BIOMATH
www.biomathforum.org/biomath/index.php/biomath
Biomath Forum

ORIGINAL ARTICLE

Efficient Implicit Runge-Kutta Methods for Fast-Responding Ligand-Gated Neuroreceptor Kinetic Models

Edward T. Dougherty
Department of Mathematics
Rowan University
Glassboro, NJ, USA
Email: doughertye@rowan.edu

Received: 11 May 2015, accepted: 31 December 2015, published: 19 January 2016

***Abstract*—Neurophysiological models of the brain typically utilize systems of ordinary differential equations to simulate single-cell electrodynamics. To accurately emulate neurological treatments and their physiological effects on neurodegenerative disease, models that incorporate biologically-inspired mechanisms, such as neurotransmitter signalling, are necessary. Additionally, applications that examine populations of neurons, such as multiscale models, can demand solving hundreds of millions of these systems at each simulation time step. Therefore, robust numerical solvers for biologically-inspired neuron models are vital. To address this requirement, we evaluate the numerical accuracy and computational efficiency of three L-stable implicit Runge-Kutta methods when solving kinetic models of the ligand-gated glutamate and γ-aminobutyric acid (GABA) neurotransmitter receptors. Efficient implementations of each numerical method are discussed, and numerous performance metrics including accuracy, simulation time steps, execution speeds, Jacobian calculations, and LU factorizations are evaluated to identify appropriate strategies for solving these models. Comparisons to popular explicit methods are presented and highlight the advantages of the implicit methods. In addition, we show a machine-code compiled implicit Runge-Kutta method implementation that possesses exceptional accuracy and superior computational efficiency.**

***Keywords*-implicit Runge-Kutta; neuroreceptor model; numerical stiffness; ODE simulation**

I. Introduction

Mathematical modeling and computational simulation provide an *in silico* environment for investigating cerebral electrophysiology and neurological therapies including neurostimulation. Traditionally, volume-conduction models have been used to emulate electrical potentials and currents within the head cavity. In particular, these models can reproduced electroencephalograph (EEG) surface potentials [1]–[3], and have been successful in predicting cerebral current density distributions from neurostimulation administrations [1], [4]–[7]. As these models become more refined, their utility in diagnosing, treating, and comprehending neurological disorders greatly increases.

Progress in field of computational neurology has motivated a migration towards models that incorporate cellular-level bioelectromagnetics. For example, bidomain based models have been used to simulate the effects of extracellular electrical current on cellular transmembrane voltage(s) [8]–[13]. In addition, multiscale models have reproduced EEG measurements originating from action potentials [14], [15], and have also demonstrated an ability to simulate the influence of transcranial electrical stimulation on neuronal depolarization [16].

These models typically utilize a system of ordinary differential equations (ODEs) to emulate cellular-level electrophysiology. While the computational expense of simulating a single cell is essentially negligible, this is not the case with large-scale applications that may include hundreds of millions of cells; in multiscale applications, solving this set of ODEs is the computational bottleneck [17]. In these applications, choosing an appropriate numerical solver and using efficient implementation approaches become paramount.

Alterations in neurotransmitter signalling is a hallmark of many neurodegenerative conditions and treatments. Parkinson's disease (PD), for example, which affects approximately one million individuals in the United States alone [18], culminates with pathological glutamate and γ-aminobutyric acid (GABA) binding activity throughout the basal ganglia-thalamocortical network [19], [20]. As a treatment for PD, deep brain stimulation (DBS) electrically stimulates areas of the basal ganglia, such as the subthalamatic nucleus (STN) [21], to restore normal glutamate and GABA synaptic concentrations [22]–[24]. Therefore, models that incorporate fundamental neurotransmitter-based signalling provide utility to the neurological research community.

Models of metabotropic and slow-responding ligand-gated receptors, such as the GABA_B and N-methyl-D-aspartate (NMDA) glutamate receptors, can be efficiently solved with explicit Runge-Kutta (ERK) methods [25]. On the contrary, fast-responding ionotropic receptors, such as the α-amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid receptor (AMPAR) and the GABA_A receptor (GABA_AR) result in models that are classified as stiff [26], which is an attribute of an ODE system that demands relatively small step sizes in portions of the numerical solution [27]. For these ODE systems, L-stable implicit Runge-Kutta (IRK) solvers with adaptive time-stepping are ideal given their exceptional stability properties [28].

In this paper, we examine L-stable IRK methods when solving models that represent the AMPA and GABA_A neuroreceptors. Three L-stable IRK methods that are highly effective at solving stiff ODE systems were selected and implemented with custom Matlab [29] programming. Features including adaptive step-sizing, embedded error estimation, error-based step size selection, and simplified Newton iterations are incorporated [30]. Numerical experiments were then used to identify the optimal maximum number of inner Newton iterations for each method. Then, for both the AMPAR and GABA_AR models, simulation time step results of each IRK method are compared to commonly used ERK methods. In addition, the numerical accuracy and computational efficiency of each IRK method is compared to one other, as well as the highly-popular fifth order, variable step size Dormand-Prince method. Finally, a C++ based IRK implementation demonstrates exceptionally accurate and expedient performances, showcasing its potential to support large-scale multi-cellular brain simulations.

II. Materials and Methods

A. Neuroreceptor models

1) AMPA: Glutamate is the single most abundant neurotransmitter in the human brain [31]. It is produced by glutamatergic neurons, and is classified as excitatory in the sense that it predominately depolarizes post-synaptic neurons towards generating action potentials [32]. Given the large concentration of glutamate in the nervous system, alterations in its production are associated with many neurodegenerative diseases and treatments. In PD patients, for example, stimulating the STN with DBS causes a cascade of cellular effects within the basal-ganglia thalamocortical pathway through its afferent and efferent projections, including increased glutamate secretion to the globus pallidus external (GPe), globus pallidus internal (GPi), and substantia nigra pars reticulata (SNr) [23].

Ligand-gated AMPA receptors for glutamate are permeable to sodium and potassium, have a reversal potential of 0 mV, and possess fast channel opening rates. Therefore, these receptors produce fast excitatory post-synaptic currents [33]. Figure 1a displays the Markov kinetic binding model for the ligand-gated AMPAR that was utilized in this paper [34]. In this network, there is the unbound AMPAR form C_0, singly and doubly bound receptor forms C_1 and C_2, which can lead to desensitized states D_1 and D_2, respectively, and the open receptor form O [35]. In addition, variable T represents neurotransmitter concentration. Mass action kinetics gives the following system of ODEs for the AMPA neuroreceptor model:

$$\frac{dC_0}{dt} = -k_b C_0 T + C_1 k_{u1}, \tag{1a}$$

$$\frac{dC_1}{dt} = k_b C_0 T + k_{u2} C_2 + k_{ud} D_1 - k_{u1} C_1 - k_b C_1 T - k_d C_1, \tag{1b}$$

$$\frac{dC_2}{dt} = k_b C_1 T + k_{ud} D_2 + k_c O - k_{u2} C_2 - k_d C_2 - k_o C_2, \tag{1c}$$

$$\frac{dD_1}{dt} = k_d C_1 - k_{ud} D_1, \tag{1d}$$

$$\frac{dD_2}{dt} = k_d C_2 - k_{ud} D_2, \tag{1e}$$

$$\frac{dO}{dt} = C_2 k_o - k_c O, \tag{1f}$$

$$\frac{dT}{dt} = -k_b C_0 T + k_{u1} C_1 - k_b C_1 T + k_{u2} C_2. \tag{1g}$$

State transition rates were assigned as follows: $k_b = 1.3 \text{ x } 10^7$, $k_o = 2.7 \text{ x } 10^3$, $k_c = 200$, $k_{u1} = 5.9$, $k_{u2} = 8.6 \text{ x } 10^4$, $k_d = 900$, and $k_{ud} = 64$, each with units [1/sec]. Initial concentrations of C_1, C_2, D_1, D_2, and O were set to 0 M [33], and initial values for C_0 and T were computed from a nonlinear least squares fit of the model to the whole cell recording data in Destexhe et al. [35].

2) GABA: GABA is the most abundant inhibitory neurotransmitter in the human brain [36]. Like glutamate, GABA concentrations are altered by neurological disease and treatment. In STN DBS, for example, increased glutamate to the GPe increases GABA secretion to the GPi and SNr, resulting in greater GABA neuroreceptor binding in these regions [24].

There are two main categories of GABA neuroreceptors. Metabotropic GABA$_B$ receptors are slow-responding due to the secondary messenger biochemical network cascade necessary for ion channel activation. On the contrary, ligand-gated GABA$_\mathrm{A}$ receptors are fast-responding due to their expedient ion channel opening rates. GABA$_\mathrm{A}$ receptors are selective to chlorine with a reversal potential of approximately -70 mV. In addition, this receptor has two bound forms that can both trigger channel activation [35].

Figure 1b displays the kinetic binding model for the GABA$_\mathrm{A}$ receptor that was utilized in this paper [26]. In this model, there is the unbound receptor form C_0, singly and doubly bound receptor forms C_1 and C_2, slow and fast desensitized states D_s and D_f, and singly open and doubly open receptor forms O_1 and O_2. This model incorporates the minimal forms needed to accurately reproduce GABA$_\mathrm{A}$R kinetics [37]. Mass action kinetics gives the following ODE system for the GABA$_\mathrm{A}$ neuroreceptor model:

$$\frac{dC_0}{dt} = -2k_b C_0 T + k_u C_1, \tag{2a}$$

$$\frac{dC_1}{dt} = 2k_b C_0 T - k_u C_1 + k_{uDs} D_s - k_{Ds} C_1 + 2k_u C_2 - k_b C_1 T + k_{c1} O_1 - k_{o1} C_1, \tag{2b}$$

$$\frac{dC_2}{dt} = k_b C_1 T - 2k_u C_2 + k_{c2} O_2 - k_{o2} C_2 + k_{uDf} D_f - k_{Df} C_2, \tag{2c}$$

$$\frac{dD_s}{dt} = k_{fs} D_f - k_{sf} D_s T + k_{Ds} C_1 - k_{uDs} D_s, \tag{2d}$$

$$\frac{dD_f}{dt} = k_{sf} D_s T - k_{fs} D_f + k_{Df} C_2 - k_{uDf} D_f, \tag{2e}$$

$$\begin{tikzcd}
C_0 \arrow[r, shift left, "k_bT"] & C_1 \arrow[l, shift left, "k_{u1}"] \arrow[r, shift left, "k_bT"] \arrow[d, shift right, "k_d"'] & C_2 \arrow[l, shift left, "k_{u2}"] \arrow[r, shift left, "k_o"] \arrow[d, shift right, "k_d"'] & O \arrow[l, shift left, "k_c"] \\
 & D_1 \arrow[u, shift right, "k_{ud}"'] & D_2 \arrow[u, shift right, "k_{ud}"'] &
\end{tikzcd}$$

(a) AMPA receptor

$$\begin{tikzcd}
 & D_s \arrow[r, shift left, "k_{sf}T"] \arrow[d, shift left, "k_{uDs}"] & D_f \arrow[l, shift left, "k_{fs}"] \arrow[d, shift left, "k_{uDf}"] \\
C_0 \arrow[r, shift left, "2k_bT"] & C_1 \arrow[l, shift left, "k_u"] \arrow[r, shift left, "k_bT"] \arrow[u, shift left, "k_{Ds}"] \arrow[d, shift right, "k_{o1}"'] & C_2 \arrow[l, shift left, "2k_u"] \arrow[u, shift left, "k_{Df}"] \arrow[d, shift right, "k_{o2}"'] \\
 & O_1 \arrow[u, shift right, "k_{c1}"'] & O_2 \arrow[u, shift right, "k_{c2}"']
\end{tikzcd}$$

(b) GABA receptor

Fig. 1: Kinetic models for ligand-gated neuroreceptors.

$$\frac{dO_1}{dt} = k_{o1}C_1 - k_{c1}O_1, \tag{2f}$$

$$\frac{dO_2}{dt} = k_{o2}C_2 - k_{c2}O_2, \tag{2g}$$

$$\begin{aligned}\frac{dT}{dt} &= k_uC_1 - 2k_bC_0T + 2k_uC_2 - k_bC_1T \\ &+ k_{fs}D_f - k_{sf}D_sT.\end{aligned} \tag{2h}$$

Transition rates for the GABA$_\mathrm{A}$R ODE system were assigned as follows: $k_b = 5$ x 10^6, $k_u = 131$, $k_{uDs} = 0.2$, $k_{Ds} = 13$, $k_{c1} = 1100$, $k_{o1} = 200$, $k_{c2} = 142$, $k_{o2} = 2500$, $k_{uDf} = 25$, $k_{Df} = 1250$, $k_{fs} = 0.01$, and $k_{sf} = 2$, each with units [1/sec]. Initial values of C_1, C_2, D_s, D_f, O_1, and O_2 were set 0 M, and C_0 and T were assigned the values 1 x 10^{-6} M and 4096 x 10^{-6} M, respectively [26].

B. Stiff ordinary differential equations

The stiffness ratio is defined as

$$L = \frac{\max|Re(\lambda_i)|}{\min|Re(\lambda_i)|},$$

where λ_i is the i_{th} eigenvalue of the local Jacobian matrix [38], given by

$$J_{ij} = \frac{\partial f_i(t,\bar{y})}{\partial y_j}.$$

A general non-linear ODE system is stiff when $L \gg 1$. For each neuroreceptor model, we estimated the eigenvalues numerically; a local Jacobian matrix is computed at each simulation time step using finite differences, and then its eigenvalues are computed using Matlab's `eig` function [39]. For the AMPAR model $L = 1.6$ x 10^{11}, and for the GABA$_\mathrm{A}$R model $L = 3.5$ x 10^{11}. Thus, both of these systems are classified as stiff.

C. Implicit Runge-Kutta methods

Runge-Kutta methods are a family of numerical integrators that solve ODE systems with trial steps within the time step. These methods can be expressed with the following formulas:

$$\bar{Z}_i = h\sum_{j=1}^{s} a_{ij}\bar{F}(t_n + c_jh, \bar{y}_n + \bar{Z}_j), \quad i = 1, ..., s \tag{3a}$$

$$\bar{y}_{n+1} = \bar{y}_n + h\sum_{j=1}^{s} b_j\bar{F}(t_n + c_jh, \bar{y}_n + \bar{Z}_j), \tag{3b}$$

where $\bar{y}_n$ is the current solution at time t_n, h is the current time step, $[a_{ij}]$ is the Runge-Kutta matrix, $\bar{F}$ is the ODE system, $[c_j]$ represents inter-time trial step nodes, $[b_j]$ is the trial step solution weights, s is the number of stages, and $\bar{y}_{n+1}$ is the numerical solution at time t_{n+1} [28]. A Runge-Kutta method can be fully defined with a Butcher table, i.e. a specific $[a_{ij}]$, $[b_j]$, and $[c_j]$ [40].

L-stable IRK methods are highly effective at solving stiff ODE systems [30]; these methods have no step size constraint to maintain numerical stability and quickly converge [41]. Methods with second and third order accuracy were considered as these orders best match the numerical accuracy of fractional step algorithms typically employed with partial differential equation based multiscale models [16], [39].

The following L-stable IRK methods were selected for examination: SDIRK(2/1) [42], ESDIRK23A [17], and RadauIIa(3/2) [30], [43]. Each has demonstrated accuracy and computational efficiency when solving extremely stiff ODE systems. In addition, each provide an efficient local error estimator that enables error-based adaptive time-stepping. For simplicity, these solvers will be referred to as SDIRK, ESDIRK, and Radau for the remainder of this paper. Butcher tables for these methods are displayed in Fig. 2.

γ	γ	
1	$1-\gamma$	γ
b	$1-\gamma$	γ
$\hat{b}$	$1-\hat{\gamma}$	$\hat{\gamma}$

(a) SDIRK(2/1)

$\frac{1}{3}$	$\frac{5}{12}$	$-\frac{1}{12}$
1	$\frac{3}{4}$	$\frac{1}{4}$
b	$\frac{3}{4}$	$\frac{1}{4}$
$\hat{b}$	$\frac{3}{4}-\frac{\sqrt{6}}{4}$	$\frac{1}{4}+\frac{\sqrt{6}}{12}$

(b) RadauIIA(3/2)

0	0			
2γ	γ	γ		
1	$\hat{b}_1$	$\hat{b}_2$	γ	
1	b_1	b_2	b_3	γ
b	$\frac{6\gamma-1}{12\gamma}$	$\frac{-1}{(24\gamma-12)\gamma}$	$\frac{-6\gamma^2+6\gamma-1}{6\gamma-3}$	γ
$\hat{b}$	$\frac{-4\gamma^2+6\gamma-1}{4\gamma}$	$\frac{-2\gamma+1}{4\gamma}$	γ	0

(c) ESDIRK23A

Fig. 2: Butcher tables for the three implicit Runge-Kutta methods evaluated in this paper. In Fig. 2a, $\gamma = 1-\frac{\sqrt{2}}{2}$ and $\hat{\gamma} = 2-\frac{5}{4}\sqrt{2}$, and in Fig. 2c, $\gamma = 0.4358665215$. In each Butcher table, $\hat{b}$ specifies the lower-order trial step solution weights.

The SDIRK method is second order with an embedded first order formula for local error estimation. Each trial step, $\bar{Z}_i$, of the SDIRK solver can be solved for sequentially. Specifically, since $a_{12} = 0$ (see Fig. 2a), the first stage of this method can be written as $\bar{Z}_1 = h\left(a_{11}\bar{F}(t_n + c_1 h, \bar{y}_n + \bar{Z}_1)\right)$, and $\bar{Z}_1$ can be solved for first and used directly in the solution of $\bar{Z}_2 = h\big(a_{21}\bar{F}(t_n + c_1 h, \bar{y}_n + \bar{Z}_1) + a_{22}\bar{F}(t_n + c_2 h, \bar{y}_n + \bar{Z}_2)\big)$.

The Radau method has two stages like the SDIRK method (see Fig. 2b), but has third order accuracy with a second order error formula. This method's Runge-Kutta matrix is full, therefore the trial stages are solved as a coupled implicit system:

$$\begin{aligned}\bar{Z}_1 &= h[a_{11}\bar{F}(t_n + c_1 h, \bar{y}_n + \bar{Z}_1) + \\ &\qquad a_{12}\bar{F}(t_n + c_2 h, \bar{y}_n + \bar{Z}_2)],\\ \bar{Z}_2 &= h[a_{21}\bar{F}(t_n + c_1 h, \bar{y}_n + \bar{Z}_1) + \\ &\qquad a_{22}\bar{F}(t_n + c_2 h, \bar{y}_n + \bar{Z}_2)].\end{aligned}$$

Trial steps in the ESDIRK method are solved sequentially like the SDIRK method, after the initial explicit first stage (see Fig. 2c). This method is third order with an embedded second order formula for local error estimation, similar to the Radau solver.

D. Implementation

The three IRK methods were programmed in Matlab using principles specified in [30] and [44]; we refer these resources for a detailed explanation of Runge-Kutta method implementation and in this section provide just a brief overview of key aspects utilized in our implementations.

For each IRK method, Newton's method is used in solving system (3a). Typically, each inner Newton iteration involves computing the local Jacobian matrix and performing an LU factorization. To greatly decrease run-time, at each time step the Jacobian computation and LU factorization are performed just once on the first Newton iteration and retained for all remaining iterations. Execution time is further decreased by retaining the Jacobian in the subsequent time step if the IRK method converges with just one Newton iteration, or $\frac{\|\bar{Z}^{k+1}-\bar{Z}^{k}\|}{\|\bar{Z}^{k}-\bar{Z}^{k-1}\|} \leq 10^{-3}$, where k is the number of inner iterations for convergence and $\|\cdot\|$ is an error-normalized 2-norm [30], [45].

Efficient starting values for each Newton iteration are produced via a Lagrange interpolation polynomial of degree s [30], [42]. For the Radau method, for example, we use the data points: $q(0) = 0, q(\frac{1}{3}) = \bar{Z}_1$, and $q(1) = \bar{Z}_2$, and obtain the following Lagrange polynomial:

$$\begin{aligned}q(w) =& q(0)\frac{(w-\frac{1}{3})(w-1)}{(0-\frac{1}{3})(0-1)}+\\ & q\left(\frac{1}{3}\right)\frac{(w-0)(w-1)}{(\frac{1}{3}-0)(\frac{1}{3}-1)}+\\ & q(1)\frac{(w-0)(w-\frac{1}{3})}{(1-0)(1-\frac{1}{3})}\\ =& \frac{w(w-1)}{\frac{-2}{9}}\bar{Z}_1 + \frac{w(w-\frac{1}{3})}{\frac{2}{3}}\bar{Z}_2.\end{aligned}$$

Newton iteration starting values are then given by:

$$\begin{aligned}&\bar{Z}_1 = q(1+wc_1) + \bar{y}_n - \bar{y}_{n+1},\\ &\bar{Z}_2 = q(1+wc_2) + \bar{y}_n - \bar{y}_{n+1}, \text{where } w = \frac{h_{new}}{h_{old}}.\end{aligned}$$

For each time step, local error is calculated and used for (i) step acceptance and (ii) subsequent step size prediction. The error at time step t_{n+1} can be computed by $\overline{err} = \hat{y}_{n+1} - \bar{y}_{n+1}$, where

$$\begin{aligned}\hat{y}_{n+1} = \bar{y}_n &+ \hat{b}_0 h\bar{F}(t_n, \bar{y}_n) +\\ & h\sum_{j=1}^{s}\hat{b}_j\bar{F}(t_n + c_j h, \bar{Z}_j + \bar{y}_n).\end{aligned} \tag{4}$$

The error calculations in the SDIRK and ESDIRK methods are suitable for stiff systems [39], [41]. For the Radau method, however, $\hat{y}_{n+1} - \bar{y}_{n+1}$ will become unbounded and is therefore not appropriate for stiff systems [46]. Instead, we use the formula $\overline{err} = (I - h\hat{b}_0 J)^{-1}(\hat{y}_{n+1} - \bar{y}_{n+1})$ which is equivalent to

$$\begin{aligned}\overline{err} =& (I - h\hat{b}_0 J)^{-1}[\hat{b}_0 h\bar{F}(t_n, \bar{y}_n) +\\ & (\hat{b}_1 - b_1)h\bar{F}(t_n + c_1 h, \bar{Z}_1 + \bar{y}_n) +\\ & (\hat{b}_2 - b_2)h\bar{F}(t_n + c_2 h, \bar{Z}_2 + \bar{y}_n)],\end{aligned} \tag{5}$$

where I is the identity matrix, J is the Jacobian, and $\hat{b}_0 = \frac{\sqrt{6}}{6}$ [46].

We can write $\hat{y}_{n+1} - \bar{y}_{n+1}$ as follows [47]:

$$\hat{y}_{n+1} - \bar{y}_{n+1} = \hat{b}_0 h\bar{F}(t_n, \bar{y}_n) + e_1\bar{Z}_1 + e_2\bar{Z}_2. \tag{6}$$

To identify the coefficients e_1 and e_2, we substitute $\bar{Z}_1$ and $\bar{Z}_2$ (3a) into (6):

$$\begin{aligned}\hat{y}_{n+1} - \bar{y}_{n+1} =& \ \hat{b}_0 h\bar{F}(t_n, \bar{y}_n)\\ & + e_1[ha_{11}\bar{F}(t_n + c_1 h, \bar{Z}_1 + \bar{y}_n) +\\ & \quad ha_{12}\bar{F}(t_n + c_2 h, \bar{Z}_2 + \bar{y}_n)]\\ & + e_2[ha_{21}\bar{F}(t_n + c_1 h, \bar{Z}_1 + \bar{y}_n) +\\ & \quad ha_{22}\bar{F}(t_n + c_2 h, \bar{Z}_2 + \bar{y}_n)].\end{aligned}$$

Collecting terms gives:

$$\begin{aligned}\hat{y}_{n+1} - \bar{y}_{n+1} =& \ \hat{b}_0 h\bar{F}(t_n, \bar{y}_n) +\\ & (e_1 a_{11} + e_2 a_{21})h\bar{F}(t_n + c_1 h, \bar{Z}_1 + \bar{y}_n) +\\ & (e_1 a_{12} + e_2 a_{22})h\bar{F}(t_n + c_2 h, \bar{Z}_2 + \bar{y}_n).\end{aligned} \tag{7}$$

From (5) and (7), we end up with the following system of equations:

$$\begin{aligned}\hat{b}_1 - b_1 &= e_1 a_{11} + e_2 a_{21},\\ \hat{b}_2 - b_2 &= e_1 a_{12} + e_2 a_{22}.\end{aligned}$$

Using the Radau Butcher table (Fig. 2b) gives $(e_1, e_2) = \hat{b}_0\left(\frac{-9}{2}, \frac{1}{2}\right)$. The error estimation is used to predict step size via the strategy proposed by Gustafsson [45]. Further, step size following a rejected step due to excessive local error, namely $\|\overline{err}\| > 1$, is $\frac{1}{3}h$.

For large-scale simulations, e.g. multiscale applications, hundreds of millions of ODE systems may be solved at each time step. For these computationally intensive simulations, scripting languages such as Matlab are not ideal, and machine-compiled programs are generally necessary to achieve simulation results within reasonable computing time [48]. Due to its superior accuracy in solving both the GABA$_\mathrm{A}$R and AMPAR models (see Sec. III), we selected the Radau method and configured a C++ implementation of it. Execution results of this version provide a measure of optimally expected computational performance.

We validated the implementation of each IRK method by comparing their GABA$_\mathrm{A}$R simulation results to those presented in Qazi et al. [37], and their AMPAR simulation results to whole cell recording data in Destexhe et al. [35].

E. Simulations

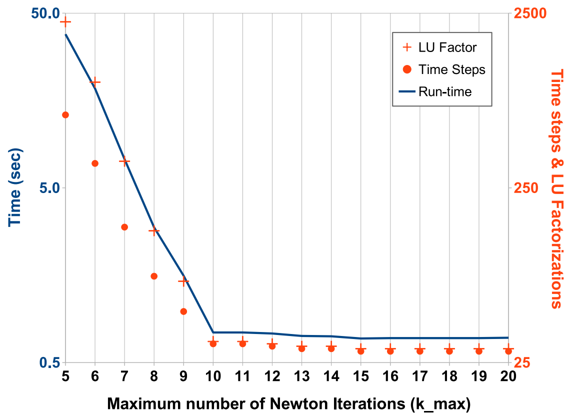
Numerical simulations were performed to assess the robustness of the IRK methods when solving the AMPAR and GABA$_A$R models. Simulations were one second in duration, with rates and initial conditions as specified in Section II-A. Absolute and relative error tolerances were both set to 10^{-8}, and initial step size, h, was set to 10^{-4}. For each IRK method, the optimal number of maximum Newton iterations, k_{max}, was identified by solving the AMPAR and GABA$_A$R models with $k_{max} = 5, 6, ..., 20$. For each value of k_{max}, the mean execution time of five simulations was computed, and the value of k_{max} that produced the lowest mean execution time was selected. Figure 3a displays the k_{max} values selected for each model and method.

For each method, it was observed that a threshold value of k_{max} exists, such that higher values do not result in faster simulations. Therefore, we selected the minimum k_{max} value associated with the fastest execution speed. For example, for the Radau method solving the GABA$_A$R model, simulation times begin to plateau for $k_{max} \geq 10$, and simulation times with $k_{max} \geq 15$ were the same (see Fig. 3b). Therefore, for this model and IRK method, $k_{max} = 15$ was selected.

Figure 3b also shows that faster run times correlate with fewer solution time steps and LU factorizations, until a floor is reached; in the case of the Radau method solving the GABA$_A$R model, this floor is 29 time steps and 30 LU factorizations. To a point, higher values of k_{max} increase the probability of Newton method convergence, resulting in fewer time steps and fewer computationally expensive LU factorizations [30]. For the Radau method solving the GABA$_A$R model, values of $k_{max} \geq 15$ yield the fewest number of simulation times steps in addition to no steps where the Newton iteration fails to converge. Thus, when $k_{max} = 15$, time steps and associated LU factorizations are minimized, yielding the fastest execution speeds.

Model	SDIRK	ESDIRK	Radau
GABA$_A$R	7	10	15
AMPAR	14	12	17

(a) Values of k_{max} selected for each model and method

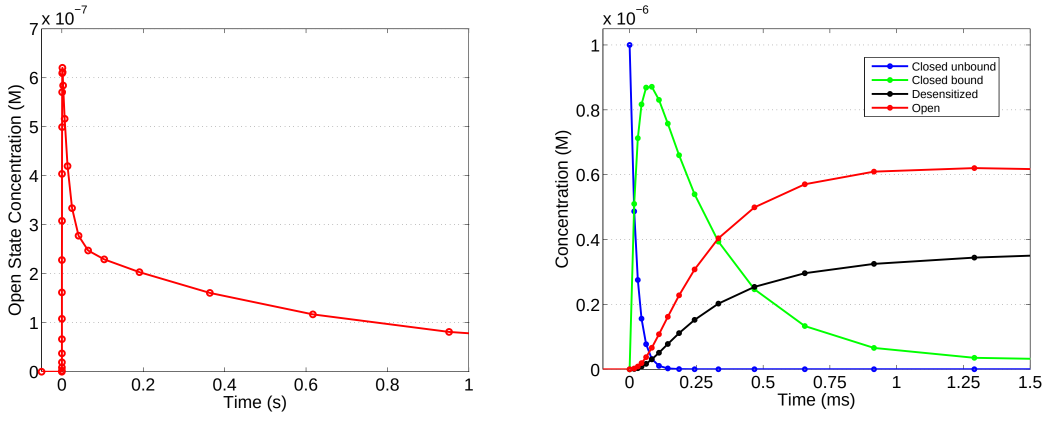


(b) Radau method solving the GABA$_A$R model: run time, time steps, and LU factorizations, for $k_{max} = 5, 6, ..., 20$

Fig. 3: Maximum Newton iteration metrics and results.

To evaluate the advantages that IRK methods have when solving fast-responding neuroreceptor models, we first compare the total number of simulation time steps and simulation step sizes of each IRK method to the following commonly used ERK methods: forward euler (FE), midpoint method (Mid), and 4^{th} order Runge-Kutta (RK4). Next, to compare each IRK method to one another and to the adaptive 5th order Dormand-Prince method (DP5) [49], metrics including local and global error, total simulation time steps, step sizes, execution times, and numbers of Jacobian computations and LU factorizations were evaluated. Absolute and relative error tolerances of the DP5 method were set to 10^{-8}, matching the tolerances of the three implicit methods.

To more comprehensively assess performance differences amoung the IRK methods, work-precision diagrams using solution run times and scd values, where scd = -log$_{10}$($\|$relative error at $t = 1.0$ sec $\|_\infty$), were then generated [50]. For the work-precision diagrams, relative error tolerances



(a) Open state concentration solution, $O_1 + O_2$

(b) Solution of all receptor forms: Closed unbound = C_0; Closed bound = $C_1 + C_2$; Desensitized = $D_s + D_f$; Open = $O_1 + O_2$

Fig. 4: SDIRK method solution of GABA$_A$R model.

were set to $rtol = 10^{-(4+\frac{m}{5})}$, $m = 0, 1, ..., 25$, absolute error tolerance was set to $10^{-4} \cdot rtol$, and initial step size was 10^{-4}. In addition, solution run times presented in these diagrams are the mean of five runs. For all accuracy calculations, solutions with a 5^{th} order adaptive time-stepping L-stable implicit Runge-Kutta method with a maximum step size of 10^{-6} and both absolute and relative tolerances set to 10^{-14} were used as true solutions.

Finally, the execution time of the Radau C++ implementation when solving both neuroreceptor models was assessed. All simulations were run on a Linux machine with an Intel i7 processor with a clock speed of 2.40 GHz.

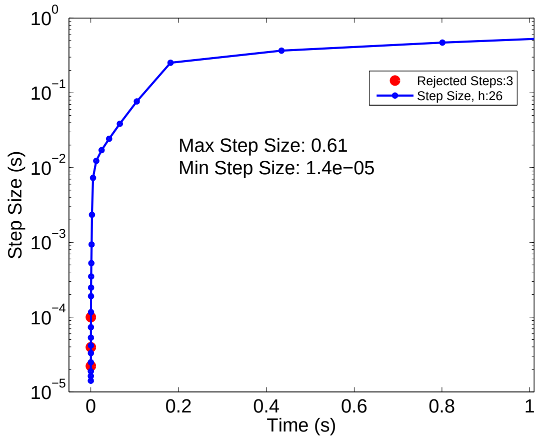
III. Results and Discussion

A. GABA$_A$R Model

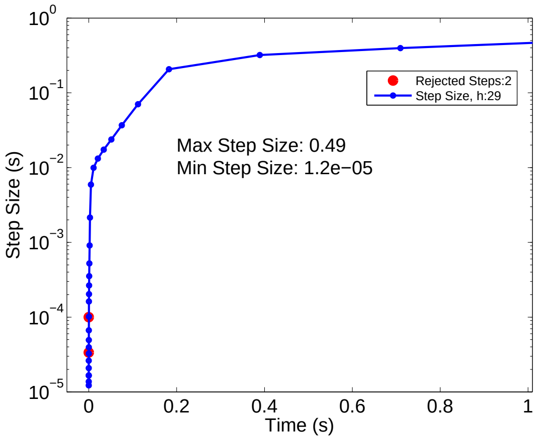
Figure 4 presents the solution of the GABA$_A$R model with the SDIRK method; ESDIRK and Radau solutions look identical. The sharp transition in the total open state concentration, $O_1(t) + O_2(t)$, at the onset of neurotransmitter stimulus at $t = 0$ displays the necessity for smaller time steps in this region of the solution (Fig. 4a). Upon examining all receptor forms during the first 1.5 ms of the simulation, it is observed that both the unbound closed form, $C_0(t)$, and total bound closed form, $C_1(t) + C_2(t)$, possess concentration transitions even greater than the open receptor form (Fig. 4b). These results show the stiffness possessed by the GABA$_A$R system.

Table I displays simulation time step metrics for the three IRK methods and the FE, Mid, and RK4 ERK methods. The maximum step size of each explicit method was calculated with the GABA$_A$R model stiffness index and the method's stability region [28], giving the largest step that can be taken while maintaining numerical stability. Then, the number of time steps required for each ERK method was computed by dividing the simulation duration by the maximum step size. The FE and Mid methods both require 2.1 x 10^4 time steps, and the RK4 method requires 1.5 x 10^4, which is lower than the FE and Mid methods due to its larger stability region [30]. On the contrary, each implicit method requires less than 30 simulation time steps. As displayed in Figure 4a, the majority of these time steps for the SDIRK method occur at the beginning of the simulation, within the region of rapid solution transition.

Similarly, the ESDIRK and Radau solvers demand noticeably more time steps at the onset of neurotransmitter stimulation (Fig. 5). Rejected steps, totalling three for the ESDIRK method (Fig.



(a) ESDIRK method



(b) Radau method

Fig. 5: Simulation step sizes for the GABA$_A$R model.

TABLE I: Simulation time steps results for the ERK and IRK methods when solving the GABA$_A$R model.

Method (Order)	Max Step Size (s)	Time Steps
FE (1)	4.8 x 10^{-5}	2.1 x 10^4
Mid (2)	4.8 x 10^{-5}	2.1 x 10^4
RK4 (4)	6.8 x 10^{-5}	1.5 x 10^4
SDIRK (2/1)	Adaptive	28
ESDIRK (3/2)	Adaptive	26
Radau (3/2)	Adaptive	29

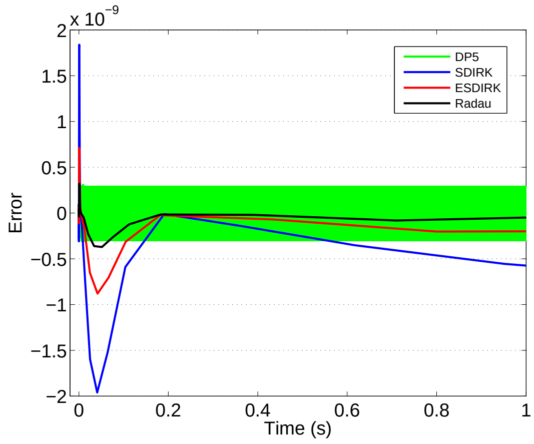


Fig. 6: GABA$_A$R model open state concentration solution error.

5a) and two for the Radau method (Fig. 5b) all occur at time $t = 0$; once the solution in this region has been accurately resolved, no further rejected steps occur. In addition, for all three IRK methods, all Newton iterations converged, which was facilitated by identifying optimal k_{max} values (see Sec. II-E). Further, the *smallest* step sizes of the IRK methods, namely 1.4 x 10^{-5} for the SDIRK and ESDIRK methods and 1.2 x 10^{-5} for the Radau method, have the same order of magnitude as the *largest* stable step sizes of the ERK methods.

Next the accuracy and computational efficiency of the IRK methods were compared to one another and with the DP5 method (Table II). While the DP5 method possesses the lowest maximal local true solution deviation (3.2 x 10^{-10}), the 2-norm of its global error is one to two orders of magnitude higher than all three IRK methods. These results are explained by the fact that the solution of the DP5 solver oscillates around the true solution (Fig. 6). In addition, the DP5 method requires approximately 50,000 simulation time steps and takes 49.0 seconds to run. In comparison, the

TABLE II: Accuracy and simulation run-time metrics of the DP5 and IRK methods when solving the GABA$_A$R model. Boldface font denotes best results of each column.

Method (Order)	$\|\|$ Error $\|\|_2$	Max $\|$Error$\|$	Time Steps	Run Time (s)
DP5 (5/4)	252.0 x 10^{-10}	**3.2 x 10^{-10}**	5.0 x 10^4	49.0
SDIRK (2/1)	45.6 x 10^{-10}	19.6 x 10^{-10}	28	**0.21**
ESDIRK (3/2)	18.3 x 10^{-10}	8.8 x 10^{-10}	**26**	0.27
Radau (3/2)	**8.7 x 10^{-10}**	3.7 x 10^{-10}	29	0.69

ESDIRK method requires 26 time steps and the SDIRK method executes in 0.21 seconds. DP5 solution accuracy can be improved with either stricter error tolerances or a decreased time step [51], however, these approaches will result in even greater run times.

The Radau method has the greatest execution time of the three IRK methods, at 0.69 seconds. While the number of simulation time steps amoung the IRK methods are comparable, two factors contribute to the longer run time of the Radau method. First, this solver generally requires a greater number of iterations for Newton's method to converge (Fig. 3a). Second, the Radau method requires 30 Jacobian computations, versus just four for the SDIRK and ESDIRK methods.

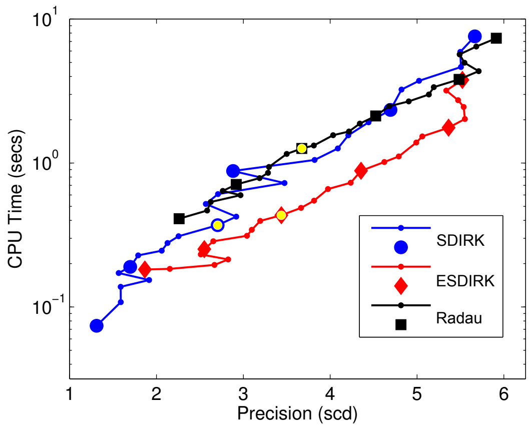


Fig. 7: GABA$_A$R work-precision diagram with solver run time vs. scd for each IRK method. Integer exponential tolerances, i.e. 10^{-4}, 10^{-5}, ..., are presented with enlarged symbols. The symbol for $rtol = 10^{-6}$ is distinguished by the yellow circle.

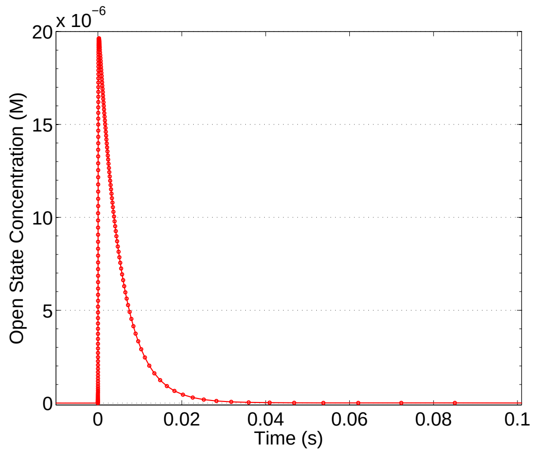
Despite its run time disadvantages amongst the IRK methods, the accuracy of the Radau method stands out as superior. It has the lowest global error 2-norm (8.7 x 10^{-10}), and its maximal deviation from the true solution (-3.7 x 10^{-10}) is comparable to that of the 5^{th} order DP5 method, the only IRK method examined where this is the case. Further, the Radau method has greater accuracy at every time step than both the SDIRK and ESDIRK methods.

These findings are reinforced by the work-precision diagram for the three IRK methods when solving the GABA$_A$R model (Fig. 7). This diagram highlights the higher precisions attained by the third order methods, and in addition, also confirms the slower execution speeds achieved by the Radau method. However, when comparing graph points of similar relative tolerances, such as the symbols marked in yellow that represent $rtol = 10^{-6}$, the Radau method is consistently more accurate.

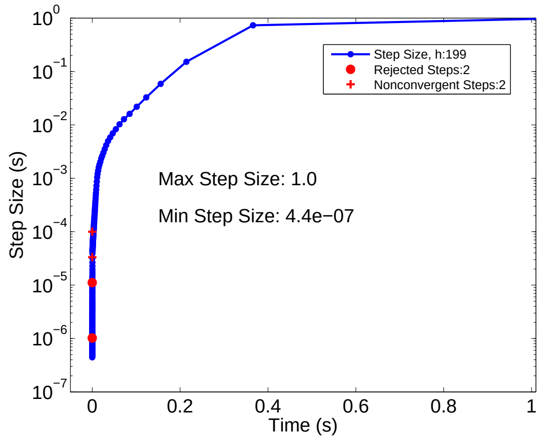
B. AMPAR Model

Figure 8 presents solution results of the AMPAR model solved with the Radau method. Like the GABA$_A$R model, the rapid transition in the open state concentration upon neurotransmitter stimulation demands a greater number of time steps (Fig. 8a). Specifically, the first 10% of the simulation (0.1 sec) encompasses approximately 96% of the simulation time steps. Once beyond this initial region, step size eventually increases by seven orders of magnitude (Fig. 8b). Similar to the GABA$_A$R model, both unbound closed and bound closed forms contribute to the system's stiffness.

A noticeable difference, compared to the GABA$_A$R simulation results, is the number of time



(a) Open state concentration solution, O



(b) Simulation step sizes

Fig. 8: Radau method solution of the AMPAR model.

steps needed by the implicit methods to solve the AMPAR model. The Radau method, for example, requires 199 time steps (Fig. 8b), a 586% increase from the 29 steps needed to solve the GABA$_A$R model. Similar increases are observed with the SDIRK and ESDIRK solvers, most notably the 531 steps required by the SDIRK method (Table III). In addition, the smallest step sizes of the IRK methods are two orders of magnitude lower with the AMPAR model (Fig. 8b), due to the stiffness index of the AMPAR system [27]. Despite the elevated simulation time step counts, each IRK method still outperforms the explicit methods (Table III); maximum stable step sizes and simulation time steps for the explicit methods were again computed with their stiffness indices and stability regions [28].

While greater k_{max} values eliminated non-convergent Newton iterations in the GABA$_A$R model, this is not the case with the AMPAR model. Each IRK method has two instances where Newton's method did not converge. In addition, the SDIRK method has four rejected steps, and the ESDIRK and Radau methods each have two, all occurring at time $t = 0$.

Table IV displays accuracy and execution efficiency results for the IRK methods. An interesting

TABLE III: Simulation time steps results for the ERK and IRK methods when solving the AMPAR model.

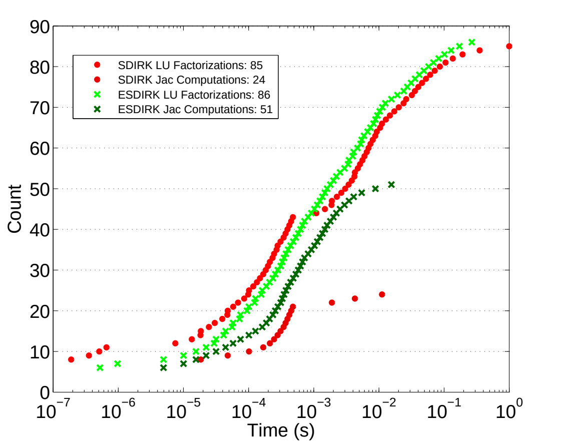
Method (Order)	Max Step Size (s)	Time Steps
FE (1)	1.7 x 10^{-5}	5.9 x 10^4
Mid (2)	1.7 x 10^{-5}	5.9 x 10^4
RK4 (4)	2.4 x 10^{-5}	4.2 x 10^4
SDIRK (2/1)	Adaptive	531
ESDIRK (3/2)	Adaptive	211
Radau (3/2)	Adaptive	199

result is the seemingly uncorrelated relationship between simulation time steps and run time. For example, despite having the lowest number of simulation time steps, the Radau method has the longest run time. Along these same lines, the Radau method has less than 50% of the simulation time steps of the SDIRK method, yet no noticeable computational advantage. Moreover, the ESDIRK method has approximately 40% of the SDIRK method's time steps, yet it requires 72% of its run-time.

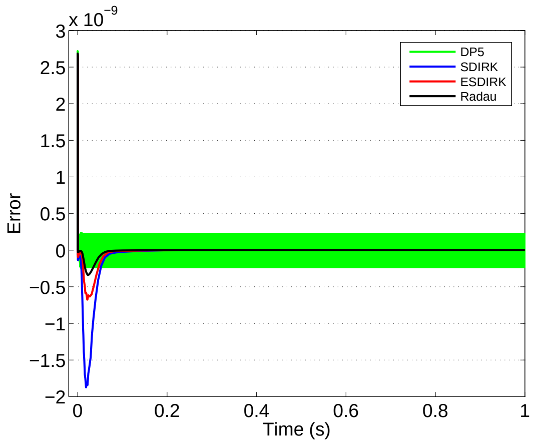
With a comparable number of rejected and non-convergent steps (Table V), a culprit for this behavior is the number of Jacobian computations

TABLE IV: Accuracy and simulation run-time metrics of the DP5 and IRK methods when solving the AMPAR model. Boldface font denotes best results of each column.

Method (Order)	$\|\| \text{Error} \|\|_2$	Max $\|\text{Error}\|$	Time Steps	Run Time (s)
DP5 (5/4)	3.3×10^{-8}	2.7×10^{-9}	1.1×10^{5}	32.4
SDIRK (2/1)	3.0×10^{-8}	2.7×10^{-9}	531	1.34
ESDIRK (3/2)	1.7×10^{-8}	2.7×10^{-9}	211	**0.97**
Radau (3/2)	**1.6×10^{-8}**	2.7×10^{-9}	**199**	1.38



(a) SDIRK and ESDIRK Jacobian computations and LU factorizations



(b) Open state concentration solution error

Fig. 9: Method comparison when solving the AMPAR model.

performed by these solvers. Figure 9a displays the Jacobian computations and LU factorizations of the SDIRK and ESDIRK methods. Each method has a near identical number of LU factorizations, however, the ESDIRK method requires 51 Jacobian computations, which is more than double the 24 performed by the SDIRK method. In addition, the Radau method requires 162 Jacobian computations. Therefore, despite having a lower number of simulation time steps, the computational advantages of the ESDIRK and Radau methods are diminished due to this elevated number of Jacobian computations.

Once again, the accuracy and computational performances of the IRK methods were compared to the DP5 method (Table IV). As observed with the GABA$_A$R model, the DP5 method has inferior execution performance, requiring 1.1×10^5 simulation time steps and 32.4 seconds for a numerically stable solution, both of which are significantly greater than results attained with the IRK methods. All four methods generate the same maximum local error (2.7×10^{-9}), which occurs at $t = 0$ for all methods. Also, differences among the global errors are relatively smaller with the AMPAR model. The oscillatory nature of the DP5 solution around the true solution (Fig. 9b) contributes to its

TABLE V: Number of rejected and non-convergent steps for each IRK method when solving the AMPAR model.

Model	Rejected	Non-convergent
SDIRK	2	2
ESDIRK	4	2
Radau	2	2

global error 2-norm (3.3×10^{-8}), which is again larger than those of the three IRK methods. The Radau method once again has the lowest global error 2-norm (1.6×10^{-8}) of all methods inspected.

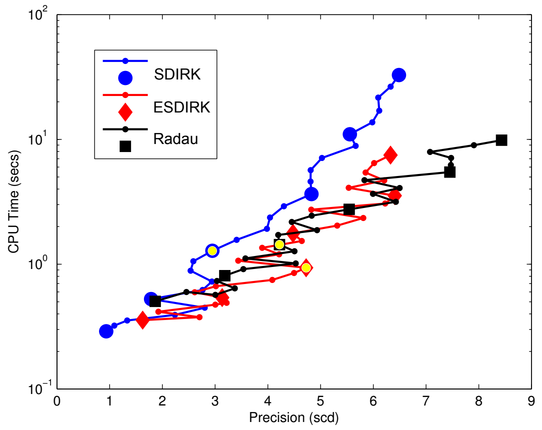


Fig. 10: AMPAR work-precision diagram with solver run time vs. scd for each IRK method. Integer exponential tolerances, i.e. 10^{-4}, 10^{-5}, ..., are presented with enlarged symbols. The symbol for $rtol = 10^{-6}$ is distinguished by the yellow circle.

The work-precision diagram for the AMPAR model (Fig. 10) again confirms the higher precision achieved by the third order ESDIRK and Radau solvers. More noticeable in this graph are the differences in the "slopes" of the curves, where "flatter" curves, i.e. ESDIRK and Radau, have more precision per unit CPU time [30]. For the AMPAR model, the Radau method is slower than the ESDIRK method at all work-precision tolerances examined, yet at relative tolerances greater than 10^{-6}, the Radau method becomes faster than the SDIRK method. Further, the Radau method is generally the most accurate of all three IRK methods.

C. C++ Radau Implementation

The Radau method consistently demonstrates the greatest accuracy of the methods examined, however, its main disadvantage is execution speed. For this reason, we selected the Radau method and configured a C++ implementation of it. Table VI displays execution times for the previous Radau Matlab implementation, as well as the new C++ version.

As expected, the C++ version is significantly faster. Specifically, the $\text{GABA}_\text{A}\text{R}$ model has a 99.6% decrease in execution time, and the AMPAR model has a 99.7% decrease in execution time. Because the implementation algorithms between the two versions are the same, the C++ version maintains the accuracy of the Matlab prototype.

TABLE VI: Run times (seconds) for the Matlab and C++ Radau method when solving the $\text{GABA}_\text{A}\text{R}$ and AMPAR models.

Implementation	$\text{GABA}_\text{A}\text{R}$	AMPAR
Matlab	0.69	1.38
C++	**2.7 x 10^{-3}**	**3.5 x 10^{-3}**

IV. Conclusions

Computational neurology is a valuable contributor in the diagnosis, treatment, and comprehension of neurological disease. To provide maximal utility to the scientific community, computational simulations should incorporate highly-detailed, neurotransmitter-based neuron models. Therefore, large-scale simulations involving populations of neurons will inevitably produce computational challenges. In this paper, we have shown that appropriate numerical solvers with efficient implementation strategies can alleviate computational difficulties.

Commonly used explicit methods are capable in solving a limited number of fast-responding ligand-gated neuroreceptor models. However, we have shown that poor stability properties make them non-ideal for large-scale applications. Rather, by addressing the stiffness possessed by these models, we show that implicit methods are highly advantageous. In particular, we demonstrate that L-stable implicit Runge-Kutta methods offer superior accuracy and run-time efficiency compared to their explicit siblings when solving biologically-based AMPA and GABA$_A$ neuroreceptor models. To accelerate solutions, we utilize a range of strategies including embedded error estimators and simplified Newton iterations. In addition, we show that optimal execution times are achieved when costly Jacobian computations and LU factorizations are minimized.

The third order Radau IRK method demonstrates exceptional local and global accuracy compared to all other explicit and implicit methods examined. In addition, its numerical stability properties yield a relatively low number of simulation time steps and efficient step sizes when solving the AMPA and GABA$_A$ neuroreceptor models. Further, a C++ implementation of the Radau solver displays the computational faculty to enable large-scale multi-cellular simulations. In future work, we plan to continue our investigation of numerical solvers for neurotransmitter-based neuron models by comparing the IRK methods to multi-step methods and exponential integrators.

Acknowledgment

The author is grateful to Professor Jeff Borggaard and Professor James Turner for useful discussions related to this manuscript, and Frank Vogel for assistance with the Radau C++ code.

References

[1] A. Datta, X. Zhou, Y. Su, L. C. Parra, and M. Bikson, "Validation of finite element model of transcranial electrical stimulation using scalp potentials: implications for clinical dose," *Journal of Neural Engineering*, vol. 10, no. 3, p. 036018, may 2013. [Online]. Available: http://dx.doi.org/10.1088/1741-2560/10/3/036018

[2] R. Plonsey and D. B. Heppner, "Considerations of quasi-stationarity in electrophysiological systems," *Bulletin of Mathematical Biophysics*, vol. 29, no. 4, pp. 657–664, dec 1967. [Online]. Available: http://dx.doi.org/10.1007/bf02476917

[3] S. Lew, C. Wolters, T. Dierkes, C. Rer, and R. MacLeod, "Accuracy and run-time comparison for different potential approaches and iterative solvers in finite element method based EEG source analysis," *Applied Numerical Mathematics*, vol. 59, no. 8, pp. 1970–1988, aug 2009. [Online]. Available: http://dx.doi.org/10.1016/j.apnum.2009.02.006

[4] T. Neuling, S. Wagner, C. H. Wolters, T. Zaehle, and C. S. Herrmann, "Finite-element model predicts current density distribution for clinical applications of tDCS and tACS," *Front. Psychiatry*, vol. 3, 2012. [Online]. Available: http://dx.doi.org/10.3389/fpsyt.2012.00083

[5] F. Gasca, L. Marshall, S. Binder, A. Schlaefer, U. G. Hofmann, and A. Schweikard, "Finite element simulation of transcranial current stimulation in realistic rat head model," in *2011 5th International IEEE/EMBS Conference on Neural Engineering*. IEEE, apr 2011. [Online]. Available: http://dx.doi.org/10.1109/ner.2011.5910483

[6] P. C. Miranda, M. Lomarev, and M. Hallett, "Modeling the current distribution during transcranial direct current stimulation," *Clinical Neurophysiology*, vol. 117, no. 7, pp. 1623–1629, jul 2006. [Online]. Available: http://dx.doi.org/10.1016/j.clinph.2006.04.009

[7] M. Åstrm, L. U. Zrinzo, S. Tisch, E. Tripoliti, M. I. Hariz, and K. Wårdell, "Method for patient-specific finite element modeling and simulation of deep brain stimulation," *Medical & Biological Engineering & Computing*, vol. 47, no. 1, pp. 21–28, oct 2008. [Online]. Available: http://dx.doi.org/10.1007/s11517-008-0411-2

[8] R. Sadleir, "A Bidomain Model for Neural Tissue," *International Journal of Bioelectromagnetism*, vol. 12, no. 1, pp. 2–6, 2010.

[9] E. Mandonnet and O. Pantz, "The role of electrode direction during axonal bipolar electrical stimulation: a bidomain computational model study," *Acta Neurochirurgica*, vol. 153, no. 12, pp. 2351–2355, sep 2011. [Online]. Available: http://dx.doi.org/10.1007/s00701-011-1151-x

[10] W. Ying and C. S. Henriquez, "Hybrid finite element method for describing the electrical response of biological cells to applied fields," *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 4, pp. 611–620, apr 2007. [Online]. Available: http://dx.doi.org/10.1109/tbme.2006.889172

[11] A. Agudelo-Toro and A. Neef, "Computationally efficient simulation of electrical activity at cell membranes interacting with self-generated and externally imposed electric fields," *Journal of Neural Engineering*, vol. 10, no. 2, p. 026019, mar 2013. [Online]. Available: http://dx.doi.org/10.1088/1741-2560/10/2/026019

[12] K. W. Altman and R. Plonsey, "Development of a model for point source electrical fibre bundle stimulation," *Med. Biol. Eng. Comput.*, vol. 26, no. 5, pp. 466–475, sep 1988. [Online]. Available: http://dx.doi.org/10.1007/bf02441913

[13] R. Szmurlo, J. Starzynski, S. Wincenciak, and A. Rysz, "Numerical model of vagus nerve electrical stimulation," *COMPEL*, vol. 28, no. 1, pp. 211–220, jan 2009. [Online]. Available: http://dx.doi.org/10.1108/03321640910919002

[14] R. Szmurlo, J. Starzynski, B. Sawicki, and S. Wincenciak, "Multiscale finite element model of the electrically active neural tissue," in *EUROCON*

2007 - The International Conference on "Computer as a Tool". IEEE, 2007. [Online]. Available: http://dx.doi.org/10.1109/eurcon.2007.4400409

[15] R. Szmurlo, J. Starzynski, B. Sawicki, S. Wincenciak, and A. Cichocki, "Bidomain formulation for modeling brain activity propagation," in *2006 12th Biennial IEEE Conference on Electromagnetic Field Computation*. IEEE, 2006. [Online]. Available: http://dx.doi.org/10.1109/cefc-06.2006.1633138

[16] E. T. Dougherty, J. C. Turner, and F. Vogel, "Multiscale coupling of transcranial direct current stimulation to neuron electrodynamics: modeling the influence of the transcranial electric field on neuronal depolarization," *Comput Math Methods Med*, vol. 2014, pp. 1–14, 2014. [Online]. Available: http://dx.doi.org/10.1155/2014/360179

[17] J. Sundnes, G. T. Lines, and A. Tveito, "Efficient solution of ordinary differential equations modeling electrical activity in cardiac cells," *Mathematical Biosciences*, vol. 172, no. 2, pp. 55–72, aug 2001. [Online]. Available: http://dx.doi.org/10.1016/s0025-5564(01)00069-4

[18] C. N. V. S. Reports, "Deaths: Preliminary Data for 2011," *NVSS*, vol. 61, no. 6, 2012.

[19] J.-A. Girault and P. Greengard, "The neurobiology of dopamine signaling," *Archives of Neurology*, vol. 61, no. 5, p. 641, may 2004. [Online]. Available: http://dx.doi.org/10.1001/archneur.61.5.641

[20] D. Tarsy, J. L. Vitek, P. A. Starr, and M. S. Okun, Eds., *Deep Brain Stimulation in Neurological and Psychiatric Disorders*. Humana Press, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-1-59745-360-8

[21] S. Miocinovic, S. Somayajula, S. Chitnis, and J. L. Vitek, "History, applications, and mechanisms of deep brain stimulation," *JAMA Neurol*, vol. 70, no. 2, p. 163, feb 2013. [Online]. Available: http://dx.doi.org/10.1001/2013.jamaneurol.45

[22] S. Miocinovic, C. C. McIntyre, M. Savasta, and J. L. Vitek, "Mechanisms of deep brain stimulation," in *Deep Brain Stimulation in Neurological and Psychiatric Disorders*. Humana Press, 2008, pp. 151–177. [Online]. Available: http://dx.doi.org/10.1007/978-1-59745-360-8_8

[23] M. D. Johnson, S. Miocinovic, C. C. McIntyre, and J. L. Vitek, "Mechanisms and targets of deep brain stimulation in movement disorders," *Neurotherapeutics*, vol. 5, no. 2, pp. 294–308, apr 2008. [Online]. Available: http://dx.doi.org/10.1016/j.nurt.2008.01.010

[24] F. Windels, N. Bruet, A. Poupard, C. Feuerstein, A. Bertrand, and M. Savasta, "Influence of the frequency parameter on extracellular glutamate and gamma-aminobutyric acid in substantia nigra and globus pallidus during electrical stimulation of subthalamic nucleus in rats," *Journal of Neuroscience Research*, vol. 72, no. 2, pp. 259–267, apr 2003. [Online]. Available: http://dx.doi.org/10.1002/jnr.10577

[25] E. Suli and D. F. Mayers, *An Introduction to Numerical Analysis*. Cambridge University Press, 2003. [Online]. Available: http://dx.doi.org/10.1017/cbo9780511801181

[26] S. Qazi, M. Caberlin, and N. Nigam, "Mechanism of psychoactive drug action in the brain: Simulation modeling of GABAA receptor interactions at non-equilibrium conditions," *Current Pharmaceutical Design*, vol. 13, no. 14, pp. 1437–1455, may 2007. [Online]. Available: http://dx.doi.org/10.2174/138161207780765972

[27] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, jan 1998. [Online]. Available: http://dx.doi.org/10.1137/1.9781611971392

[28] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I*. Springer Berlin Heidelberg, 1987. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-12607-3

[29] MATLAB, *version 8.2.0.701 (R2013b)*. Natick, Massachusetts: The MathWorks Inc., 2013.

[30] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II*. Springer Berlin Heidelberg, 1996. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-05221-7

[31] R. Sapolsky, *Biology and Human Behavior: The Neurological Origins of Individuality, 2nd Edition*. American Psychological Association (APA). [Online]. Available: http://dx.doi.org/10.1037/e526622012-001

[32] B. S. Meldrum, "Glutamate as a neurotransmitter in the brain: review of physiology and pathology," *J. Nutr.*, vol. 130, no. 4S Suppl, pp. 1007S–15S, Apr 2000.

[33] A. Destexhe, Z. F. Mainen, and T. J. Sejnowski, "Synthesis of models for excitable membranes, synaptic transmission and neuromodulation using a common kinetic formalism," *Journal of Computational Neuroscience*, vol. 1, no. 3, pp. 195–230, aug 1994. [Online]. Available: http://dx.doi.org/10.1007/bf00961734

[34] D. K. Patneau and M. L. Mayer, "Kinetic analysis of interactions between kainate and AMPA: Evidence for activation of a single receptor in mouse hippocampal neurons," *Neuron*, vol. 6, no. 5, pp. 785–798, may 1991. [Online]. Available: http://dx.doi.org/10.1016/0896-6273(91)90175-y

[35] A. Destexhe, Z. F. Mainen, and T. J. Sejnowski, "Kinetic models of synaptic transmission," in *Methods in Neuronal Modeling: From Synapse to Networks*, C. Koch and I. Segev, Eds. MIT press, 1998, pp. 1–25.

[36] A. Meir, S. Ginsburg, A. Butkevich, S. G. Kachalsky, I. Kaiserman, R. Ahdut, S. Demirgoren, and R. Rahamimoff, "Ion channels in presynaptic nerve terminals and control of transmitter release," *Physiol. Rev.*, vol. 79, no. 3, pp. 1019–1088, Jul 1999.

[37] S. Qazi, A. Beltukov, and B. A. Trimmer, "Simulation modeling of ligand receptor interactions at non-equilibrium conditions: processing of noisy inputs by ionotropic receptors," *Mathematical Biosciences*, vol. 187, no. 1, pp. 93–110, jan 2004. [Online]. Available: http://dx.doi.org/10.1016/j.mbs.2003.01.001

[38] J. D. Lambert, *Numerical methods for ordinary differential systems : the initial value problem*. Chichester New York: Wiley, 1991.

[39] J. Sundnes, G. T. Lines, X. Cai, F. N. Bjorn, K. A. Mardal, and A. Tveito, *Computing the Electrical Activity in the Heart*. Springer Berlin Heidelberg, 2006. [Online]. Available: http://dx.doi.org/10.1007/3-540-33437-8

[40] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University Press, 2008. [Online]. Available: http://dx.doi.org/10.1017/cbo9780511995569

[41] A. Kværnø, "Singly diagonally implicit runge–kutta methods with an explicit first stage," *BIT Numerical Mathematics*, vol. 44, no. 3, pp. 489–502, aug 2004. [Online]. Available: http://dx.doi.org/10.1023/b:bitn.0000046811.70614.38

[42] L. M. Skvortsov, "An efficient scheme for the implementation of implicit runge-kutta methods," *Computational Mathematics and Mathematical Physics*, vol. 48, no. 11, pp. 2007–2017, nov 2008. [Online]. Available: http://dx.doi.org/10.1134/s0965542508110092

[43] L. Brugnano, F. Iavernaro, and C. Magherini, "Efficient implementation of radau collocation methods," *Applied Numerical Mathematics*, vol. 87, pp. 100–113, jan 2015. [Online]. Available: http://dx.doi.org/10.1016/j.apnum.2014.09.003

[44] J. J. de Swart, "A simple ODE solver based on 2-stage radau IIA," *Journal of Computational and Applied Mathematics*, vol. 84, no. 2, pp. 277–280, oct 1997. [Online]. Available: http://dx.doi.org/10.1016/s0377-0427(97)00141-6

[45] K. Gustafsson, "Control-theoretic techniques for stepsize selection in implicit runge-kutta methods," *ACM Trans. Math. Softw.*, vol. 20, no. 4, pp. 496–517, dec 1994. [Online]. Available: http://dx.doi.org/10.1145/198429.198437

[46] J. Wang, J. Rodriguez, and R. Keribar, "Integration of flexible multibody systems using radau IIA algorithms," *J. Comput. Nonlinear Dynam.*, vol. 5, no. 4, p. 041008, 2010. [Online]. Available: http://dx.doi.org/10.1115/1.4001907

[47] N. Guglielmi and E. Hairer, "Implementing radau IIA methods for stiff delay differential equations," *Computing*, vol. 67, no. 1, pp. 1–12, jul 2001. [Online]. Available: http://dx.doi.org/10.1007/s006070170013

[48] H. P. Langtangen, *Computational Partial Differential Equations: Numerical Methods and Diffpack Programming*, ser. Texts in Computational Science and Engineering. Springer Berlin Heidelberg, 2003.

[49] J. Dormand and P. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, mar 1980. [Online]. Available: http://dx.doi.org/10.1016/0771-050x(80)90013-3

[50] F. Mazzia and C. Magherini, "Testset for initial value problem solvers, release 2.4," http://www.dm.unipi.it/testset/testsetivpsolvers/, University of Bari and INdAM, Tech. Rep., 02 2008.

[51] M. Caberlin, *Stiff Ordinary and Delay Differential Equations in Biological Systems*, ser. McGill theses. McGill University, 2002.