UCLA
COMPUTATIONAL AND APPLIED MATHEMATICS

Accelerated Solutions of Nonlinear Equations Using Stabilized Runge-Kutta Methods

Christopher R. Anderson

Christopher J. Elion

May 2004

CAM Report 04-26

Department of Mathematics
University of California, Los Angeles
Los Angeles, CA. 90095-1555

ACCELERATED SOLUTIONS OF NONLINEAR EQUATIONS USING STABILIZED RUNGE-KUTTA METHODS*

CHRISTOPHER R. ANDERSON† AND CHRISTOPHER J. ELION‡

Abstract. In this paper we discuss the use of stabilized Runge-Kutta methods to accelerate the solution of systems of nonlinear equations. The general idea is to seek solutions as steady state solutions of an associated system of ordinary differential equations. A class of stabilized Runge-Kutta methods are derived that can be used to efficiently evolve the associated system to steady state. Computational results for a set of reaction-diffusion equations and a set of Schroediner-Poisson equations are presented.

Key words. nonlinear equations, ordinary differential equations, explicit Runge–Kutta methods

AMS subject classifications. 47J25, 65L06

For large coupled systems of nonlinear equations, typically arising from discretization of nonlinear partial differential equations, it is common practice to obtain solutions by evolving an associated system of ordinary differential equations to a steady state. In its simplest form, the solution of a system

$$\vec{F}\,(\vec{y}) = 0 \tag{0.1}$$

where $\vec{F} : \mathcal{R}^n \to \mathcal{R}^n$ is obtained by solving the ordinary differential equation

$$\frac{d\vec{y}}{dt} = \vec{F}\,(\vec{y}) \tag{0.2}$$

to steady state. Variants of this procedure include evolving a "preconditioned" system to steady state; e.g. if $\vec{P}$ is a non-singular function, then one evolves

$$\frac{d\vec{y}}{dt} = \vec{P}(\vec{F}\,(\vec{y})) \tag{0.3}$$

to steady state. Obtaining solutions to nonlinear equations via steady state calculations is particularly attractive if the resulting ODE's can be successfully solved using explicit schemes. For, in such cases, the solution is obtained without the need to compute gradients of $\vec{F}$ (or in the case of the preconditioned system, gradients of $\vec{P}(\vec{F}\,(\vec{y}))$) and the solution procedure is not difficult to implement.

One drawback of this approach is that the resulting ODE's may be "stiff" and, consequently, commonly used explicit methods require that a large number of timesteps be taken to achieve the steady state solution. One can try to remedy this situation by choosing appropriate preconditioners. For example, the ODE system (0.2) is stiff when the eigenvalues of the Jacobian of $\vec{F}$ differ by several orders of magnitude — thus one tries to pick the preconditioner $\vec{P}$ so that the eigenvalues of the Jacobian

*Research supported by DARPA through the Quantum Information Science and Technology (QuIST) program under Army Research Office contract number DAAD-19-01-C-0077.

†UCLA Mathematics Department Box 951555, Los Angeles, CA 90095-1555 (anderson@math.ucla.edu).

‡UCLA Mathematics Department Box 951555, Los Angeles, CA 90095-1555 (elion@math.ucla.edu).

of $\vec{P}(\vec{F}\,(\vec{y}))$ are tightly clustered. However, even with preconditioning, the resulting system is often mildly stiff and standard explicit methods can still require a large number of timesteps to achieve a steady state solution.

A family of methods known as "stabilized Runge-Kutta methods" have proven effective as a means of evolving mildly stiff differential equations [3][5][6]. (See [5] for good review of these procedures and references to previous work.) The stabilized methods differ from the standard Runge-Kutta methods in that they are of moderate accuracy and have extra stages chosen so that they provide solutions without requiring excessively small timesteps. These methods are thus excellent candidates as methods to evolve systems of differential equations to steady state. The primary purpose of this paper is to derive a family of stabilized Runge-Kutta methods that are particularly suited to steady state calculations. One can use the methods developed in [4][8][9] directly, however, by taking advantage of the fact that we are only interested in obtaining steady state solutions, one can develop a family of methods that have large stability regions, high damping properties, and have coefficients that are relatively easy to construct.

Another technique for solving sets of nonlinear equations consists of transforming the equations into an equivalent fixed point problem

$$\phi = \mathrm{H}(\phi) \tag{0.4}$$

and then using the standard fixed point iteration

$$\phi_{\,m+1} = \mathrm{H}(\phi_{\,m})$$

to obtain a solution. When this simple iteration strategy does not converge, one often introduces a relaxation factor γ and uses instead the iteration

$$\phi^{\,*} = \mathrm{H}(\phi_{\,m})$$

$$\phi_{\,m+1} = \gamma\ \phi^{\,*} + (1-\gamma)\ \phi_{\,m} \tag{0.5}$$

For many problems convergence can be obtained with a sufficiently small value of γ. However, even though a convergent iteration is obtained, the iteration may require a large number of iterations to achieve a solution. The starting point for a method to accelerate slowly convergent iterations is to recognize that fixed point iteration with a relaxation factor is equivalent to solving the ordinary differential equation

$$\frac{d\phi}{dt} = H(\phi) - \phi \tag{0.6}$$

using Euler's method. Specifically, eliminating $\phi^{\,*}$ from the iteration (0.5) we have

$$\begin{aligned}
\phi_{\,m+1} &= \gamma H(\phi_{\,m}) + (1-\gamma)\phi_{\,m} \\
\Rightarrow \phi_{\,m+1} - \phi_{\,m} &= \gamma H(\phi_{\,m}) - \gamma\phi_{\,m} \\
\Rightarrow \frac{\phi_{\,m+1} - \phi_{\,m}}{\gamma} &= H(\phi_{\,m}) - \phi_{\,m}
\end{aligned}$$

The last expression is just Euler's method applied to (0.6) with a timestep dt = γ. The need to use a small relaxation factor and a large number of iterations to obtain a solution with the iteration (0.5) occurs when (0.6) is a relatively stiff differential equation. Therefore, a means of accelerating the convergence of fixed point iterations consists of simply applying a stabilized Runge-Kutta method to (0.6) directly.

In the first section we outline general properties that an explicit one-step ODE method should possess if it is to be used for the computation of steady state solutions and then derive a family of stabilized Runge-Kutta methods that satisfy these properties. In the second section we discuss an adaptive time-step selection strategy followed by computational examples demonstrating the effectiveness of using stabilized Runge-Kutta methods to compute steady state solutions. Two of the examples concern the solution of a set of reaction diffusion equations and the third concerns the use of stabilized Runge-Kutta methods to accelerate a fixed point iteration that computes solutions to a set of Schreodinger-Poisson equations occurring in quantum device simulations.

1. Stabilized Runge-Kutta methods for steady state calculations. When using an explicit one step method to evolve solutions of an ODE to steady state, three desirable properties of the method are

(i) The method is convergent and at least first order accurate.

(ii) For large dt, the numerical solution locally (in time) exhibits qualitatively correct behavior as the true solution.

(iii) (Numerical fixed point condition) The method has the form $\vec{y}_{m+1} = \vec{G}_{dt}(\vec{y}_m)$ with $\vec{G}_{dt}$ having the property that if $\vec{F}(\vec{y}^*) = 0$ then $\vec{y}^*$ is a fixed point of the numerical method, e.g. $\vec{y}^* = \vec{G}_{dt}(\vec{y}^*)$ for all dt.

The first condition ensures that in the limit as the timestep tends to zero, the behavior of the numerical solutions and the exact solutions of the ODE coincide. Hence, if a true solution approaches a steady state solution then we have confidence that by taking sufficiently small timesteps the numerical approximation to this solution will also approach the steady state solution. Of course, when actually using the method to compute the steady state solutions, we attempt to use as large a timestep as possible. As one evolves to a steady state using such large timesteps, the numerical solution may not be particularly accurate and the second condition is a reflection of the fact that with such timesteps we at least desire that the numerical solution exhibit locally correct qualitative behavior. In particular, if locally the solution trajectories of the differential equation are converging to a single trajectory then we would like for the numerical trajectories to have the same behavior. The third condition is desired because it implies that the steady state solution is a fixed point of the numerical method for non-zero dt. Thus, if the numerical method is evolved to steady state, $\vec{y}^* = \vec{G}_{dt}(\vec{y}^*)$, then the accuracy with which $\vec{y}^*$ is calculated does not depend upon dt.

There are many explicit one step methods that satisfy properties (i) and (iii) above — for example, Euler's method or the standard Runge-Kutta methods. It is the creation of methods that satisfy condition (ii) that presents the most difficulty. To determine methods that satisfy condition (ii) requires an understanding of the region of absolute stability associated with an ODE method [1]. The region of absolute

stability of a method is that region, R, in the complex plane so that if $dt\lambda \in$R, then for any initial condition y_0, the numerical solution to

$$\frac{dy}{dt} = \lambda\ y \tag{1.1}$$

obtained with the method has the property that $y_m \to 0$ as $m \to \infty$. For a system of ordinary differential equations, the behavior of solution trajectories near a steady state is determined by the linearized system

$$\frac{d\vec{z}}{dt} = \mathrm{J}F\ \ \vec{z}$$

where $\mathrm{J}F$ is the Jacobian of $\vec{F}\,(\vec{y})$. If all of the eigenvalues, λ_i, of $\mathrm{J}F$ have negative real parts, then solution trajectories of the linearized system decay to the steady state solution as $t \to \infty$. The condition that the numerical solution also exhibits this property requires that the timestep be chosen sufficiently small so that for all λ_i, $dt\lambda_i$ is contained within the region of absolute stability of the method. Thus, if we are to have methods that exhibit locally correct behavior for large dt, we seek explicit methods that have regions of absolute stability that contain a large portion of the negative real axis.

A family of explicit methods that have very large regions of absolute stability are the stabilized (or Chebyshev) Runge Kutta methods [3][5][6]. These methods differ from the standard Runge-Kutta methods in that they are of moderate accuracy and have extra stages chosen so that their regions of absolute stability contain a large portion of the negative real axis. One can directly employ the methods presented in [4][8][9], however, as we will discuss next, methods that are designed specifically for evolution to steady state can be constructed. In particular, one can obtain methods that have large stability regions, good damping properties, and satisfy the third (fixed point) condition described above.

In our construction of a class of stabilized Runge-Kutta methods appropriate for steady state problems, we follow the same general procedure as that used in [5]. When an n-stage Runge-Kutta method is applied to the model problem (1.1) the method can be expressed as $y_{m+1} = p_n(dt\lambda)y_m$ where $p_n(s)$ is a polynomial of degree n. The region of absolute stability of the method is the set $\{s \in \mathbf{C} \mid |p_n(s)| < 1\}$. We refer to p_n as the "absolute stability polynomial" associated with the method. The first step in the construction of the stabilized methods is to identify polynomials of degree n, $p_n(s)$, that satisfy constraints imposed by accuracy considerations and for which the set $\{s \in \mathbf{C} \mid |p_n(s)| < 1\}$ contains as large a portion of the negative real axis as possible. The second step consists of creating explicit n stage Runge-Kutta type methods that have as their absolute stability polynomials the identified polynomials $p_n(s)$.

The first order accuracy condition imposes on $p_n(s)$ the constraints $p_n(0) = 1$ and $p_n'(0) = 1$. The shifted Chebyshev polynomial, $\mathrm{T^*}_n(s) = \mathrm{T}_n(-1 + 2\frac{(s+\mathrm{M})}{\mathrm{M}})$, with $\mathrm{T}_n(x)$ being the nth Chebyshev polynomial, is a polynomial of degree n that satisfies $p_n(0) = 1$ and $p_n'(0) = 1$. Moreover, $|\mathrm{T^*}_n(s)| \leq 1$ for $s \in [-2n^2, 0]$, so that a method whose absolute stability polynomial is the nth shifted Chebyshev polynomial will have a region of absolute stability that contains the interval $[-2n^2, 0]$ except at the $n+1$ locations $s_j = n^2(\cos(\frac{j\pi}{n}) - 1)$, $j = 0 \ldots n$, where $|\mathrm{T^*}_n(s_j)| = 1$. To create methods

that have the property that $|p_n(s)| < 1$ over a large interval of the negative real axis, we follow [5] and for a given value of $\gamma < 2$ consider the nth degree polynomial

$$\bar{\mathrm{T}}_n^{\gamma}(s) = \frac{\mathrm{T}_n(-1 + 2\dfrac{(s+\gamma n^2)}{\gamma n^2\text{-}\delta}\ )}{\mathrm{T}_n(-1 + \dfrac{2\gamma n^2}{\gamma n^2\text{-}\delta})}$$

with δ chosen so that $\bar{\mathrm{T}}_n^{\gamma\,\prime}(0) = 1$. For a given γ, such a δ is readily found using standard root finding techniques. The resulting polynomial satisfies the conditions that $\bar{\mathrm{T}}_n^{\gamma}(0) = 1$, $\bar{\mathrm{T}}_n^{\gamma\,\prime}(0) = 1$ and over the interval $[-\gamma n^2, -\delta]$ has the maximal value $\dfrac{1}{\mathrm{T}_n(-1 + \dfrac{2\gamma n^2}{\gamma n^2\text{-}\delta})} < 1$. Since $\mathrm{T}_n(-1 + 2\dfrac{(s+\gamma n^2)}{\gamma n^2\text{-}\delta}\ )$ grows extremely fast outside of the interval $[-\gamma n^2, -\delta]$ [7] one gets a rather dramatic reduction in the size of $\max\limits_{s\in[-M,-\delta]} |p_n(s)|$ as a function of γ. In Table 1 we give the maximal value of the stability polynomial for various stage orders and stability bound factors γ.

	$\gamma = 1.0$	$\gamma = 1.25$	$\gamma = 1.5$	$\gamma = 1.75$	$\gamma = 2.0$
$n = 2$	0.236	0.415	0.605	0.801	1.00
$n = 3$	0.266	0.437	0.620	0.808	1.00
$n = 4$	0.276	0.445	0.625	0.810	1.00
$n = 5$	0.280	0.448	0.627	0.811	1.00

Table 1 : $\max|\bar{T}_n^{\gamma}(s)|$ for s $\in [-\ \gamma n^2, -\delta_\gamma]$.

The target stability polynomial we use in the construction of our numerical methods is $\bar{\mathrm{T}}_n^{\gamma}(s)$ where $\bar{\mathrm{T}}_n^{\gamma}(s)$ has been constructed using a value of specified stability bound factor $\gamma \leq 2$. There are a variety of ways to formulate Runge-Kutta methods whose absolute stability polynomials match the polynomials $\bar{\mathrm{T}}_n^{\gamma}(s)$. After some experimentation, the method that we settled on has the general form

$$\begin{aligned}
g_1 &= dt * f(y_m) \\
g_2 &= dt * f(y_m + \alpha_1^1\ g_1) \\
g_3 &= dt * f(y_m + \alpha_1^2\ g_1 + \alpha_2^2\ g_2) \\
&\vdots \\
g_n &= dt * f(y_m + \alpha_1^{n-1}\ g_1 + \alpha_2^{n-1}\ g_2 + + \cdots + \alpha_{n-1}^{n-1}\ g_{n-1}) \\
y_{m+1} &= y_m + \alpha_1^n\ g_1 + \alpha_2^n\ g_2 + \cdots + \alpha_n^n\ g_n
\end{aligned} \tag{1.2}$$

This form was selected so that the coefficients of the intermediate stages could be determined so that when $s \in [-\gamma k^2, 0]$ (e.g. the stability interval for $\bar{\mathrm{T}}_k^{\gamma}(s)$) then s is also contained within the stability intervals associated with each intermediate stage (the argument of f in the calculation of each g_k). Of course, the use of $r-1$ non-zero stage coefficients in the rth stage requires the storage and accumulation of all $r-1$ previous stages. When n is large this can be computational expensive; however, in

our experiments we found that modest order methods ($n \leq 10$) were quite effective and thus the extra cost of storing and accumulating the stages was not significant. The method (1.2) also has the property that when $f(y_m) = 0$ the mapping from y_m to y_{m+1} is the identity operator so that the fixed point condition of the numerical method is satisfied. One can create a Runge-Kutta type method that directly utilizes the recurrence property of the Chebyshev polynomial; however, this was not done because when $\gamma \neq 2$ the resulting method does not satisfy the fixed point property (iii).

If we let $\mathrm{s} = \mathrm{dt}\lambda$ then the stability polynomial, $p_n(s)$, associated with an n stage method of the form (1.2) can be constructed recursively;

$$\begin{aligned}
z_1(s) &= s \\
p_1(s) &= 1 + \alpha_1^1 \; z_1(s) \\
z_2(s) &= s \; p_1(s) \\
p_2(s) &= 1 + \alpha_1^2 \; z_1(s) + \alpha_2^2 \; z_2(s) \\
&\vdots \\
p_n(s) &= 1 + \alpha_1^n \; z_1(s) + \alpha_2^n \; z_2(s) + \cdots + \alpha_n^n \; z_n(s)
\end{aligned}$$

The stage coefficients are determined by matching the monomial coefficients of $p_k(s)$ with the coefficients of $\bar{\mathrm{T}}_k^\gamma \left(s \left(\frac{\mathrm{M}_k}{\mathrm{M}n} \right) \right)$ where $\mathrm{M}_k = \gamma \, k^2$ determines the bound for the stability region associated with $\mathrm{T}_k^\gamma(s)$. This choice of coefficients ensures that for each stage we have $|p_k(s)| < 1$ when $s \in [-\gamma n^2, 0]$ and $p_n(s) = \bar{\mathrm{T}}_n^\gamma(s)$. In the determination of the stage coefficients, the polynomials $\bar{\mathrm{T}}_k^\gamma(s)$ have constant term 1 and so one need only match the coefficients of monomials of degree greater than 1.

To construct the stage coefficient equations it is useful to identify the relation between the coefficients for the ith stage and those at the i+1st stage. Let $\mathrm{A}^i \vec{\alpha}^i$ be the linear mapping from the ith stage coefficients, $\vec{\alpha}^{\,i}$, to the coefficients of the monomials $s, s^{\,2}, s^{\,3}, \ldots s^{\,i}$ occurring in

$$p_i(s) = 1 + \alpha_1^i \; z_1(s) + \alpha_2^i \; z_2(s) + \cdots + \alpha_i^i \; z_i(s)$$

Since

$$p_{i+1}(s) = 1 + \alpha_1^{i+1} \; z_1(s) + \alpha_2^{i+1} \; z_2(s) + \cdots + \alpha_i^{i+1} \; z_i(s) + \alpha_{i+1}^{i+1} \; z_{i+1}(s)$$

the mapping from the values ${\alpha_1}^{i+1}$,${\alpha_2}^{i+1}$, $\ldots {\alpha_i}^{i+1}$ to the monomials of $s, s^{\,2}, s^{\,3}, \ldots s^{\,i}$ occurring in $p_{i+1}(s)$ is identical to that for $\vec{\alpha}^{\,i}$. Also, since $z_{i+1}(s) = s \; p_i(s)$, the mapping from α_{i+1}^{i+1} to the monomials $s, s^2, s^3, \ldots s^i, s^{i+1}$ occurring in $p_{i+1}(s)$ is just α_{i+1}^{i+1} times the monomial coefficients $\mathrm{A}^i \vec{\alpha}^i$ shifted down in index. Thus, we have that A^{i+1} given by

$$\mathrm{A}^{i+1} \; \vec{\alpha}^{\,i+1} = \left(\begin{array}{c|c} \mathrm{A}^i & 1 \\ & A^i \vec{\alpha}^{\,i} \\ \hline 0 & \end{array} \right) \left(\begin{array}{c} \\ \vec{\alpha}^{\,i+1} \\ \\ \end{array} \right)$$

represents the mapping from the values ${\alpha_1}^{i+1}$,${\alpha_2}^{i+1}$, $\ldots {\alpha_{i+1}}^{i+1}$to the monomials of $s, {s}^{\,2}, {s}^{\,3}, \ldots {s}^{\,i+1}$ occurring in $p_{i+1}(s)$.

Let $\tilde{c}_1^{\,k}$, $\tilde{c}_2^{\,k}$ $\ldots \tilde{c}_k^{\,k}$ be the coefficients of $s, {s}^{\,2}, {s}^{\,3}, \ldots {s}^{\,k}$ occurring in $\bar{\mathrm{T}}_k^{\gamma}(s)$. The equations that determine the stage coefficients can be represented as

$$\begin{aligned} \mathrm{A}^1\ \vec{\alpha}^1 &= \vec{c}^{\,1} \\ \mathrm{A}^2\ \vec{\alpha}^2 &= \vec{c}^{\,2} \\ \mathrm{A}^3\ \vec{\alpha}^2 &= \vec{c}^{\,3} \\ &\vdots \\ \mathrm{A}^n\ \vec{\alpha}^n &= \vec{c}^{\,n} \end{aligned}$$

where

$$\vec{c}^{\,k} = \begin{pmatrix} \tilde{c}_1^{\,k}\left(\frac{\mathrm{M}_k}{\mathrm{M}n}\right) \\ \tilde{c}_2^{\,k}\left(\frac{\mathrm{M}_k}{\mathrm{M}n}\right)^2 \\ \tilde{c}_3^{\,k}\left(\frac{\mathrm{M}_k}{\mathrm{M}n}\right)^3 \\ \vdots \\ \tilde{c}_k^{\,k}\left(\frac{\mathrm{M}_k}{\mathrm{M}n}\right)^k \end{pmatrix}$$

However, using the fact that $\mathrm{A}^1 = 1$ and, for each i, the last column of A^{i+1} is just

$$\begin{pmatrix} 1 \\ A^{\,i}\vec{\alpha}^{\,i} \end{pmatrix} = \begin{pmatrix} 1 \\ \vec{c}^{\,i} \end{pmatrix}$$

The system of equations for the kth stage can be expressed as

$$\begin{pmatrix} 1 & 1 & 1 & & 1 \\ & \tilde{c}_1^{\,1}\left(\frac{\mathrm{M}_1}{\mathrm{M}n}\right) & \tilde{c}_1^{\,2}\left(\frac{\mathrm{M}_2}{\mathrm{M}n}\right) & \cdots & \tilde{c}_1^{\,k-1}\left(\frac{\mathrm{M}_{k-1}}{\mathrm{M}n}\right) \\ & & \tilde{c}_2^{\,2}\left(\frac{\mathrm{M}_2}{\mathrm{M}n}\right)^2 & \cdots & \tilde{c}_2^{\,k-1}\left(\frac{\mathrm{M}_{k-1}}{\mathrm{M}n}\right)^2 \\ & & & \ddots & \\ & & & & \tilde{c}_{k-1}^{\,k-1}\left(\frac{\mathrm{M}_{k-1}}{\mathrm{M}n}\right)^{k-1} \end{pmatrix} \begin{pmatrix} {\alpha_1}^k \\ {\alpha_2}^k \\ {\alpha_3}^k \\ \vdots \\ {\alpha_k}^k \end{pmatrix} = \begin{pmatrix} \tilde{c}_1^{\,k}\left(\frac{\mathrm{M}_k}{\mathrm{M}n}\right) \\ \tilde{c}_2^{\,k}\left(\frac{\mathrm{M}_k}{\mathrm{M}n}\right)^2 \\ \tilde{c}_3^{\,k}\left(\frac{\mathrm{M}_k}{\mathrm{M}n}\right)^{k-1} \\ \vdots \\ \tilde{c}_k^{\,k}\left(\frac{\mathrm{M}_k}{\mathrm{M}n}\right)^k \end{pmatrix}$$

Or, using the fact that $\mathrm{M}_k = \gamma\, k^2$ and simplifying, we have that the coefficients for the kth stage are given by

$$\begin{pmatrix} 1 & 1 & 1 & & 1 \\ & \tilde{c}_1{}^1\left(\frac{1}{k^2}\right) & \tilde{c}_1{}^2\left(\frac{2^2}{k^2}\right) & \cdots & \tilde{c}_1{}^{k-1}\left(\frac{(k-1)^2}{k^2}\right) \\ & & \tilde{c}_2{}^2\left(\frac{2^2}{k^2}\right)^2 & \cdots & \tilde{c}_2{}^{k-1}\left(\frac{(k-1)^2}{k^2}\right)^2 \\ & & & \ddots & \\ & & & & \tilde{c}_{k-1}{}^{k-1}\left(\frac{(k-1)^2}{k^2}\right)^{k-1} \end{pmatrix} \begin{pmatrix} \alpha_1{}^k \\ \alpha_2{}^k \\ \alpha_3{}^k \\ \vdots \\ \alpha_k{}^k \end{pmatrix} = \left(\frac{k^2}{n^2}\right) \begin{pmatrix} \tilde{c}_1{}^k \\ \tilde{c}_2{}^k \\ \tilde{c}_3{}^k \\ \vdots \\ \tilde{c}_k{}^k \end{pmatrix}$$

where $\tilde{c}_i{}^k$is the coefficient of $s^{\,i}$ occurring in $\bar{\mathrm{T}}_k^{\gamma}(s)$. For any k, $\tilde{c}_k^{\,k}$ is the coefficient of the degree k monomial in $\bar{\mathrm{T}}_k^{\gamma}(s)$, and hence non-zero, thus the equations for the kth stage coefficients are non-singular and the coefficients can be obtained quickly using back-substitution.

2. Timestep Selection. The use of stabilized RK methods requires the specification of a timestep size, stability region bound factor, and stage order. In the experiments described below we used a fixed stage order, a fixed stability region bound factor, and a rather simple adaptive timestep selection strategy. For the problem that motivated this work, the solution of a set of Schroedinger-Poisson equations, this simple strategy worked reasonably well. However, to create a code that can automatically determine stage order, stability region bound and timestep size requires the creation of procedures similar to those developed for the stabilized RK methods used to compute time-accurate solutions [4][9]. Work on such procedures is in progress.

The central idea behind the adaptive timestep selection strategy is to choose as large a timestep as possible that leads to a decrease in the size of the residual. The process begins with the determining a first timestep so that there is an initial decrease in the size of the residual. This stepsize is readily determined using a bisection process. After the first timestep is determined, the solution is advanced and the size of the residual monitored. If the residual of the solution computed with a given timestep is smaller than the previous residual, then the solution and timestep are accepted. If the residual increases with any given stepsize, then that step is rejected and the timestep is reduced by a fixed factor β, ($dt \leftarrow \beta\, dt$). This reduction in timestep is repeated until it leads to a solution with a decreased residual or some set number of reductions have been preformed. Periodically (every few steps) the timestep is increased by a fixed factor α, ($dt \leftarrow \alpha\, dt$). If this new timestep leads to a reduced residual then it is accepted and the solution advanced; otherwise the solution is rejected and the timestep reduced.

3. Computational Examples. Our first two test problems consist of finding solutions to the discretization of

$$\epsilon\, \frac{d^2u}{dx^2} - (u - \frac{1}{2})^3 = 0 \qquad x \in [0,1] \tag{3.1}$$

$$u(0) = 1 \qquad u(1) = 0$$

These equations arise as the steady state solution of the reaction-diffusion equation.

$$\frac{du}{dt} = \epsilon \frac{d^2u}{dx^2} - (u - \frac{1}{2})^3 \tag{3.2}$$

The standard second order finite difference approximation $\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$ was employed to approximate $\frac{d^2u}{dx^2}$ in the discretization of (3.1). The mesh used was uniform with 50 panels ($h = \frac{1}{50}$) and, ϵ, the diffusion coefficient was taken to be .001. After discretization the system of equations has the form

$$\epsilon \text{ M } \vec{u} - (\vec{u} - \frac{1}{2})^3 + \vec{f} = 0 \tag{3.3}$$

where M is a tridiagonal matrix arising from the discretization of $\frac{d^2u}{dx^2}$, $(\vec{u} - \frac{1}{2})^3$ formally represents the vector whose ith component is $(u_i - \frac{1}{2})^3$, and $\vec{f}$ contains the specification of the boundary conditions $u(0) = 1$, $u(1) = 0$.

The first experiment consisted of using methods with $\gamma = 1.75$ and stage orders from $n = 1$ to $n = 10$ to solve

$$\frac{d\vec{u}}{dt} = \epsilon \text{ M } \vec{u} - (\vec{u} - \frac{1}{2})^3 + \vec{f} \tag{3.4}$$

to steady state. The computation was stopped when the maximum value of all components of the residual were less than 10^{-6}. As a measure of the efficiency of the methods, we monitored the total number of function evaluations required. (This total includes function evaluations used in the adaptive determination of the timestep.) The results are listed in Table 2.

stage order	# steps	# function evaluations
$n = 1$	1709	2943
$n = 2$	413	1224
$n = 4$	129	728
$n = 6$	61	498
$n = 8$	39	416
$n = 10$	30	360

Table 2 : The total number of function evaluations required to obtain a steady state solution of (3.4) using a stabilized RK method with $\gamma = 1.75$.

The ODE system is mildly stiff, and the benefit of using a stabilized Runge-Kutta method on this problem clearly seen. It is also interesting to note that with orders greater than 6, the reduction in computational cost obtainable with an increase in stage order is not that significant.

The second experiment consisted of using the inverse of ϵM as a preconditioner. Specifically, instead of directly seeking the solution (3.3) we seek the solution of

$$(\epsilon\ \mathrm{M})^{-1}\left[(\vec{u}-\frac{1}{2})^3+\vec{f}\right]-\vec{u}=0 \tag{3.5}$$

The solution of these equations was obtained by solving

$$\frac{d\vec{u}}{dt}=(\epsilon\ \mathrm{M})^{-1}\left[(\vec{u}-\frac{1}{2})^3+\vec{f}\right]-\vec{u} \tag{3.6}$$

to steady state. A stability bound factor of $\gamma = 1.75$ was used, and the iteration was stopped when the maximum value of all residual components was less than 10^{-6}. The results are listed in Table 3.

stage order	# steps	# function evaluations
$n = 1$	39	62
$n = 2$	20	58
$n = 4$	28	148
$n = 6$	21	144
$n = 8$	20	176
$n = 10$	18	200

Table 3 : The total number of function evaluations required to obtain a steady state solution of (3.6) using a stabilized RK method with $\gamma = 1.75$.

The results of this latter example demonstrate that, for some problems, the use of stabilized Runge-Kutta methods is not required (and in fact, using a method with more than two stages should be avoided). The reason for this behavior is that the preconditioned system is not particularly stiff and a standard one or two stage Runge-Kutta method is an efficient method for obtaining steady state solutions.

The third computational problem, and that which motivated this work, consisted of computing solutions to a set of Schroedinger-Poisson equations used in quantum device modeling. For a one-dimensional simulation, these equations have the form

$$\frac{d}{dz}(\varepsilon(z)\frac{d\phi}{dz})=\sum_s \delta_s+\sum_{\lambda_k<0} G(\Psi_k,\lambda_k) \tag{3.7}$$

$$\frac{d}{dz}(\frac{\hbar^2}{2m}\frac{d\Psi_k}{dz})+(\phi+U)\Psi_k=\lambda_k\Psi_k \tag{3.8}$$

Here ϕ is the potential, Ψ_k the single election wavefunctions and their associated energies λ_k, $\varepsilon(z)$ the dielectric constant, $\hbar$ Plank's constant, m the reduced mass,

U the band offset, G the nonlinear density of states functional accounting for the change in potential induced by occupied states and δ_s represents the sources due to doping. The self-consistent solution of these equations consists of a potential ϕ and a collection of wavefunctions Ψ_k and associated energies λ_k so that (3.7) and (3.8) are simultaneously satisfied.

To find such a self-consistent solution we first formally eliminate Ψ_k and associated energies λ_k and just consider (3.7) as a non-linear equation for ϕ

$$\frac{d}{dz}(\varepsilon(z)\frac{d\phi}{dz}) = \delta_s + \sum_{\lambda_k(\phi) < 0} G(\Psi_k(\phi), \lambda_k(\phi)) \tag{3.9}$$

Or, letting $L(\phi)$ denote the linear Poisson operator and $S(\phi)$ the nonlinear contribution due to the occupied states, the equation takes the form

$$L(\phi) = \delta_s + S(\phi) \tag{3.10}$$

Since the evaluation of $S(\phi)$ requires a computationally expensive eigenvalue/eigenvector calculation, it is critical that any solution process use as few evaluations of $S(\phi)$ as possible. Also, because of the complexity of $S(\phi)$ and the fact that $G(\Psi_k(\phi), \lambda_k(\phi))$ is a non-smooth function of its arguments, methods that rely on derivatives (e.g. Newton like methods) are difficult to implement and tend to have limited robustness.

Therefore, to solve (3.10) we convert it into an equivalent fixed point problem by applying L^{-1} to both sides; this yields

$$\phi = L^{-1}\left[\delta_s + S(\phi)\right] \tag{3.11}$$

One can use standard fixed point iteration with a suitably small relaxation factor to (3.11) to successfully compute solutions. However, the size of the relaxation factor can be very small and convergence slow. For example, in a typical semi-conductor design problem the relaxation factor required was .011 and 821 iterations were needed to reduce the size of the residual to a maximal value of less than 10^{-4}.

When the stabilized RK methods described above were used to evolve

$$\frac{d\phi}{dt} = L^{-1}\left[\delta_s + S(\phi)\right] - \phi \tag{3.12}$$

to steady state the results were considerably better. The necessity of using a small relaxation factor indicated that the system (3.12) was mildly stiff, so that using a method more than two stages was effective. In Table 4 we show the results obtained using various stages.

degree	# steps	# function evaluations
$n = 1$	263	402
$n = 2$	112	348
$n = 4$	46	284
$n = 6$	21	180
$n = 8$	17	184
$n = 10$	20	280

Table 4 : The total number of function evaluations required to obtain a steady state solution of (3.12) using a stabilized RK method with $\gamma = 1.5$.

In all of these experiments the same initial guess for the timestep was used, and the factor for the stability bound, γ, was chosen as 1.5.

As with the first computational experiment, the computational results indicate the utility of using a stabilized Runge-Kutta method with a moderate number of stages. When compared to the original fixed point iteration with an optimally chosen relaxation factor, there was a factor of four improvement; with a more advanced timestep/stage order selection procedure, this can be expected to be improved even further.

4. Conclusions. For many nonlinear equations, especially those arising from the discretization of partial differential equations, it is possible to associate with them sets of ordinary differential equations whose solutions evolve to a steady state solution that is also a solution of the original system. When the associated ordinary differential equations are moderately stiff, the number of steps to converge to the steady state can be quite large. As shown in the examples given above, one can make good use of stabilized Runge-Kutta methods to reduce the number of required steps. We have also shown that stabilized Runge-Kutta methods can also be used to accelerate slowly convergent fixed point iterations. The stabilized methods we derived here were developed explicitly for the solution of steady state problems — they were chosen to have large stability regions, good damping factors, and coefficients that are relatively easy to construct. Additionally, the methods were constructed so that the accuracy of the solution of the nonlinear equations was not dependent upon the size of the timestep. We have by no means exhausted the types of stabilized methods that might be used for computing steady state solutions; in particular, other methods could conceivably be constructed that do not require the accumulation of all previous stages.

REFERENCES

[1] Stoer,J. and Bulirsch, R., Introduction to Numerical Analysis, Springer-Verlag, New York, pp. 490-491, (1993) .

[2] Atkinson, K., An Introduction to Numerical Analysis, John Wiley and Sons, New York, pg. 229.

[3] Abdulle, A.,Medovikiv, A, *Second order Chebyshev methods based on orthogonal polynomials*, Numer. Math. 90: 1-18 (2001).

[4] Abdulle, A., *Fourth order Chebyshev methods with Recurrence relation*, SIAM J. Sci. Comput. 23, pp. 2042-2055, (2002).

[5] Verwer, J.W.,*Explicit Runge-Kutta methods for parabolic partial differential equations*, Runge-Kutta centennial, special issue of Appl. Num. Math., 22 (1996), pp. 359-379.

[6] Lebedev, V.I., *How to solve stiff systems of differential equaions by explicit methods*, Numerical Methods and Applications, CRC Press, Boca Raton, (1994), pp. 45-80.

[7] Rivlin, J., *An Introduction to the Approximation of Functions*, Dover, New York, pp. 31-32, (1981).
[8] Medovicof, A.A., *High Order Explicit Methods for Parabolic Equations*, BIT, 38, pp. 372-390, (1998).
[9] Sommeijer, B.P., Shampine, L.F., Verwer, J.G., *RKC: An explicit solver for parabolic PDE's*, J. Comput. and Appl. Math., 88, pp 315-326, (1997).