An Alternating Direction Explicit (ADE) Scheme for Time-Dependent Evolution Equations

Shingyu Leung* and Stanley Osher†

June 9, 2005

Abstract

We propose an explicit finite difference scheme to solve nonlinear time-dependent equations. While applying to linear equations, this scheme is proven to be second order accurate in time and unconditionally stable for arbitrary time step size. Unlike the Crank-Nicloson scheme or other implicit schemes, however, this scheme can easily be implemented for a wide range of equations. We will give numerical examples to demonstrate the simplicity and the computational efficiency of the method.

1 Introduction

Time-dependent evolution equations are important in modeling many physical phenomena. When there is an evolution of some physical property, denoted by $T(t, \mathbf{x})$, one can model the change in this quantity by the following evolution equation,

$$\frac{\partial T(t,\mathbf{x})}{\partial t} = \mathcal{L}[T(t,\mathbf{x})]\,, \tag{1}$$

with some differential operator $\mathcal{L}$. For simple cases when this operator is linear, the resulting time-dependent evolution equations can be solved relatively easily. Over the years, there have been a large number of schemes proposed [16].

On the other hand, for nonlinear operator $\mathcal{L}$, it is still not clear how to solve the resulting nonlinear evolution equation efficiently. One possible way is to follow a similar approach to linear cases. Using the idea of method of line, we can first discretize the operator $\mathcal{L}$ and get a system of ODE,

$$\frac{dT}{dt} = A(T)T\,, \tag{2}$$

where $A(T)$ is a nonlinear operator involving T, and $T = (T_1, T_2, \ldots, T_N)^T$ with $T_i = T(x_i)$.

Different ways to treat the term $A(T)T$ yield different numerical methods. The simplest one is the explicit scheme, which discretizes the equation in this way

$$T^{n+1} = [I + \Delta t\, A(T^n)]T^n\,, \tag{3}$$

where I is the identity operator. As is well-known, such an explicit scheme makes for ease in programming with a code for an explicit scheme usually containing only a few lines even for high dimension problems. However, there is a trade-off in computational time coming from the stability requirement. Inherited from the explicit scheme for solving the equations with linear $\mathcal{L}$, this explicit scheme for nonlinear equation also has a stability

*Department of Mathematics, UCLA, Los Angeles, CA 90095-1555. Email: **syleung@math.ucla.edu**
†Department of Mathematics, UCLA, Los Angeles, CA 90095-1555. Email: **sjo@math.ucla.edu**

condition on the time step size one can use. In many cases, this condition could be relatively restrictive and this could make the method impractical.

One way to relax this stability restriction is to use the method of regularization, sometimes called the method of precondition. In [15], the author has proposed stabilizing the evolution equation by adding and subtracting a Laplacian operator. For example, it is possible to introduce a new operator by rewriting the evolution equation as

$$\frac{\partial T(t,\mathbf{x})}{\partial t} = \beta \Delta T + \tilde{\mathcal{L}}(T)\,, \tag{4}$$

with $\tilde{\mathcal{L}} = \mathcal{L} - \beta\Delta$. This modified equation could then be solved by

$$\frac{T_i^{n+1} - T_i^n}{\Delta t} = \frac{\beta}{2}(\Delta T^{n+1} + \Delta T^n) + \tilde{\mathcal{L}}(T^n)\,. \tag{5}$$

One advantage of doing these manipulations is that the resulting numerical scheme is unconditionally stable for arbitrary time step size. This idea may also be applied to a wide range of nonlinear evolution equations. However, the main drawback of this approach is that it is not clear how to relate the parameter β and the nonlinear operator $\mathcal{L}$.

Other than regularization, the time step restriction may also be relaxed by using implicit, or semi-implicit, schemes. The most-used representatives from this class are motivated by applying the backward Euler scheme or the Trapezoidal rule in time. The schemes are given by

$$T^{n+1} = [I - \Delta t\, A(T^n)]^{-1} T^n\,, \tag{6}$$

and (Crank-Nicloson)

$$T^{n+1} = \left[I - \frac{\Delta t}{2}\, A(T^n)\right]^{-1} \left[I + \frac{\Delta t}{2}\, A(T^n)\right] T^n\,. \tag{7}$$

Even though these numerical schemes are unconditionally stable, in many cases, especially in high dimensional problems, it is not trivial how to invert the operator $(I - \lambda\, A)$ computationally efficiently.

There are two general approaches to invert the operator $(I - \lambda\, A)$. The first group uses Multiplicative Operator Splitting (MOS) methods. A common feature of these methods is splitting the complicated operator into a product of some simple operators, so that each of these operators can easily be inverted. Mathematically, we approximate equation (2) by

$$T^{n+1} = \prod_{k=1}^{K} [I - \Delta t A_k(T^n)]^{-1} T^n\,. \tag{8}$$

The second group of methods is called Additive Operator Splitting (AOS) methods [17, 2, 10]. The idea is to split the operator $(I - \Delta t\, A)$ into a summation, rather than multiplication as in MOS, of some simple operators. Mathematically, we have

$$T^{n+1} = \frac{1}{m} \sum_{k=1}^{K} \left[I - \Delta t A_k(T^n)\right]^{-1} T^n\,. \tag{9}$$

There are a few advantages of using AOS schemes over MOS schemes. Perhaps the most important one is that we can easily apply parallel computing. Computationally, each simple operator can first be inverted by a different processor separately. Then T^{n+1} can be computed by summing over all these solutions. This makes the inversion highly efficient.

However, both of these types of operator splitting schemes require a clever choice of splitting

$$A(T^n) = \sum_{k=1}^{K} A_k(T^n)\,. \tag{10}$$

The method for splitting might be equation-specific or might not be easy to generalize in high dimensions.

In this paper, we propose applying a simple operator splitting method based on the so-called Alternative Direction Explicit (ADE) scheme [7, 1]. Similar to the AOS schemes, this proposed numerical scheme can be easily parallelized. More importantly, this ADE scheme can not only be implemented in a way similar to an explicit method, but is also unconditionally stable similar to an implicit method.

The paper is organized as follows. In Section 2, we first revisit the Alternative Direction Explicit (ADE) scheme proposed in [7, 1], which was developed to solve the heat equation. Then we will generalize the scheme in Section 3 to solve other evolution equations by deriving an explicit expression to split the operator A. In the same section, we will also show the accuracy and the stability of applying the ADE scheme to a wider range of $\mathcal{L}$. In Section 4, we use some numerical examples to demonstrate the simplicity of applying the ADE scheme to higher order and nonlinear equations in both one- and two-dimensional spaces. Finally in Section 5, we give numerical examples to show the improvement in the time step size.

2 The ADE Scheme for the Heat Equation

In this section, we review the previous analysis in [7, 1] applying the ADE scheme to solve the heat equation. For simplicity, we only deal with the one-dimensional case. Similar analysis can be applied to higher dimensional equations.

The following ADE scheme was first used to solve the heat equation

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} \tag{11}$$

with the boundary conditions $T(x_0) = T_0$ and $T(x_{N+1}) = T_{N+1}$.

Given the solution T^n at $t = t^n$, we use the following algorithm to determine T^{n+1}, defined at t^{n+1}, on grid point $x = x_i$ with $i = 1, 2, \ldots, N$,

1. Set $u_i^n = T_i^n$ and $v_i^n = T_i^n$ for $i = 1, 2, \ldots, N$;
2. For $i = 1, 2, \ldots, N$, solve

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{u_{i-1}^{n+1} - u_i^{n+1} - u_i^n + u_{i+1}^n}{\Delta x^2}\,; \tag{12}$$

3. For $i = N, N-1, \ldots, 1$, solve

$$\frac{v_i^{n+1} - v_i^n}{\Delta t} = \frac{v_{i-1}^n - v_i^n - v_i^{n+1} + v_{i+1}^{n+1}}{\Delta x^2}\,; \tag{13}$$

4. Compute

$$T_i^{n+1} = \frac{u_i^{n+1} + v^{n+1}}{2}\,. \tag{14}$$

Notice that according to the index-sweeping-like procedures in Step 2 and Step 3, both u_i^{n+1} and v_i^{n+1} can be computed explicitly. When we calculate u_i^{n+1} in Step 2, because the index is in the ascending order, we have already known the value u_{i-1}^{n+1} on the right hand side of (12). Similarly, when we calculate v_i^{n+1} in Step 3, because the index is now in the descending order, we have already known the value v_{i+1}^{n+1} on the right hand side of (13). As a result, the whole method is fully explicit.

Other than its explicitness, another interesting property of the ADE scheme concerns the accuracy. In truncational analysis, although the accuracies in computing u and v in Step 2 and Step 3 are both $O[\Delta t^2, \Delta x^2, (\Delta t/\Delta x)^2]$, the averaging step in Step 4 will

cancel out all the terms in the form of $(\Delta t/\Delta x)^r$. Therefore, the overall scheme will have the following leading order terms from the truncational analysis

$$-\frac{\Delta t^2}{12}T_{ttt} - \frac{\Delta x^2}{12}T_{xxxx}\,. \tag{15}$$

Perhaps the most important property is that the Von Neumann stability analysis shows that this numerical scheme is unconditionally stable for all $\Delta t > 0$.

3 Operator Form

In this section, we will generalize the above technique to a wider class of linear equations in higher dimensions. Given a linear time-dependent equation,

$$\frac{\partial T}{\partial t} = \mathcal{L}(T)\,, \tag{16}$$

we first discretize the right hand side of the equation, which implies

$$\frac{\partial T}{\partial t} = AT + b\,. \tag{17}$$

Now, we decompose the matrix A into $L+D+U$, where D is the diagonal part of A, and L and U are the strictly lower and the strictly upper triangular part of A, respectively. Further, defining B and C by

$$B = L + \frac{1}{2}D \;\text{ and }\; C = U + \frac{1}{2}D\,, \tag{18}$$

we can express the ADE scheme as

$$\begin{aligned} T^{n+1} &= \frac{1}{2}\left[(I-\Delta tB)^{-1}(I+\Delta tC) + (I-\Delta tC)^{-1}(I+\Delta tB)\right]T^n + \\ &\quad \frac{1}{2}\left[(I-\Delta tB)^{-1} + (I-\Delta tC)^{-1}\right]b\,. \end{aligned} \tag{19}$$

The matrices $(I \pm \Delta tB)$ and $(I \pm \Delta tC)$ are all either lower triangular or upper triangular. Thus their corresponding inverses can be found easily by using simple forward or backward substitution. In the algorithm mentioned in Section 2, these two substitutions are simply done by switching the indexing direction. Therefore, the computational complexity for each time step would be the same as the standard explicit scheme, i.e. $O(N)$ where N is the number of grid points in the space direction.

For the rest of this section, we will use these matrix notations to analyze the accuracy and the stability of the ADE scheme.

Theorem 3.1. *If the diagonal elements of A are all non-positive, the ADE scheme (19) is second order accurate in time.*

Proof. For simplicity, we take $b = 0$. Because the diagonal elements of A are all non-positive, the matrices $(I - \Delta tB)$ and $(I - \Delta tC)$ in (19) are both nonsingular. Therefore, we can expand these inverses and get

$$\begin{aligned} T^{n+1} &= \frac{1}{2}\left\{\left[I + \sum_{m=1}^{\infty}\Delta t^m(B^m + B^{m-1}C)\right] + \left[I + \sum_{m=1}^{\infty}\Delta t^m(C^m + C^{m-1}B)\right]\right\}T^n \\ &= \left\{I + \Delta t\,A + \frac{\Delta t^2}{2}\,A^2 + \frac{1}{2}\Delta t^3\,(B^3 + B^2C + C^2B + C^3) + O(\Delta t^4)\right\}T^n\,. \end{aligned} \tag{20}$$

We complete the proof by comparing this expression with the exact solution of (17), given by

$$\begin{aligned} T^{n+1} &= \exp(\Delta t\,A)T^n \\ &= \sum_{m=0}^{\infty}\frac{1}{m!}(\Delta t\,A)^m T^n\,. \end{aligned} \tag{21}$$

□

Here we give two proofs of the stability property of the ADE scheme.

Theorem 3.2. *If A is symmetric negative definite, the ADE scheme (19) is unconditionally stable.*

Proof. For simplicity, we will prove only the stability of the operator $(I - \Delta t\, B)^{-1}(I + \Delta t\, C)$. The stability of $(I-\Delta t\, C)^{-1}(I+\Delta t\, B)$ can be proven similarly. And, the averaging of these two operators will still be stable.

Defining $\tilde{B} = I - \Delta t\, B$ and $\tilde{C} = I + \Delta t\, C$, we have

$$\tilde{B}^T + \tilde{B} = 2I - \Delta t\,(B + B^T) = 2I - \Delta t\, A\,. \tag{22}$$

Because $-A$ is positive definite, we can define $\|.\|_{-A}$ as the norm induced by the matrix $-A$. The induced norm of $\tilde{B}^{-1}\tilde{C}$ is then given by

$$\|\tilde{B}^{-1}\tilde{C}\|^2_{-A} = \sup_{x\neq 0} \frac{(\tilde{B}^{-1}\tilde{C}x, -A\tilde{B}^{-1}\tilde{C}x)}{(x, -Ax)} = \sup_{x\neq 0} \frac{(x, -\tilde{C}^T\tilde{B}^{-T}A\tilde{B}^{-1}\tilde{C}x)}{(x, -Ax)}\,. \tag{23}$$

Replacing C by $(A - B)$, the matrix $\tilde{C}^T\tilde{B}^{-T}A\tilde{B}^{-1}\tilde{C}$ can be rewritten as

$$\begin{aligned}
\tilde{C}^T\tilde{B}^{-T}A\tilde{B}^{-1}\tilde{C} &= (I - \Delta t\, B + \Delta t\, A)^T\tilde{B}^{-T}A\tilde{B}^{-1}(I - \Delta t\, B + \Delta t\, A)\\
&= A + \Delta t\, A(I - \Delta t\, B)^{-1}A + \Delta t\, A(I - \Delta t\, B)^{-T}A\\
&\quad +\Delta t^2 A(I - \Delta t\, B)^{-T}A(I - \Delta t\, B)^{-1}A\\
&= A + \Delta t\, A\tilde{B}^{-T}(\Delta t\, A + \tilde{B}^T + \tilde{B})\tilde{B}^{-1}A\\
&= A + 2\Delta t\,(\tilde{B}^{-1}A)^T(\tilde{B}^{-1}A)\,.
\end{aligned} \tag{24}$$

This implies

$$\|\tilde{B}^{-1}\tilde{C}\|^2_{-A} = 1 - 2\Delta t \sup_{x\neq 0} \frac{\|\tilde{B}^{-1}Ax\|_2^2}{\|x\|^2_{-A}} < 1\,, \tag{25}$$

for all $\Delta t > 0$. Thus, we have

$$\rho(\tilde{B}^{-1}\tilde{C}) \leq \|\tilde{B}^{-1}\tilde{C}\| < 1 \tag{26}$$

for all $\Delta t > 0$. □

Theorem 3.3. *If A is lower-triangular with all diagonal elements negative, the ADE scheme (19) is unconditionally stable.*

Proof. Let $d_j < 0$ be the diagonal elements of the matrix A. The spectral radius of $\tilde{B}^{-1}\tilde{C}$ is then given by

$$\rho(\tilde{B}^{-1}\tilde{C}) = \max_j \left|\frac{1 + \Delta t\, d_j/2}{1 - \Delta t\, d_j/2}\right| < 1 \tag{27}$$

for all $\Delta t > 0$. And, this implies stability.

Using the same idea, if A is upper-triangular with all diagonal elements negative, the ADE scheme (19) is also unconditionally stable. □

4 Further Applications

In this section, we generalize the ADE scheme and apply it to both higher order and nonlinear equations. Section 4.1 deals with linear equations. A generalization to nonlinear equations will be given in Section 4.2

4.1 Linear Equations

4.1.1 Heat Equation with Variable Coefficients

Applying the ADE technique to the equation

$$\frac{\partial T}{\partial t} = \nabla \cdot [k(x)\nabla T] \tag{28}$$

where $k(x) > 0$ and using central differencing to approximate the gradient operators, we have

$$B = \frac{1}{\Delta x^2}\begin{pmatrix} k_1 & 0 & 0 & \cdots & \cdots \\ k_{3/2} & k_2 & 0 & 0 & \cdots \\ 0 & k_{5/2} & k_3 & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & k_{(2n-1)/2} & k_n \end{pmatrix} \tag{29}$$

where $k_i = -[k(x_{i-1/2}) + k(x_{i+1/2})]/2$ or $-k((x_{i-1/2} + x_{i-1/2})/2)$, and $C = B^T$.

Therefore, A is symmetric negative definite. Using Theorem 3.2, we can easily prove that the resulting ADE scheme is unconditionally stable for arbitrary $\Delta t > 0$.

4.1.2 Linear Advection Equation

Another interesting example is the linear advection equation. Applying the forward differencing in time and the central differencing in space, we obtain

$$u_i^{n+1} = u_i^n - \frac{a\Delta t}{2\Delta x}(u_{i+1}^n - u_{i-1}^n)\,, \tag{30}$$

which is well-known to be unconditionally **unstable** for all $\Delta t > 0$. However, employing the idea of the ADE scheme, we obtain an unconditionally **stable** scheme by slightly modifying (30). Numerically, we only need to replace either u_{i+1}^n or u_{i-1}^n in the above scheme by u^{n+1} using

1. Set $\tilde{u}_i^n = u_i^n$ and $\hat{u}_i^n = u_i^n$ for $i = 1, 2, \ldots, N$;
2. For $i = 1, 2, \ldots, N$, solve

$$\tilde{u}_i^{n+1} = \tilde{u}_i^n - \frac{a\Delta t}{\Delta}(\tilde{u}_{i+1}^n - \tilde{u}_{i-1}^{n+1})\,; \tag{31}$$

3. For $i = N, N-1, \ldots, 1$, solve

$$\hat{u}_i^{n+1} = \hat{u}_i^n - \frac{a\Delta t}{\Delta}(\hat{u}_{i+1}^{n+1} - \hat{u}_{i-1}^n)\,; \tag{32}$$

4. Compute

$$u_i^{n+1} = \frac{\tilde{u}_i^{n+1} + \hat{u}^{n+1}}{2}\,. \tag{33}$$

We can also easily generalize this type of central scheme to a higher degree of accuracy. For example, we can have

$$u_i^{n+1} = u_i^n + a_i\Delta t\left(\sum_{j=1}^{i-1}\alpha_j u_j^* + \sum_{j=i+1}^{N}\alpha_j u_j^*\right)\,, \tag{34}$$

where $\alpha_{i+k} = -\alpha_{i-k}$ and the $*$-state can be n or $(n+1)$, depending on the direction of the indexing. Here we briefly analyze the resulting method. We consider only half of the ADE scheme, in which we use u_j^{n+1} for $j < i$ and u_j^n for $j > i$. The other half can be

analyzed in the same way. Using the Fourier analysis, with $I = \sqrt{-1}$, we can compute the amplifying factor of the scheme which is given by

$$\xi = \frac{1 - a_i \Delta t \sum_{j=i+1}^{N} \alpha_j \exp[Ik(j-i)\Delta x]}{1 + a_i \Delta t \sum_{j=1}^{i-1} \alpha_j \exp[Ik(j-i)\Delta x]} . \tag{35}$$

Thus $\|\xi\| = 1$, which implies stability.

Although stable, these schemes are non-dispersive. This means that oscillations could be generated near discontinuity. Therefore, this type of central ADE scheme is not recommended for solving hyperbolic-type equations with discontinuity in the solution.

In addition to the central differencing used above, we can also use the upwind differencing to discretize the space derivative. Guaranteed by Theorem 3.3, the resulting scheme is stable, and therefore convergent, for all $\Delta t > 0$.

4.1.3 Fourth-order Diffusion Equation

For the higher order diffusion equation

$$\frac{\partial T}{\partial t} = -T_{xxxx} , \tag{36}$$

the stability condition for the standard fully explicit scheme is $\Delta t = O(\Delta x^4)$. It is possible to relax this condition using the fully implicit scheme or the Crank-Nicloson-type scheme. However, for higher dimensions or problems with complicated geometry, any dimensional splitting techniques, including the popular ADI method, may no longer be useful. The main difficulty is that the operator in each direction cannot be solved as easily as the Laplace operator.

Of course, one could use FFT to solve this type of equation. However, it is still necessary to solve the system of ODE's which governs the evolution of each mode,

$$\frac{d\tilde{T}_k}{dt} = -k^4 \tilde{T}_k . \tag{37}$$

For higher frequency modes, these equations are stiff and, therefore, require a very small time step Δt.

Applying the ADE scheme to this equation, we have

$$B = \frac{1}{\Delta x^4} \begin{pmatrix} -3 & 0 & 0 & \cdots & \cdots \\ 4 & -3 & 0 & 0 & \cdots \\ -1 & 4 & -3 & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & \cdots & -1 & 4 & -3 \end{pmatrix} \tag{38}$$

and $C = B^T$. More importantly, guaranteed by Theorem 3.2, this resulting numerical scheme is unconditionally stable for any $\Delta t > 0$.

4.2 Nonlinear Equations

In this section, we will mainly emphasize the following two classes of nonlinear equations. These classes have wide applications in many fields of applied mathematics. Solving them efficiently is then especially important.

The first type is the Hamilton-Jacobi equations. These equations are hyperbolic and, in general, the stability condition for any explicit scheme is only $O(\Delta x)$. Compared with those elliptic equations as in previous sections, this condition is relatively less restrictive and therefore the improvement in using ADE is not obvious. The second class is more

important. They are nonlinear diffusion equations which for example govern motions of thin film flows or remove noise in image processing.

A general way to deal with these nonlinear equations is to express them as

$$\frac{\partial T}{\partial t} = A(T)T + b(T)\,, \tag{39}$$

where $A(T)$ is now a nonlinear operator involving T. This operator and the source term $b(T)$ will then be linearized by using $T = T^n$. For example, a simple implicit scheme can be given by

$$\frac{T^{n+1} - T^n}{\Delta t} = A(T^n)T^{n+1} + b(T^n)\,. \tag{40}$$

Here, we follow a similar approach.

4.2.1 H-J Equations

These equations are widely used in Level Set Methods [11] or image segmentation. For example

1. Level Set Equation
$$\phi_t + v_n|\nabla\phi| = 0\,. \tag{41}$$
2. Reinitialization equation
$$\phi_t + S(\phi_0)(|\nabla\phi| - 1) = 0\,. \tag{42}$$

To solve these equations using the ADE scheme, we first rewrite them as

$$\frac{\partial \phi}{\partial t} + H(\nabla\phi) = 0\,, \tag{43}$$

with a corresponding Hamiltonian. Traditionally, the time-derivative can be approximated using a high order TVD Runge-Kutta method [13], while the Hamiltonian can be discretized using the Osher-Sethian scheme [11], the Lax-Friedrichs scheme (LF) [4], the Local Lax-Friedrichs (LLF) scheme [14], the Roe-Fixed scheme [14], or the Godunov scheme [3]. Among them all, LF, or LLF, is the easiest to implement. For a general Hamiltonian, we have the numerical Hamiltonian

$$\hat{H}(\nabla\phi) = H\left(\frac{\phi_x^+ + \phi_x^-}{2}\right) - \alpha\left(\frac{\phi_x^+ - \phi_x^-}{2}\right)\,, \tag{44}$$

where $\phi_x^\pm$ are the forward and the backward difference approximations of the derivative which can be obtained using a high order ENO [13, 14] or WENO [6] reconstruction, and α is the dissipation coefficient given by

$$\alpha = \max\left|\frac{\partial H(p)}{\partial p}\right|\,. \tag{45}$$

For simplicity, we first consider the case where $\phi_x^\pm$ are obtained using the first order forward and backward difference, respectively. Therefore, we have the following approximation for the H-J equation,

$$\frac{\partial \phi_i}{\partial t} + H\left(\frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x}\right) - \alpha\left(\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{2\Delta x}\right) = 0\,. \tag{46}$$

Now, we apply the idea of the ADE scheme. To make the operator $(I - \Delta t B)^{-1}$ simple to calculate, all values of ϕ inside the nonlinear Hamiltonian will be taken on $t = t^n$. This gives

$$\left(1 + \frac{\alpha\Delta t}{2\Delta x}\right)\phi_i^{n+1} = \left(1 - \frac{\alpha\Delta t}{2\Delta x}\right)\phi_i^n + \frac{\alpha\Delta t}{2\Delta x}(\phi_{i+1}^* + \phi_{i-1}^*) - \Delta t H\left(\frac{\phi_{i+1}^n - \phi_{i-1}^n}{2\Delta x}\right)\,, \tag{47}$$

where ϕ^* can either be ϕ^{n+1} or ϕ^n depending on the direction of the indexing. This resulting scheme is unconditionally stable for all $\Delta t > 0$, proven by Theorem 3.2, and second order accurate in time and first order accurate in space.

In evaluating the Hamiltonian, we can actually implement a high order ENO or WENO scheme for the derivative, i.e.

$$\left(1+\frac{\alpha\Delta t}{2\Delta x}\right)\phi_i^{n+1} = \left(1-\frac{\alpha\Delta t}{2\Delta x}\right)\phi_i^n + \frac{\alpha\Delta t}{2\Delta x}(\phi_{i+1}^* + \phi_{i-1}^*) - \Delta t H\left(\frac{(\phi_i^n)_x^+ + (\phi_i^n)_x^-}{2}\right). \quad (48)$$

However, because of the Δx order diffusion term, the overall accuracy in space will not be significantly improved.

Notice that this technique can also be applied to the linear advection equation by treating $H(\nabla\phi) = \mathbf{u}\cdot\nabla\phi$.

4.2.2 Nonlinear Diffusion Equations

Another class of nonlinear equations is the nonlinear diffusion equations. They can be found in image processing or thin film flow modeling. We list only a few of them here.

1. The modified lubrication equation

$$\frac{\partial h}{\partial t} + h^\alpha \frac{\partial^4 h}{\partial x^4} = 0 \quad (49)$$

for some $\alpha \in \mathbb{R}$. One possible numerical scheme is to follow the one in Section 4.1.3 where the diffusion coefficient can be approximated by $(h^n)^\alpha$.
One variation of the above equation is the lubrication equation, given by

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}\left(h^\alpha \frac{\partial^3 h}{\partial x^3}\right) = 0\,. \quad (50)$$

More details in solving this equation will be given later in Section 5.3.

2. The equation for Total Variation (TV) denoising [12] is given by

$$\frac{\partial u}{\partial t} = \nabla\cdot[g(|\nabla u|)\nabla u] - \lambda(u-u_0)\,, \quad (51)$$

with $g(|\nabla u|) = 1/|\nabla u|$. This equation can be solved easily using a scheme similar to the one in Section 4.1.1. More details in solving this equation will be given later in Section 5.4.

3. Texture/Cartoon decomposition in image processing [8]

$$\frac{\partial u}{\partial t} = -\frac{1}{2\lambda}\Delta\left[\nabla\cdot\left(\frac{\nabla u}{|\nabla u|}\right)\right] - (u-u_0)\,. \quad (52)$$

To discretize the equation, we use a similar approach as in the TV-denoising equation. Again, we can treat $g(|\nabla u|) = 1/|\nabla u|$, while the term in the square bracket $[\cdot]$ can be approximated by

$$\alpha_{i,j}^{x-}u_{i-1,j} + \alpha_{i,j}^{x+}u_{i+1,j} + \alpha_{i,j}^{y-}u_{i,j-1} + \alpha_{i,j}^{y+}u_{i,j+2} + \alpha_{i,j}^{0}u_{i,j}\,. \quad (53)$$

Next, we can apply a standard central differencing for the Laplacian, giving an expression based on the following 13 values $u_{i,j}$, $u_{i\pm1,j}$, $u_{i\pm2,j}$, $u_{i,j\pm1}$, $u_{i,j\pm2}$ and $u_{i\pm1,j\pm1}$. It is then straight forward to apply the ADE techniques.

5 Test Cases

The computations below are done in a cluster system with each node having a 3.0 GHz hyperthreading Pentium IV processor and 2GB of memory.

5.1 The Fourth-order Heat Equation

We solve

$$T_t = -T_{xxxx}\,, \tag{54}$$

with the initial condition $T = 1$ when $x > 0.5$ and $T = 0$ otherwise. The boundary conditions are $T(x = 1) = 1$, $T(x = 0) = 0$ and $T_x = 0$ at both end points. Using different numerical schemes, we compute the solution at $t = 6.25 \times 10^{-8}$.

Figure 1 shows the solutions obtained using the standard fully explicit scheme with $\Delta t = 0.1\Delta x^4$ and using the ADE scheme with $\Delta t = \Delta x^4, 2\Delta x^4$ and $10\Delta x^4$ respectively. Notice that using the standard explicit scheme, the stability condition is given by $\Delta t < 0.125\Delta x^4$.

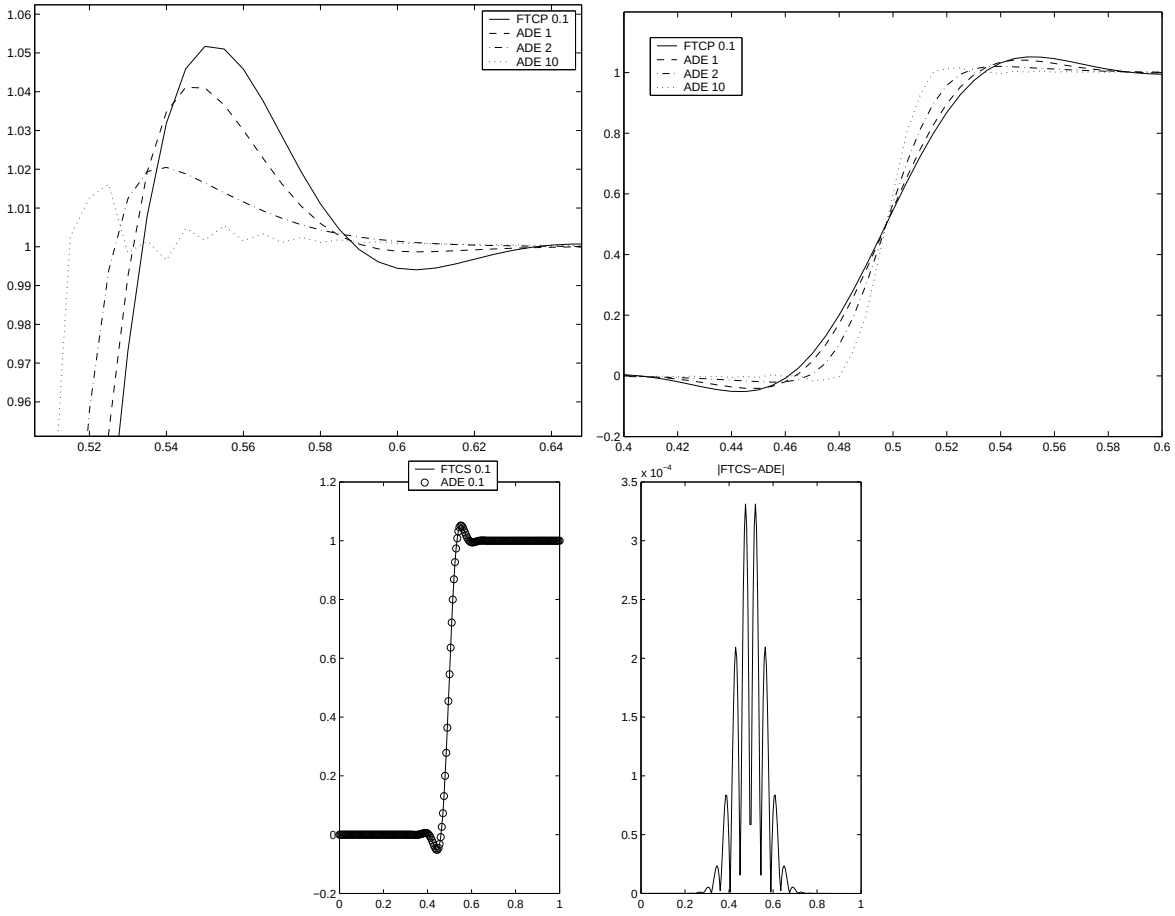

Figure 1: (Example 5.1) First row shows solutions using the standard explicit scheme with $\Delta t = 0.1\Delta x^4$ and the proposed ADE scheme with much larger step sizes. First figure on the second row shows a comparison of two schemes using the same $\Delta t = 0.1\Delta x^4$.

We also performed a convergence test for the same equation. The initial condition we chose is

$$T(x, t = 0) = \cos(2\pi x) \;\text{ in } x \in [0, 1]. \tag{55}$$

The exact solution can easily be computed,

$$T_{\text{exact}}(x, t) = \exp(-16\pi^4 t)\cos(2\pi x)\,. \tag{56}$$

N \ Δt	$0.5/400^4$	$1.0/400^4$	$2.0/400^4$	$4.0/400^4$
21	0.00214777236180	0.00214777236586	0.00214777237287	0.00214777239804
41	0.00052974323039	0.00052974332658	0.00052974371361	0.00052974526166
81	0.00013114337822	0.00013114944076	0.00013117370438	0.00013127075711
161	0.00003273907817	0.00003312413074	0.00003466433462	0.00004082485682

Table 1: (Example 5.1) L_2 Errors.

N \ Δt	$0.5/400^4$	$1.0/400^4$	$2.0/400^4$	$4.0/400^4$
21	-	1.66381009134637	1.71177376531252	-
41	-	2.00071013686408	1.99944714308720	-
81	-	2.00072867857121	1.99997111723545	-
161	-	1.99999479735959	1.99993994756767	-

Table 2: (Example 5.1) Order of convergence in the t-direction, $(r_1)_{k,l}$.

Table 1 shows the L_2 errors defined by

$$E_{k,l} = \sqrt{\sum_i (T_i - T_{\text{exact}}(x_i, t_{\text{final}}))^2 \Delta x_l} \tag{57}$$

at $t_{\text{final}} = 1/100^2$ using different step sizes Δt_k. To compute the accuracy of the method, we assume the leading order of the error can be expressed in the form

$$E_{k,l} = \omega_1 \Delta t_k^{r_1} + \omega_2 \Delta x_l^{r_2} \,. \tag{58}$$

Using the data from Table 1, we define

$$(r_1)_{k,l} = \frac{1}{\log 2} \log \left(\frac{E_{k,l} - E_{k-1,l}}{E_{k+1,l} - E_{k,l}} \right) \text{ and } (r_2)_{k,l} = \frac{1}{\log 2} \log \left(\frac{E_{k,l} - E_{k,l-1}}{E_{k,l+1} - E_{k,l}} \right) . \tag{59}$$

Their numerical values are given in Table 2 and 3. It is clear that the numerical scheme is second order accurate in both the t- and x-directions.

5.2 A Hamilton-Jacobi Equation

We solve

$$\phi_t + H(\phi_x) = 0 \,, \tag{60}$$

N \ Δt	$0.5/400^4$	$1.0/400^4$	$2.0/400^4$	$4.0/400^4$
21	-	-	-	-
41	2.12099072959563	2.12101174641717	2.12109585752083	2.12143234224636
81	2.04909457955820	2.05460090325181	2.07684012570413	2.16941638856231
161	-	-	-	-

Table 3: (Example 5.1) Order of convergence in the x-direction, $(r_2)_{k,l}$.

with $H(p) = -\cos(p+1)$. We discretize the x-axis by 201 grid point and solve the equation up to $t_{\text{final}} = 0.15$. To approximate $\phi_x^{\pm}$, we use only simple forward and backward differences.

On the left hand side of Figure 2, we show the solutions using the ADE scheme with $\Delta t = 0.01\Delta x$ and $15\Delta x$. Because $15\Delta x = 0.15 = t_{\text{final}}$, the solution ADE_{15} is essentially obtained by using only one time marching step. On the right hand side of the same figure, we compare the solutions at $t = t_{\text{final}}$ using different Δt.

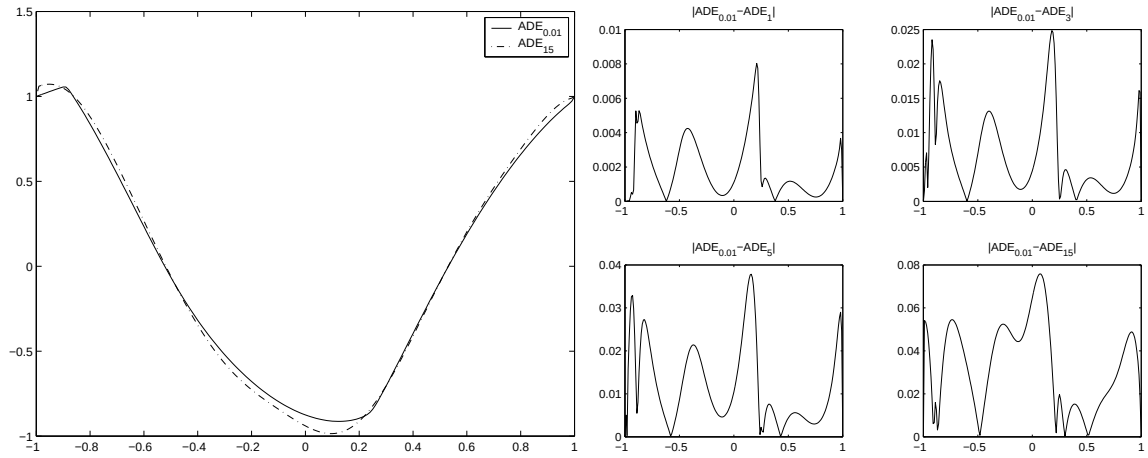

Figure 2: (Example 5.2) Solutions using the proposed ADE scheme with $\Delta t = 0.01\Delta x^2$ and $15\Delta x^2$ at $t_{\text{final}} = 15\Delta x^2$. Comparisons of the solutions using different step sizes are shown on the right hand side.

5.3 The Lubrication Equation

This equation was studied extensively in [18]

$$\frac{\partial h}{\partial t} + \frac{\partial}{\partial x}\left(f(h)\frac{\partial^3 h}{\partial x^3}\right) = 0\,, \tag{61}$$

where $f(h) = h^{1/2}$. To regularize the equation, one can replace $f(h)$ by

$$f_\epsilon(h) = \frac{h^4 f(h)}{\epsilon f(h) + h^4}\,, \tag{62}$$

with $\epsilon = 10^{-11}$. The initial condition of this equation is given by

$$h(x) = 0.8 - \cos(\pi x) + 0.25\cos(2\pi x)\,. \tag{63}$$

On both end points at $x = 0$ and $x = 1$, we impose reflective boundary conditions, i.e. we have $h' = h'' = 0$. Numerically, on the left boundary, we set $h_{-1} = h_1$ and $h_{-2} = h_2$. Similar boundary conditions are imposed on the other end point at $x = 1$. In this numerical test, we use 501 grid points to discretize the x-direction, or $\Delta x^4 = 1.6 \times 10^{-11}$, and $\Delta t = 10\Delta x^4$.

Using the proposed ADE approach, we have

$$\left(1 + \frac{\beta\alpha_0}{2}\right) h_i^{n+1} = \left(1 - \frac{\beta\alpha_0}{2}\right) h_i^n - \beta(\alpha_{+2}h_{i+2}^* + \alpha_{+1}h_{i+1}^* + \alpha_{-1}h_{i-1}^* + \alpha_{-2}h_{i-2}^*)\,, \tag{64}$$

where

$$\beta = \frac{\Delta t}{\Delta x^4}\,,\ \alpha_0 = 3(f^n_{i+1/2} + f^n_{i-1/2})\,,$$
$$\alpha_{-2} = f^n_{i-1/2}\,,\ \alpha_{-1} = -(f^n_{i+1/2} + 3f^n_{i-1/2})\,,$$
$$\alpha_{+1} = -(3f^n_{i+1/2} + f^n_{i-1/2})\,,\ \alpha_{+2} = f^n_{i+1/2}\,. \tag{65}$$

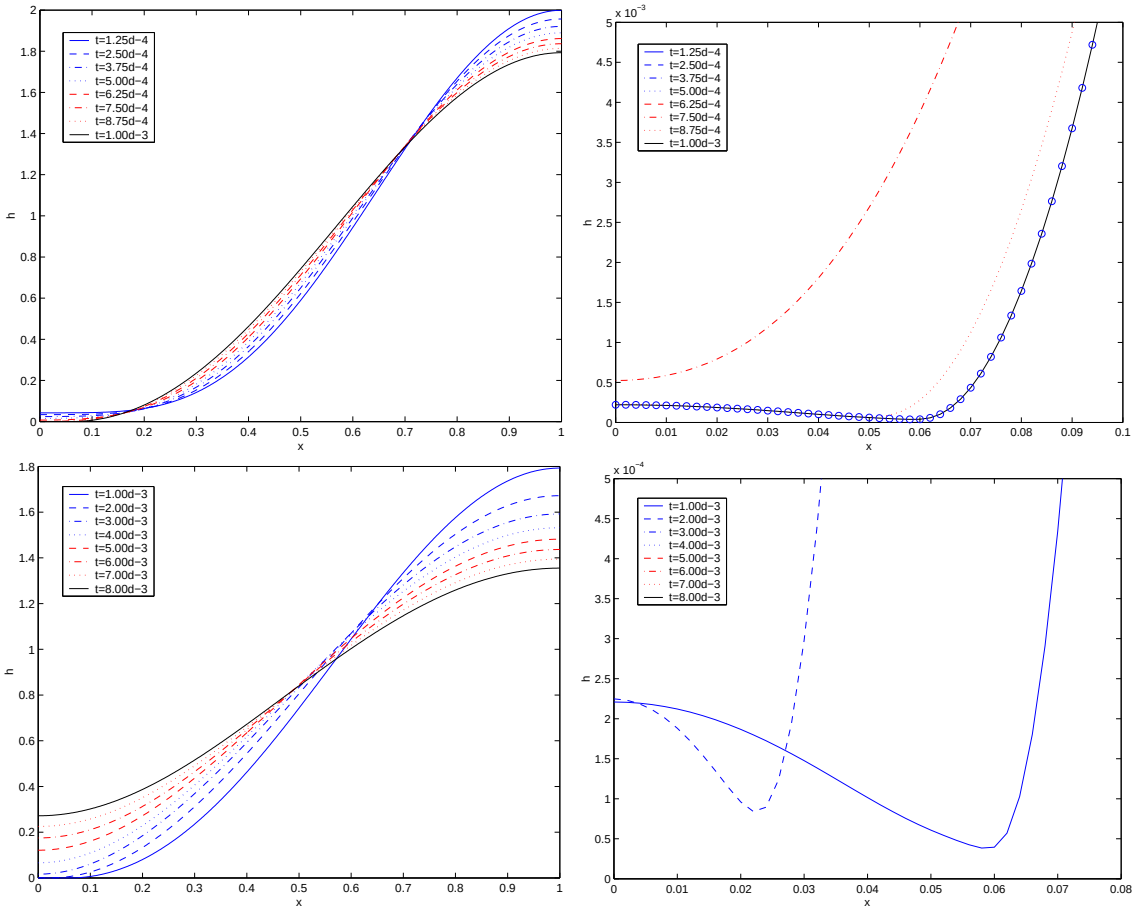

Figure 3: (Example 5.3) Figures on the first row shows the solution of the lubrication equation at $t = k \times 1.25 \times 10^{-4}$ for $k = 1, \ldots, 8$. Second rows show the solution at a later time $t = k \times 10^{-4}$ for $k = 1, \ldots, 8$.

Solutions are plotted in Figure 3. Although the exact solution for this problem is not available, our solutions look reasonably close to those in [18].

5.4 TV denoising

Here we study in detail the method of applying TV regularization in image processing [12] using the proposed ADE scheme. Given a noisy image u_0, we solve the following equation to remove the noise,

$$\frac{\partial u}{\partial t} = \nabla \cdot (g(u_x, u_y)\nabla u) - \lambda(u - u_0)\,, \tag{66}$$

where

$$g(u_x, u_y) = \frac{1}{\sqrt{u_x^2 + u_y^2 + \epsilon}}\,. \tag{67}$$

In this numerical test, we take $\Delta x = \Delta y = 1$, $\epsilon = 10^{-10}$. The test image we used is a 512-by-512 pixels image of Elaine, as shown in Figure 4. This image is taken from the USC-SIPI Image Database [5]. To obtain a noisy version of the original clean image, we add 20% random uniform noise, i.e.

$$u_0^{\text{new}} = u_0^{\text{old}} + 0.2\beta\chi\,, \tag{68}$$

where β is the maximum gray level of the image, which equals 256, and χ is a uniform random variable from $[-1, 1]$.

Figure 4: (Example 5.4) Original noisy image of Elaine and the one with 20% more noise.

For the standard fully explicit scheme, the stability condition is given by

$$\Delta t = O(\sqrt{u_x^2 + u_y^2}\Delta x^2)\,. \tag{69}$$

In practise, this condition is too restrictive in flat regions where $u_x^2 + u_y^2 \simeq 0$. To relax this restriction, [9] has proposed multiplying the right hand side of equation (66) by a factor of $|\nabla u|$. The paper has argued that away from the places where $g(\cdot)$ is singular, the steady state solution from solving this new equation is the same as the one obtained from the original equation. With this extra factor, the stability condition can be relaxed to

$$\Delta t = O(\Delta x^2)\,. \tag{70}$$

However, in the current paper, we do not implement this idea.

Violating this stability condition, we first implement the standard explicit scheme using $\Delta t = 0.1\Delta x^2$. The numerical results after 10^4 iterations are shown in Figure 5.

Applying the ADE approach to equation (66) directly, we have

$$\begin{aligned}\left[1 + \frac{\alpha}{2}(g_x^+ + g_x^- + g_y^+ + g_y^-) + \frac{\lambda\Delta t}{2}\right] u_{i,j}^{n+1} = \\ \left[1 - \frac{\alpha}{2}(g_x^+ + g_x^- + g_y^+ + g_y^-) - \frac{\lambda\Delta t}{2}\right] u_{i,j}^n + \alpha g_x^+ u_{i+1,j}^* + \alpha g_x^- u_{i-1,j}^* \\ + \alpha g_y^+ u_{i,j+1}^* + \alpha g_y^- u_{i,j-1}^* + \lambda\Delta t(u_0)_{i,j}\,,\end{aligned} \tag{71}$$

where $\alpha = \alpha_x = \alpha_y = \Delta t/(\Delta x \Delta y)$,

$$g_x^\pm = g(D_x^\pm u_{i,j}^n, D_y^0 u_{i,j}^n) \text{ and } g_y^\pm = g(D_x^0 u_{i,j}^n, D_y^\pm u_{i,j}^n)\,. \tag{72}$$

Figure 5: (Example 5.4) (From left to right, top to bottom) Steady state solutions using the standard fully explicit scheme with $\Delta t = 0.1\Delta x^2$ and $\lambda = 0.005, 0.010, 0.020$ and 0.050.

With this discretization, we have computed the solution using $\Delta t = 0.1\Delta x^2$. The solution is shown in Figure 6. It is similar to the one we obtained previously. However, because the ADE scheme eliminates the stability restriction, we have also implemented the cases where $\Delta t = 10\Delta x^2$ and $40\Delta x^2$. The numerical solutions are shown in Figures 7 to 8. From the plot of the change in the TV, we see that these steady state solutions are obtained using as few as 50 iterations.

Because we are able to use a much larger time step without violating any stability condition, we can obtain the steady state solution in much fewer iterations. This speedup in the computational efficiency is clearly shown in Figure 9 and 10. From these plots, we see that for those cases with smaller step sizes, steady state solutions are obtained using approximately 3000 iterations. For those cases with larger step sizes, the steady state solution is reached after only around 50 iterations. However, from Figure 11, as we further increase the step size Δt from $10\Delta x^2$ to $40\Delta x^2$, there is no extensive improvement in the speed of convergence. One possible reason is that the error from the numerical scheme has significantly grown when we increase Δt.

Figure 6: (Example 5.4) (From left to right, top to bottom) Steady state solutions using the proposed ADE scheme with $\Delta t = 0.1\Delta x^2$ and $\lambda = 0.005, 0.010, 0.020$ and 0.050.

To compare the computational speed of using different methods, we have also calculated the total CPU time for 10^4 iterations using the standard explicit method and the ADE scheme, respectively. On average, the standard explicit scheme requires 0.7851 second/iteration, while the ADE scheme requires 0.8631 second/iteration. It seems like the ADE scheme takes longer. However, if we look at the total computational time to reach the steady state solution, we see that it takes approximately 45 seconds for the ADE scheme but approximately 4000 seconds for the standard fully explicit scheme. The ADE scheme is still overall much faster and speeds up the computations by a larger factor.

The final point we want to make is that one needs to be very cautious in using the standard fully explicit scheme with a stability violating Δt. Different from the situation in Section 5.1, the numerical solution from the explicit scheme may not blow up. We have performed tests to see how the TV of the image varies with the step size. Figure 12 shows the solutions using some larger Δt with $\lambda = 0.010$. The TV of the image actually decreases for the first few iterations, goes up a little bit for the next few and finally stays relatively flat. More importantly, the TV-norm of these *steady state solutions* depends on the Δt

Figure 7: (Example 5.4) (From left to right, top to bottom) Steady state solutions using the proposed ADE scheme with $\Delta t = 10\Delta x^2$ and $\lambda = 0.005, 0.010, 0.020$ and 0.050.

we have chosen. This is a subtle issue computationally in using a conditionally stable scheme. On one hand, if we follow the stability condition of the standard explicit scheme, Δt has to be in the order of $\sqrt{u_x^2 + u_y^2}$, which makes the computations impractical. On the other hand, if we ignore this condition and use a relatively small (intuitionally) step size, a wrong solution may unfortunately still stay stable. Therefore, it is not clear if the *steady state solutions* we obtained above in Figure 5 using the standard explicit scheme really correspond to the **true** steady state solutions of equation (66). In other words, one cannot be sure how much the *steady state solution* is affected by the stability in the numerical scheme.

6 Conclusion

We have successfully applied an unconditionally stable, while still explicit, scheme to nonlinear evolution equations. This resulting method has shown a significant improvement

Figure 8: (Example 5.4) (From left to right, top to bottom) Steady state solutions using the proposed ADE scheme with $\Delta t = 40\Delta x^2$ and $\lambda = 0.005, 0.010, 0.020$ and 0.050.

in the computational efficiency, as clearly demonstrated by different numerical examples.

Acknowledgment

S. Leung is supported by (...). He would also like to thank both J.J. Xu for valuable discussions on the section of image processing and the UCLA Department of Mathematics for providing the computing facilities supported by NSF SCREMS grant DMS-0112330 and the Regents of the University of California.

References

[1] H.Z. Barakat and J.A. Clark. On the solution of diffusion equation by numerical methods. *J. Heat Transfer*, 88:421–427, 1966.

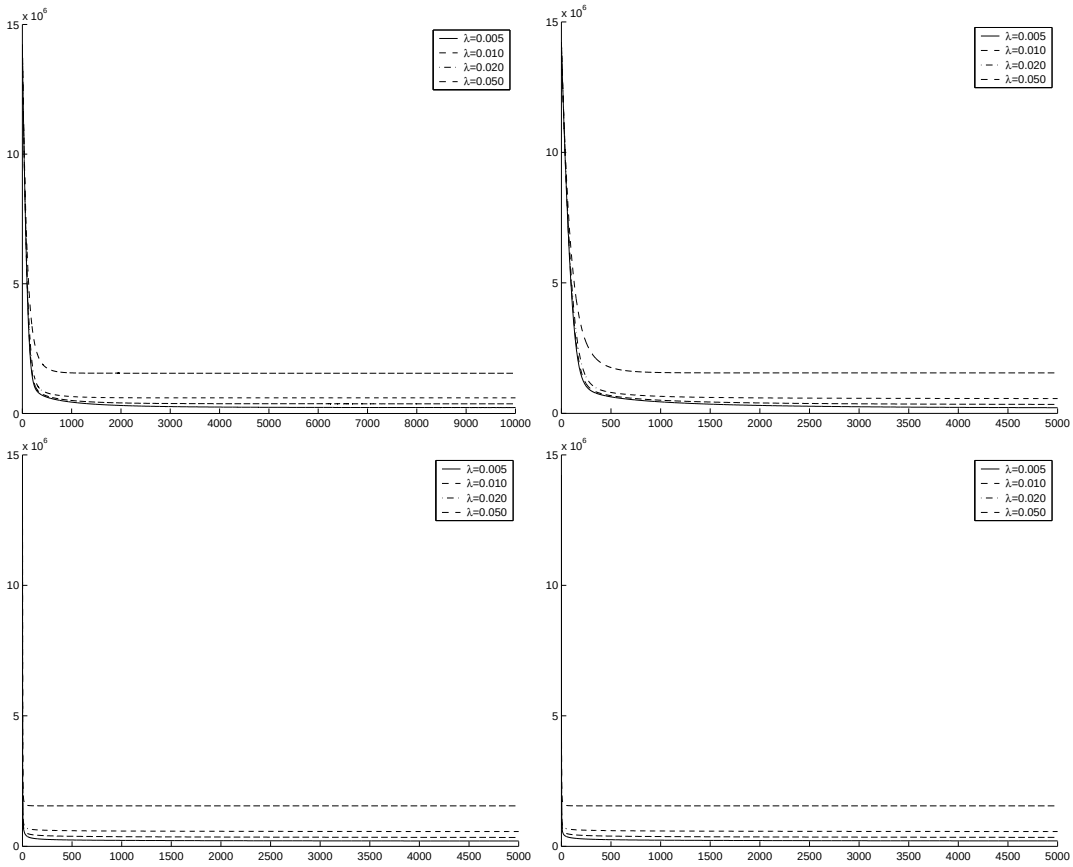

Figure 9: (Example 5.4) (From left to right, top to bottom) Change in the TV of the image using the standard explicit scheme with $\Delta t = 0.1\Delta x^2$ and the proposed ADE scheme with $\Delta t = 0.1\Delta x^2, 10\Delta x^2$ and $40\Delta x^2$.

[2] D. Barash, M. Israeli, and R. Kimmel. An accurate operator splitting scheme for nonlinear diffusion filtering. *Proceedings of the Third International Conference on Scale-Space and Morphology in Computer Vision*, pages 281–289, 2001.

[3] M. Bardi and S. Osher. The nonconvex multidimensional riemann problem for hamilton-jacobi equations. *SIAM J. Math. Anal.*, 22:344–351, 1991.

[4] M. G. Crandall and P. L. Lions. Two approximations of solutions of Hamilton-Jacobi equations. *Math. Comp.*, 43:1–19, 1984.

[5] USC-SIPI Image Database. http://sipi.usc.edu/services/database/database.html.

[6] G. S. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21:2126–2143, 2000.

[7] B.K. Larkin. Some stable explicit difference approximations to the diffusion equation. *Math. Comp.*, 18:196–202, 1964.

[8] J. Osher, A. Solé, and L. Vese. Image decomposition and restoration using total variation minimization and the h^{-1} norm. *SIAM J. Multi. Model. & Simul.*, 1(3):349–370, 2003.

[9] S. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces.* Springer-Verlag, New York, 2003.

[10] S. Osher and N. Paragios (Eds.). *Geometric Level Set Methods in Imaging, Vision and Graphics.* Springer, 2003.

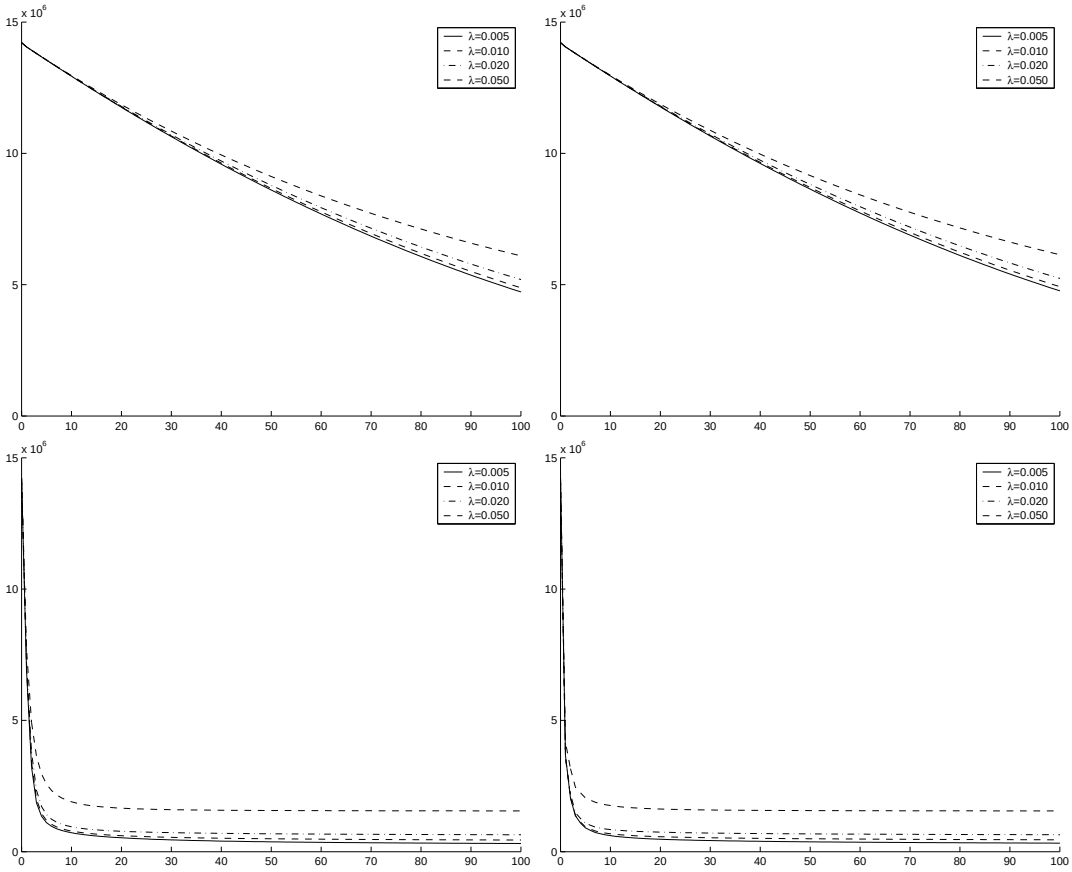

Figure 10: (Example 5.4) (From left to right, top to bottom) A close-up in the change in the TV of the image using the standard explicit scheme with $\Delta t = 0.1\Delta x^2$ and the proposed ADE scheme with $\Delta t = 0.1\Delta x^2, 10\Delta x^2$ and $40\Delta x^2$.

[11] S. J. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[12] L. Rudin, S.J. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[13] C. W. Shu and S. J. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

[14] C.W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes 2. *J. Comput. Phys.*, 83:32–78, 1989.

[15] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *J. Sci. Comput.*, 1-3:439–456, 2003.

[16] J. C. Strikwerda. *Finite difference schemes and partial differential equations.* Wadsworth-Brooks/Cole, Pacific Grove, California, 1989.

[17] J. Weickert, B.M. ter Haar Romeny, and M.A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7:398–410, 1998.

[18] L. Zhornitshaya and A.L. Bertozzi. Positivity-preserving numerical schemes for lubrication-type equations. *SIAM J. Num. Anal.*, 37:523–555, 2000.

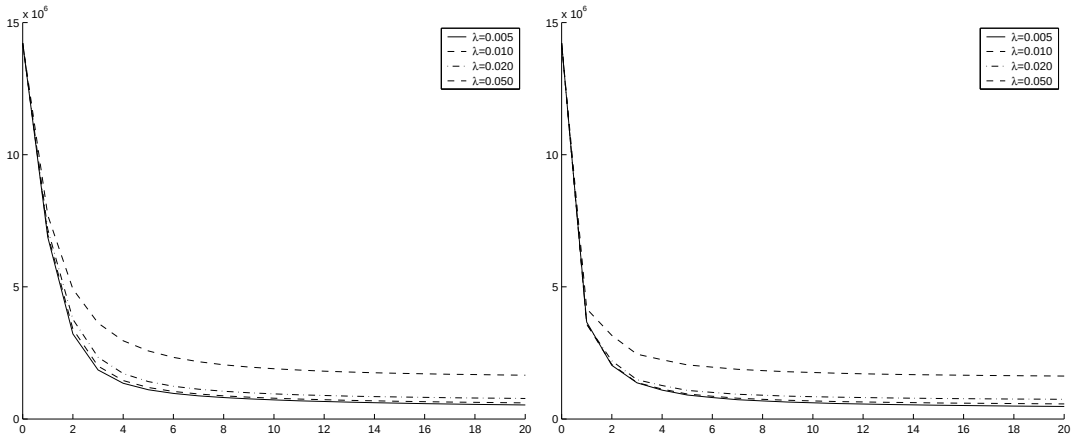

Figure 11: (Example 5.4) Another close-up in the change in the TV of the image using the proposed ADE scheme with $\Delta t = 10\Delta x^2$ and $40\Delta x^2$.

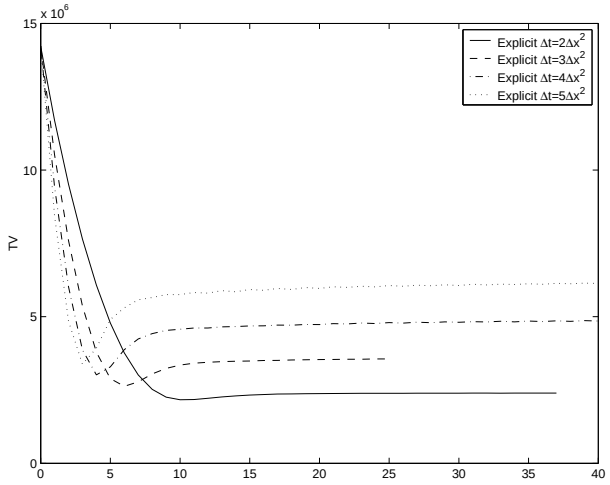

Figure 12: (Example 5.4) Changes in the TV of the image using the standard explicit scheme with $\lambda = 0.010$ and $\Delta t = 2\Delta x^2, 3\Delta x^2, 4\Delta x^2$ and $5\Delta x^2$.