# SUPER-TIME-STEPPING ACCELERATION OF EXPLICIT SCHEMES FOR PARABOLIC PROBLEMS

VASILIOS ALEXIADES* GENEVIÈVE AMIEZ† AND PIERRE-ALAIN GREMAUD‡

**Abstract.** The goal of this paper is to bring to the attention of the computational community a long overlooked, very simple, acceleration method that impressively speeds up explicit time-stepping schemes, at essentially no extra cost. We explain the basis of the method, namely stabilization via wisely chosen inner steps (stages), justify it for linear problems, and spell out how simple it is to incorporate in any explicit code for parabolic problems. Finally, we demonstrate its performance on the (linear) heat equation as well as on the (nonlinear) classical Stefan problem, by comparing it with standard implicit schemes (employing SOR or Newton iterations). The results show that super-time-stepping is more efficient than the implicit schemes in that it runs at least as fast, it is of comparable or better accuracy, and it is of course much easier to program (and to parallelize for distributed computing).

**Key words.** Explicit scheme, implicit scheme, time-stepping, acceleration, Chebyshev, parabolic, heat equation, Stefan problem

**AMS(MOS) subject classifications.** Primary 65M20, 65M10

**1. Introduction.** Super-time-stepping is a very simple and effective way to speed up explicit time-stepping schemes for parabolic problems. The method is not new. Conceptually, it belongs to a class known in the ODE Numerical Analysis community as Runge-Kutta-Chebyshev methods, see [6], [9] and references therein. Such methods have a long history and are continually being rediscovered. A variation of the method also exists in Numerical Linear Algebra (Chebyshev semi-iterative method, see e.g. [8]), which is not surprising since many iterative algorithms for solving linear systems admit some kind of temporal interpretation. However, and in spite of the fact that the method is almost 20 years old, [4], [7], it seems that hardly anyone knows about it in the computational PDE world. This is regrettable, since the method is not only very simple to use, but also impressively effective. In fact, Super-Time-Stepping (STS) frees the explicit scheme from the stability restriction on the time-step, rendering it as usable as any implicit scheme, while retaining its simplicity and better accuracy. Our computational results bear this out, showing that super-time-stepping is more efficient than the standard implicit schemes (using SOR or Newton iterations), in that it runs at least as fast, it is of comparable or better accuracy, and it is of course much easier to program (and to parallelize for distributed computing).

Our goal in this paper is twofold. First, to present a direct development of (a variant of) the method and its justification for linear equations. The Super-Time-Stepping algorithm is based on relaxing the requirement of (strong) stability at the end of every time step, to requiring it only at the end of a cycle of $N$ of them, [3], [6]. The scheme presented here is a variant of Gentzsch's method, [3]. Second, to

* Mathematics Dept., University of Tennessee, Knoxville, TN 37996-1300, vasili@math.utk.edu, and Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6367.
Partially supported by Science Alliance, a Centers of Excellence Program of the State of Tennessee.
† Laboratoire de Calcul Scientifique, Université de Franche-Comté, 25030 Besançon, France.

‡ Department of Mathematics and Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC 27695, gremaud@dali.math.ncsu.edu.
Partially supported by the Army Research Office through grant DAAH04-91-1-0419 and by the Minnesota Supercomputer Institute while at stay at the School of Mathematics of the University of Minnesota.

demonstrate the performance of the method and compare it to implicit schemes. We present detailed comparisons of performance on the (linear) heat equation, and on the (nonlinear) Stefan problem. The main point of interest here is that STS performs extremely well not only in the linear case, but also for nonlinear problems, even though, in the latter case, no theoretical results are available. The good behavior of STS-like algorithms for this kind of degenerate nonlinear parabolic problems was also observed in [2].

The outline of the paper is as follows. In section 2, we derive and analyze the super-time-stepping scheme in the case of a linear parabolic equation. Implementation related questions are addressed in section 3. Section 4 is devoted to performance evaluations in both the linear and nonlinear case. Conclusions are offered in section 5.

**2. The super-time-stepping scheme.** Let us consider the following time dependent problem

$$\frac{dU}{dt}(t) + AU(t) = 0 \qquad t > 0 \qquad U(0) = U_0, \tag{2.1}$$

together with the corresponding standard explicit scheme

$$U^{n+1} = U^n - \Delta t\, A\, U^n \qquad n = 0, 1, \ldots \qquad U^0 = U_0, \tag{2.2}$$

where $A$ is a $M \times M$ symmetric positive definite matrix, $\Delta t > 0$ is the time step, and $U_0$ is a given vector in $\mathbb{R}^M$. As is well known, the above algorithm is subject to the restrictive stability condition

$$\rho(I - \Delta t A) < 1, \tag{2.3}$$

where $\rho(\cdot)$ denotes the spectral radius. If $\lambda_{\max}$ stands for the largest eigenvalue of $A$, the latter condition is equivalent to

$$\Delta t < \Delta t_{expl} =: \frac{2}{\lambda_{\max}}. \tag{2.4}$$

This is the famous Courant-Friedrichs-Lewy (CFL) stability condition. For example, in the case of the 1-dimensional heat equation $u_t = \alpha u_{xx}$, discretized by standard second-order differences on a uniform mesh, we have $\lambda_{\max} = 4\alpha/\Delta x^2$, so that $\Delta t_{expl} = \Delta x^2/(2\alpha)$.

In order to relax the restrictive character of the latter condition, we do not require stability at the end of each time-step $\Delta t$, but rather at the end of a cycle of $N$ time-steps, thus leading to a Runge-Kutta–like method with $N$ stages. We introduce a *superstep* $\Delta T$ consisting of $N$ time-steps $\tau_1,\ \tau_2, \ldots, \tau_N$. The idea is now to ensure stability over the superstep $\Delta T$, while trying to maximize its duration $\Delta T = \sum_{j=1}^{N} \tau_j$.

It should be emphasized at this point that only the values at the end of each superstep $\Delta T$ approximate the solution of the problem. The inner values have no approximating properties and should only be considered as intermediate calculations.

The new algorithm can be written as

$$U^{n+1} = \left(\prod_{j=1}^{N} (I - \tau_j A)\right) U^n \qquad n = 0, 1, \ldots. \tag{2.5}$$

The corresponding stability condition is

$$\rho\Big(\prod_{j=1}^{N} (I - \tau_j A)\Big) < 1. \tag{2.6}$$

This relation is satisfied if

$$|\prod_{j=1}^{N}(1-\tau_j\lambda)| < 1 \qquad \forall\lambda\in[\lambda_{\min},\lambda_{\max}], \tag{2.7}$$

where $0 < \lambda_{\min}$ denotes the smallest eigenvalue of $A$. Among the infinity of choices for the $\tau_j$'s satisfying (2.7), we are looking for an optimal one, namely one that makes

$$\Delta T = \sum_{j=1}^{N}\tau_j \text{ as large as possible.}$$

However, the above problem is not well posed as stated, because in (2.7) we have $<$ and not $\leq$.

In order to obtain "strong" stability, we replace the condition (2.7) by

$$|\prod_{j=1}^{N}(1-\tau_j\lambda)| \leq K \qquad \forall\lambda\in[\mu,\lambda_{\max}], \tag{2.8}$$

where $\mu$ is some number in the interval $(0,\lambda_{min}]$, and $K$ is some number $0<K<1$. The problem of finding the "optimal"values for the $\tau_j$'s can then be reformulated as

*Find* $\tau_1,\tau_2,\ldots,\tau_N$ *such that* $p_N(\lambda)=\prod_{j=1}^{N}(1-\tau_j\lambda)$ *satisfies*

$$\begin{aligned} &|p_N(\lambda)|\leq K && \forall\lambda\in[\mu,\lambda_{\max}] && \text{(STABILITY)},\\ &|p'_N(0)| = \sum_{j=1}^{N}\tau_j \quad \text{maximal} && && \text{(OPTIMALITY)}. \end{aligned}$$

Using the remarkable optimality properties of the Chebyshev polynomials $T_N(\cdot)$ of degree $N$, Markoff [5] (1892!), we have that if $K$ is given by

$$K = 1\,/\,T_N(\frac{\lambda_{\max}+\mu}{\lambda_{\max}-\mu})$$

then the optimal values of the $\tau_j$'s are those for which

$$p_N(\lambda) = \frac{T_N(\dfrac{\lambda_{\max}+\mu-2\lambda}{\lambda_{\max}-\mu})}{T_N(\dfrac{\lambda_{\max}+\mu}{\lambda_{\max}-\mu})}.$$

Notice that, if desired, $K$ may be chosen arbitrarily close to 1, by choosing $\mu$ small enough. Moreover, the $\tau_j$'s corresponding to the above polynomial $p_N$ are explicitly given by

$$\tau_j = 2\left((-\lambda_{\max}+\mu)\cos(\frac{2j-1}{N}\frac{\pi}{2})+\lambda_{\max}+\mu\right)^{-1} \qquad j=1,\ldots N,$$

which can be written more conveniently as

$$\tau_j = \Delta t_{expl}\left((-1+\nu)\cos(\frac{2j-1}{N}\frac{\pi}{2})+1+\nu\right)^{-1} \qquad j=1,\ldots N, \tag{2.9}$$

where $\nu = \mu/\lambda_{\max}$, $0 < \nu < \lambda_{min}/\lambda_{\max}$, $\Delta t_{expl} = 2/\lambda_{\max}$ as in (2.4). One can show the relation

$$\Delta T = \sum_{j=1}^{N} \tau_j = \Delta t_{expl} \frac{N}{2\sqrt{\nu}} \left( \frac{(1+\sqrt{\nu})^{2N} - (1-\sqrt{\nu})^{2N}}{(1+\sqrt{\nu})^{2N} + (1-\sqrt{\nu})^{2N}} \right), \tag{2.10}$$

which yields

$$\Delta T \underset{\nu \to 0}{\longrightarrow} N^2 \Delta t_{expl}. \tag{2.11}$$

Noting that $N$ explicit steps, each of length $\Delta t_{expl}$, cover time $N\Delta t_{expl}$, we see that executing a superstep consisting of $N$ substeps covers a time interval $N$ times longer (when $\nu \approx 0$). Thus, superstepping is (up to) $N$ times faster than the standard explicit scheme, at essentially the same cost! This is where the speed up comes from.

We observe that the error between the exact and approximate solutions is given by

$$\begin{aligned}
\|E^k\| &\equiv \|U(k\Delta T) - U^k\| \\
&= \|e^{-\Delta T A} U((k-1)\Delta T) - \prod_{j=1}^{N} (I - \tau_j A) U^{k-1}\| \\
&\leq \|e^{-\Delta T A} - \prod_{j=1}^{N} (I - \tau_j A)\| \, \|U((k-1)\Delta T)\| \\
&+ \| \prod_{j=1}^{N} (I - \tau_j A)\| \, \|U((k-1)\Delta T) - U^{k-1}\|.
\end{aligned}$$

Since $A$ is positive definite, we have

$$\|U((k-1)\Delta T)\| = \|e^{-(k-1)\Delta T A} U_0\| \leq \|U_0\|.$$

On the other hand, we notice that $A$ being symmetric, so is $\prod_{j=1}^{N}(I - \tau_j A)$, and thus the stability condition (2.6) is equivalent to $\| \prod_{j=1}^{N}(I - \tau_j A)\| < 1$. Putting all this together yields

$$\begin{aligned}
\|E^k\| &\leq \|e^{-\Delta T A} - \prod_{j=1}^{N} (I - \tau_j A)\| \, \|U_0\| + \|E^{k-1}\| \\
&\leq k \, \|e^{-\Delta T A} - \prod_{j=1}^{N} (I - \tau_j A)\| \, \|U_0\|
\end{aligned}$$

Expanding and keeping only the lowest order terms, we get

$$\|E^k\| \leq k \frac{\lambda_{max}}{2} (\sum_{j=1}^{N} \tau_j^2) \|U_0\|.$$

As excepted, the method is essentially of order one with respect to $\Delta T$. If $\nu$ is increased, the length of each of the $\tau_j$'s is decreased and, in view of the last relation, so is the error, at the expense of more computations. The length of the superstep $\Delta T$ (which is determined by the spectral properties of $A$, and the *choice* of $N$ and $\nu$) is only restricted by accuracy, just like with any unconditionally stable implicit scheme.

We remark that the case $\mu = \nu = 0$ corresponds to the stability limit. Therefore choosing $\nu$ too small can lead to a method very sensitive to round-off errors. The situation can be improved by slightly increasing $\nu$, or by using some suitable reordering of the inner steps (see e.g. [3]).

It is worth noting that although we justify the method only in case the operator $A$ appearing in (2.1) is a symmetric positive definite matrix, such an assumption does not appear to be always required in practice. An important practical case is that of an iteration matrix $A$ admitting complex eigenvalues, as may be the case in convection-diffusion problems. We show in the following example that the price to pay for the acceleration of the method is only a slight flattening of the stability region in the complex plane.

Indeed, let us consider the case of a superstep with two inner steps vs. two steps of the standard explicit scheme. Let $G$ denote the discrete iteration operator appearing in (2.5), in terms of which the region of stability on the complex plane is $\{z \in \mathbb{C}; |G(z)| < 1\}$. Roughly speaking, the variable $z$ now plays the role of $\Delta t_{expl} A$. Using (2.9), we have

$$G_{expl}(z) = (1-z)^2 \qquad \text{for two steps of the explicit scheme,}$$
$$G_\nu(z) = (1-\delta_1 z)(1-\delta_2 z) \qquad \text{for the STS method with two inner steps,}$$

where $\delta_{1,2} = 1/((1+\nu) \pm (-1+\nu)/\sqrt{2})$. In this normalized framework, the length of the superstep is larger than two standard steps if $\delta_1 + \delta_2 > 2$. Figure 1.a shows the region of stability for $G_{expl}$ (circle) and $G_\nu$, for $\nu = 0, 0.04, 0.08, 0.12, 0.16, 0.20$, for which $\delta_1 + \delta_2$ is respectively equal to $4, 3.35, 2.90, 2.58, 2.34, 2.14$.

The stability regions are increasing with $\nu$. Again, $\nu = 0$ corresponds to the stability limit, has the smallest stability region ($\infty$-shaped domain), and is clearly not

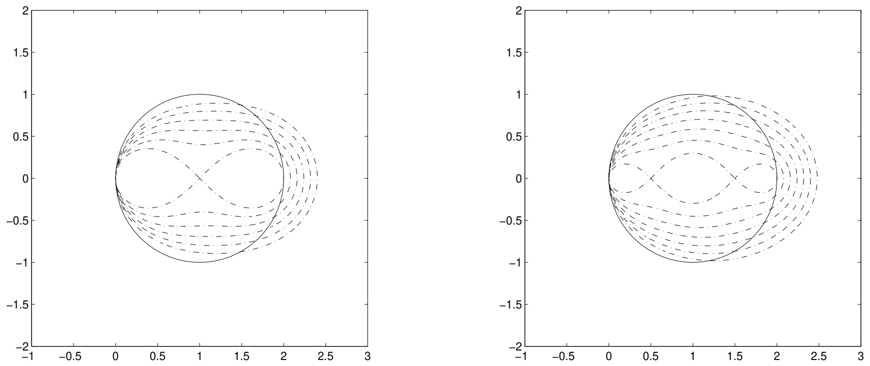

FIGURE 1. *Stability region for the standard explicit scheme (circle) and for STS with various values of $\nu$ for $N = 2$ (Fig. 1.a), and for $N = 3$ (Fig. 1.b).*

to be considered for any practical purposes. Figure 1.b, shows the stability regions for $N = 3$.

In other words, the acceleration, up to a factor $N$, provided by superstepping, is obtained at the very small price of a slight reduction of the stability region along the imaginary axis (accompanied by an increase along the real axis).

Finally, as will be clear from the numerical experiments presented in the next sections, STS performs very well not only in the linear case, which we have justified here, but even when $A$ represents a nonlinear operator. The theoretical difficulties in justifying the nonlinear case come from the fact that for a time dependent family of matrices $A(t)$ the symmetry of the products $\prod_{j=1}^{N}(I-\tau_j A(\cdot))$ is no longer guaranteed, nor is the commutativity of the terms entering in those products.

**3. Implementation.**

Super-time-stepping may be applied easily to any explicit time-stepping algorithm of the form (2.2), so in particular to parabolic problems.

Even though the spectral properties of the operator $A$ play an important role in the theoretical justification presented in section 2, no precise knowledge of those properties is required for implementation purposes. The method is robust in this respect. This is true not only for the examples presented below, but also in general, as, for instance, in the three-dimensional industrial computations in [2]. One determines the explicit time-step $\Delta t_{expl}$ in the usual way to satisfy the CFL condition, but instead of executing steps of length $\Delta t_{expl}$, one executes supersteps of length $\Delta T$ as follows: choose $N$, $\nu$ and execute the $N$ substeps $\tau_1, \tau_2, \ldots, \tau_N$, of (2.9), without outputing until the end of each super-step. The only additional expense is the trivial computation in (2.9), while the execution is accelerated by (up to) a factor of $N$.

If one is to integrate (2.1) up to a time $T$, the standard explicit method requires at least $n_{expl} = T/\Delta t_{expl}$ explicit steps, all the computed values being approximations to the solution. The small time-step, forced upon the scheme by stability, yields high accuracy, which often exceeds the user's requirements, and is obtained at high cost. On the other hand, both STS and the implicit methods allow the choice of a (super) time-step $\Delta T$ as large as desired. The choice here is based on accuracy rather than stability requirements. STS with $N$ inner steps executes at least $N\,T/\Delta T$ explicit steps, where $\Delta T$ is determined by (2.10), and yields $T/\Delta T$ approximate values of the solution. An implicit method yielding an equal number of approximate values only requires $T/\Delta T$ implicit steps, but each of them is computed by an indeterminate number of iterations. Thus, effectively, super-time-stepping frees the explicit scheme from the stability restriction on the time-step, rendering it as usable as any implicit scheme, while retaining its simplicity and better accuracy.

Let us remark that our approach, as the one in [2,3,4], differs from the approach advocated in [6,7,9]. We consider a factorized implementation into a sequence of explicit steps, as opposed to using a three-term recursion and so-called diagonal implementation. The latter may yield better inner stability, i.e. stability of the inner steps. However, as shown in the following examples and as observed in [2,4], inner stability problems do not seem to corrupt the solution beyond very acceptable levels. The factorized implementation has the great advantage that it can be incorporated directly in an existing explicit code, as described above. In addition, being a single-step method, it requires less storage than the diagonal implementation, which uses two steps. Finally, since our implementation does not rely on precise knowledge of spectral properties, it may be applied to complicated problems and not to just a few textbook problems.

**4. Performance on linear and nonlinear problems.**

We examine the performance of the Super-Time-Stepping scheme on two exactly solvable model problems: a plain heat conduction problem (linear) and the classical 2-phase Stefan problem (nonlinear), in one space dimension. In each case, we compare with standard versions of the Crank-Nicolson and/or full implicit scheme, one using SOR and the other Newton iterations for solving the resulting system. The model problems are the following.

**Problem 1:** Consider the heating of a slab in $x > 0$, initially at (normalized) temperature $u(x, 0) = 0$, when $u = 1$ is imposed at $x = 0$. Assuming constant thermophysical properties, in dimensionless variables the temperature $u(x, t)$ satisfies the heat equation $u_t = u_{xx}$. The problem admits an exact (similarity) solution in terms of the error function, given by

$$u(x,t) = 1 - \operatorname{erf}(x/\,2\sqrt{t}), \qquad x > 0, \quad t > 0. \tag{4.1}$$

**Problem 2:** Consider the melting of a slab in $x > 0$, initially solid at (normalized) temperature $u(x, 0) = -1$, due to $u = +1$ imposed at $x = 0$. A melt front $x = X(t)$ originates from $x = 0$ at $t = 0$, separating liquid in $0 \le x < X(t)$ from solid in $X(t) < x$, $t > 0$. Assuming constant thermophysical properties and, for simplicity here, the same in both phases, in dimensionless variables the unknown temperature $u(x, t)$ and interface $X(t)$ satisfy (see [1]) the heat equation $u_t = u_{xx}$ in both phases, and the interface conditions: $u(X(t), t) = 0$, and $X'(t) = St[-u_x(X(t)^-, t) + u_x(X(t)^+, t)]$, where $St$ denotes the Stefan number of the process (ratio of sensible to latent heat, here equal to one over latent heat due to our normalizations). This is one of the few exactly solvable phase-change problems, admitting a similarity solution (known as the Neumann solution) in terms of the error function, [1]. Problem 1 is simply the case of zero latent heat (and initial temperature equal to zero).

The most convenient, general, and effective numerical method for treating such problems is the **enthalpy method**, [1], which approximates a weak formulation of the problem (expressing directly the physics), known as the enthalpy formulation. It is a fixed-domain method, in which only the enthalpy (energy) is updated from the conservation law and it determines the phase and temperature. Thus, it is a "front capturing" scheme, as opposed to "front tracking". Its discretization by the finite volume method (integrated finite differences) is particularly simple and robust, retaining conservation at the discrete level. It is described and studied in great detail in [1], where the performance of the standard explicit scheme is compared with two versions of the implicit scheme (SOR and Newton iterations), using the Neumann solution to determine accuracy. In what follows, we present a similar comparison with the Super-Time-Stepping scheme.

We solve the general enthalpy formulation of these model problems numerically on the interval $0 \le x \le 1$ using $M = 100$ equi-spaced nodes. At the back face $x = 1$ we impose the values of the exact solution ((4.1) or the Neumann solution) itself evaluated at $x = 1$ at any desired time. This way we are solving numerically the same problems the exact solutions solve and we can find the error in the numerical solutions by direct comparison with the exact solutions. The Dirichlet boundary conditions restrict the time-step of the explicit scheme to $\Delta t_{expl} := \Delta x^2/3$.

The schemes are applicable to general conduction/diffusion problems with non-constant (even u-dependent) coefficients, i.e. they are not customized or optimized in any sense for the simple problems under consideration here.

We discretize the partial differential equation by (integrated) finite-differences, and compare the performance of the explicit Super-Time-Stepping scheme and the fully implicit and/or Crank-Nicolson scheme with SOR iterations and with Newton iterations (using the direct tridiagonal solver for the Newton step).

In the explicit STS scheme we vary $N$ and $\nu$, whereas in the implicit schemes we increase the time-step $\Delta t$ in multiples of the explicit time-step $\Delta t_{expl} := \Delta x^2/3$, setting $\Delta t = factor \times \Delta t_{expl}$ with $factor = 10, 20, 40, \ldots$. The SOR extrapolation parameter $\omega$ is found from linear SOR theory for the heat equation, and by trial for the Stefan problem since linear theory is inapplicable there. We report a value of $\omega$ that minimized the number of iterations.

To determine the accuracy of each scheme we perform a large number of direct comparisons with the exact solution. As remarked earlier, one cannot compare more frequently than $N$. We tried to choose the frequency of comparisons so that the total number of comparisons (3rd column of the tables) would be roughly the same on all schemes. In Tables 1 and 2, in addition to the values of the parameters $N$, $\nu$ or $factor$, $\omega$, we report the following quantities for each run.

cmps = *number of comparisons with exact solution*
$E\_max$ = *maximum error in temperature at the fixed locations:*
*0.,.1, ... , .9, 1 (linearly interpolated from nodal values)*
*over all the comparisons,*
$E\_L^1$ = *maximum $L^1$-error in temperature at the nodes over all the*
*comparisons (computed via trapezoidal rule integration),*
$E\_front$ = *maximum error in melt-front location over all the comparisons*
*(defined only for the Stefan problem),*
s-steps = *number of supersteps in the STS scheme,*
t-steps = *total number of time-steps,*
iters = *number of iterations performed.*

Iterations of the implicit schemes correspond to time-steps of the explicit scheme, so they are listed in the same column. We used a tolerance of 1.E-6 for convergence in the iterative schemes. Since the temperature range is normalized to $0 \leq u \leq 1$ and also $0 \leq X \leq 1$, the reported errors may be viewed as percent errors.

As $\Delta t$ is increased the errors grow, and only runs with both errors less than about 6% are considered. The computations were performed, in single precision, on a SUN Sparc 10 Model 40 workstation, and we record the CPU units (user time) reported by the UNIX time command. Coming from a networked unix machine, this is a rather unreliable number, but in conjunction with the total number of time-steps or iterations, it is a useful measure of efficiency. Note that $N = 1$, $\nu = 0$ is the standard explicit scheme itself.

The best performance of STS for this problem was achieved for $N = 9$, $\nu = .001$; it integrated the PDE to the desired time in only 20495 time-steps (down from 166494 of the standard explicit scheme), an 8-fold speed-up, with $E\_max = 0.087$ and $E\_L^1 = 0.022$. By increasing the parameter $\nu$ to 0.005, the errors were reduced to 0.025 and 0.0004 respectively, at somewhat higher cost (27585 time-steps).

Among the implicit schemes, Crank-Nicolson / SOR with $\Delta t = 200 \times \Delta t_{expl}$ was fastest (30055 iterations). Its accuracy is comparable to that of the fastest STS, but its speed is 30% slower. On the other hand, its cost is comparable to STS with $N = 9$, $\nu = 0.005$, but its errors are at least 2.5 times worse.

With Newton iterations, far fewer iterations were performed, but each Newton iteration is so much more expensive that efficiency suffers considerably, and in fact it deteriorated for $factor > 120$. With large $\Delta t$ (i.e. $factor > 120$), Newton iterations on the fully implicit (backward Euler) did better than on Crank-Nicolson, but it still falls far short of STS in both cost and accuracy.

TABLE 1
*Performance of STS and Implicit Schemes on the Heat Equation during $0 \leq t \leq 5$, with $M = 100$ nodes.*

| $N$ | $\nu$ | cmps | $E\_max$ | $E\_L^1$ | s-steps | t-steps | CPU |
|---|---|---|---|---|---|---|---|
| SuperStep | | | | | | | |
| 1 | 0.0 | 833 | 0.0006 | 0.00003 | 166494 | 166494 | $27.8u$ |
| 7 | 0.0015 | 933 | 0.096 | 0.015 | 3729 | 26103 | $4.6u$ |
| 9 | 0.001 | 759 | 0.087 | 0.0022 | 2280 | 20495 | $3.7u$ |
| 9 | 0.005 | 767 | 0.025 | 0.0004 | 3067 | 27585 | $4.8u$ |
| 20 | 0.006 | 649 | 0.043 | 0.0086 | 1301 | 25968 | $4.4u$ |
| factor | $\omega$ | cmps | $E\_max$ | $E\_L^1$ | t-steps | iters | CPU |
| C-N SOR | | | | | | | |
| 20. | 1.40 | 834 | 0.023 | 0.005 | 8349 | 73242 | $11.4u$ |
| 40. | 1.50 | 834 | 0.022 | 0.006 | 4174 | 56597 | $8.6u$ |
| 80. | 1.60 | 695 | 0.022 | 0.009 | 2087 | 44002 | $6.6u$ |
| 120. | 1.60 | 695 | 0.022 | 0.014 | 1391 | 35084 | $5.3u$ |
| 200. | 1.70 | 834 | 0.065 | 0.025 | 834 | 30055 | $4.6u$ |
| C-N Newton | | | | | | | |
| 20. | | 834 | 0.000 | 0.001 | 8349 | 8591 | $33.0u$ |
| 40. | | 834 | 0.001 | 0.004 | 4174 | 4418 | $17.2u$ |
| 80. | | 695 | 0.010 | 0.010 | 2087 | 2377 | $9.4u$ |
| 120. | | 695 | 0.041 | 0.017 | 1391 | 1979 | $8.4u$ |
| 160. | | 521 | 0.059 | 0.020 | 1043 | 2603 | $10.2u$ |
| 200. | | 834 | 0.069 | 0.029 | 834 | 3177 | $12.2u$ |
| Impl Newt | | | | | | | |
| 20. | | 834 | 0.013 | 0.002 | 8349 | 8591 | $33.1u$ |
| 40. | | 834 | 0.024 | 0.004 | 4174 | 4446 | $17.3u$ |
| 80. | | 695 | 0.041 | 0.007 | 2087 | 2474 | $9.8u$ |
| 120. | | 695 | 0.058 | 0.010 | 1391 | 1938 | $8.2u$ |
| 160. | | 521 | 0.064 | 0.011 | 1043 | 1661 | $6.5u$ |
| 200. | | 834 | 0.086 | 0.016 | 834 | 1600 | $6.5u$ |

Table 2 shows similar comparisons on the Stefan Problem. In addition to the pointwise and $L^1$ errors on temperature, we show the error in interface location, $E\_front$, again found by comparison with the exact (Neumann) solution. The phase-change renders the problem nonlinear and degrades the performance of all the schemes considerably.

The best performance of STS now is for $N = 5$, $\nu = .008$; it reduced the 166494 steps of the standard explicit scheme to 41801 steps, a 4-fold speedup. Again, by increasing the parameter $\nu$ to 0.04, the errors become lower than even those of the explicit scheme itself, at a respectable speedup by a factor of 2.4.

The Crank-Nicolson scheme with either SOR or Newton iterations shows unacceptably large errors for large $\Delta t$. On the other hand, the fully implicit / SOR scheme becomes competitive with STS when $factor = 200$, at which it achieves the same front- and $L^1$-errors but its temperature-error ($E\_max$) is 3 times that of STS. Newton iterations again prove to be too expensive.

At the same cost, the best performing implicit scheme is the fully implicit / SOR

TABLE 2
*Performance of STS and Implicit Schemes on the Stefan Problem For $St = 0.1$ during $0 \le t \le 5$, with $M = 100$ nodes.*

| $N$ | $\nu$ | cmps | $E\_front$ | $E\_max$ | $E\_L^1$ | s-steps | t-steps | CPU |
|---|---|---|---|---|---|---|---|---|
| SupStep | | | | | | | | |
| 1 | 0.0000 | 834 | 0.0005 | 0.038 | 0.003 | 166494 | 166494 | $36.4u$ |
| 5 | 0.0080 | 834 | 0.0100 | 0.038 | 0.021 | 8346 | 41801 | $9.3u$ |
| 5 | 0.0400 | 862 | 0.0001 | 0.026 | 0.003 | 13808 | 69164 | $14.9u$ |
| 7 | 0.0165 | 810 | 0.0100 | 0.056 | 0.024 | 6457 | 45276 | $10.0u$ |
| 10 | 0.0400 | 833 | 0.0200 | 0.080 | 0.016 | 6674 | 66848 | $14.5u$ |
| 20 | 0.1000 | 878 | 0.0300 | 0.039 | 0.028 | 5271 | 105627 | $22.5u$ |
| factor | $\omega$ | cmps | $E\_front$ | $E\_max$ | $E\_L^1$ | t-steps | iters | CPU |
| C-N SOR | | | | | | | | |
| 20. | 1.50 | 834 | 0.005 | 0.071 | 0.011 | 8349 | 106195 | $25.8u$ |
| 40. | 1.60 | 834 | 0.005 | 0.069 | 0.015 | 4174 | 93567 | $22.0u$ |
| 80. | 1.60 | 695 | 0.005 | 0.062 | 0.029 | 2087 | 61882 | $14.5u$ |
| 200. | 1.70 | 834 | 0.029 | 0.170 | 0.077 | 834 | 44064 | $10.4u$ |
| Impl SOR | | | | | | | | |
| 20. | 1.50 | 834 | 0.006 | 0.071 | 0.012 | 8349 | 106883 | $25.9u$ |
| 40. | 1.60 | 834 | 0.006 | 0.071 | 0.012 | 4174 | 83530 | $19.9u$ |
| 80. | 1.60 | 695 | 0.009 | 0.067 | 0.016 | 2087 | 60944 | $14.3u$ |
| 200. | 1.70 | 834 | 0.009 | 0.077 | 0.021 | 834 | 41655 | $9.9u$ |
| C-N Newt | | | | | | | | |
| 20. | | 834 | 0.000 | 0.038 | 0.006 | 8349 | 17069 | $134.8u$ |
| 40. | | 834 | 0.001 | 0.041 | 0.014 | 4174 | 8616 | $68.2u$ |
| 80. | | 695 | 0.000 | 0.065 | 0.026 | 2087 | 4561 | $36.2u$ |
| 200. | | 834 | 0.005 | 0.162 | 0.089 | 834 | 2172 | $17.6u$ |
| Impl Newt | | | | | | | | |
| 20. | | 834 | 0.001 | 0.038 | 0.005 | 8349 | 17367 | $137.0u$ |
| 40. | | 834 | 0.001 | 0.038 | 0.006 | 4174 | 8855 | $70.1u$ |
| 80. | | 695 | 0.002 | 0.034 | 0.006 | 2087 | 4859 | $38.5u$ |
| 200. | | 834 | 0.003 | 0.077 | 0.019 | 834 | 2323 | $18.7u$ |

at $factor = 80$, but its errors are considerably worse.

**5. Conclusions.** The paper describes a simple method that stabilizes and accelerates the explicit (forward Euler) scheme for parabolic problems. The method is a variant of Gentzsch's super-time-stepping scheme and it belongs to the Runge-Kutta-Chebyshev class of methods.

The analysis and computations presented show that super-time-stepping is not only very simple to implement in an existing explicit code, but also impressively effective. It runs at least as fast, it achieves comparable or better accuracy, and it is of course much easier to program (and to parallelize for distributed computing) than standard implicit schemes for parabolic problems.

It should be noted that STS is independent of space dimensionality, so it applies as well in higher dimensions. Given the increased complexity and cost of higher dimensional implicit schemes, STS is even more effective there.

**Acknowledgements.** We are grateful to Steve Campbell and to the referees for several remarks that led to a better presentation of this paper.

REFERENCES

[1] V. Alexiades & A.D. Solomon, *Mathematical Modeling of Melting and Freezing Processes*, Hemisphere Publ. Co., Washington DC, 1993.

[2] J.-J. Droux, *Three-dimensional numerical simulation of solidification by an improved explicit scheme*, Comp. Meth. Appl. Mech. Engrg., 85 (1991), pp. 57–74.

[3] W. Gentzsch, *Numerical solution of linear and non-linear parabolic differential equations by a time-discretisation of third order accuracy*, in Proceeding of the Third GAMM-Conference on Numerical Methods in Fluid Mechanics, E.H. Hirschel, ed., Friedr. Vieweg & Sohn, 1979, pp. 109–117.

[4] W. Gentzsch & A. Schluter, *On one-step methods with cyclic stepsize changes for solving parabolic differential equations (German)*, Z. Angew. Math. Mech., 58 (1978), pp. T415–T416.

[5] W. Markoff, *Über Polynome, die in einem gegebenen Intervall möglichst wenig von Null abweichen*, Ann., 77 (1892 (translation and condensation by J. Grossman of Russian article published in 1892)), pp. 213–258.

[6] P.J. van der Houwen, *Construction of Integration Formulas for Initial Value Problems*, North-Holland, 1977.

[7] P.J. van der Houwen, *The development of Runge-Kutta methods for partial differential equations*, preprint, presented at the 14th IMACS World Congress, Atlanta, GA. (1994).

[8] R.S. Varga, *Matrix Iterative Analysis*, Prentice-Hall, 1962.

[9] J.G. Verwer, W.H. Hundsdorfer & B.P. Sommeijer, *Convergence properties of the Runge-Kutta-Chebyshev method*, Numer. Math., 57 (1990), pp. 157–178.