Solving Nonlinear Systems of First Order Ordinary Differential Equations Using a Galerkin Finite Element Method

AHMAD AL-OMARI1, HEINZ-BERND SCHÜTTLER2, JONATHAN ARNOLD3, AND THIAB TAHA4

1Institute of Bioinformatics, University of Georgia, Athens, GA 30602, USA
2Department of Physics and Astronomy, University of Georgia, Athens, GA 30602, USA
3Department of Genetics, University of Georgia, Athens, GA 30602, USA
4Department of Computer Science, University of Georgia, Athens, GA 30602, USA

Corresponding author: A. Al-Omari (aomari@uga.edu)

This work was supported in part by the NSF under Grants NSF QSB-0425762 and the NSF DBI-1062213 and the Department of Systems Engineering and Medical Bioinformatics, Yarmouk University, Irbid, Jordan.

ABSTRACT A new numerical technique to solve nonlinear systems of initial value problems for nonlinear first-order differential equations (ODEs) that model genetic networks in systems biology is developed. This technique is based on finding local Galerkin approximations on each sub-interval at a given time grid of points using piecewise hat functions. Comparing the numerical solution of the new method for a single nonlinear ODE with an exact solution shows that this method gives accurate solutions with relative error $1.88 \times 10^{-11}$ for a time step $1 \times 10^{-6}$. This new method is compared with the adaptive Runge Kutta (ARK) method for solving systems of ODEs, and the results are comparable for a time step $2 \times 10^{-4}$. It is shown that the relative error of the Galerkin method decreases approximately linearly with the log of the number of hat functions used. Unlike the ARK method, this new method has the potential to be parallelizable and to be useful for solving biological problems involving large genetic networks. An NSF commissioned video illustrating how systems biology helps us understand that a fundamental process in cells is included.


I. INTRODUCTION

In the new cross-disciplinary field of systems biology merging genomics, bioinformatics and engineering the focus is on using networks of genes and their products to predict fundamental processes in the cell [1]. The field began in the 1990s with the assembly of biochemical pathways to describe the functioning of entire cells [2]–[4]. The field was transformed with the development of new genomics technologies [1], [5], [6] to measure how many genes and proteins behave simultaneously in cells. We are now poised to describe the cellular dynamics of an entire cellular network [7], [8]. The challenge is to be able to simulate such large networks. The dynamics of these cellular networks are often described by very large systems of ordinary differential equations [9]. One of the major problems in systems biology is solving large systems of ordinary differential equations describing how genetic networks behave [10], a challenge arising in other areas of science and engineering as well [11]. The Galerkin method has been employed for solving different kinds of ordinary differential equations [12]–[19]. Here we show how Galerkin’s method can be used in conjunction with Finite Element Method (FEM) piecewise hat functions to solve systems of nonlinear first-order ordinary differential equations (ODEs). Here our method is applied to systems of ODEs describing several genetic networks [20], [21]. The importance of these networks to our daily lives is summarized in an NSF commissioned video attached [22]. The idea behind the method is to find local Galerkin approximations to the solutions of the ODEs on each sub-interval of a given mesh using a collection of hat functions. In addition to the fact that this method is a new method for solving any nonlinear system of ODEs with high accuracy and stability that is comparable with the ARK method, it has the potential to be parallelizable and to be useful for solving biological problems that depend
on solving systems of nonlinear ODEs modeling genetic networks [10]. Since the data to identify such networks are sparse and noisy (error being 10% of values measured or larger), such biological problems can be solved quickly and with acceptable accuracy and high stability when a small number of hat functions is used as shown in (Fig. 1). The high accuracy of the ARK method, as an example, is not needed for these biological problems [23]. Our new approach achieves the required biological accuracy and if so desired, gives results as accurate as the ARK method. The basic idea of the new approach is to approximate each element in the solution of a system of nonlinear first-order ODEs by a piecewise hat function on one subinterval at a time. In this paper, this method is illustrated by solving an initial value problem of: 1) a single nonlinear first-order differential equation; 2) a system of nonlinear first-order differential equations for a genetic network describing the toggle switch [24]; and 3) a system of nonlinear first-order differential equations for a genetic network for the biological clock of the model fungal system, Neurospora crassa [25] described in the video.

The latter two initial value problems are central to systems biology. It is to be noted that parallelizing the computations of a system of nonlinear first-order differential equations with a piecewise linear function as a sum of basis functions (Hat Functions). By using FEM and a weak formulation of the approximation method, which transfers the problem from a system of ODEs to a system of algebraic equations, we find the solution for the ODEs by solving these algebraic equations using the Newton-Raphson method. Three initial value problems are considered to show the accuracy of our method. The first problem involves solving only a single nonlinear first-order differential equation, and the other two cases involve solving two systems of nonlinear first-order differential equations. In the first example the new method is as accurate as the ARK method, and the other two examples solutions by the new method are comparable with the ARK method.

A. GALERKIN ALGORITHM FOR SOLVING SYSTEMS OF ODEs
A system of initial-value problems for nonlinear first order ODEs over a solution’s interval [0 L] can be defined as

\[ y' = V(y, t); \text{ where } y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_s-1 \\ y_s \end{bmatrix}, \quad V = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{s-1} \\ V_s \end{bmatrix}, \quad y(a) = b; \]

where S is the number of variables in a system of ODEs and in particular, the number of molecular species in a genetic network.

Note that a single nonlinear first order ODE problem considered above can be solved as a special case of the above system.

An approximate solution is expanded in terms of basis functions \( \phi_j(t) \) as

\[ y_n(t) = \sum_{j=0}^{N} p_{n,j} \phi_j(t) \]  

(1)

N is the number of hat functions; \( p_{n,j} \) is a vector of unknown expansion amplitudes that we are solving for; and \( n \) labels the different molecular species; and the \( \phi_j(t) \) is a finite-element basis function (hat function) defined on a grid of time points \( t_j \) by

\[ \phi_j(t) = \begin{cases} \frac{t - t_{j-1}}{t_j - t_{j-1}}, & t_{j-1} \leq t \leq t_j \\ \frac{t_j - t}{t_{j+1} - t_j}, & t_j \leq t \leq t_{j+1} \\ 0, & \text{otherwise} \end{cases} \]

For example, the initial condition of the first species is given by \( y_1(0) = p_{1,0} \) and the solution for a specific species \( n \) at a specific time point \( j \) is given by \( y_n(t_j) = p_{n,j} \), since \( \phi_j(t_j) = 1 \) and \( \phi_k(t_j) = 0 \) for \( k \neq j \).

An alternative to hat functions is using compactly supported wavelets [26] or other types of finite-element basis functions, such as Hermite finite elements [23].

Using the residual form

\[ R_n(t) = \sum_{j=0}^{N} p_{n,j} \phi_j(t) - V_n(Y_n(t), t) \]  

(2)

we impose a weak Galerkin formulation of an approximate solution to solve for \( p_{n,j} \) as weight variables

\[ 0 = \int_{t_{k-1}}^{t_{k+1}} \phi_k(t) R_n dt. \]  

(3)

B. The Algorithm

\[ 0 = \int_{t_{k-1}}^{t_{k+1}} \phi_k(t) R_n dt \]  

(4)
be described by for solving this system of nonlinear algebraic equations can method [27]. The procedure for a Newton-Raphson scheme \( f(p) = 0 \) where \( p = p^{k+1} \)

\[
\begin{bmatrix}
  F_{k,1}(p^{k-1}, p^k, p) \\
  F_{k,2}(p^{k-1}, p^k, p) \\
  \vdots \\
  F_{k,L}(p^{k-1}, p^k, p)
\end{bmatrix}
\begin{bmatrix}
  f_1(p) \\
  f_2(p) \\
  \vdots \\
  f_L(p)
\end{bmatrix}
\]

where \( p_{1,k} \) are the solution points for the first ODE (species \( f_1 \)), \( p_{S,k} \) is the solution points for the last ODE (species \( f_S \)); \( k = 1, \ldots, L - 1 \).

Solving for \( f(p) = 0 \) is done by using the Newton-Raphson method [27]. The procedure for a Newton-Raphson scheme for solving this system of nonlinear algebraic equations can be described by

1) Setting the initial iteration value to zero and assigning initial values for each variable,

2) Calculating the Jacobian matrix \( J \),

\[
J(p) = \begin{bmatrix}
    \frac{\partial f_1}{\partial p_1} & \frac{\partial f_1}{\partial p_2} & \cdots & \frac{\partial f_1}{\partial p_n} \\
    \frac{\partial f_2}{\partial p_1} & \frac{\partial f_2}{\partial p_2} & \cdots & \frac{\partial f_2}{\partial p_n} \\
    \vdots & \vdots & \ddots & \vdots \\
    \frac{\partial f_L}{\partial p_1} & \frac{\partial f_L}{\partial p_2} & \cdots & \frac{\partial f_L}{\partial p_n}
\end{bmatrix}
\]

3) Using the Newton-Raphson scheme to solve the system of algebraic equations, which is obtained from \( f(p) = 0 \), the solution is defined by

\[
p_{i+1} = p_i - G_i \quad (6)
\]

\[
G_i = J^{-1}(p_i)f(p_i) \quad (7)
\]

\[
J(p_i)G_i = f(p_i) \quad (8)
\]

The matrix \( G_i \) can be obtained by using the Gaussian elimination method with scale partial pivoting [27].

A central finite-difference formula has been used to find an approximation to the partial derivatives of the Jacobian matrix.

For example, calculating a given value in the Jacobian is done by

\[
\frac{\partial f_T}{\partial p_u} \approx \frac{\Delta p_u}{\Delta f_T} = \frac{f_T(p_u + \delta) - f_T(p_u - \delta)}{2\delta} ; \quad T = 0 \ldots S, u = 0 \ldots N \quad (9)
\]

The \( \delta \) has been assigned a small value such as \( 1 \times 10^{-4} \), so that the final ODEs solutions for the above system do not change much by further reducing the \( \delta \) value.

4) Calculating \( f_T(p_u + \delta), f_T(p_u - \delta) \) by using the Gauss quadrature rule [27] since \( f_T \) is an integration function in this approach.

5) Iterating until the convergence of all variables is achieved. Tolerance of \( 1 \times 10^{-6} \) is sufficient for solving the above system by using the Newton-Raphson method.

6) Note that solving such a system of algebraic equations is obtained sequentially. For example, the solution of the vector \( f(p^{k+1}) \) at fixed time is obtained from the solution of the vectors \( f(p^{k-1}) \) and \( f(p^k) \).

What makes this Galerkin approach so attractive is the stability properties of the algorithm and the ability to control rigorously the error [14], [15], [18], [19].

In (Fig. 2) the solution of such systems is shown for the first initial value problem described below using our proposed method, and the solution oscillates around the exact solution. Therefore, to achieve a reasonably accurate solution with the lowest possible number of hat functions, we propose that the initial guess for the Newton-Raphson method on the oscillation time to be the average of \( p^{k-1} \) and \( p^k \) instead of just \( p^k \). We check for an oscillation on \( p^{k-2} \), \( p^{k-1} \) and \( p^k \) and we assign the average of \( p^{k-1} \) and \( p^k \) if the oscillation happens on those points. The accuracy of the final solution is based on the number of hat functions used for the solution (as the number of hat functions increases, the accuracy increases).

![Figure 2](image-url)

**FIGURE 2.** The solutions for the single nonlinear ODE and the effect of averaging on the numerical solution.

(Fig. 2) below shows part of the solution for the nonlinear single ODE during the interval \([0 \ 10]\) using 1000 hat functions. The numerical solution without averaging is shown in blue, the exact solution in red, and the solution with averaging in green.

C. THE THREE INITIAL VALUE PROBLEMS CONSIDERED ARE

1) An initial value problem of a single nonlinear first ODE

\[
y' = -y - y^2; \quad (10)
\]

with initial condition \( y(0) = 1 \)
that has the exact solution:

\[ y(t) = \frac{1}{(-1 + 2e^t)} \]  

(11)

2) An initial value problem of a system of nonlinear first order ODEs for a genetic network of the toggle switch [24] as specified by

\[
\begin{align*}
\frac{du}{dt} &= \alpha_1 / (1 + v^\beta) - u \\
\frac{dv}{dt} &= \alpha_2 / (1 + u^\gamma) - v
\end{align*}
\]

(12) (13)

where; \( u(0) = 0, v(0) = 0 \), \( \alpha_1 = 2, \alpha_2 = 4, \beta = 2, \gamma = 2 \)

3) An initial value problem of a system of nonlinear first order ODEs for a genetic network of the biological clock of \textit{Neurospora crassa} [25] as specified by

\[
\begin{align*}
\frac{df_1}{dt} &= A(f_{\text{G}} - f_1)w^n - Af_1 \\
\frac{df_r}{dt} &= S_3(f_{\text{G}} - f_1) + S_4f_1 - D_3fr \\
\frac{df_p}{dt} &= L_3fr - D_6fp \\
w' &= E_2up - D_8w - nA(f_{\text{G}} - f_1)w^n + nAf_1 - Pw(t)^m \\
u_p &= L_1u_1 - D_4up - E_2up \\
u_1' &= C_1u_0 - D_7u_1fr \\
u_0' &= V_1 - D_6u_0 - C_1u_0fp
\end{align*}
\]

All of the parameter values of this clock network problem are given in Tables 1 and 2 [25].

### III. RESULTS AND DISCUSSIONS

The new method yields solutions for fixed and specific time steps, and the accuracy is as high as the ARK method if a large number of hat functions are considered as shown in Table 3 and (Fig. 3). On the other hand, the accuracy of the solution is still acceptable for biological problems if a fewer number of hat functions is considered as shown in (Fig. 4–Fig. 5). The maximum global relative error has been computed for the single nonlinear first ODE, for the genetic network of the toggle switch, and for the genetic network of the biological clock of \textit{Neurospora crassa} using the following formula:

\[
\text{Max global relative error} (14) = \frac{\text{Max}_{k} \left| Y_{\text{Aprx}}(t_n) - Y_{\text{Exact}}(t_n) \right|}{\text{Max}_{n} \left| Y_{\text{Aprx}}(t_n) \right| + \text{Max}_{n} \left| Y_{\text{Exact}}(t_n) \right|} (15)
\]

### TABLE 1. Initial conditions at t = 0 for clock.

<table>
<thead>
<tr>
<th>Species</th>
<th>Initial Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_1 )</td>
<td>0.00040782</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>0.181388</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>1.37307</td>
</tr>
<tr>
<td>( w )</td>
<td>0.0000663227</td>
</tr>
<tr>
<td>( u_1 )</td>
<td>0.0000362815</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>0.212505</td>
</tr>
<tr>
<td>( u_0 )</td>
<td>0.0000000252030</td>
</tr>
</tbody>
</table>

### TABLE 2. Parameters values.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>0.0000462010</td>
<td>( D_r )</td>
<td>0.00285475</td>
</tr>
<tr>
<td>( A' )</td>
<td>0.566108</td>
<td>( C_r )</td>
<td>1.66501</td>
</tr>
<tr>
<td>( S_1 )</td>
<td>9.22739</td>
<td>( P )</td>
<td>3.55829</td>
</tr>
<tr>
<td>( S_2 )</td>
<td>0.00353803</td>
<td>( A_r )</td>
<td>5.57336</td>
</tr>
<tr>
<td>( S_3 )</td>
<td>0.000000136553</td>
<td>( B_r )</td>
<td>1.82043</td>
</tr>
<tr>
<td>( S_4 )</td>
<td>9.07295</td>
<td>( S_r )</td>
<td>0.0149985</td>
</tr>
<tr>
<td>( D_1 )</td>
<td>1.35911</td>
<td>( L_r )</td>
<td>0.011332</td>
</tr>
<tr>
<td>( D_2 )</td>
<td>2.77832</td>
<td>( D_{3r} )</td>
<td>0.268920</td>
</tr>
<tr>
<td>( D_3 )</td>
<td>0.223231</td>
<td>( D_{5r} )</td>
<td>0.269409</td>
</tr>
<tr>
<td>( C_1 )</td>
<td>0.0545178</td>
<td>( v_p )</td>
<td>0.120699</td>
</tr>
<tr>
<td>( L_1 )</td>
<td>59.7062</td>
<td>( u_1 )</td>
<td>0.0124268</td>
</tr>
<tr>
<td>( L_2 )</td>
<td>35.3755</td>
<td>( l_0 )</td>
<td>0.692213</td>
</tr>
<tr>
<td>( L_3 )</td>
<td>0.798222</td>
<td>( n )</td>
<td>4</td>
</tr>
<tr>
<td>( D_4 )</td>
<td>0.00000947792</td>
<td>( m )</td>
<td>4</td>
</tr>
<tr>
<td>( D_5 )</td>
<td>0.00000179706</td>
<td>( E_2 )</td>
<td>( \nu_{e}C_2 )</td>
</tr>
<tr>
<td>( D_6 )</td>
<td>0.159737</td>
<td>( f_{G} )</td>
<td>( f_{G} + f_{1} )</td>
</tr>
<tr>
<td>( D_7 )</td>
<td>0.192918</td>
<td>( V_1 )</td>
<td>( S_{1}u_{1} )</td>
</tr>
</tbody>
</table>

### TABLE 3. Errors in the solution over the time period [0 10] are shown as a function of the number of hat functions for the single ODE (Example 1). The maximum relative errors have been calculated using different numbers of hat functions and the solutions compared with the ARK method (with absolute and relative errors for ARK equal to 1 \( \times 10^{-13} \)).

<table>
<thead>
<tr>
<th># of hat functions</th>
<th>Max relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 E+02</td>
<td>0.0114304</td>
</tr>
<tr>
<td>1.0 E+03</td>
<td>0.000190308</td>
</tr>
<tr>
<td>1.0 E+04</td>
<td>3.34528064273916E-06</td>
</tr>
<tr>
<td>1.0 E+05</td>
<td>5.946046468561576E-08</td>
</tr>
<tr>
<td>1.0 E+06</td>
<td>1.05735181433986E-09</td>
</tr>
<tr>
<td>1.0 E+07</td>
<td>1.87646163866028E-11</td>
</tr>
</tbody>
</table>
where \( k = 1, 2, \ldots, S \) = number of species.

The single nonlinear first ODE has an exact solution, and for the other two cases we have considered the approximate solutions of ARK with absolute and relative errors equal to \( 1 \times 10^{-13} \) as exact solutions for them.

A. THE THREE CASES THAT HAVE BEEN CONSIDERED TO SHOW THE ACCURACY OF OUR METHOD

1) SOLVING AN INITIAL VALUE PROBLEM OF A SINGLE NONLINEAR FIRST ORDER ODE

We used the proposed algorithm to find solutions for the single nonlinear ODE with various numbers of hat functions in the fixed interval [0 10]. It has been found that the accuracy increases approximately two orders of magnitude as the number of hat functions increases by one order of magnitude as is shown in Table 3, (Fig. 3), and (Fig. 4).

2) SOLVING ODES OF THE GENETIC NETWORK OF THE TOGGLE SWITCH

We used the proposed algorithm to find solutions for the toggle switch genetic network with various numbers of hat functions and time steps over the solution interval [0 10]. The solution is comparable with the ARK method (with absolute and relative errors for ARK being \( 1 \times 10^{-13} \)) as is shown in Table 4, (Fig. 6), (Fig. 7), and (Fig. 8).

From Table 4 above, we found that the errors coming from solving the \( v \) species in the toggle switch are larger than the ones coming from the \( u \) species with varying numbers of hat functions. Thus, the maximum global relative error equals to the maximum local relative error of the \( v \) species.

The Galerkin and the ARK methods give comparable solutions for both variables \( u(t) \) and \( v(t) \) as shown in (Fig. 7) and (Fig. 8). The two solutions by different methods are virtually indistinguishable.

3) SOLVING OF THE GENETIC NETWORK OF THE BIOLOGICAL CLOCK OF NEUROSPORA CRASSA

The proposed Galerkin algorithm yields the solution for the biological clock of \textit{Neurospora crassa} genetic network with various numbers of hat functions and time steps over a larger solution interval [0 200]. The solution is comparable with the ARK method (with absolute and relative errors for ARK equal to \( 1 \times 10^{-13} \)) as it is shown in Table 5 and (Fig. 9), (Fig. 10), (Fig. 11), (Fig. 12), (Fig. 13), (Fig. 14), (Fig. 15), and (Fig. 16).

Note that the accuracy is less than the second case 2) because we use the same number of hat functions over a
TABLE 4. Errors in the solution over the time period [0 10] are shown as a function of number of hat functions for the toggle switch (Example 2). The max local and global relative errors have been calculated using different number of hat functions and the solutions compared with the ARK method (with absolute and relative errors for ARK equal to $1 \times 10^{-13}$).

<table>
<thead>
<tr>
<th>Number of Hat Functions</th>
<th>Maximum Local Relative Error ($u$ Species)</th>
<th>Maximum Local Relative Error ($v$ Species)</th>
<th>Maximum Global Relative Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1.0 \times 10^2$</td>
<td>2.82661552086085E-02</td>
<td>3.87113886066362E-02</td>
<td>3.87113886066362E-02</td>
</tr>
<tr>
<td>$1.0 \times 10^3$</td>
<td>7.088438084833E-04</td>
<td>2.17995473439916E-03</td>
<td>2.17995473439916E-03</td>
</tr>
<tr>
<td>$1.0 \times 10^4$</td>
<td>1.09812819646773E-05</td>
<td>3.77341709775271E-05</td>
<td>3.77341709775271E-05</td>
</tr>
<tr>
<td>$1.0 \times 10^5$</td>
<td>1.12067920569916E-07</td>
<td>3.56779968791140E-07</td>
<td>3.56779968791140E-07</td>
</tr>
<tr>
<td>$1.0 \times 10^6$</td>
<td>4.01110985582986E-09</td>
<td>1.28771772065253E-08</td>
<td>1.28771772065253E-08</td>
</tr>
</tbody>
</table>

TABLE 5. Maximum local relative errors of the Galerkin solution over the time period [0 200] for varied numbers of hat functions for the clock model (Example 3). The maximum local and global relative errors have been calculated using different numbers of hat functions and the solutions, compared with the ARK method (with absolute and relative errors for ARK equal to $1 \times 10^{-13}$).

<table>
<thead>
<tr>
<th># of Hat Functions</th>
<th>Max Local Rel. Error $uf$</th>
<th>Max Local Rel. Error $fr$</th>
<th>Max Local Rel. Error $fp$</th>
<th>Max Local Rel. Error $uf$ error</th>
<th>Max Local Rel. Error $urf$</th>
<th>Max Local Rel. Error $urp$</th>
<th>Max Local Rel. Error $ur0$</th>
<th>Max Global Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2.0 \times 10^2$</td>
<td>1.12E-00</td>
<td>9.30E-01</td>
<td>7.28E-01</td>
<td>1.01E+00</td>
<td>1.25E-01</td>
<td>9.35E-02</td>
<td>4.53E-02</td>
<td>1.1155125069628E+00</td>
</tr>
<tr>
<td>$1.0 \times 10^3$</td>
<td>2.34E-01</td>
<td>1.91E-01</td>
<td>3.08E-01</td>
<td>1.21E-02</td>
<td>7.47E-03</td>
<td>5.60E-03</td>
<td>3.07789282736857E-01</td>
<td></td>
</tr>
<tr>
<td>$1.0 \times 10^4$</td>
<td>5.14E-02</td>
<td>3.71E-02</td>
<td>1.66E-02</td>
<td>6.92E-02</td>
<td>2.31E-03</td>
<td>1.17E-03</td>
<td>9.91E-04</td>
<td>6.915397734511926E-02</td>
</tr>
<tr>
<td>$1.0 \times 10^5$</td>
<td>6.20E-03</td>
<td>4.55E-03</td>
<td>2.28E-03</td>
<td>8.49E-03</td>
<td>2.84E-04</td>
<td>1.44E-04</td>
<td>1.22E-04</td>
<td>8.48562558931092E-03</td>
</tr>
<tr>
<td>$1.0 \times 10^6$</td>
<td>6.77E-04</td>
<td>5.00E-04</td>
<td>2.52E-04</td>
<td>9.33E-04</td>
<td>2.76E-05</td>
<td>1.43E-05</td>
<td>1.35E-05</td>
<td>9.33489762577023E-04</td>
</tr>
</tbody>
</table>

FIGURE 7. The solutions of $u(t)$ using the Galerkin method with $1 \times 10^2$ hat functions and the ARK method over the time period [0 10] for the genetic network of the toggle switch.

FIGURE 8. The solutions of $v(t)$ using the Galerkin method with $1 \times 10^2$ hat functions and the ARK method over the time period [0 10] for the genetic network of the toggle switch.

larger solution interval [0 200] for the genetic network of the biological clock of *Neurospora crassa* instead of [0 10] for the former two cases. On the other hand, we still can see in Table 5 that a total of 10,000 hat functions is sufficient to obtain a relative error that is 0.07 or 7% or less. Again in (Fig. 9) there is a linear relation between the maximum global relative error and the number of hat functions on a log-log plot.

The solutions for this dynamical system in (Fig. 10), (Fig. 11), (Fig. 12), (Fig. 13), (Fig. 14), (Fig. 15), and (Fig. 16) using the Galerkin and ARK methods are indistinguishable using 1000 hat functions. Although the max global relative error using 1000 hat functions over the interval [0 200] is of order of 30% as it is shown in Table 5, in the figures we show that the solution is sufficiently good for biological problems.

### B. A POTENTIAL PARALLELIZATION SCHEME FOR THE GALERKIN METHOD

Unlike the ARK method, which is inherently sequential for solving systems of ODEs, the Galerkin method as stated before can be parallelized by parallelizing the Jacobian matrix’s calculation and the integration functions. This parallelization will speed up the numerical method of solving a system of nonlinear first-order differential equations. More-
The maximum global relative error (on a log scale) among the whole species (seven species) for ODEs of the genetic network of the biological clock of *Neurospora crassa* decreases approximately linearly with the log of the number of hat functions compared with the ARK method (with absolute and relative errors for ARK equal to $1 \times 10^{-13}$). Different numbers of hat functions ($2 \times 10^2$, $1 \times 10^3$, $1 \times 10^4$, $1 \times 10^5$, $1 \times 10^6$, and $1 \times 10^7$) over solution interval [0 200] are used.

The solution of $f_1(t)$ using the Galerkin and the ARK method using $1 \times 10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa*.

The solution of $f_2(t)$ using the Galerkin and the ARK methods using $1 \times 10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa*.

The solution of $f_3(t)$ using the Galerkin and the ARK methods using $1 \times 10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa*.

Over, this new method allows us to parallelize the ensemble method [25] for identifying genetic networks from real data on each variable (or species). Briefly, the ensemble method suggests that instead of identifying one unique parameterization of the model, we aim to identify an ensemble of models consistent with available experimental data and use Monte Carlo simulation techniques to generate random samples of model parameterizations (an ensemble) that represent the data well. This sampling process is captured in an animation within the associated NSF commissioned video [22]. In other words, in the ODE solving scenario a unique solution will be found by specifying the initial conditions. In contrast, in the ensemble method since we don’t know the initial conditions and other parameters values, which are required for solving systems of ODE, a Monte Carlo procedure is used to generate several initial conditions and parameters values, and while the Monte Carlo runs, it finds parameters that make the predicted solution closer to the experimental data. Finding the parameters could be done by using the Metropolis procedure [25], which minimizes the Chi-squared statistic comparing the experimental data and the predicted solution. Mainly, there are two stages in the ensemble method: the equilibration stage that is used to find parameters values that make the ODE solution converge to the experimental data and the accumulation stage, which is used to accumulate many sets of these parameters (i.e., the ensemble) that fit the experimental data well. Averaging over the ensemble allows an assessment of fit to the experimental data. Thus, averaging several solutions of the ODEs with different initial conditions that fit the experimental data will be found from a random sample of parameters that reproduce the experimental data.
Using ARK in the ensemble method implies that the system of ODEs should be re-solved for each proposed ensemble Monte Carlo updating step, and solving for the time step $t+h$ requires the solution at the prior $t$. To apply a parallelized Galerkin method version instead of the ARK method version, suppose there are $(n)$ hat functions subintervals with $(k)$ test grid points. The purpose of these test grid points is to sample the quality of the solution, for each of the subintervals as shown in (Fig. 17). On the one hand, Monte Carlo simulation will propose a set of parameters values and hat function amplitudes, which are required for solving the system of ODEs using FEM explained in this paper, for each subinterval. On the other hand, one or more subinterval(s) could be assigned to one slave processor that will solve for the system of ODEs given these parameters on its subinterval(s). Then for each of these test grid points within this subinterval(s), the method evaluates the left hand side of the differential equation and the right hand side of the differential equation and from the difference between the left hand side and the right hand side will find the residual, which is given in Equation 13. After that each processor will calculate its chi-...
square statistic and send the result back to a master processor. The master processor adds the resulting chi-squared statistics up and either accepts or rejects the proposed parameters and amplitudes based on the Metropolis procedure. The potential parallelizing procedures for the Galerkin method are either through Message Passing Interface (MPI) or MPI with Graphics Processing Units (GPUs). The result is a new parallel ensemble method, which we call the super-ensemble method [23] because it combines the Monte Carlo search for parameters with an approximation to the ODE solution (by the Galerkin Method).

IV. CONCLUSION
A new method for solving systems of initial-value problems for nonlinear First-order Ordinary Differential Equations using the Galerkin finite elements method piecewise hat functions has been developed that gives as accurate a solution as the Adaptive Runge Kutta method when a large number of hat functions are used, and acceptable accuracy for the biological problems when a fewer number of hat functions is used. On the other hand, unlike the adaptive Runge Kutta method, this method has the potential to be parallelizable and to be useful for solving biological problems that depend on solving large systems of nonlinear ODEs describing genetic networks and other systems in engineering. Moreover, this method yields solutions not for arbitrary time steps but for desirable fixed time steps.

As shown above we produce trajectories from the Galerkin Method comparable with the adaptive Runge Kutta method (with relative and absolute errors for the ARK are equal to $1 \times 10^{-13}$) by using a low number of hat functions (100 hat functions for the first two cases over the interval [0 10] and 1000 hat functions for the last case over the interval [0 200]). Developing this method to be faster than the traditional ODE solvers is a potential study in the future especially when biological problems with large networks are considered. Identifying large networks is complicated by having many parameters and limited data [28]. One solution to this problem is the use of ensemble methods [25]. A parallelized ODE solver enables faster sampling of the parameter space in ensemble methods to identify what we know (i.e., is supported across the ensemble) and what we do not know (i.e., is not supported across the ensemble) about a large system of ODEs.

REFERENCES


AHMAD AL-OMARI received the B.Sc. degree in electrical and computer engineering from Yarmouk University, Irbid, Jordan, in 2004. He was a Teaching Assistant and a Lab Engineer with the Department of Electrical and Computer Engineering, Yarmouk University, from 2004 to 2010. He worked on more than 30 different projects on computers, communication, and electronic engineering. He received a grant of $100,000 from Yarmouk University to pursue the Ph.D. degree in bioinformatics. He is currently a Research Assistant and a Ph.D. degree in bioinformatics under the supervision of Prof. J. Arnold with the University of Georgia, Atlanta, GA, USA, on systems biology and genetic networks. His current research interests include parallel computation, systems biology, finite elements method, machine learning and pattern recognition, biological circuits and gene networks, and numerical analysis.

HEINZ-BERND SCHÜTTLER received the Diploma degree in physics from Technische Universität München, Garching, Germany, in 1981, and the Ph.D. degree in physics from the University of California, Los Angeles, CA, USA, in 1984. After post-doctoral appointments with the University of California, Santa Barbara, CA, USA, and the Argonne National Laboratory, Lemont, IL, USA, he joined the Faculty of the Department of Physics and Astronomy, University of Georgia, Atlanta, GA, USA, in 1987, where he currently serves as a Professor of physics. His current research interests include the applications of computational statistical mechanics in biology and condensed matter physics. His recent notable contributions in computational biology include the development of the ensemble network simulation (ENS) method for the reconstruction of biological circuits and ENS-based methods for maximally informative next experiment design.

THIAB TAHA received the Ph.D. degree from Clarkson University, Potsdam, NY, USA, in 1982. In 1982, he joined the Computer Science Department, University of Georgia, Atlanta, GA, USA. His current research interests include scientific and distributed computing and software development for solving problems in nonlinear waves, optical fiber communication systems, and biochemical reaction networks. He has authored more than 60 research papers. He is a Senior Editor of Mathematics and Computers in Simulation Journal and the Vice President of IMACS. He received the M. G. Michael Award for Research in the Sciences at UGA in 1985 and was a Fulbright Scholar from 1995 to 1996. Currently, he is the Director of the CUDA Teaching Center at UGA.

JONATHAN ARNOLD received the B.S. degree in mathematics from Yale University, New Haven, CT, in 1975, and the M.Phil. and Ph.D. degrees in statistics from Yale University, in 1978 and 1982, respectively. In 1982, he joined the Departments of Statistics and Genetics, College of Arts and Sciences, University of Georgia, Atlanta, GA, USA, where he was a Professor of genetics, statistics, and physics and astronomy. He has authored or co-authored over 130 papers in journals, book chapters, and conference proceedings in the subjects of statistical genetics, population genetics, computational biology, and fungal genomics. He has served on the NSF Computational Biology Panel, DOE Genome Panel, and as a Charter Member on the NIH Genetic Variation and Evolution Study Section from 2005 to 2008. His current research interests include the development and identification of genetic networks of fundamental processes in the model system Neurospora crassa, i.e., Computing Life. He was elected an AAAS Fellow in 2011.

VOLUME 1, 2013