Journal of Computational Physics **170,** 415–447 (2001)
doi:10.1006/jcph.2001.6744, available online at http://www.idealibrary.com on IDEAL®

Design and Implementation of Runge–Kutta Methods for MAS NMR Lineshape Calculations

J. H. Kristensen,*[,1] G. L. Hoatson,* and R. L. Vold†

**Department of Physics and †Department of Applied Science, College of William and Mary, P.O. Box 8795, Williamsburg, Virginia 23187-8795*
E-mail: jorgen00@esc.cam.ac.uk; gina@nmr.physics.wm.edu; rlv@nmr.physics.wm.edu

Received December 10, 1999; revised July 31, 2000

The magic angle spinning (MAS) nuclear magnetic resonance (NMR) technique has proven successful for obtaining information about molecular structure and motion in solids. The most advanced description of the MAS NMR experiment involves the density operator and the stochastic Liouville–von Neumann equation. In this equation, the effects of the anisotropic nuclear spin interactions are represented by the Hamiltonian, while the motion involves a stochastic operator. For many systems, molecular motion may be described by a discrete Markov process. In order to obtain a solution for a discrete Markov process we have used a Lie algebra formalism to rewrite the stochastic Liouville–von Neumann equation in the form of a linear homogeneous system of coupled first-order differential equations. This system has periodically time-dependent coefficients and can only be solved by numerical methods. In this paper, we discuss the use of different Runge–Kutta methods to approximate the solution. These methods have the advantages of simplicity and relatively high efficiency and may be implemented with automatic stepsize control. The study involves the most important classes of Runge–Kutta methods including explicit, semi-implicit, and implicit schemes. The results have shown that the choice of the Runge–Kutta method depends on the motion. It is found that explicit Runge–Kutta methods are the most efficient schemes in the slow and intermediate motion regimes. However, in the fast motion regime, the stochastic Liouville–von Neumann equation becomes stiff. In this regime, we have used semi-implicit and implicit Runge–Kutta methods with better stability characteristics. For many semi-implicit or implicit Runge–Kutta methods, the stiff components are represented inaccurately, and the methods are shown to be relatively inefficient. In order to improve the efficiency we have designed and implemented stiffly A-stable Runge–Kutta methods. These have better stability and accuracy characteristics and are useful in the fast motion regime. In another approach, we have rewritten the stochastic Liouville–von

[1] Present address: Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ, United Kingdom.

0021-9991/01 $35.00

Neumann equation in a form with a vanishing stiffness ratio. Based on this form we have designed modified explicit Runge–Kutta methods, which are stable and accurate in the fast motion regime. The results have shown that the Runge–Kutta methods improve the computational efficiency for small systems by a factor between two and five compared with the eigenvalue method. The efficiency increases with the dimension of the system, and for large systems the improvement approaches an order of magnitude. This is an important result because of the long computation times involved in MAS NMR lineshape calculations.

Key Words: nuclear magnetic resonance (NMR); magic angle spinning (MAS); density operator theory; stochastic Liouville–von Neumann equation; Runge–Kutta methods; stiff differential systems.

1. INTRODUCTION

The technique of solid-state nuclear magnetic resonance (NMR) spectroscopy [1–3] has become one of the most important methods for investigating molecular structure and motion in solid materials. At present, the most successful approach is magic angle spinning (MAS) NMR spectroscopy [4–9]. In this method the spectra are reduced to a manifold of spinning sidebands which improves the resolution and sensitivity. In the MAS NMR experiment, the intensity distribution in the manifold of spinning sidebands is defined by the coupling parameters of the anisotropic nuclear spin interactions. These parameters often correlate with structural parameters, such as bond angles and bond lengths, and are important sources of information about molecular structure. In addition to its usefulness for structural analysis, the MAS NMR experiment has been introduced as a valuable method for investigating molecular motion in solids [10–14]. In the presence of molecular motion, the spectra exhibit lineshape modulations that may be analyzed to determine the motional mechanism.

The most advanced model for describing the MAS NMR experiment is based on the stochastic Liouville–von Neumann equation [10–16]. This describes the time evolution of the density operator in the presence of molecular motion. The density operator accounts for all the alignments and coherences in a nuclear spin ensemble and is necessary in order to calculate effects of multiple-quantum transitions and finite pulse widths in MAS NMR experiments. The stochastic Liouville–von Neumann equation includes the nuclear spin Hamiltonian and a stochastic operator. The Hamiltonian is defined by anisotropic nuclear spin interactions, while the form of the stochastic operator depends on the details of the motion. The simplest description of molecular motion is based on a discrete Markov process [17]. This involves large amplitude transitions between a finite set of motional states. In this case, the stochastic Liouville–von Neumann equation can be expressed as a linear homogeneous system of coupled first-order differential equations among the alignments and coherences. Because of the time dependence induced by MAS, it is impossible to solve this system explicitly. The design and implementation of numerical methods is therefore an important part of the theoretical investigations.

For polycrystalline samples, the calculation of MAS NMR spectra involves two stages that are equally important in determining the overall efficiency of the computation. In the first stage, the stochastic Liouville–von Neumann equation is integrated in time to produce the time-domain signal for a single crystallite. The second stage is a powder integration whereby the time-domain signal is integrated over all possible crystallite orientations. The

powder integration usually involves a sum of time-domain signals calculated for a suitable distribution of crystallite orientations [18, 19]. The choice of time integration method is an important problem that depends on the exact form of the equations. In this paper we show that the stochastic Liouville–von Neumann equation is stiff, implying that the numerical solution of the problem is very difficult.

Note that our initial studies showed that many of the usual methods [20–22] for solving stiff systems are relatively inefficient when applied to the stochastic Liouville–von Neumann equation. This includes Gear's method [20], which has previously been applied successfully for many stiff systems. The inefficiency of Gear's method is probably the result of the computational overhead involved in the predictor-corrector algorithms. Clearly, in the motional regimes where the stiffness vanishes, it is inefficient to implement a stiff method. As a result it is unlikely to find a method that is optimum for all motional regimes. It is more useful to consider several different integration methods and determine the most efficient scheme in each motional regime.

In this paper, we have elaborated on the use of different Runge–Kutta methods [23–26] which are interesting because of their simplicity and relatively high efficiency. In the first part of this paper we demonstrate how the stochastic Liouville–von Neumann equation may be reduced to a linear homogeneous system of coupled first-order differential equations. Following this introduction we consider the principles of Runge–Kutta methods. This is intended to establish notation and to summarize the concepts of local and global truncation error that are essential in order to understand the accuracy and stability characteristics of the methods.

We have investigated the performance of a set of representative Runge–Kutta methods including explicit, semi-implicit, and implicit methods. These have been used to calculate motional effects on deuteron MAS NMR spectra [8, 9, 12, 14]. We have elaborated on deuteron experiments because these are used extensively for investigations of molecular motion. The results have shown that explicit Runge–Kutta methods are most efficient for slow and intermediate molecular motion. However, for fast motion, the equations become stiff, making explicit methods inefficient. We show that there are two different strategies that may be used for these stiff systems. The most obvious approach is to implement semi-implicit or implicit Runge–Kutta methods that are sufficiently stable and accurate. Another strategy is to rewrite the stochastic Liouville–von Neumann equation in a form with a vanishing stiffness. In this paper, we introduce a transformation that eliminates the stiffness completely. Because the transformed system may be integrated with explicit methods, the computational efficiency is improved significantly.

2. PROBLEM FORMULATION

2.1. *The Stochastic Liouville–von Neumann Equation*

In modern Fourier transform NMR spectroscopy, the experimental method is based on measuring the response of an ensemble of nuclear spin systems to an external magnetic perturbation [1–3]. This is usually a sequence of radio frequency pulses designed to create observable coherences in the ensemble, which evolve in time to generate the observable time-domain signal. The action of anisotropic nuclear spin interactions combined with the effect of molecular motion determines the characteristics of the response. From the form of the time-domain signal, it is therefore possible to deduce many details about molecular structure and motion.

The description is usually concerned with a statistical ensemble because the experiment is performed on a sample with a large number of randomly distributed spin systems. The physical state of the ensemble is described by the density operator $\sigma(t)$, which contains all the information about the observable properties of the system [27]. In the presence of molecular motion, the density operator evolves according to the stochastic Liouville–von Neumann equation [10–16]

$$\frac{\partial}{\partial t}|\sigma(t)\rangle = A(t)|\sigma(t)\rangle, \tag{2.1}$$

which includes the coefficient operator

$$A(t) = -iAd(H(t)) + \Xi, \tag{2.2}$$

where $Ad(H(t))$ is the adjoint Hamiltonian and Ξ the stochastic operator.

In order to solve the stochastic Liouville–von Neumann equation, it is useful to implement matrix representations spanned by the group generators $\{I_m(\xi_n) \mid m = 1, \ldots, M, n = 1, \ldots, N\}$ where M is the number of alignments and coherences and N the number of motional states [28–30]. Each group generator is given by the value ξ_n of a stochastic variable that defines the motional state. The group generators define the basis vectors $\{|I_1(\xi_n)\rangle, \ldots, |I_M(\xi_n)\rangle\}$ which satisfy the orthogonality condition

$$\frac{\langle I_k(\xi_m)|I_l(\xi_n)\rangle}{\langle I_k(\xi_m)|I_k(\xi_m)\rangle} = \delta_{kl}\delta_{mn} \tag{2.3}$$

and completeness relation

$$\sum_{m=1}^{M}\sum_{n=1}^{N}\frac{|I_m(\xi_n)\rangle\langle I_m(\xi_n)|}{\langle I_m(\xi_n)|I_m(\xi_n)\rangle} = E. \tag{2.4}$$

The closure property of the group generators is expressed by the commutation relation

$$Ad(I_k(\xi_n))|I_l(\xi_n)\rangle = |[I_k(\xi_n), I_l(\xi_n)]\rangle = \sum_{m=1}^{M} c_{kl}^m |I_m(\xi_n)\rangle, \tag{2.5}$$

where c_{kl}^m are the structure constants [28–30]. The commutation relation may be rewritten in the form

$$\frac{\langle I_m(\xi_n)|Ad(I_k(\xi_n))|I_l(\xi_n)\rangle}{\langle I_m(\xi_n)|I_m(\xi_n)\rangle} = c_{kl}^m, \tag{2.6}$$

which defines the matrix representation of all adjoint operators. The completeness implies that

$$\sigma(t) = \sum_{n=1}^{N}\sigma(\xi_n, t) = \sum_{m=1}^{M}\sum_{n=1}^{N}\sigma_m(\xi_n, t)I_m(\xi_n), \tag{2.7}$$

$$H(t) = \sum_{n=1}^{N}H(\xi_n, t) = \sum_{m=1}^{M}\sum_{n=1}^{N}H_m(\xi_n, t)I_m(\xi_n), \tag{2.8}$$

where the expansion coefficients

$$\sigma_m(\xi_n, t) = \frac{\langle I_m(\xi_n)|\sigma(t)\rangle}{\langle I_m(\xi_n)|I_m(\xi_n)\rangle}, \tag{2.9}$$

$$H_m(\xi_n, t) = \frac{\langle I_m(\xi_n)|H(t)\rangle}{\langle I_m(\xi_n)|I_m(\xi_n)\rangle}. \tag{2.10}$$

The stochastic Liouville–von Neumann equation may be developed by inserting the expansion of the density operator (Eq. (2.7)) and using the orthogonality (Eq. (2.3)) of the group generators. This leads to the linear homogeneous system of coupled first-order differential equations

$$\frac{\partial}{\partial t}|\boldsymbol{\sigma}(t)\rangle = \mathbf{A}(t)|\boldsymbol{\sigma}(t)\rangle, \tag{2.11}$$

where $\boldsymbol{\sigma}(t) = \{\sigma_k(\xi_r, t)\}$ is an MN-dimensional vector and $\mathbf{A}(t) = \{a_{kl}(\xi_r, \xi_s, t\}$ a time-dependent (MN, MN)-dimensional coefficient matrix. The elements of the coefficient matrix are

$$a_{kl}(\xi_r, \xi_s, t) = \frac{\langle I_k(\xi_r)|A(t)|I_l(\xi_s)\rangle}{\langle I_k(\xi_r)|I_k(\xi_r)\rangle} = \frac{\langle I_k(\xi_r)|\Xi|I_l(\xi_s)\rangle}{\langle I_k(\xi_r)|I_k(\xi_r)\rangle} - i\frac{\langle I_k(\xi_r)|Ad(H(t))|I_l(\xi_s)\rangle}{\langle I_k(\xi_r)|I_k(\xi_r)\rangle}, \tag{2.12}$$

where

$$\frac{\langle I_k(\xi_r)|Ad(H(t))|I_l(\xi_s)\rangle}{\langle I_k(\xi_r)|I_k(\xi_r)\rangle} = \delta_{rs}\sum_{m=1}^{M} H_m(\xi_r, t)c_{ml}^k, \tag{2.13}$$

$$\frac{\langle I_k(\xi_r)|\Xi|I_l(\xi_s)\rangle}{\langle I_k(\xi_r)|I_k(\xi_r)\rangle} = \delta_{kl}\,\Xi(\xi_r, \xi_s) \tag{2.14}$$

specify the elements of the adjoint Hamiltonian and the stochastic operator. These equations lead to

$$a_{kl}(\xi_r, \xi_s, t) = \delta_{kl}\,\Xi(\xi_r, \xi_s) - i\delta_{rs}\sum_{m=1}^{M} H_m(\xi_r, t)c_{ml}^k, \tag{2.15}$$

which gives an explicit expression for the elements of the coefficient matrix.

The stochastic operator is defined by the details of the motion. For many systems molecular motion may be represented by a discrete Markov process [17]. In this model, the motion is described by transitions between a finite set of motional states, and the elements of the stochastic operator are

$$\begin{aligned}\Xi(\xi_m, \xi_n) &= k_{nm},\\ \Xi(\xi_m, \xi_m) &= -\sum_{n=1}^{N}[1 - \delta_{mn}]k_{mn},\end{aligned} \tag{2.16}$$

where k_{mn} is the rate constant for the transition from the ξ_m to ξ_n motional state. This model is sufficiently accurate for systems with high activation energies and low temperatures.

The stochastic Liouville–von Neumann equation is usually solved subject to an initial condition. In this case, the formal solution may be expressed by

$$|\boldsymbol{\sigma}(t)\rangle = \mathbf{F}(t_0, t)|\boldsymbol{\sigma}(t_0)\rangle, \tag{2.17}$$

where the propagator

$$\mathbf{F}(t_0, t) = T \exp\left(\int_{t_0}^{t} \mathbf{A}(\tau)\, d\tau\right) \tag{2.18}$$

is a time-ordered exponential. The time ordering implies that there is no explicit solution, and it is necessary to implement numerical integration methods. There are several different techniques that may be used to solve the stochastic Liouville–von Neumann equation. In one approach, the system is solved directly to give the alignments and coherences. This has the advantage that the dimension of the system is at a minimum. Another method is based on the equation

$$\frac{\partial}{\partial t}\mathbf{F}(t_0, t) = \mathbf{A}(t)\mathbf{F}(t_0, t), \tag{2.19}$$

which may be solved for the propagator. This is then used to determine the time evolution of the alignments and coherences. In this method the dimension of the system is larger. However, because of the periodicity of the MAS NMR Hamiltonian the propagator obeys

$$\mathbf{F}(t_0, t) = \mathbf{F}(t_0 + nT_r, t + nT_r), \tag{2.20}$$

where $T_r = \frac{2\pi}{\omega_r}$ is the rotation period [4–6]. This demonstrates that it is only necessary to evaluate the propagator for one rotation period. The result is an improvement in computational efficiency that often more than compensates for the increased dimension. For a polycrystalline sample, the solution must be integrated over all crystallite orientations. In the case of MAS, the dependence of the propagator on the crystallite orientation may be transcribed into a time shift

$$\mathbf{F}(t_0, t, \alpha, \beta, \gamma) = \mathbf{F}\left(t_0 + \frac{\gamma}{\omega_r}, t + \frac{\gamma}{\omega_r}, \alpha, \beta, 0\right), \tag{2.21}$$

which leads to an additional improvement in computational efficiency [31].

2.2. Example Application

In this paper we apply the above formalism to deuteron MAS NMR spectroscopy [8, 9, 12, 14]. This provides an example of widespread interest and importance that is sufficiently general to reveal most of the properties of the stochastic Liouville–von Neumann equation. Because deuterons are SU(3) nuclear spin systems, the alignments and coherences may be represented by irreducible cartesian SU(2) tensor operators for SU(3) [30]. It is noted that there are other group generators [32–34] but the irreducible cartesian SU(2) tensor operators for SU(3) are optimum for the description of nonselective and symmetric experiments. In

the absence of rf irradiation, the deuteron MAS NMR Hamiltonian is

$$H(t) = \sum_{n=1}^{N} H_Q^{(1)}(\xi_n, t), \tag{2.22}$$

which includes the first-order quadrupole interaction [6, 7]. This is given in terms of irreducible cartesian SU(2) tensor operators for SU(3) by

$$H_Q^{(1)}(\xi_n, t) = \sqrt{\frac{2}{3}}\omega_Q^{(1)}(\xi_n, t) I_{z^2}(\xi_n), \tag{2.23}$$

where the angular first-order quadrupole frequency

$$\omega_Q^{(1)}(\xi_n, t) = \sqrt{\frac{3}{2}} A_{z^2}^Q(\xi_n, t). \tag{2.24}$$

This is specified in terms of the second-rank irreducible cartesian quadrupole tensor

$$\mathbf{A}_Q^{(2)}(\xi_n, t) = \boldsymbol{\rho}_Q^{(2)}(\xi_n)\boldsymbol{\Gamma}^{(2)}(\boldsymbol{\Omega}_1(\xi_n))\Gamma^{(2)}(\boldsymbol{\Omega}_2)\boldsymbol{\Gamma}^{(2)}(\boldsymbol{\Omega}_3(t)), \tag{2.25}$$

where

$$\boldsymbol{\rho}_Q^{(2)}(\xi_n) = \frac{\pi C_Q(\xi_n)}{\sqrt{2}I[2I-1]}\{-\eta_Q(\xi_n), 0, \sqrt{3}, 0, 0\} \tag{2.26}$$

is the principal second-rank irreducible cartesian quadrupole tensor. This model includes the quadrupole coupling constant $C_Q(\xi_n)$ and asymmetry parameter $\eta_Q(\xi_n)$. The second-rank irreducible cartesian representation matrix $\boldsymbol{\Gamma}^{(2)}(\boldsymbol{\Omega})$ defines the transformations. The motional states are specified by the Euler angles $\boldsymbol{\Omega}_1(\xi_n) = \{\alpha_1(\xi_n), \beta_1(\xi_n), \gamma_1(\xi_n)\}$. These represent the transformation from the principal axis system of the quadrupole tensor to the crystallite fixed axis system. The Euler angles $\boldsymbol{\Omega}_2 = \{\alpha_2, \beta_2, \gamma_2\}$ specify the relative orientation of the crystallite and rotor fixed axis systems. The transformation from the rotor to the laboratory fixed axis system is given by $\boldsymbol{\Omega}_3(t) = \{\omega_r t, \theta, 0\}$, where ω_r is the angular rotation frequency and θ is the angle between the rotation axis and the magnetic field.

By implementing the irreducible cartesian SU(2) tensor operators for SU(3), the density operator may be represented by an $8N$-dimensional state vector with elements

$$\begin{aligned}
&[\boldsymbol{\sigma}(t)]_{8(m-1)+1} = \boldsymbol{\sigma}_x(\xi_m, t), \quad [\boldsymbol{\sigma}(t)]_{8(m-1)+2} = \boldsymbol{\sigma}_z(\xi_m, t),\\
&[\boldsymbol{\sigma}(t)]_{8(m-1)+3} = \boldsymbol{\sigma}_y(\xi_m, t), \quad [\boldsymbol{\sigma}(t)]_{8(m-1)+4} = \boldsymbol{\sigma}_{x^2-y^2}(\xi_m, t),\\
&[\boldsymbol{\sigma}(t)]_{8(m-1)+5} = \boldsymbol{\sigma}_{xz}(\xi_m, t), \quad [\boldsymbol{\sigma}(t)]_{8(m-1)+6} = \boldsymbol{\sigma}_{z^2}(\xi_m, t),\\
&[\boldsymbol{\sigma}(t)]_{8(m-1)+7} = \boldsymbol{\sigma}_{yz}(\xi_m, t), \quad [\boldsymbol{\sigma}(t)]_{8(m-1)+8} = \boldsymbol{\sigma}_{xy}(\xi_m, t).
\end{aligned} \tag{2.27}$$

where $m = 1, \ldots, N$. The elements $\boldsymbol{\sigma}_z(\xi_n, t)$ and $\boldsymbol{\sigma}_{z^2}(\xi_n, t)$ are the dipole and quadrupole alignments, respectively. The dipole single-quantum coherences are $\boldsymbol{\sigma}_x(\xi_n, t)$ and $\boldsymbol{\sigma}_y(\xi_n, t)$, while $\boldsymbol{\sigma}_{xz}(\xi_n, t)$ and $\boldsymbol{\sigma}_{yz}(\xi_n, t)$ specify the quadrupole single-quantum coherences. The

quadrupole double-quantum coherences are $\boldsymbol{\sigma}_{x^2-y^2}(\xi_n, t)$ and $\boldsymbol{\sigma}_{xy}(\xi_n, t)$. In this case, the elements of the $(8N,8N)$-dimensional coefficient matrix are

$$\begin{aligned}
[\mathbf{A}(t)]_{8(m-1)+7,8(m-1)+1} &= [\mathbf{A}(t)]_{8(m-1)+3,8(m-1)+5} = \omega_Q^{(1)}(\xi_m, t), \\
[\mathbf{A}(t)]_{8(m-1)+1,8(m-1)+7} &= [\mathbf{A}(t)]_{8(m-1)+5,8(m-1)+3} = -\omega_Q^{(1)}(\xi_m, t), \\
[\mathbf{A}(t)]_{8(m-1)+k,8(n-1)+k} &= \Xi(\xi_m, \xi_n),
\end{aligned} \tag{2.28}$$

where $m, n = 1, \ldots, N$ and $k = 1, \ldots, 8$. It has been shown that the $(8N,8N)$-dimensional coefficient matrix may be reduced into eight irreducible (N,N)-dimensional coefficient matrices [14]

$$\begin{aligned}
[\mathbf{A}_1]_{m,n} &= [\mathbf{A}_2]_{m,n} = [\mathbf{A}_3]_{m,n} = [\mathbf{A}_4]_{m,n} = \Xi(\xi_m, \xi_n), \\
[\mathbf{A}_5]_{m,n} &= [\mathbf{A}_7]_{m,n} = \Xi(\xi_m, \xi_n) - i\omega_Q^{(1)}(\xi_m, t)\delta_{mn}, \\
[\mathbf{A}_6]_{m,n} &= [\mathbf{A}_8]_{m,n} = \Xi(\xi_m, \xi_n) + i\omega_Q^{(1)}(\xi_m, t)\delta_{mn},
\end{aligned} \tag{2.29}$$

where $m, n = 1, \ldots, N$. The systems defined by the time-independent coefficient matrices are solved most efficiently by the eigenvalue method [12–14]. For the time-dependent systems, there is no explicit solution, and it is necessary to implement numerical integration methods.

3. THE RUNGE–KUTTA FORMALISM

The results of the previous section have demonstrated that the stochastic Liouville–von Neumann equation is equivalent to a linear homogeneous system of coupled first-order differential equations. In this paper we consider the application of Runge–Kutta methods to solve this system [23–26]. The Runge–Kutta methods are designed to provide a solution to the system

$$\frac{\partial}{\partial t}\mathbf{y}(t) = \mathbf{f}(t, \mathbf{y}(t)), \tag{3.1}$$

where $\mathbf{y}(t) = \{y_k(t)\}$ and $\mathbf{f}(t, \mathbf{y}(t)) = \{f_k(t, \mathbf{y}(t))\}$ are subject to an initial condition. It is usually impossible to obtain an explicit solution. The object is therefore to find for a sequence of independent variables t_k, sequence of approximate solutions $\hat{\mathbf{y}}(t_k)$. In the Runge–Kutta formalism the solution is

$$\hat{\mathbf{y}}(t_{k+1}) = \hat{\mathbf{y}}(t_k) + \delta t_k \hat{\boldsymbol{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k), \tag{3.2}$$

where $t_{k+1} = t_k + \delta t_k$ is given by the stepsize δt_k. The approximate increment function takes the form

$$\hat{\boldsymbol{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) = \sum_{r=1}^{q} w_r \mathbf{g}_r, \tag{3.3}$$

where w_r $(r = 1, \ldots, q)$ are weighting coefficients and

$$\mathbf{g}_r = \mathbf{f}\left(t_k + \alpha_r \delta t_k, \hat{\mathbf{y}}(t_k) + \delta t_k \sum_{s=1}^{q} \beta_{rs}\mathbf{g}_s\right), \tag{3.4}$$

is given in terms of the parameters α_r $(r = 1, \ldots, q)$ and β_{rs} $(r, s = 1, \ldots, q)$. It is useful to classify the Runge–Kutta methods by the structure of the matrix $\boldsymbol{\beta} = \{\beta_{rs}\}$. For an explicit Runge–Kutta method, $\boldsymbol{\beta}$ is lower triangular and any $\mathbf{g}_r$ may be obtained in terms of $\mathbf{g}_1, \ldots, \mathbf{g}_{r-1}$. A semi-implicit Runge–Kutta method has elements on the diagonal of $\boldsymbol{\beta}$ and any $\mathbf{g}_r$ is defined by $\mathbf{g}_1, \ldots, \mathbf{g}_r$. Implicit Runge–Kutta methods have elements in the upper right triangle of $\boldsymbol{\beta}$, and any $\mathbf{g}_r$ is given in terms of $\mathbf{g}_1, \ldots, \mathbf{g}_q$.

4. STABILITY AND ACCURACY OF RUNGE–KUTTA METHODS

4.1. *Introductory Remarks*

In order to understand the performance of Runge–Kutta methods it is necessary to consider the stability and accuracy characteristics [23–26]. For simplicity we restrict the discussion to the system

$$\frac{\partial}{\partial t}|\mathbf{y}(t)\rangle = \mathbf{A}(t)|\mathbf{y}(t)\rangle, \tag{4.1}$$

where $\mathbf{y}(t) = \{y_k(t)\}$ is an N-dimensional vector and $\mathbf{A}(t) = \{a_{kl}(t)\}$ is an (N,N)-dimensional coefficient matrix. The equation is subject to an initial condition in which case the solution obeys

$$|\mathbf{y}(t_s)\rangle = T\prod_{k=r}^{s-1}\mathbf{F}(t_k, t_{k+1})|\mathbf{y}(t_r)\rangle. \tag{4.2}$$

The Runge–Kutta formalism leads to an approximate solution

$$|\widehat{\mathbf{y}}(t_{k+1})\rangle = T\prod_{k=r}^{s-1}\widehat{\mathbf{F}}(t_k, t_{k+1})|\widehat{\mathbf{y}}(t_r)\rangle, \tag{4.3}$$

where $\widehat{\mathbf{F}}(t_k, t_{k+1})$ is an approximate propagator. For a small integration interval, the propagator is

$$\mathbf{F}(t_k, t_{k+1}) = \exp(\mathbf{A}\delta t_k). \tag{4.4}$$

The approximate propagator is given for any Runge–Kutta method by the rational function

$$\widehat{\mathbf{F}}(t_k, t_{k+1}) = \left[\sum_{r=0}^{m} a_r[\mathbf{A}\delta t_k]^r\right]\left[\sum_{s=0}^{n} b_s[\mathbf{A}\delta t_k]^s\right]^{-1}, \tag{4.5}$$

where a_r and b_s are constants. By implementing the eigenvalue equation $\mathbf{AQ} = \mathbf{Q\Lambda}$, where $\mathbf{Q} = \{\boldsymbol{\lambda}_m\}$ is an eigenvector matrix and $\boldsymbol{\Lambda} = \{\lambda_m\delta_{mn}\}$ an eigenvalue matrix, the propagators may be rewritten as

$$\mathbf{F}(t_k, t_{k+1}) = \mathbf{Q}\exp(\boldsymbol{\Lambda}\delta t_k)\mathbf{Q}^{-1}, \tag{4.6}$$

$$\widehat{\mathbf{F}}(t_k, t_{k+1}) = \mathbf{Q}\left[\sum_{r=0}^{m} a_r[\boldsymbol{\Lambda}\delta t_k]^r\right]\left[\sum_{s=0}^{n} b_s[\boldsymbol{\Lambda}\delta t_k]^s\right]^{-1}\mathbf{Q}^{-1}. \tag{4.7}$$

From these equations one finds that the components of the solution are defined by

$$y_i(t_s) = \sum_{j=1}^{N} c_{ij}(t_r) \prod_{k=r}^{s-1} \exp(\lambda_j \delta t_k), \tag{4.8}$$

where $c_{ij}(t_r)$ are constants. The components of the approximate solution are

$$\widehat{y}_i(t_s) = \sum_{j=1}^{N} c_{ij}(t_r) \prod_{k=r}^{s-1} \mu(\lambda_j \delta t_k), \tag{4.9}$$

where the characteristic roots

$$\mu(\lambda_j \delta t_k) = \left[\sum_{r=0}^{m} a_r [\lambda_j \delta t_k]^r\right] \left[\sum_{s=0}^{n} b_s [\lambda_j \delta t_k]^s\right]^{-1}. \tag{4.10}$$

4.2. *The Local Truncation Error*

The accuracy of a Runge–Kutta method is defined by the local truncation error that represents the difference between the exact and approximate solutions obtained in one step from an exact initial condition [23–26]. For a linear homogeneous system of coupled first-order differential equations, the local truncation error may be obtained from

$$|\mathbf{T}(t_{k+1}, \delta t_k)\rangle = [\mathbf{F}(t_k, t_{k+1}) - \widehat{\mathbf{F}}(t_k, t_{k+1})]|\mathbf{y}(t_k)\rangle. \tag{4.11}$$

Because the exact and approximate solutions must be close, it is evident that the local truncation error is $\mathbf{T}(t_{k+1}, \delta t_k) = \mathbf{O}(\delta t_k^{p+1})$ where the order p is a measure of the accuracy of the method. This shows that any scheme involving q stages may be classified as a (p, q) Runge–Kutta method [24]. The elements of the local truncation error may be rewritten in the form

$$T_i(t_{k+1}, \delta t_k) = \sum_{j=1}^{N} c_{ij}(t_k)[\exp(\lambda_j \delta t_k) - \mu(\lambda_j \delta t_k)], \tag{4.12}$$

which reveals that the characteristic roots must approximate the exact solution components closely in order to obtain a valid numerical solution. This relationship is expressed by the error function

$$O\left(\delta t_k^{p+1}\right) = \exp(\lambda_j \delta t_k) - \mu(\lambda_j \delta t_k), \tag{4.13}$$

which may be plotted as function of $\lambda \delta t$ in order to visualize the accuracy.

4.3. *The Global Truncation Error*

The performance of Runge–Kutta methods depends not only on the accuracy but also on the stability characteristics [23–26]. The stability refers to the behavior of the global

truncation error

$$|\varepsilon(t_s)\rangle = \left[T \prod_{k=0}^{s-1} \mathbf{F}(t_k, t_{k+1}) - T \prod_{k=0}^{s-1} \widehat{\mathbf{F}}(t_k, t_{k+1}) \right] |\mathbf{y}(t_0)\rangle, \tag{4.14}$$

which is the difference between the exact and approximate solutions obtained in several steps from an exact initial condition [23–26]. The global truncation error may be rewritten in the form

$$\varepsilon_i(t_s) = \sum_{j=1}^{N} c_{ij}(t_0)[\exp(s\lambda_j \delta t) - \mu(\lambda_j \delta t)^s], \tag{4.15}$$

where it has been assumed that all stepsizes are equal. It is obvious that a valid solution must satisfy

$$|\varepsilon_i(t_s)| \ll \sum_{j=1}^{N} |c_{ij}(t_0)||\exp(s\lambda_j \delta t)|, \tag{4.16}$$

which indicates that the global truncation error remains small relative to the exact solution. Because the characteristic roots must approximate the exact solution components closely, it is also required that

$$|\varepsilon_i(t_s)| \ll \sum_{j=1}^{N} |c_{ij}(t_0)||\mu(\lambda_j \delta t)|^s. \tag{4.17}$$

When these equations are satisfied the method is said to be numerically stable [23, 24]. A linear homogeneous system of coupled first-order differential equations is inherently stable if $\mathrm{Re}(\lambda_j) \leq 0$. For these systems $|\mu(\lambda_j \delta t)| \leq 1$ defines the requirement for numerical stability. When this condition is satisfied the method is called absolute stable. The stability characteristics of a method may be visualized by its region of absolute stability $\{\lambda\delta t||\mu(\lambda\delta t)| \leq 1\}$. A method with a finite region of absolute stability is called conditionally stable, while a method with an infinite region of absolute stability is denoted unconditionally stable. If the region of absolute stability contains the entire negative-half-plane, the method is A-stable. For a conditionally stable method, the stepsize δt must be chosen carefully in order that $\lambda_j \delta t$ is contained within the region of absolute stability. However, there is no guarantee that all values of δt for which $\lambda_j \delta t$ are within the region of absolute stability will yield an accurate solution. The relationship between stability and accuracy is thus fundamental to the choice of stepsize.

5. SPECIFIC RUNGE–KUTTA METHODS

5.1. *Explicit Methods Based on Newton–Cotes Quadrature Formulas*

Because the efficiency of a Runge–Kutta method depends on the number of stages q, it is desirable to choose the smallest value of q consistent with the order p. For explicit methods of order$1 \leq p \leq 4$ the smallest number of stages is $q = p$. As the order is increased a larger number of stages is required. The smallest number of stages is $q = p + 1$ for $5 \leq p \leq 6$ and $q = p + 2$ for $7 \leq p$ [35]. This does not detract from the usefulness of higher-order methods since these may be used with a larger stepsize [36–39]. A representative third-order

scheme is the classical explicit (3,3) Runge–Kutta method [40]

$$\begin{aligned}\hat{\boldsymbol{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) &= \frac{1}{6}[\mathbf{g}_1 + 4\mathbf{g}_2 + \mathbf{g}_3],\\ \mathbf{g}_1 &= \mathbf{f}(t_k, \hat{\mathbf{y}}(t_k)),\\ \mathbf{g}_2 &= \mathbf{f}\left(t_k + \frac{1}{2}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{2}\delta t_k \mathbf{g}_1\right),\\ \mathbf{g}_3 &= \mathbf{f}(t_k + \delta t_k, \hat{\mathbf{y}}(t_k) + \delta t_k[2\mathbf{g}_2 - \mathbf{g}_1]),\end{aligned} \tag{5.1}$$

and a useful fourth-order scheme is the classical (4,4) Runge–Kutta method [40]

$$\begin{aligned}\hat{\boldsymbol{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) &= \frac{1}{6}[\mathbf{g}_1 + 2\mathbf{g}_2 + 2\mathbf{g}_3 + \mathbf{g}_4],\\ \mathbf{g}_1 &= \mathbf{f}(t_k, \hat{\mathbf{y}}(t_k)),\\ \mathbf{g}_2 &= \mathbf{f}\left(t_k + \frac{1}{2}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{2}\delta t_k \mathbf{g}_1\right),\\ \mathbf{g}_3 &= \mathbf{f}\left(t_k + \frac{1}{2}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{2}\delta t_k \mathbf{g}_2\right),\\ \mathbf{g}_4 &= \mathbf{f}(t_k + \delta t_k, \hat{\mathbf{y}}(t_k) + \delta t_k \mathbf{g}_3).\end{aligned} \tag{5.2}$$

In order to achieve fifth-order accuracy for an explicit Runge–Kutta method, a minimum of six stages is required. An example is the Butcher explicit (5,6) Runge–Kutta method [37]

$$\begin{aligned}\hat{\boldsymbol{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) &= \frac{1}{90}[7\mathbf{g}_1 + 32\mathbf{g}_3 + 12\mathbf{g}_4 + 32\mathbf{g}_5 + 7\mathbf{g}_6],\\ \mathbf{g}_1 &= \mathbf{f}(t_k, \hat{\mathbf{y}}(t_k)),\\ \mathbf{g}_2 &= \mathbf{f}\left(t_k + \frac{1}{4}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{4}\delta t_k \mathbf{g}_1\right),\\ \mathbf{g}_3 &= \mathbf{f}\left(t_k + \frac{1}{4}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{8}\delta t_k[\mathbf{g}_1 + \mathbf{g}_2]\right),\\ \mathbf{g}_4 &= \mathbf{f}\left(t_k + \frac{1}{2}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{2}\delta t_k[-\mathbf{g}_1 + 2\mathbf{g}_2]\right),\\ \mathbf{g}_5 &= \mathbf{f}\left(t_k + \frac{3}{4}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{16}\delta t_k[3\mathbf{g}_1 + 9\mathbf{g}_4]\right),\\ \mathbf{g}_6 &= \mathbf{f}\left(t_k + \delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{7}\delta t_k[-3\mathbf{g}_1 + 2\mathbf{g}_2 + 12\mathbf{g}_3 - 12\mathbf{g}_4 + 8\mathbf{g}_5]\right).\end{aligned} \tag{5.3}$$

The stability and accuracy characteristics are illustrated in Fig. 1, which shows that the methods are conditionally stable. The accuracy is seen to increase by an order of magnitude in going from the (3,3) to the (4,4) method and by another order of magnitude in going from the (4,4) to the (5,6) method.

5.2. *Implicit Methods Based on Gauss–Legendre Quadrature Formulas*

The advantages of implicit Runge–Kutta methods are high orders for the number of stages and desirable stability characteristics. In cases where stability is more important

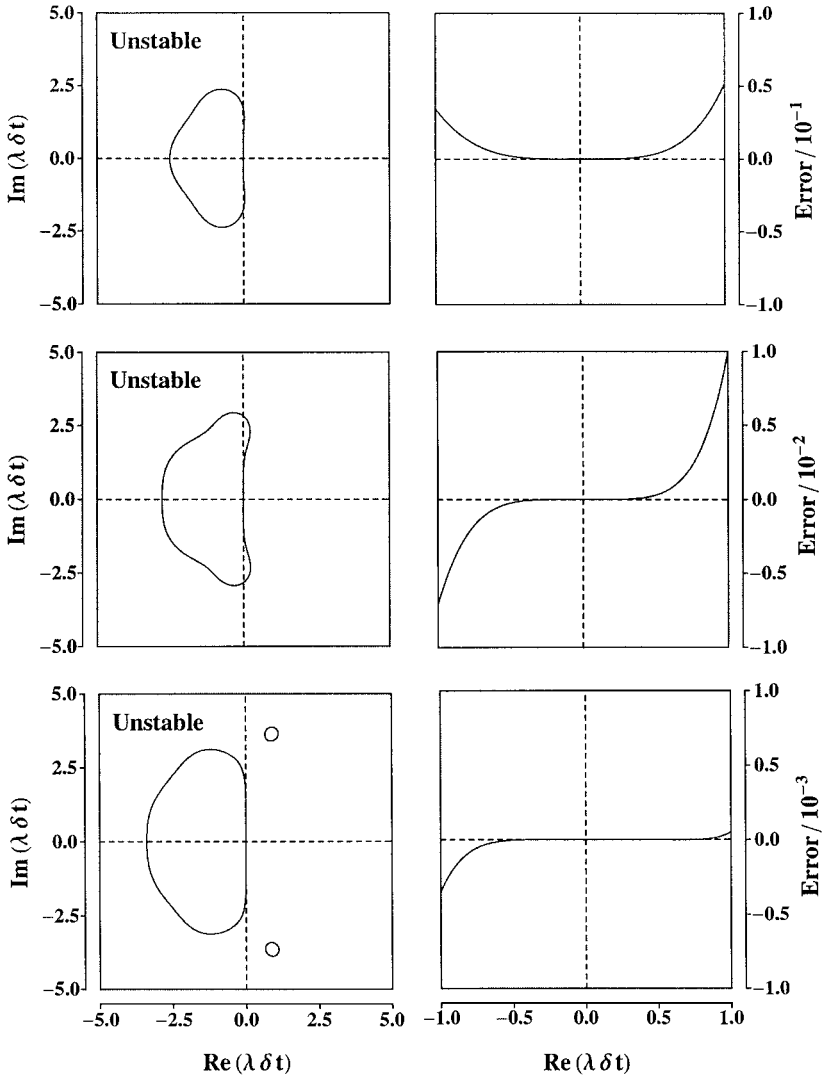

FIG. 1. Region of absolute stability (first column) and error function (second column) for explicit Runge–Kutta methods. The figure includes the classical explicit (3,3) Runge–Kutta method (first row), the classical explicit (4,4) Runge–Kutta method (second row), and the Butcher explicit (5,6) Runge–Kutta method (third row).

than accuracy, implicit methods may perform better than explicit methods. A useful class of implicit methods is defined by the Gauss–Legendre quadrature formulas. For each such formula, there is an implicit Runge–Kutta method of order $p = 2q$ [41]. An example is the implicit (4,2) Runge–Kutta method of Gauss type

$$\hat{\mathbf{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) = \frac{1}{2}[\mathbf{g}_1 + \mathbf{g}_2],$$

$$\mathbf{g}_1 = \mathbf{f}\left(t_k + \frac{[3-\sqrt{3}]}{6}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{4}\delta t_k \mathbf{g}_1 + \frac{[3-2\sqrt{3}]}{12}\delta t_k \mathbf{g}_2\right), \quad (5.4)$$

$$\mathbf{g}_2 = \mathbf{f}\left(t_k + \frac{[3+\sqrt{3}]}{6}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{[3+2\sqrt{3}]}{12}\delta t_k \mathbf{g}_1 + \frac{1}{4}\delta t_k \mathbf{g}_2\right).$$

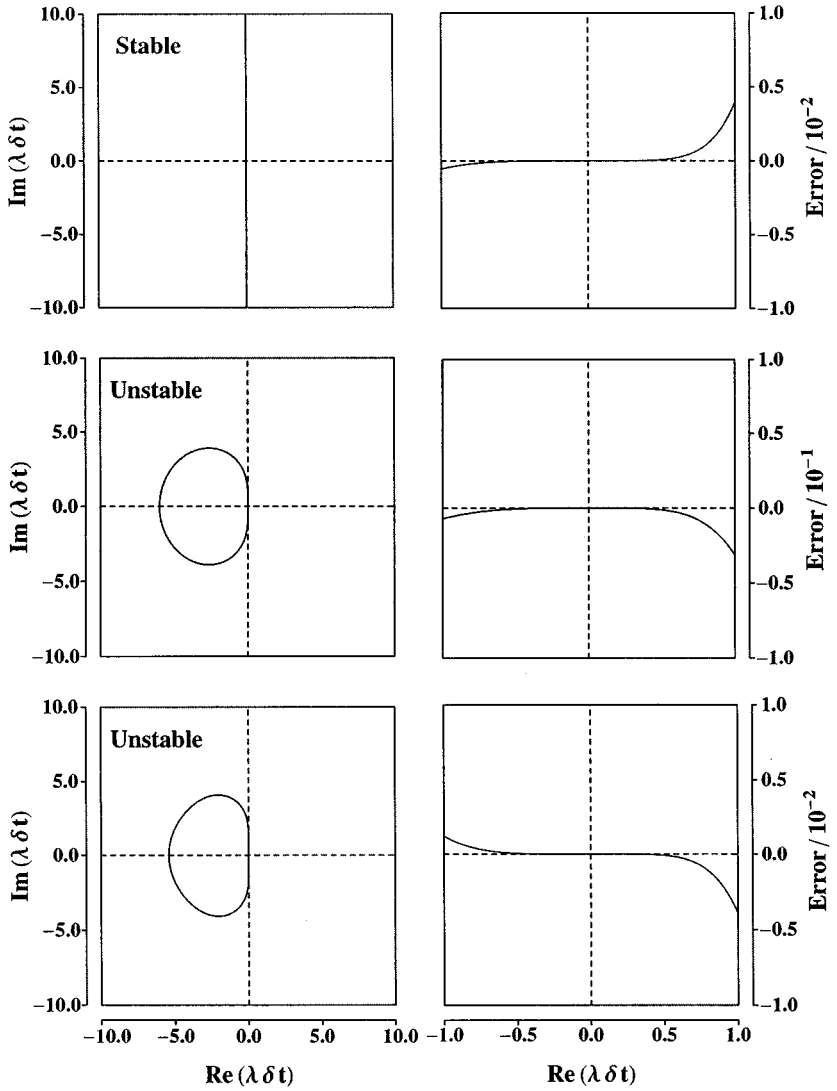

FIG. 2. Region of absolute stability (first column) and error function (second column) for semi-implicit and implicit Runge–Kutta methods. The figure includes the implicit (4,2) Runge–Kutta method of Gauss type (first row), the semi-implicit (3,2) Runge–Kutta method of first Radau type (second row), and the semi-implicit (4,3) Runge–Kutta method of Lobatto type (third row).

The accuracy and stability are revealed by Fig. 2, which shows that the scheme is A-stable. The error function demonstrates that the method has a relatively high accuracy for the number of stages.

5.3. *Implicit Methods Based on Radau Quadrature Formulas*

Another important class of implicit Runge–Kutta methods is derived from the Radau quadrature formulas. For each such formula there is a Runge–Kutta method of order $p = 2q - 1$ [42]. In the first type of Radau quadrature formulas, the ordinates are subject to the constraint that $\alpha_1 = 0$ which is the same as saying that the derivatives must be evaluated at

the initial point. A typical example is the semi-implicit (3,2) Runge–Kutta method of first Radau type

$$\begin{aligned}\hat{\mathbf{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) &= \frac{1}{4}[\mathbf{g}_1 + 3\mathbf{g}_2],\\ \mathbf{g}_1 &= \mathbf{f}(t_k, \hat{\mathbf{y}}(t_k)),\\ \mathbf{g}_2 &= \mathbf{f}\left(t_k + \frac{2}{3}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{3}\delta t_k[\mathbf{g}_1 + \mathbf{g}_2]\right).\end{aligned} \tag{5.5}$$

The second type of Radau quadrature formulas is obtained by imposing the condition that $\alpha_q = 1$ which means that the derivatives must be evaluated in the final point. A representative example is the semi-implicit (3,2) Runge–Kutta method of second Radau type

$$\begin{aligned}\hat{\mathbf{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) &= \frac{1}{4}[3\mathbf{g}_1 + \mathbf{g}_2],\\ \mathbf{g}_1 &= \mathbf{f}\left(t_k + \frac{1}{3}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{3}\delta t_k \mathbf{g}_1\right),\\ \mathbf{g}_2 &= \mathbf{f}(t_k + \delta t_k, \hat{\mathbf{y}}(t_k) + \delta t_k \mathbf{g}_1).\end{aligned} \tag{5.6}$$

The accuracy and stability properties are shown in Fig. 2, which demonstrates that the methods are conditionally stable. The accuracy is low and corresponds to the explicit (3,3) Runge–Kutta method.

5.4. *Implicit Methods Based on Lobatto Quadrature Formulas*

The Lobatto quadrature formulas define another class of implicit Runge–Kutta methods. The ordinates are subject to the constraints that $\alpha_1 = 0$ and $\alpha_q = 1$ which means that the derivatives must be evaluated in the first and final points. For each Lobatto formula,there is a Runge–Kutta method of order $p = 2q - 2$ [42]. A useful example is the implicit (4,3) Runge–Kutta method of Lobatto type

$$\begin{aligned}\hat{\mathbf{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) &= \frac{1}{6}[\mathbf{g}_1 + 4\mathbf{g}_2 + \mathbf{g}_3],\\ \mathbf{g}_1 &= \mathbf{f}(t_k, \hat{\mathbf{y}}(t_k)),\\ \mathbf{g}_2 &= \mathbf{f}\left(t_k + \frac{1}{2}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{4}\delta t_k[\mathbf{g}_1 + \mathbf{g}_2]\right),\\ \mathbf{g}_3 &= \mathbf{f}(t_k + \delta t_k, \hat{\mathbf{y}}(t_k) + \delta t_k \mathbf{g}_2).\end{aligned} \tag{5.7}$$

The accuracy and stability properties are shown in Fig. 2. This reveals that the method is conditionally stable with an accuracy that corresponds to the implicit (4,2) Runge–Kutta method of Gauss type.

6. NUMERICAL EXPERIMENTS

6.1. *Preliminaries*

The usefulness of the Runge–Kutta formalism depends not only on the integration method but also on the system of differential equations. This implies that Runge–Kutta methods

that have proven useful for some systems may be inferior in other cases. In this section we evaluate the performance of different Runge–Kutta methods for solving the stochastic Liouville–von Neumann equation applied to deuteron MAS NMR spectroscopy. In this case, the coefficient matrix may be reduced into eight independent systems (Eqs. (2.29)). The time-independent systems are solved most readily by the eigenvalue method. For the periodically time-dependent systems, we have applied Runge–Kutta methods to obtain a solution.

The results are for a deuteron system subject to molecular motion and MAS conditions with the rotation frequency $\nu_r = 5$ kHz. It is assumed that the quadrupole coupling constant $C_Q = 200$ kHz and asymmetry parameter $\eta_Q = 0.10$. These parameters reflect the small and nearly axially symmetric quadrupole interaction that characterizes most deuteron systems. The motion involves discrete rotation about a crystallite fixed axis with rotation angles $\xi_n = \frac{2\pi[n-1]}{N}$ $(n = 1, \ldots, N)$. These define the values of a discrete stochastic variable which evolves according to a discrete Markov process with rate constants $k_{mn} = k_{nm} = \delta_{m,n+1}k$ $(n = 1, \ldots, N-1)$ and $k_{1N} = k_{N1} = k$. The relative orientation of the principal axis system of the quadrupole tensor and the crystallite fixed axis system is defined by the Euler angles $\mathbf{\Omega}(\xi_n) = \{0, \arctan(\sqrt{2}), \xi_n\}$ $(n = 1, \ldots, N)$. This model represents molecular motion in many important systems and has the advantage of computational simplicity.

The performance of the Runge–Kutta methods was evaluated by comparing computation times used to calculate a sequence of approximate solutions $\widehat{\mathbf{y}}(t_n)$ corresponding to the independent variables $t_n = 2n$ (μs). The computation time depends on the number of points in the sequence, and it is noted that the results are for the calculation of a sequence with one thousand points. In this case the computation time includes contributions from both the evaluation of the propagator and the evolution of the density operator. Both of these elements are critical factors in terms of execution time. It is evident that the computation time depends on the crystallite orientation. The numerical results were therefore averaged over a uniform distribution of crystallites. This implies that the computation time for a polycrystalline sample may be estimated by multiplying the results by the number of crystallites.

The calculations were performed with Fortran 95 programs on a 400 MHz dual Pentium II computer with 256 Mb RAM. It is noted that the execution time depends on both the operating system and the Fortran compiler. We have compared computation times for the Windows NT and Linux operating systems using the Compaq Visual Fortran 95 and PGI Fortran 90 and HPF compilers. The results have demonstrated that the Compaq Visual Fortran 95 compiler executing under Windows NT is the most efficient environment on our system. The programs were all subject to full serial optimization to improve the performance.

6.2. *Comparison of Runge–Kutta Methods*

The classification of the Runge–Kutta methods is important because the individual classes have different stability and accuracy characteristics. As representative explicit schemes we consider the classical explicit (3,3) Runge–Kutta method (Eqs. (5.1)), the classical explicit (4,4) Runge–Kutta method (Eqs. (5.2)), and the Butcher explicit (5,6) Runge–Kutta method (Eqs. (5.3)). These methods were implemented in two different algorithms using either fixed stepsizes or automatic stepsize control [23–26]. For fixed stepsizes, the classical explicit (3,3) Runge–Kutta method is Method 1a, the classical explicit (4,4) Runge–Kutta method is Method 2a, and the Butcher explicit (5,6) Runge–Kutta method is Method 3a. For automatic

stepsize control, the methods are denoted Method 1b, Method 2b, and Method 3b. Some examples of implicit schemes are the implicit (4,2) Runge–Kutta method of Gauss type (Eqs. (5.4)), the semi-implicit (3,2) Runge-Kutta method of first Radau type (Eqs. (5.5)), and the semi-implicit (4,3) Runge–Kutta method of Lobatto type (Eqs. (5.7)). In the case of fixed stepsizes, the implicit (4,2) Runge–Kutta method of Gauss type is Method 4a, the semi-implicit (3,2) Runge–Kutta method of first Radau type is Method 5a, and the semi-implicit (4,3) Runge–Kutta method of Lobatto type is Method 6a. For automatic stepsize control, the methods are Method 4b, Method 5b, and Method 6b.

The computation times were obtained for the stepsizes $\delta t = 0.1\,\mu$s, $\delta t = 0.2\,\mu$s, and $\delta t = 0.4\,\mu$s which are representative for calculations of deuteron MAS NMR spectra. The effects on the accuracy of decreasing the stepsize depend on the specific Runge–Kutta method. As an example, we have investigated the accuracy of Method 2a for different fixed stepsizes. The results shown in Fig. 3 reveal that the accuracy improves significantly by decreasing the stepsize. However, it is expected that the increased round-off error resulting from a decreasing stepsize will eventually reduce the accuracy. This suggests that there may be both a minimum and maximum value of the allowed stepsize. Because a fixed stepsize may result in an inaccurate and inefficient calculation, it is interesting to compare the results with automatic stepsize control. The algorithms for automatic stepsize control were based on the Richardson extrapolation procedure for estimating the local truncation error [23–26]. The methods used the local error tolerances $T_m^{\max} = 10^{-3}$, $T_m^{\max} = 10^{-4}$, and $T_m^{\max} = 10^{-5}$, which are typical for simulations of deuteron MAS NMR spectra. The

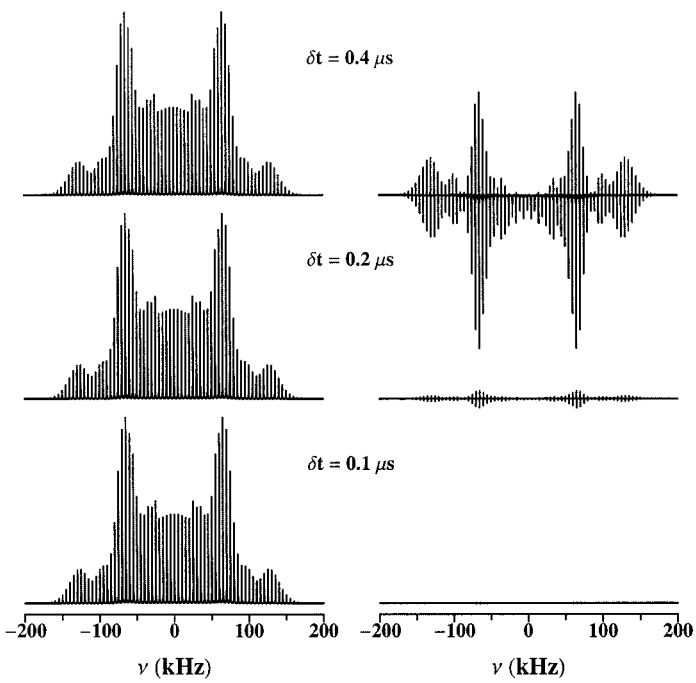

FIG. 3. Theoretical deuteron MAS NMR spectra (left column) and absolute error (right column) calculated with the classical explicit (4,4) Runge–Kutta method for different fixed stepsizes. The results are for a deuteron quadrupole tensor that reorients between two orientations $\Omega(\xi_1) = \{0, \arctan(\sqrt{2}), 0\}$ and $\Omega(\xi_2) = \{0, \arctan(\sqrt{2}), \pi\}$. The calculations used the quadrupole coupling constant $C_Q = 200$ kHz, asymmetry parameter $\eta_Q = 0.10$, rotation frequency $\nu_r = 5.0$ kHz, and rate constant $k = 10^2$ Hz.

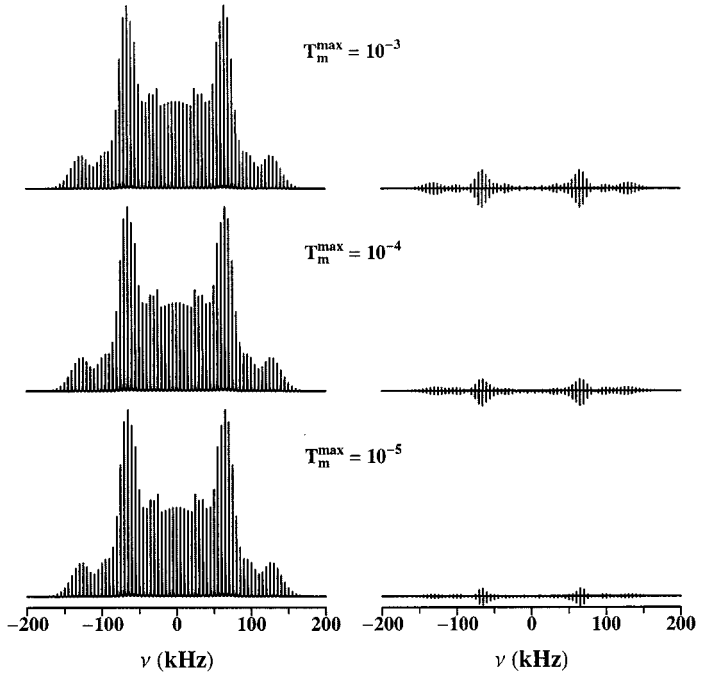

FIG. 4. Theoretical deuteron MAS NMR spectra (left column) and absolute error (right column) calculated with the classical explicit (4,4) Runge–Kutta method for different local error tolerances. The results are for a deuteron quadrupole tensor that reorients between two orientations $\Omega(\xi_1) = \{0, \arctan(\sqrt{2}), 0\}$ and $\Omega(\xi_2) = \{0, \arctan(\sqrt{2}), \pi\}$. The calculations used the quadrupole coupling constant $C_Q = 200$ kHz, asymmetry parameter $\eta_Q = 0.10$, rotation frequency $\nu_r = 5.0$ kHz, and rate constant $k = 10^2$ Hz.

effects of decreasing the local error tolerances are shown in Fig. 4 for Method 2b. The results demonstrate that the accuracy improves as the local error tolerances are decreased. However, the differences are too small to justify the use of very small error tolerances. The execution times were determined as function of the number of motional states N for a fixed rate constant $k = 10^2$ Hz. The results are listed in Tables I through VII and illustrated in Figs. 5 through 7. For automatic stepsize control, the execution times were determined for $N = 2$ as function of the rate constant k. The results are listed in Table VIII and shown in Figs. 8 and 9.

From the computational results the stability characteristics of the Runge–Kutta methods become apparent. It is recalled that the stability is determined by the eigenvalue structure of the coefficient matrix. For the systems discussed in this paper, the coefficient matrix is of the form $\mathbf{A}(t) = \Xi + i\boldsymbol{\Omega}(t)$, where $\boldsymbol{\Xi} = \{\Xi(\xi_m, \xi_n)\}$ is the stochastic matrix and $\boldsymbol{\Omega}(t) = \{\omega_Q^{(1)}(\xi_n, t)\delta_{mn}\}$. For a system with two motional states, the eigenvalues are

$$\begin{aligned}\lambda_1 &= \frac{i}{2}\left[\omega_Q^{(1)}(t,\xi_1) + \omega_Q^{(1)}(t,\xi_2)\right] + \frac{1}{2}\sqrt{4k^2 - \left[\omega_Q^{(1)}(t,\xi_1) - \omega_Q^{(1)}(t,\xi_2)\right]^2} - k,\\ \lambda_2 &= \frac{i}{2}\left[\omega_Q^{(1)}(t,\xi_1) + \omega_Q^{(1)}(t,\xi_2)\right] - \frac{1}{2}\sqrt{4k^2 - \left[\omega_Q^{(1)}(t,\xi_1) - \omega_Q^{(1)}(t,\xi_2)\right]^2} - k,\end{aligned} \tag{6.1}$$

which show that the real parts are always less than or equal to zero. This is valid for any coefficient matrix, and the stochastic Liouville–von Neumann equation is therefore an

TABLE I

Absolute CPU Times (s) for the Eigenvalue Method

N	$\delta t = 0.4\,\mu$s	$\delta t = 0.2\,\mu$s	$\delta t = 0.1\,\mu$s
2	0.0284	0.0555	0.1095
3	0.0545	0.1072	0.2125
4	0.0786	0.1549	0.3073
5	0.1175	0.2321	0.4608
6	0.1628	0.3213	0.6387
7	0.2231	0.4415	0.8784
8	0.2863	0.5667	1.1281
9	0.3852	0.7633	1.5196
10	0.4811	0.9541	1.9003

Note. The results are for a deuteron quadrupole tensor whose principal axis system reorients between N different orientations. $\Omega(\xi_n) = \{0, \arctan(\sqrt{2}), \frac{2\pi[n-1]}{N}\}(n = 1, \ldots, N)$. The motion is described by a discrete Markov model with rate constants $k_{mn} = k_{nm} = \delta_{m,n+1}10^2$ Hz $(n = 1, \ldots, N-1)$ and $k_{1N} = k_{N1} = 10^2$ Hz. The calculations used the quadrupole coupling constant $C_Q = 200$ kHz, asymmetry parameter $\eta_Q = 0.10$, and rotation frequency $\nu_r = 5.0$ kHz.

TABLE II

Absolute CPU Times (s) for Runge–Kutta Methods Using Fixed Stepsizes $\delta t = 0.4\,\mu$s

N	Method 1a	Method 2a	Method 3a	Method 4a	Method 5a	Method 6a	Method 7a	Method 8a	Method 9a	Method10a
2	0.0059	0.0075	0.0113	0.0336	0.0137	0.0151	0.0330	0.0098	0.0117	0.0449
3	0.0091	0.0117	0.0175	0.0573	0.0234	0.0249	0.0571	0.0136	0.0176	0.0605
4	0.0124	0.0158	0.0243	0.0883	0.0361	0.0400	0.0892	0.0156	0.0215	0.0664
5	0.0199	0.0255	0.0393	0.1351	0.0542	0.0605	0.1363	0.0254	0.0352	0.0957
6	0.0278	0.0357	0.0554	0.1933	0.0786	0.0874	0.1953	0.0332	0.0410	0.1192
7	0.0397	0.0509	0.0786	0.2638	0.1040	0.1182	0.2683	0.0449	0.0566	0.1582
8	0.0495	0.0638	0.0973	0.3465	0.1338	0.1553	0.3545	0.0527	0.0605	0.1758
9	0.0679	0.0876	0.1348	0.4573	0.1821	0.2051	0.4688	0.0703	0.0859	0.2363
10	0.0866	0.1121	0.1750	0.5832	0.2261	0.2563	0.5976	0.0879	0.1054	0.2969

TABLE III

Absolute CPU Times (s) for Runge–Kutta Methods Using Fixed Stepsizes $\delta t = 0.2\,\mu$s

N	Method 1a	Method 2a	Method 3a	Method 4a	Method 5a	Method 6a	Method 7a	Method 8a	Method 9a	Method 10a
2	0.0108	0.0140	0.0214	0.0669	0.0264	0.0297	0.0647	0.0156	0.0234	0.0898
3	0.0169	0.0219	0.0356	0.1172	0.0444	0.0483	0.1124	0.0234	0.0352	0.1152
4	0.0231	0.0298	0.0469	0.1768	0.0708	0.0771	0.1761	0.0293	0.0410	0.1367
5	0.0372	0.0485	0.0759	0.2710	0.1050	0.1182	0.2704	0.0508	0.0645	0.1855
6	0.0563	0.0682	0.1070	0.3862	0.1528	0.1709	0.3864	0.0625	0.0801	0.2344
7	0.0752	0.0977	0.1528	0.5268	0.2041	0.2324	0.5304	0.0879	0.1094	0.3125
8	0.0948	0.1232	0.1900	0.6934	0.2622	0.3047	0.7034	0.0977	0.1172	0.3457
9	0.1299	0.1693	0.2632	0.9092	0.3574	0.4033	0.9288	0.1367	0.1642	0.4668
10	0.1667	0.2174	0.3424	1.1602	0.4458	0.5049	1.1858	0.1660	0.1973	0.5879

TABLE IV

Absolute CPU Times (s) for Runge–Kutta Methods Using Fixed Stepsizes $\delta t = 0.1\,\mu$s

N	Method 1a	Method 2a	Method 3a	Method 4a	Method 5a	Method 6a	Method 7a	Method 8a	Method 9a	Method 10a
2	0.0208	0.0270	0.0418	0.1323	0.0508	0.0571	0.1235	0.0313	0.0469	0.1758
3	0.0323	0.0424	0.0657	0.2319	0.0864	0.0947	0.2197	0.0469	0.0664	0.2305
4	0.0446	0.0580	0.0919	0.3516	0.1392	0.1519	0.3472	0.0586	0.0781	0.2676
5	0.0717	0.0944	0.1491	0.5386	0.2075	0.2334	0.5366	0.0957	0.1270	0.3672
6	0.1024	0.1333	0.2106	0.7681	0.3018	0.3384	0.7656	0.1230	0.1543	0.4668
7	0.1463	0.1913	0.3012	1.0469	0.4028	0.4595	1.0586	0.1719	0.2129	0.6172
8	0.1853	0.2418	0.3753	1.3799	0.5186	0.6035	1.3960	0.1875	0.2305	0.6914
9	0.2536	0.3326	0.5195	1.8130	0.7056	0.7993	1.8423	0.2656	0.2503	0.9219
10	0.3266	0.4276	0.6769	2.3120	0.8823	1.0024	2.3620	0.3662	0.3926	1.1660

TABLE V
Absolute CPU Times (s) for Runge–Kutta Methods Using Automatic Stepsize Control with the Local Error Tolerances $T_m^{\max} = 10^{-3}$

N	Method 1b	Method 2b	Method 3b	Method 4b	Method 5b	Method 6b	Method 7b	Method 8b	Method 9b	Method 10b
2	0.0075	0.0084	0.0129	0.0469	0.0166	0.0190	0.0386	0.0117	0.0156	0.0547
3	0.0128	0.0132	0.0203	0.0874	0.0278	0.0303	0.0664	0.0176	0.0215	0.0722
4	0.0193	0.0182	0.0282	0.1611	0.0444	0.0479	0.1147	0.0254	0.0273	0.0820
5	0.0334	0.0291	0.0455	0.2495	0.0664	0.0737	0.1660	0.0449	0.0410	0.1133
6	0.0504	0.0409	0.0639	0.3975	0.1260	0.1040	0.2544	0.0566	0.0508	0.1445
7	0.0758	0.0584	0.0912	0.5757	0.1401	0.1430	0.3599	0.0820	0.0703	0.1914
8	0.0959	0.0723	0.1133	0.7554	0.1875	0.1860	0.4736	0.0918	0.0742	0.2109
9	0.1332	0.0993	0.1571	1.0186	0.2422	0.2432	0.6147	0.1270	0.1035	0.2871
10	0.1741	0.1286	0.2065	1.3232	0.3174	0.3135	0.8340	0.1621	0.1270	0.3555

inherently stable system. In the slow ($k_{mn} \leq 10^4$ Hz) and intermediate (10^4 Hz $\leq k_{mn} \leq 10^7$ Hz) motion regimes the eigenvalues may be approximated by

$$\begin{aligned} \lambda_1 &= i\omega_Q^{(1)}(t, \xi_1) - k, \\ \lambda_2 &= i\omega_Q^{(1)}(t, \xi_2) - k, \end{aligned} \tag{6.2}$$

which are sufficiently accurate provided that $k \ll \frac{1}{2}|\omega_Q^{(1)}(t, \xi_1) - \omega_Q^{(1)}(t, \xi_2)|$. In the slow and intermediate motion regimes one finds that the magnitudes of $\mathrm{Im}(\lambda_1)$ and $\mathrm{Im}(\lambda_2)$ are larger than the magnitudes of $\mathrm{Re}(\lambda_1)$ and $\mathrm{Re}(\lambda_2)$, respectively. This shows that the magnitudes of the eigenvalues and the corresponding stepsizes are independent of the rate constant. This is verified by the computation times which are invariant (Figs. 8 and 9) in the slow and intermediate motion regimes.

The regular pattern observed in the slow and intermediate regimes changes for fast (10^7 Hz $\leq k_{mn}$) molecular motion. By considering the eigenvalues in the fast motion regime one finds that

$$\begin{aligned} \lambda_1 &= \frac{i}{2}\left[\omega_Q^{(1)}(t, \xi_1) + \omega_Q^{(1)}(t, \xi_2)\right], \\ \lambda_2 &= \frac{i}{2}\left[\omega_Q^{(1)}(t, \xi_1) + \omega_Q^{(1)}(t, \xi_2)\right] - 2k, \end{aligned} \tag{6.3}$$

which are sufficiently accurate provided that $k \gg \frac{1}{2}|\omega_Q^{(1)}(t, \xi_1) - \omega_Q^{(1)}(t, \xi_2)|$. It is noted that the magnitude of $\mathrm{Im}(\lambda_1)$ is larger than the magnitude of $\mathrm{Re}(\lambda_1)$ and that the magnitude

TABLE VI
Absolute CPU Times (s) for Runge–Kutta Methods Using Automatic Stepsize Control with the Local Error Tolerances $T_m^{\max} = 10^{-4}$

N	Method 1b	Method 2b	Method 3b	Method 4b	Method 5b	Method 6b	Method 7b	Method 8b	Method 9b	Method 10b
2	0.0103	0.0093	0.0129	0.0972	0.0205	0.0190	0.0483	0.0117	0.0176	0.0547
3	0.0213	0.0158	0.0203	0.2026	0.0371	0.0303	0.0947	0.0234	0.0273	0.0703
4	0.0330	0.0234	0.0283	0.3442	0.0664	0.0488	0.1802	0.0430	0.0430	0.0820
5	0.0582	0.0402	0.0456	0.5518	0.1084	0.0747	0.2905	0.0742	0.0742	0.1133
6	0.0893	0.0616	0.0640	0.8535	0.2114	0.1074	0.4595	0.1016	0.0996	0.1445
7	0.1321	0.0921	0.0913	1.2178	0.2422	0.1494	0.6650	0.1445	0.1406	0.1914
8	0.1677	0.1185	0.1133	1.5825	0.3267	0.1948	0.8784	0.1641	0.1582	0.2148
9	0.2336	0.1654	0.1572	2.1221	0.4351	0.2485	1.1938	0.2305	0.2227	0.2871
10	0.3068	0.2167	0.2064	2.7593	0.5537	0.3296	1.5508	0.2891	0.2793	0.3574

TABLE VII

Absolute CPU Times (s) for Runge–Kutta Methods Using Automatic Stepsize Control with the Local Error Tolerances $T_m^{max} = 10^{-5}$

N	Method 1b	Method 2b	Method 3b	Method 4b	Method 5b	Method 6b	Method 7b	Method 8b	Method 9b	Method 10b
2	0.0176	0.0118	0.0130	0.1768	0.0347	0.0200	0.0986	0.0195	0.0125	0.0957
3	0.0348	0.0218	0.0204	0.3627	0.0683	0.0356	0.1865	0.0391	0.0430	0.1758
4	0.0522	0.0340	0.0283	0.6372	0.1201	0.0635	0.3394	0.0684	0.0703	0.2246
5	0.0917	0.0599	0.0457	1.0034	0.1948	0.1016	0.5264	0.1172	0.1230	0.3516
6	0.1389	0.0912	0.0641	1.5522	0.3867	0.1587	0.8091	0.1641	0.1660	0.4746
7	0.2053	0.1352	0.0913	2.2153	0.4375	0.2280	1.1729	0.2383	0.2363	0.6152
8	0.2607	0.1716	0.1134	2.8770	0.5835	0.3081	1.5205	0.2656	0.2598	0.7129
9	0.3613	0.2388	0.1571	3.8804	0.7793	0.4072	2.0688	0.3711	0.3633	0.9531
10	0.4743	0.3144	0.2066	5.0474	0.9922	0.5386	2.6543	0.4668	0.4473	1.2186

of $\mathrm{Re}(\lambda_2)$ is much larger than the magnitude of $\mathrm{Im}(\lambda_2)$. The magnitudes of the eigenvalues are therefore determined by $\mathrm{Im}(\lambda_1)$ and $\mathrm{Re}(\lambda_2)$. Because the magnitude of $\mathrm{Re}(\lambda_2)$ is much larger than the magnitude of $\mathrm{Im}(\lambda_1)$ the eigenvalues differ significantly.

It is recalled that systems with widely different eigenvalues are characterized as stiff, and that the stiffness is measured by the stiffness ratio [20–26]. A stiff system represents a special problem for a Runge–Kutta method. More specifically, if a conditionally stable Runge–Kutta method is applied to a stiff system, the solution components defined by the eigenvalues with large negative real parts will enforce a small stepsize in order to satisfy the conditions of accuracy and stability. Furthermore, the solution components defined by the eigenvalues with small negative real parts will lead to a wide range of integration. This implies that the execution time for a conditionally stable Runge–Kutta method may become excessive. This is verified by the computation times for Methods 1a (1b) through 6a (6b) which are seen (Figs. 8 and 9) to be increasing functions of the rate constant in the fast motion regime. The explicit Runge–Kutta methods are more sensitive to the stiffness than the semi-implicit and implicit schemes, which have better stability properties. Because the stability characteristics are almost equal for the explicit Runge–Kutta methods, their performance differs only slightly. It is found that Method 6a (6b) is more sensitive to the stiffness than Method 5a (5b) and that Method 4a (4b) depends only weakly on the stiffness. This reflects the better stability of Methods 4a (4b) and 5a (5b).

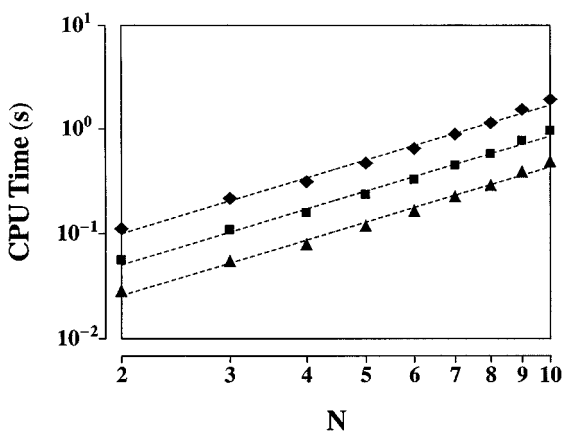

FIG. 5. Absolute computation time as function of the number of motional states N for the eigenvalue method. The method implemented fixed stepsizes given by $\delta t = 0.1\,\mu$s (◆), $\delta t = 0.2\,\mu$s (■), and $\delta t = 0.4\,\mu$s (▲).

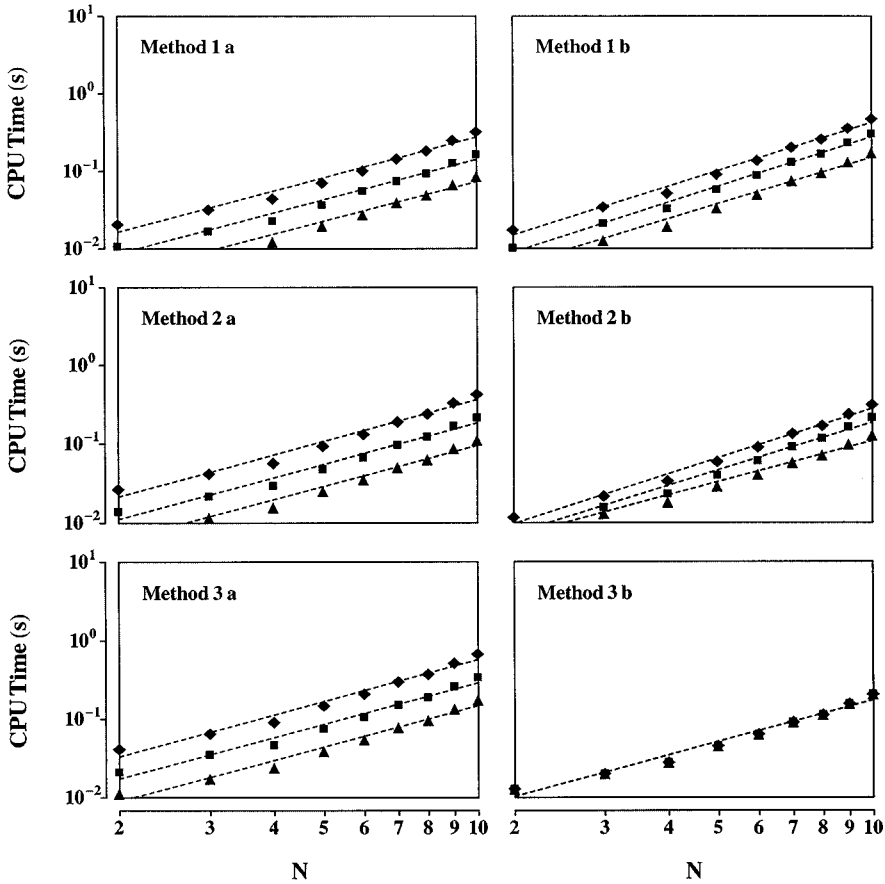

FIG. 6. Absolute computation time as function of the number of motional states N for explicit Runge–Kutta methods. The methods used fixed stepsizes (first column) given by $\delta t = 0.1\,\mu$s (◆), $\delta t = 0.2\,\mu$s (■), and $\delta t = 0.4\,\mu$s (▲) and automatic stepsize control (second column) with the local error tolerances $T_m^{\max} = 10^{-5}$ (◆), $T_m^{\max} = 10^{-4}$ (■), and $T_m^{\max} = 10^{-3}$ (◆). The figure includes the classical explicit (3,3) Runge–Kutta method (first row), the classical explicit (4,4) Runge–Kutta method (second row), and the Butcher explicit (5,6) Runge–Kutta method (third row).

The accuracy properties of explicit Runge–Kutta methods are illustrated by the behavior of Methods 1a (1b) through 3a (3b), which have comparable stability properties. The error functions shown in Fig. 1 demonstrate that the fourth-order Method 2a (2b) is more accurate by an order of magnitude compared with the third-order Method 1a (1b). In addition, it is seen that the fifth-order Method 3a (3b) is an order of magnitude more accurate than Method 2a (2b). For a fixed stepsize, the computation time depends only on the number of derivative evaluations. It is seen that Method 1a requires three, Method 2a four, and Method 3a six derivative evaluations for each step. If the derivative evaluations are rate determining this suggests that the execution times for fixed stepsizes be related by $t_{1a} = \frac{3}{4}t_{2a}$ and $t_{3a} = \frac{3}{2}t_{2a}$. These predictions agree closely enough with the actual results to suggest that most of the execution time is spent evaluating the derivatives. In the slow and intermediate motion regimes, the stepsizes are determined solely by the condition of accuracy. It is important to note that the increased accuracy in going from the third- to the fourth- to the fifth-order method is about 10 to 100, and that the computation times are similar. This indicates that the added computation time is less important than the decreasing error. From these results it can

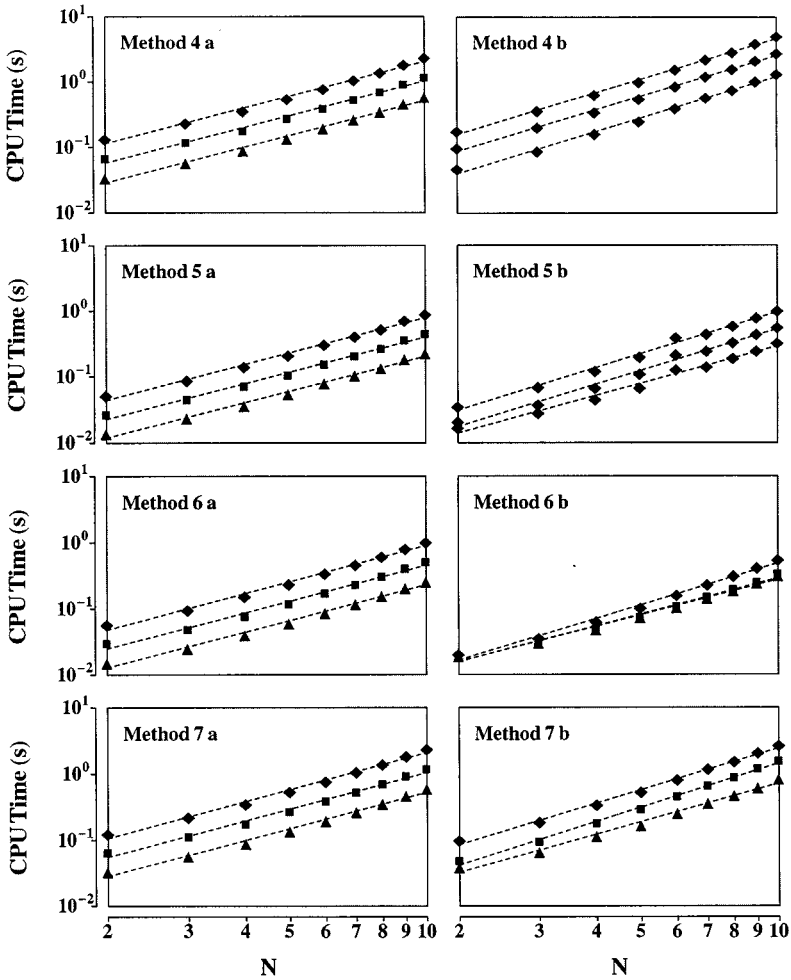

FIG. 7. Absolute computation time as function of the number of motional states N for semi-implicit and implicit Runge–Kutta methods. The methods implemented fixed stepsizes (first column) given by $\delta t = 0.1\,\mu$s (◆), $\delta t = 0.2\,\mu$s (■), and $\delta t = 0.4\,\mu$s (▲) and automatic stepsize control (second column) with the local error tolerances $T_m^{\max} = 10^{-5}$ (◆), $T_m^{\max} = 10^{-4}$ (■), and $T_m^{\max} = 10^{-3}$ (▲). The figure includes the implicit (4,2) Runge–Kutta method of Gauss type (first row), the semi-implicit (3,2) Runge–Kutta method of first Radau type (second row), the semi-implicit (4,3) Runge–Kutta method of Lobatto type (third row), and the stiffly A-stable (3,2) Runge–Kutta method of Padé type (fourth row).

be concluded that Method 3a (3b) is preferable if high accuracy is required. The computation times for Method 3b depend only weakly on the error tolerances demonstrating the high accuracy of the scheme. It is evident that Method 2a (2b) is preferred for calculations of low accuracy.

For the semi-implicit and implicit Runge–Kutta methods, the accuracy properties are illustrated by Methods 5a (5b) and 6a (6b), which have similar stability properties. The fourth-order Method 6a (6b) is more accurate by an order of magnitude than the third-order Method 5a (5b). The result is that the stepsizes used by Method 5a (5b) are smaller than for Method 6a (6b). This is verified by the computation times for Method 5b, which are longer than for Method 6b. The execution times for Method 6a are longer than for Method 5a

TABLE VIII
Relative CPU Times for Runge–Kutta Methods Using Automatic Stepsize Control with the Local Error Tolerances $T_m^{\max} = 10^{-4}$

Log(k)	Method 1b	Method 2b	Method 3b	Method 4b	Method 5b	Method 6b	Method 7b	Method 8b	Method 9b	Method 10b
2.0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2.5	1.0000	1.0000	1.0000	0.9897	1.0000	1.0000	0.9982	1.0000	1.0000	0.9634
3.0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9982	1.1709	0.8864	1.0000
3.5	0.9909	1.0000	1.0000	0.9949	1.0000	1.0270	0.9964	1.0000	1.0000	0.9287
4.0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0541	0.9909	1.0000	1.0000	1.0000
4.5	0.9909	1.0000	1.0000	0.9846	1.0000	2.0865	0.9781	1.1709	1.0000	1.0000
5.0	0.9818	0.9892	1.0000	0.9743	0.9268	3.7189	0.9162	1.0000	1.0000	1.1060
5.5	1.4273	1.1075	1.0155	1.4218	1.5707	6.4649	1.2878	1.0000	1.0000	1.3931
6.0	2.5091	2.4946	1.8993	2.6728	2.7610	10.5622	2.3953	1.1709	1.0000	1.7148
6.5	3.9000	4.1075	3.2791	3.5412	3.8829	15.4973	3.5155	1.0000	1.0000	1.2852
7.0	5.6091	7.2796	5.9922	4.1687	6.0244	20.0811	4.3461	1.0000	0.8864	1.2139
7.5	11.7182	17.0860	15.0000	4.6111	10.1767	25.5568	4.9763	1.0000	1.0000	1.2139
8.0	29.4546	42.2043	40.9070	4.9938	17.9366	32.7568	5.7395	1.0000	1.0000	1.2139
8.5	86.2000	133.1720	123.0620	5.1595	47.7561	68.3838	7.0893	1.0000	0.8864	1.2139
9.0	183.6818	277.1720	313.6434	5.5504	140.8878	187.4486	4.2222	1.1624	1.1080	1.2505

Note. The relative computation time is defined by $t_{rel}(k) = t_{abs}(k)t_{abs}^{-1}(k = 10^2\,\text{Hz})$, where $t_{abs}(k)$ is the absolute computation time.

because it involves more computation per step. However, the difference is small indicating that Method 6a (6b) is superior to Method 5a (5b) when high accuracy is required. The computation times for Method 4a (4b) are longer than for Method 6a (6b). Because these methods have similar accuracy properties, the difference derives from the fact that Method 4a (4b) involves more computation per step. This seems to suggest that Method 6a (6b) is better than Method 4a (4b) in the slow and intermediate motion regimes. However, the inferior stability characteristics of Method 6a (6b) show that Method 4a (4b) is the preferred method.

The efficiency of the Runge–Kutta methods may be expressed by the power functions listed in Tables IX and X. The results show that the most important effect of decreasing the stepsize or error tolerance is to increase the premultiplying factor while the exponent is almost constant. Another feature is that the exponent is smaller for the methods that implement fixed stepsizes compared with the methods that use automatic stepsize control.

TABLE IX
Power Function Representations of Absolute CPU Times for Runge–Kutta Methods

	$\delta t = 0.4\,\mu$s	$\delta t = 0.2\,\mu$s	$\delta t = 0.1\,\mu$s
Method 1a	$1.4371 \cdot 10^{-3}$, 1.7091	$2.5771 \cdot 10^{-3}$, 1.7431	$4.8756 \cdot 10^{-3}$, 1.7533
Method 2a	$1.8177 \cdot 10^{-3}$, 1.7179	$3.3177 \cdot 10^{-3}$, 1.7438	$6.3395 \cdot 10^{-3}$, 1.7564
Method 3a	$2.6982 \cdot 10^{-3}$, 1.7373	$5.2179 \cdot 10^{-3}$, 1.7414	$9.8132 \cdot 10^{-3}$, 1.7641
Method 4a	$8.2423 \cdot 10^{-3}$, 1.7973	$1.6692 \cdot 10^{-2}$, 1.7904	$3.2900 \cdot 10^{-2}$, 1.7953
Method 5a	$3.4818 \cdot 10^{-3}$, 1.7645	$6.5843 \cdot 10^{-3}$, 1.7819	$1.2594 \cdot 10^{-2}$, 1.7982
Method 6a	$3.6791 \cdot 10^{-3}$, 1.7978	$7.1271 \cdot 10^{-3}$, 1.8037	$1.3656 \cdot 10^{-2}$, 1.8203
Method 7a	$7.9794 \cdot 10^{-3}$, 1.8230	$1.5594 \cdot 10^{-2}$, 1.8299	$2.9609 \cdot 10^{-2}$, 1.8527
Method 8a	$2.9470 \cdot 10^{-3}$, 1.4001	$4.5951 \cdot 10^{-3}$, 1.5031	$8.9816 \cdot 10^{-3}$, 1.5125
Method 9a	$3.9252 \cdot 10^{-3}$, 1.3647	$7.9556 \cdot 10^{-3}$, 1.3345	$1.6410 \cdot 10^{-2}$, 1.2844
Method 10a	$1.6317 \cdot 10^{-2}$, 1.1723	$3.2261 \cdot 10^{-2}$, 1.1712	$6.3312 \cdot 10^{-2}$, 1.1765

Note. The representations are defined by specifying the parameters (a, b) such that $t = aN^b$.

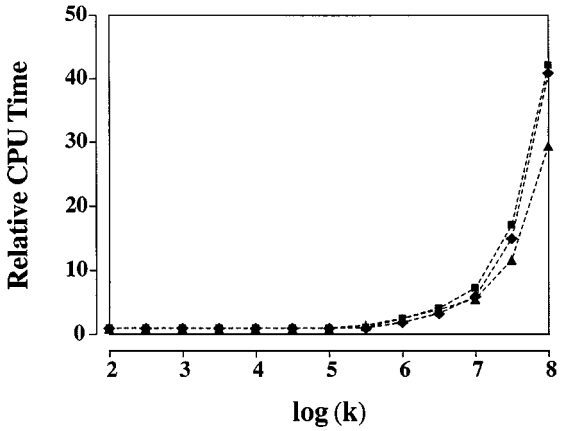

FIG. 8. Relative computation time $t_{rel}(k) = t_{abs}(k)t_{abs}^{-1}(k = 10^2\,\text{Hz})$ as function of the rate constant k for explicit Runge–Kutta methods. The methods used automatic stepsize control with the local error tolerances $T_m^{\max} = 10^{-4}$. The figure includes the classical explicit (3,3) Runge–Kutta method (▲), the classical explicit (4,4) Runge–Kutta method (■), and the Butcher explicit (5,6) Runge–Kutta method (◆).

The premultiplying factor tends to be smaller for the methods that are based on automatic stepsize control than for the methods that use fixed stepsizes. For the methods that use either fixed stepsizes or automatic stepsize control, the exponent is almost invariant while the premultiplying factor depends on the scheme. The results suggest that automatic stepsize control is the preferred method of operation. Although the execution time may be longer than for fixed stepsizes, the accuracy is expected to be higher especially for systems of large dimension.

The computation time in the slow and intermediate motion regimes is longer for the fourth-order Methods 4a (4b) and 6a (6b) compared with Method 2a (2b). Moreover, it is realized that the computation time for the third-order Method 5a (5b) is longer than for

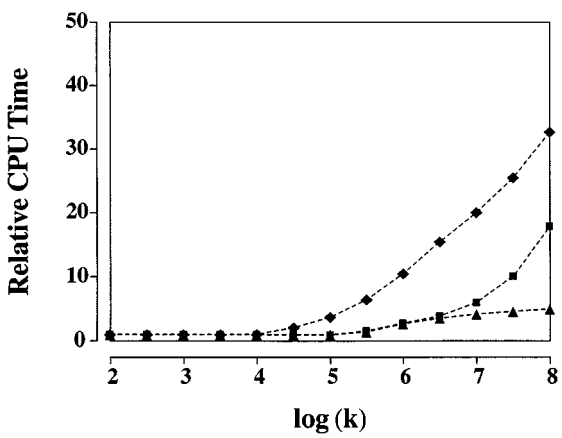

FIG. 9. Relative computation time $t_{rel}(k) = t_{abs}(k)t_{abs}^{-1}(k = 10^2\,\text{Hz})$ as function of the rate constant k for semi-implicit and implicit Runge–Kutta methods. The methods used automatic stepsize control with the local error tolerances $T_m^{\max} = 10^{-4}$. The figure includes the implicit (4,2) Runge–Kutta method of Gauss type (▲), the semi-implicit (3,2) Runge–Kutta method of first Radau type (■), and the semi-implicit (4,3) Runge–Kutta method of Lobatto type (◆).

TABLE X
Power Function Representations of Absolute CPU Times for Runge–Kutta Methods

	$T_m^{\max} = 10^{-3}$	$T_m^{\max} = 10^{-4}$	$T_m^{\max} = 10^{-5}$
Method 1b	$1.4809 \cdot 10^{-3}$, 2.0102	$2.0420 \cdot 10^{-3}$, 2.1321	$3.5653 \cdot 10^{-3}$, 2.0738
Method 2b	$2.0379 \cdot 10^{-3}$, 1.7271	$1.8103 \cdot 10^{-3}$, 2.0127	$2.3016 \cdot 10^{-3}$, 2.0827
Method 3b	$3.0635 \cdot 10^{-3}$, 1.7503	$3.0680 \cdot 10^{-3}$, 1.7499	$3.0960 \cdot 10^{-3}$, 1.7457
Method 4b	$9.2289 \cdot 10^{-3}$, 2.1187	$2.0595 \cdot 10^{-2}$, 2.0935	$3.7232 \cdot 10^{-2}$, 2.0979
Method 5b	$3.7912 \cdot 10^{-3}$, 1.8797	$4.0187 \cdot 10^{-3}$, 2.1200	$6.9972 \cdot 10^{-3}$, 2.1376
Method 6b	$4.6162 \cdot 10^{-3}$, 1.7765	$4.4979 \cdot 10^{-3}$, 1.8070	$3.8898 \cdot 10^{-3}$, 2.0948
Method 7b	$8.5315 \cdot 10^{-3}$, 1.9294	$9.1885 \cdot 10^{-3}$, 2.1962	$2.0279 \cdot 10^{-2}$, 2.0812
Method 8b	$3.0351 \cdot 10^{-3}$, 1.6730	$2.7702 \cdot 10^{-3}$, 2.0068	$4.6311 \cdot 10^{-3}$, 1.9894
Method 9b	$5.2159 \cdot 10^{-3}$, 1.3202	$4.3467 \cdot 10^{-3}$, 1.7655	$3.5139 \cdot 10^{-3}$, 2.1303
Method 10b	$1.9853 \cdot 10^{-2}$, 1.1676	$1.9471 \cdot 10^{-2}$, 1.1789	$3.0152 \cdot 10^{-2}$, 1.5539

Note. The representations are defined by specifying the parameters (a, b) such that $t = aN^b$.

Method 1a (1b). This is because Methods 4a (4b) through 6a (6b) are semi-implicit or implicit while Methods 1a (1b) through 3a (3b) are explicit. For most systems, the solution of the implicit equations is only partially compensated for by the smaller number of stages and the higher order of the semi-implicit or implicit schemes. Because the methods are of comparable accuracy and sufficiently stable in the slow and intermediate motion regimes, this is measured by the computation times, which are significantly longer for semi-implicit or implicit methods than for explicit methods. This implies that Methods 1a (1b) through 3a (3b) are superior to Methods 4a (4b) through 6a (6b) in the slow and intermediate motion regimes. Another observation is that the explicit Runge–Kutta methods in the slow and intermediate motion regimes are more efficient than the eigenvalue method by a factor between two and five. The power functions show that the efficiency increases with the dimension of the system and approaches an order of magnitude for large systems. Because the computation times involved in MAS NMR calculations may be very long, this is an important result. An additional advantage is that the solution is more accurate because automatic stepsize control is impossible for the eigenvalue method.

6.3. *Stiffly A-Stable Runge–Kutta Methods*

The stability problem is avoided in the case of an A-stable Runge–Kutta method. However, for a reasonable stepsize, the solution components defined by the eigenvalues with large negative real parts may be approximated inaccurately initially when they are nonnegligible. These inaccurate components may reduce the overall accuracy. Thus, although the components of the solution defined by the eigenvalues with large negative real parts may be of no particular interest, the condition of accuracy may enforce a small stepsize over the entire range of integration. The result is that the execution time may be prohibitive for an A-stable Runge–Kutta method. For these systems we therefore require that the characteristic roots closely approximate not only the solution components defined by the eigenvalues with small negative real parts but also the solution components defined by the eigenvalues with large negative real parts. A method with this property is characterized as stiffly A-stable [23, 24]. The literature presents relatively few examples of stiffly A-stable Runge–Kutta methods, and many schemes are based on extended formulas [43]. In order to improve the

efficiency, we have designed a set of stiffly A-stable Runge–Kutta methods by restricting the characteristic roots to be superdiagonal Padé approximants [23, 24]. An example is the stiffly A-stable (3,2) Runge–Kutta method of Padé type

$$\begin{aligned}\hat{\mathbf{\Phi}}(t_k, \hat{\mathbf{y}}(t_k), \delta t_k) &= \frac{1}{2}[\mathbf{g}_1 + \mathbf{g}_2],\\ \mathbf{g}_1 &= \mathbf{f}\left(t_k + \frac{1}{3}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{6}\delta t_k[3\mathbf{g}_1 - \mathbf{g}_2]\right),\\ \mathbf{g}_2 &= \mathbf{f}\left(t_k + \frac{2}{3}\delta t_k, \hat{\mathbf{y}}(t_k) + \frac{1}{6}\delta t_k[3\mathbf{g}_1 + \mathbf{g}_2]\right),\end{aligned} \tag{6.4}$$

which is denoted Method 7a for fixed stepsizes and Method 7b in the case of automatic stepsize control. It is noted that stiffly A-stable methods generally have a low order for the number of stages because of the restrictions on the characteristic roots. The results of numerical experiments with this method are listed in Tables II through VIII and illustrated in Fig. 7.

Because Method 4a (4b) is A-stable but not stiffly A-stable the computation time is an increasing function of the rate constants in the fast motion regime. However, it is noted that Method 4a (4b) is not as sensitive to the stiffness as Methods 5a (5b) and 6a (6b) because of the better stability. The effect of the stiffness is less pronounced for Method 7a (7b) and the results (Table VIII) indicate that it decreases for large rate constants. Since the methods involve the same number of implicit stages, the execution time for Method 4a is similar to Method 7a. However, the computation time for Method 7b is shorter than for Method 4b. This demonstrates that the form of the characteristic roots may be more important than the order in determining the efficiency of the method. The conclusion is that Method 7a (7b) is the best implicit scheme in the fast motion regime.

6.4. *Modified Runge–Kutta Methods*

Another approach to solving the stochastic Liouville–von Neumann equation is to rewrite the system in a form with a vanishing stiffness ratio. This has the advantage that one may use explicit Runge–Kutta methods to obtain a solution. In the case of the stochastic Liouville–von Neumann equation, the form of the coefficient matrix makes it possible to remove the stiffness completely. This is realized by considering the equation

$$\frac{\partial}{\partial t}|\mathbf{y}(t)\rangle = \mathbf{A}(t)|\mathbf{y}(t)\rangle \tag{6.4}$$

and rewriting the coefficient matrix in the form

$$\mathbf{A}(t) = \mathbf{S} + \mathbf{R}(t), \tag{6.5}$$

where $\mathbf{S}$ is defined by the stochastic operator and $\mathbf{R}(t)$ by the Hamiltonian. Because the approach does not depend on whether $\mathbf{S}$ and $\mathbf{R}(t)$ commute, this form can be obtained for any stochastic operator and any Hamiltonian and is particularly useful when $\mathbf{S}$ is the source of the stiffness. In order to eliminate the stiffness we consider the transformation

$$|\mathbf{z}(t)\rangle = \exp(-\mathbf{S}t)|\mathbf{y}(t)\rangle, \tag{6.6}$$

which when differentiated leads to

$$\frac{\partial}{\partial t}|\mathbf{z}(t)\rangle = \exp(-\mathbf{S}t)\mathbf{R}(t)\exp(\mathbf{S}t)|\mathbf{z}(t)\rangle. \tag{6.7}$$

It is noted that the coefficient matrix for this system has the same eigenvalue structure as $\mathbf{R}(t)$ which is determined by the Hamiltonian. This implies that the transformed system has a vanishing stiffness ratio. For a stiff system it is difficult to determine $\exp(-\mathbf{S}t)$. However, it is trivial to evaluate $\exp(\mathbf{S}t)$ using the eigenvalue method. This is sufficient to obtain a Runge–Kutta solution and usually one does not need to evaluate $\exp(-\mathbf{S}t)$. More specifically, we consider the Runge–Kutta equations

$$|\hat{\mathbf{z}}(t_{k+1})\rangle = |\hat{\mathbf{z}}(t_k)\rangle + \delta t_k \sum_{r=1}^{q} w_r |\mathbf{g}_r\rangle, \tag{6.8}$$

where

$$|\mathbf{g}_r\rangle = \exp(-[t_k+\alpha_r\delta t_k]\mathbf{S})\mathbf{R}(t_k+\alpha_r\delta t_k)\exp([t_k+\alpha_r\delta t_k]\mathbf{S})\left[|\hat{\mathbf{z}}(t_k)\rangle + \delta t_k \sum_{s=1}^{q}\beta_{rs}|\mathbf{g}_s\rangle\right] \tag{6.9}$$

includes the coefficient matrix of the transformed system. The equations are rewritten in the form

$$\begin{aligned}|\hat{\mathbf{y}}(t_{k+1})\rangle = {} & \exp(\delta t_k\mathbf{S})|\hat{\mathbf{y}}(t_k)\rangle + \delta t_k \sum_{r=1}^{q} w_r \exp([1-\alpha_r]\delta t_k\mathbf{S})\mathbf{R}(t_k+\alpha_r\delta t_k) \\ & \times [\exp(\alpha_r\delta t_k\mathbf{S})|\hat{\mathbf{y}}(t_k)\rangle + \sum_{s=1}^{q}\beta_{rs}\exp([\alpha_r-\alpha_s]\delta t_k\mathbf{S})|\mathbf{h}_s\rangle],\end{aligned} \tag{6.10}$$

where

$$|\mathbf{h}_r\rangle = \mathbf{R}(t_k+\alpha_r\delta t_k)[\exp(\alpha_r\delta t_k\mathbf{S})|\hat{\mathbf{y}}(t_k)\rangle + \delta t_k\sum_{s=1}^{q}\beta_{rs}|\mathbf{h}_s\rangle] \tag{6.11}$$

has been introduced for simplicity. These equations involve matrix exponentials that may be readily evaluated using the eigenvalue method.

Because the transformation eliminates the stiffness, the approach is likely to be successful for explicit Runge–Kutta methods. In these cases, the modified Runge–Kutta equations may be solved explicitly leading to a relatively high efficiency. The evaluation of the matrix exponentials reduces the performance of the methods but in many cases the computational overhead is acceptable. For A-stable semi-implicit or implicit methods, the stepsizes may be larger because the stiff components have been eliminated. However, the difficulty in solving the Runge–Kutta equations suggests that they are unlikely to be efficient. The modified classical explicit (3,3) Runge–Kutta method is Method 8a (8b), the modified classical explicit (4,4) Runge–Kutta method is Method 9a (9b), and the modified Butcher explicit (5,6) Runge–Kutta method is Method 10a (10b). The results of numerical experiments with these methods are listed in Tables II through VIII and illustrated graphically in Fig. 10.

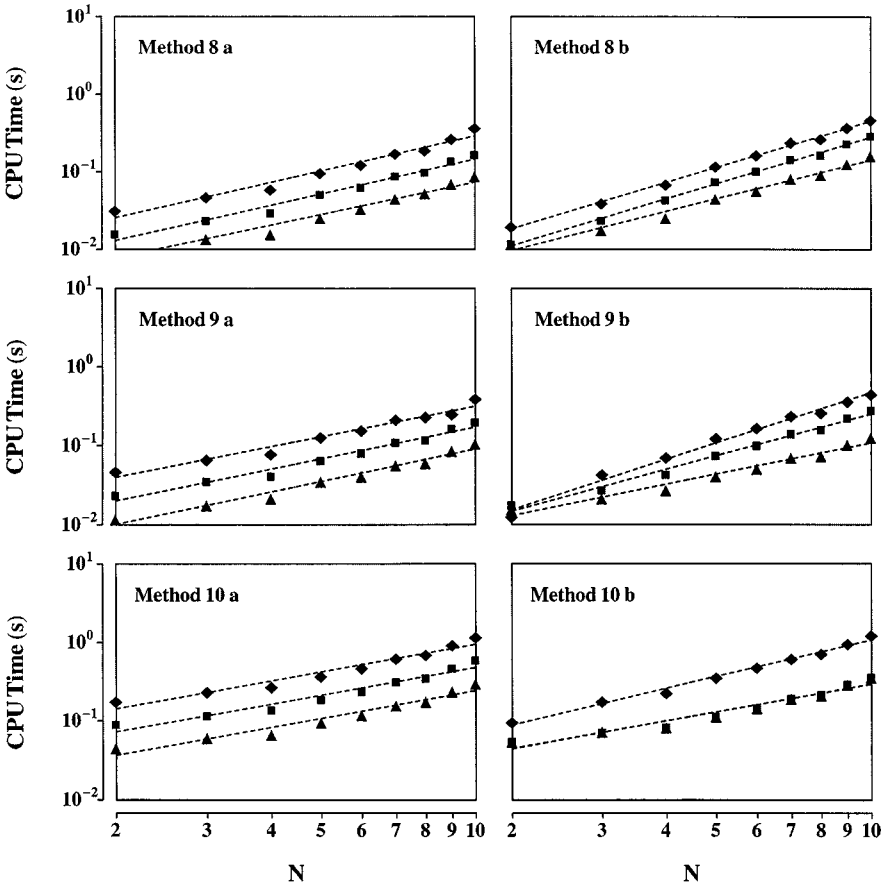

FIG. 10. Absolute computation time as function of the number of motional states N for modified explicit Runge–Kutta methods. The methods used fixed stepsizes (first column) given by $\delta t = 0.1\,\mu$s (◆), $\delta t = 0.2\,\mu$s (■), and $\delta t = 0.1\,\mu$s (▲) and automatic stepsize control (second column) with the local error tolerances $T_m^{\max} = 10^{-5}$ (◆), $T_m^{\max} = 10^{-4}$ (■), and $T_m^{\max} = 10^{-3}$ (▲). The figure includes the modified classical explicit (3,3) Runge–Kutta method (first row), the modified classical explicit (4,4) Runge–Kutta method (second row), and the modified Butcher explicit (5,6) Runge–Kutta method (third row).

The numerical results reveal that, in the slow and intermediate motion regimes, the modified explicit Runge–Kutta methods are less efficient than the explicit schemes. This is because the amount of computation involved in the modified methods is larger for each step of calculation. Because of the computational overhead, it is expected that modified explicit Runge–Kutta methods will always be less efficient than explicit schemes in the slow and intermediate motion regimes. However, it is seen that Methods 8a (8b) and 9a (9b) are only slightly less efficient than the explicit methods. Moreover, the modified explicit Runge–Kutta methods are orders of magnitude more efficient than the explicit, semi-implicit, and implicit schemes in the fast motion regime. The explicit, semi-implicit, and implicit methods are inefficient because of the stiffness while the modified explicit schemes approximate the stiff components accurately. This is reflected in the computation times, which increase with the rate constant for explicit, semi-implicit, and implicit methods, but are almost constant for the modified explicit schemes. Although the stiffly A-stable Runge–Kutta methods are useful for large stiffness ratios they are less efficient than the modified explicit Runge–Kutta

methods. An example where the modified methods would be valuable is in the calculation of motional effects for urea and thiourea inclusion compounds that usually exhibit fast motion [14].

7. SUMMARY

The MAS NMR experiment has become one of the most successful methods for obtaining information about molecular structure and motion in solid materials. The most fundamental description of the MAS NMR experiment is based on the stochastic Liouville–von Neumann equation. In this paper we have implemented a Lie algebra formalism to rewrite the stochastic Liouville–von Neumann equation in the form of a linear homogeneous system of coupled first-order differential equations. The approach includes several computationally important features and may be implemented for MAS NMR experiments on any nuclear spin system.

In the case of the MAS NMR experiment, it is impossible to obtain an explicit solution to the stochastic Liouville–von Neumann equation. As an alternative we have used Runge–Kutta methods to approximate the solution. These have the advantages of simplicity and relatively high efficiency and may be implemented with automatic stepsize control. To provide an explicit example we have applied the Runge–Kutta methods to the deuteron MAS NMR experiment, which is of widespread interest as one of the most important methods for studying molecular motion in solids.

In the case of the deuteron MAS NMR experiment, we have evaluated the performance of different Runge–Kutta methods including explicit, semi-implicit, and implicit schemes. The explicit methods examined in this paper are the classical explicit (3,3) Runge–Kutta method, the classical explicit (4,4) Runge–Kutta method, and the Butcher explicit (5,6) Runge–Kutta method. The semi-implicit and implicit methods are the implicit (4,2) Runge–Kutta method of Gauss type, the semi-implicit (3,2) Runge–Kutta method of first Radau type, and the semi-implicit (4,3) Runge–Kutta method of Lobatto type. The results of the numerical experiments have shown that no method performs better than any other for every set of rate constants. It is therefore necessary to learn about each method in order to decide which one to use for each set of rate constants. The reason is that the stochastic Liouville–von Neumann equation is stiff with a stiffness ratio that increases with the rate constants.

The results have shown that explicit Runge–Kutta methods are superior to other schemes in the slow and intermediate motion regimes. In these regimes the stiffness is sufficiently small that the computation is determined solely by the condition of accuracy. Although semi-implicit and implicit Runge–Kutta methods require fewer derivative evaluations, the solution of the simultaneous equations more than compensates for this. The performance of the methods was evaluated in the case of fixed stepsizes and automatic stepsize control. It is found that automatic stepsize control is the preferred method of operation. The difference in computation time for fixed stepsizes and automatic stepsize control is sufficiently small to justify the improved accuracy especially for systems of high dimension. Because of the increased accuracy, the classical explicit (4,4) Runge–Kutta method is found to be more efficient than the classical explicit (3,3) Runge–Kutta method. However, the Butcher explicit (5,6) Runge–Kutta method is less efficient than the classical explicit (4,4) Runge–Kutta method. The reason is that the increased number of derivative evaluations compensates for the increased accuracy. However, because the computation times are almost equal, the

classical explicit (4,4) Runge–Kutta method is best for calculations of low accuracy, while the Butcher explicit (5,6) Runge–Kutta method is to be preferred for calculations of high accuracy.

In the fast motion regime, the explicit Runge–Kutta methods are very inefficient because of the high stiffness of the stochastic Liouville–von Neumann equation. In this regime the stability of the Runge–Kutta methods becomes important. It is noted that there is no advantage of implicit methods when the sole criterion is accuracy. However, the implicit methods have desirable stability properties, which in this case is the primary consideration. The results have shown that the semi-implicit (3,2) Runge–Kutta method of first Radau type and the semi-implicit (4,3) Runge–Kutta method of Lobatto type are inefficient in the fast motion regimes. The reason is that these methods are conditionally stable. It is found that the semi-implicit (3,2) Runge–Kutta method of first Radau type is more efficient than the semi-implicit (4,3) Runge–Kutta method of Lobatto type because it has better stability properties. The results have shown that the implicit (4,2) Runge–Kutta method of Gauss type is better than the semi-implicit (3,2) Runge–Kutta method of first Radau type and the semi-implicit (4,3) Runge–Kutta method of Lobatto type in the fast motion regime. The implicit (4,2) Runge–Kutta method of Gauss type is A-stable and therefore sufficiently stable. However, the method is not stiffly A-stable, and the condition of accuracy makes the method relatively inefficient for high rate constants.

It is noted that there are two alternatives to solving the stochastic Liouville–von Neumann equation in the fast motion regime. The most obvious approach is to implement stiffly A-stable semi-implicit or implicit Runge–Kutta methods. These approximate the stiff components accurately and are less sensitive to the stiffness. We have designed stiffly A-stable Runge–Kutta methods by restricting the characteristic roots to be superdiagonal Padé approximants. These are shown to be the best implicit schemes in the fast motion regime. An alternative is to rewrite the stochastic Liouville–von Neumann equation in a form with a small or vanishing stiffness ratio. In this paper, we have introduced a transformation that eliminates the stiffness completely. This has been combined with explicit schemes to define a set of modified explicit Runge–Kutta methods. We have shown that these are the most efficient schemes in the fast motion regime. The modified explicit (3,3) and (4,4) Runge–Kutta methods are only slightly less efficient than the explicit schemes in the slow and intermediate regimes. This demonstrates that the modified explicit (3,3) and (4,4) Runge–Kutta are useful for solving the stochastic Liouville–von Neumann equation for all motional regimes.

The results of this research have shown that the Runge–Kutta formalism is a useful and efficient approach to MAS NMR lineshape calculations. The formalism provides a set of integration methods that improve the accuracy of the calculations and that lead to significantly shorter computation times. The experiments have shown that the Runge–Kutta methods improve the computational efficiency by a factor between two and five compared with the eigenvalue method. It is shown that the efficiency increases with the dimension of the system, and for large systems the improvement may be more than an order of magnitude. Because of the long computation times involved in MAS NMR calculations, this is a significant result. The possibility of implementing automatic stepsize control is another important feature, which makes the approach more accurate than the eigenvalue method. The modified Runge–Kutta methods developed in this paper extends the formalism to calculations involving fast molecular motion and stiff systems of differential equations.

ACKNOWLEDGMENTS

This research was supported by the Carlsberg Foundation (J.No. 960105/20-1220 and 970005/20-1283, JHK) and the American National Science Foundation (CHE 9701014, GLH, and RLV).

REFERENCES

1. U. Haeberlen, *High Resolution NMR in Solids, Selective Averaging* (Academic Press, New York, 1976).
2. M. Mehring, *Principles of High Resolution NMR on Solids* (Springer-Verlag, New York, 1983).
3. C. A. Fyfe, *Solid State NMR for Chemists* (C.F.C. Press, Guelph, 1983).
4. E. R. Andrew, A. Bradbury, and R. G. Eades, Nuclear magnetic resonance spectra from a crystal rotated at high speed, *Nature* **182**, 1659 (1958).
5. I. J. Lowe, Free induction decays of rotating solids, *Phys. Rev. Lett.* **2**, 285 (1959).
6. M. M. Maricq and J. S. Waugh, NMR in rotating solids, *J. Chem. Phys.* **70**, 3300 (1979).
7. A. Samoson, E. Kundla, and E. Lippmaa, High resolution MAS NMR of quadrupolar nuclei in powders, *J. Magn. Reson.* **49**, 350 (1982).
8. N. J. Clayden, Computer simulations of ^{2}H MAS NMR spinning sideband spectra, *Chem. Phys. Lett.* **131**, 517 (1986).
9. J. H. Kristensen, H. Bildsøe, H. J. Jakobsen, and N. C. Nielsen, Deuterium quadrupole couplings from least-squares computer simulations of ^{2}H MAS NMR spectra, *J. Magn. Reson*. **92**, 443 (1991).
10. A. Schmidt, S. O. Smith, D. P. Raleigh, J. E. Roberts, R. G. Griffin, and S. Vega, Chemical exchange effects in the NMR spectra of rotating solids, *J. Chem. Phys.* **85**, 4248 (1986).
11. A. Schmidt and S. Vega, NMR line shape analysis for two-site exchange in rotating solids, *J. Chem. Phys.* **87**, 6895 (1987).
12. J. H. Kristensen, H. Bildsøe, H. J. Jakobsen, and N. C. Nielsen, Theory and simulations of molecular dynamics in ^{2}H MAS NMR, *J. Magn. Reson.* **100**, 437 (1992).
13. M. J. Duer and M. H. Levitt, Time-domain calculation of chemical exchange effects in the NMR spectra of rotating solids, *Solid State Nucl. Magn. Reson.* **1**, 211 (1992).
14. J. H. Kristensen, G. L. Hoatson, and R. L. Vold, Investigation of multiaxial molecular dynamics by ^{2}H MAS NMR spectroscopy, *Solid State Nucl. Magn. Reson.* **13**, 1 (1998).
15. R. Kubo, *Fluctuation, Relaxation, and Resonance in Magnetic Systems* (Oliver and Boyd, Edinburgh, 1962).
16. R. Kubo, Stochastic processes in chemical physics, *Adv. Chem. Phys.* **15**, 101 (1969).
17. A. Abragam, *The Principles of Nuclear Magnetism* (Oxford Univ. Press, Oxford, 1961).
18. D. W. Alderman, M. S. Solum, and D. M. Grant, Methods for analyzing spectroscopic line shapes. NMR solid powder patterns, *J. Chem. Phys.* **84**, 3717 (1986).
19. A. Ponti, Simulation of magnetic resonance static powder lineshapes. A quantitative assessment of spherical codes, *J. Magn. Reson.* **138**, 288 (1999).
20. C. W. Gear, The automatic integration of stiff ordinary differential equations, *Information Processing* **68**, 187 (1969).
21. R. A. Willoughby, *Stiff Differential Equations* (Plenum, New York, 1974).
22. W. H. Enright, T. E. Hull, and B. Lindberg, Comparing numerical methods for stiff systems of ODEs, *BIT* **15**, 10 (1975).
23. C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations* (Prentice Hall, New Jersey, 1971).
24. L. Lapidus and J. H. Seinfeld, *Numerical Solution of Ordinary Differential Equations* (Academic Press, New York, 1971).
25. J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis* (Springer-Verlag, New York, 1980).
26. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes, The Art of Scientific Computing* (Cambridge Univ. Press, Cambridge, 1989).

27. U. Fano, Description of states in quantum mechanics by density matrix and operator techniques, *Rev. Mod. Phys.* **29**, 74 (1957).
28. B. G. Weybourne, *Classical Groups for Physicists* (Wiley, New York, 1974).
29. R. Gilmore, *Lie Groups, Lie Algebras, and Some of Their Applications* (Wiley, New York, 1974).
30. J. H. Kristensen, H. Bildsøe, H. J. Jakobsen, and N. C. Nielsen, Application of Lie algebra to NMR spectroscopy, *Prog. Nucl. Magn. Reson. Spectrosc.* **34**, 1 (1999).
31. M. H. Levitt, Why do spinning sidebands have the same phase, *J. Magn. Reson.* **82**, 427 (1989).
32. A. Wokaun and R. R. Ernst, Selective excitation and detection in multilevel spin systems. Application of single transition operators, *J. Chem. Phys.* **67**, 1752 (1977).
33. B. C. Sanctuary, T. K. Halstead, and P. A. Osment, Multipole NMR. IV. Dynamics of single spin, *Mol. Phys.* **49**, 753 (1983).
34. G. J. Bowden and W. D. Hutchison, Tensor operator formalism for multiple-quantum NMR. 2. Spins $\frac{2}{3}$, 2, and $\frac{5}{2}$ and general I, *J. Magn. Reson.* **67**, 403 (1986).
35. J. C. Butcher, On the attainable order of Runge–Kutta methods, *Math. Comp.* **19**, 408 (1965).
36. J. C. Butcher, Coefficients for the study of Runge–Kutta integration processes, *J. Austral. Math. Soc.* **3**, 185 (1963).
37. J. C. Butcher, On Runge–Kutta processes of high order, *J. Austral. Math. Soc.* **4**, 179 (1964).
38. H. A. Luther, Further explicit fifth-order Runge–Kutta formulas, *SIAM Rev.* **8**, 374 (1966).
39. E. Fehlberg, Classical fifth-, sixth-, seventh-, and eighth-order Runge–Kutta formulas with stepsize control, *NASA Technical Report*, NASA TR R-287 (1968).
40. W. Kutta, Beitrag zur näherungsweisen integration totaler differentialgleichungen, *Zeit. Math. Physik* **46**, 435 (1901).
41. J. C. Butcher, Implicit Runge–Kutta processes, *Math. Comp.* **18**, 50 (1964).
42. J. C. Butcher, Integration processes based on Radau quadrature formulas, *Math. Comp.* **18**, 233 (1964).
43. P. Kaps and P. Rentrop, Generalized Runge–Kutta methods of order four with step size control for stiff ordinary differential equations, *Numer. Math.* **33**, 55 (1979).