3.81e-02	0.00
30	1.e-1	11055	331650	1.99e-06	1.58
50	1.e-1	6633	331650	5.62e-06	1.58

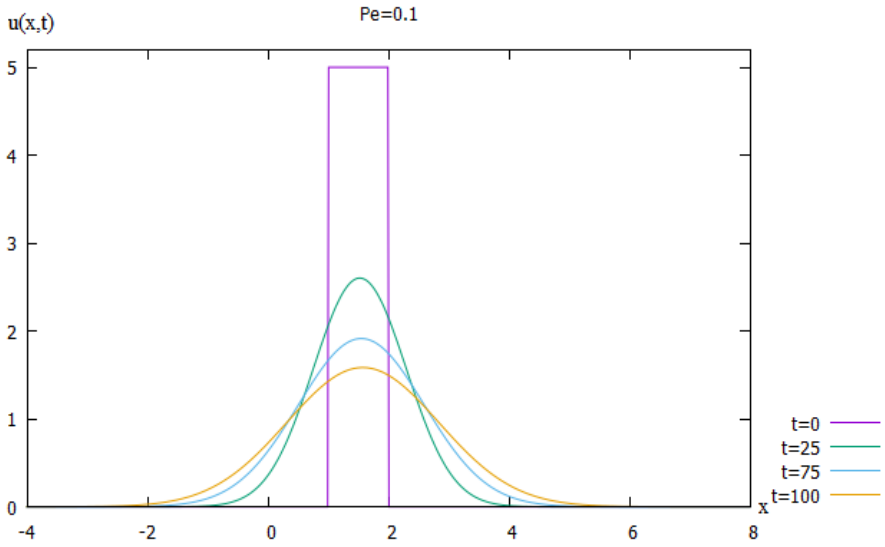

Figure 3: Pe=0.1: Initial square bump diffusing and advecting at low speed.

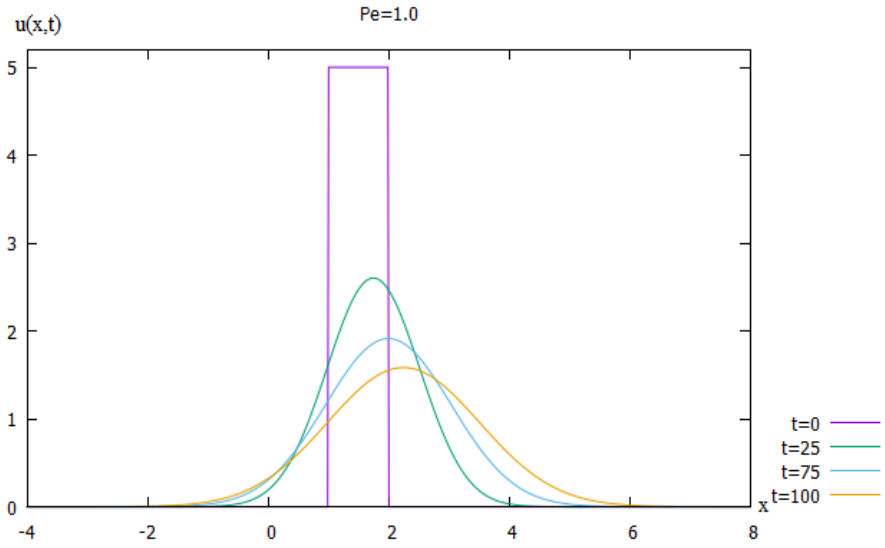

Figure 4: Pe=1: Initial square bump diffusing and advecting at moderate speed.

Table 4: 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-5	524806	524806	4.28e-07	1.00
1	1.e-4	524853	524853	4.29e-07	1.00
1	1.e-3	525326	525326	4.35e-07	1.00
1	1.e-2	530049	530049	4.76e-07	0.99
1	1.e-1	577281	577281	4.14e-07	0.91
5	0.0	20993	104965	1.16e-03	0.00
5	1.e-5	21000	105000	6.84e-04	5.00
5	1.e-4	21062	105310	3.55e-04	4.98
5	1.e-3	21681	108405	2.24e-05	4.84
5	1.e-2	27513	137565	1.62e-05	3.81
5	1.e-1	66574	332870	3.01e-06	1.58
10	0.0	5249	52490	1.71e+01	0.00
10	1.e-5	5256	52560	2.78e-03	0.00
10	1.e-4	5319	53190	7.09e-04	9.87
10	1.e-3	5929	59290	8.58e-05	8.85
10	1.e-2	10883	108830	3.15e-05	4.82
10	1.e-1	33192	331920	3.09e-06	1.58
20	0.0	1313	26260	7.29e+299	0.00
20	1.e-5	1320	26400	2.00e+300	0.00
20	1.e-4	1382	27640	1.79e+272	0.00
20	1.e-3	1948	38960	5.79e-02	0.00
20	1.e-2	5252	105040	2.64e-04	5.00
20	1.e-1	16597	331940	3.32e-06	1.58
30	1.e-1	11065	331950	5.04e-06	1.58
50	1.e-1	6639	331950	2.09e+65	0.00

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-4 with Speedup=5. With $N = 30$, could only get Speedup=1.58 at ν=1.e-1. The scheme completely fails at $N = 50$, it remains unstable for all ν tested.

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.

Table 5: Pe=10.0, a=0, b=18, MM=512

N	ν	nSupSteps	nSteps	Error	Speedup
1	0.0	529409	529409	2.39e-06	1.00
1	1.e-5	529414	529414	2.13e-06	1.00
1	1.e-4	529462	529462	2.13e-06	1.00
1	1.e-3	529938	529938	2.11e-06	1.00
1	1.e-2	534703	534703	2.00e-06	0.99
1	1.e-1	582350	582350	1.02e-06	0.91
5	0.0	21177	105885	1.18e+12	0.00
5	1.e-5	21184	105920	5.92e+14	0.00
5	1.e-4	21247	106235	5.63e-03	0.00
5	1.e-3	21872	109360	6.87e-04	4.84
5	1.e-2	27755	138775	1.31e-04	3.81
5	1.e-1	67158	335790	2.28e-05	1.58
10	0.0	5295	52950	2.43e+45	0.00
10	1.e-5	5302	53020	6.29e+40	0.00
10	1.e-4	5365	53650	4.84e-01	0.00
10	1.e-3	5981	59810	1.39e-03	0.00
10	1.e-2	10979	109790	2.66e-04	4.82
10	1.e-1	33484	334840	2.37e-05	1.58
20	1.e-1	16742	334840	2.69e-05	1.58

In Figure 4, Pe=10 ($V = 0.1$), so the bump moves 2.5 units to the right after each time interval of 25 sec, while diffusing.

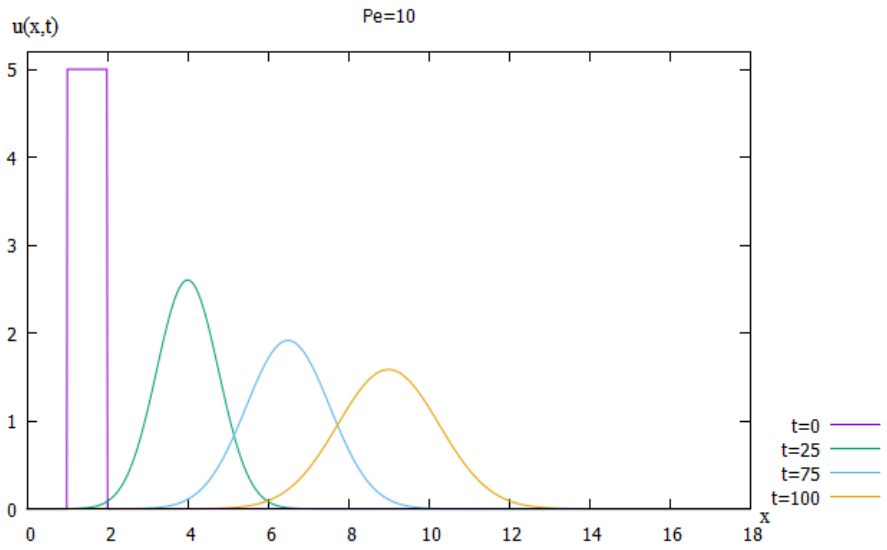

Figure 5: Pe=10: Initial square bump diffusing and advecting at high speed.

Table 6: Pe=100.0, a=0, b=110, MM=512

N	ν	nSupSteps	nSteps	Error	Speedup
1	0.0	575489	575489	1.77e-05	1.00
1	1.e-5	575495	575495	1.56e-05	1.00
1	1.e-4	575547	575547	1.56e-05	1.00
1	1.e-3	576064	576064	1.55e-05	1.00
1	1.e-2	581244	581244	1.44e-05	0.99
1	1.e-1	633038	633038	5.02e-06	0.91
5	1.e-2	30171	150855	8.96e-03	0.00
5	1.e-1	73004	365020	2.16e-04	1.58
10	1.e-2	11934	119340	8.10e-02	0.00
10	1.e-1	36398	363980	2.23e-04	1.58
20	0.0	1440	28800	1.26e+267	0.00
20	1.e-5	1447	28940	3.10e+264	0.00
20	1.e-4	1516	30320	1.76e+276	0.00
20	1.e-3	2136	42720	1.01e+299	0.00
20	1.e-1	18200	364000	2.35e-04	1.58
50	0.0	231	11550	6.01e+172	0.00
50	1.e-5	239	11950	1.01e+158	0.00
50	1.e-4	303	15150	8.23e+216	0.00
50	1.e-3	732	36600	1.11e+293	0.00

When Pe=100 (in Table 6), the process is very advective and STS achieves minimal Speedup. For $N = 1$, increasing ν improves the error at the cost of speedup. The scheme quickly gets into trouble with $N = 5$ for ν=1.0e-02 or 0.1, resulting in large *error*=8.96e-03 or 2.16e-04. The scheme does achieve Speedup=1.58 at ν=0.1, but it does not come close to N times faster. The same issue comes up with $N = 10$ and ν=0.1 where Speedup=1.58. The error quickly gets out of hand for N=20 and 50. The only speedup occurs at $N = 20$ with ν=0.1, resulting in Speedup=1.58.

In the highly advectice case of Pe=100 (so V=1.0) shown in Figure 5, the peak moves the expected 25 units to the right after each time interval of 25 sec, while diffusing.

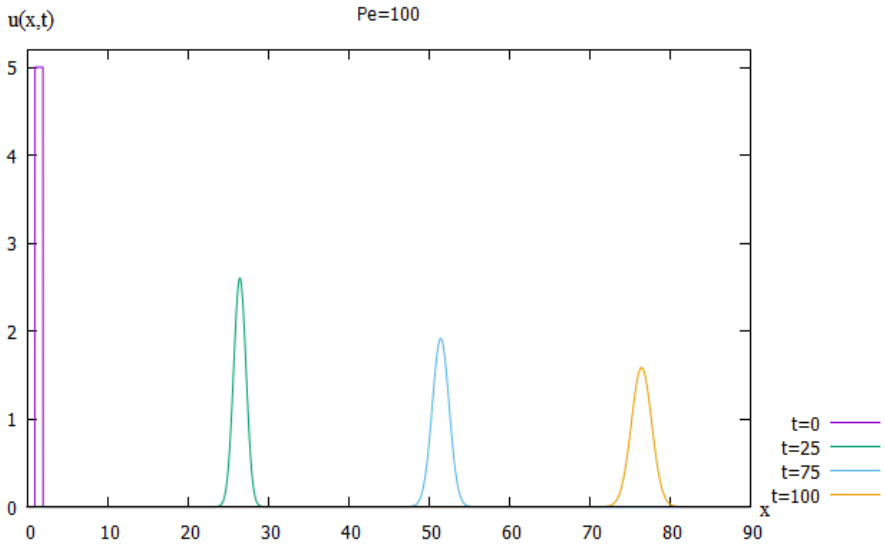

Figure 6: Pe=100: Initial square bump diffusing and advecting at very high speed.

6 Summary and Conclusions

The Super-Time-Stepping (STS) scheme is a very simple to implement. Its 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 $N = 10$. Note that 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!

7 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.