01 Aug 1994

The Multirate Method for Simulation of Power System Dynamics

J. G. Chen

Mariesa Crow
Missouri University of Science and Technology, crow@mst.edu

Recommended Citation

J. G. Chen and M. Crow, "The Multirate Method for Simulation of Power System Dynamics," *IEEE Transactions on Power Systems*, vol. 9, no. 3, pp. 1684-1690, Institute of Electrical and Electronics Engineers (IEEE), Aug 1994.
The definitive version is available at https://doi.org/10.1109/59.336087

The Multirate Method for Simulation of Power System Dynamics

M. L. Crow, Member James G. Chen
Department of Electrical Engineering
University of Missouri-Rolla
Rolla, Missouri, 65401

Abstract

In this paper, the multirate method will be introduced to analyze power system behavior for systems with widely varying time constants. Estimates for computational speed-up will be derived based on a linear system approach. The method will be applied to a small power system example and the results, both in terms of accuracy and computation time, will be compared to traditional simulation methods.

1 Introduction

Computational complexity is of timely concern in the assessment of dynamic security. Steady-state and transient stability computational methods have been studied in depth and many robust and widely-used tools are available for analyses in these time frames. Unfortunately, the development of computational tools for dynamic (mid- to long-range) analysis lags far behind. The ability to develop such methods is further complicated by the lack of appropriate models for various system components. In response, the power engineering community has tried to incorporate more detailed models into simulators. The inclusion of increasingly detailed models has further increased the complexity of the numerical calculations.

Parallel processing methods have been proposed as one means of obtaining greater computational efficiency. However, as the state of the art in compiler technology advances, much of the fine grained algorithmic parallelism may be exploited during compilation. Thus, all parallelism will be transparent and need not be explicitly algorithm dependent. For this reason, it is still imperative to pursue new algorithms which offer superior performance in a serial environment. This paper presents one such algorithm.

94 WM 173-5 PWRS A paper recommended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1994 Winter Meeting, New York, New York, January 30 - February 3, 1994. Manuscript submitted July 26, 1993; made available for printing November 29, 1993.

One family of methods which has been used for power system simulation are variable-step methods [1] [2]. Variable-step methods are integration techniques in which the time-step may vary in accordance with the fastest varying state in the system. The variable-step method is well suited for simulating dynamic systems which are primarily slow response systems, but exhibit infrequent fast decaying transients. The method is not well suited for systems in which the fast response is sustained for a large portion of the simulation interval. Unfortunately, this is the case when the power system contains induction machines under continually changing loading levels. So, while only a small portion of the entire system states are affected by fast dynamics of the induction machine loads, the integration time step must remain small, thus computational efficiency is lost. This is again the case when the power system is modelled with fast switching devices, such as FACTs devices or the converters necessary to intertie DC lines into an AC system.

Multirate methods were first proposed by Gear [3] [4] for systems with widely varying time response behavior. Recently, multirate, rather than variable-step methods have been proposed to effectively simulate power systems which exhibit time scale separation [5] [6]. Multirate methods are distinguished from variable-step methods in that the system states are aggregated into loosely coupled components which are then integrated individually with a time step dictated by the time response of the component. The coupling between components is either neglected or estimated in some way.

Computational speed-up is achieved if the number of rapidly varying components, which require small integration time steps, is small compared to the number of slowly varying components, while retaining both the gross and specific behavior of the system. The potential of this method for power system simulation is great.

In this paper, a study of multirate methods has established the viability of this type of numerical method for efficient simulation of power system dynamics. As a first approach, the multirate method has been applied to a generalized linear system which may encompass a separation into n distinct time scales. One of the main results of the preliminary linear system study is the development of a formula to estimate the possible obtainable speed-up given any number of time scales and the separation between them. The multirate method is then extended to a small nonlinear power system example which exhibits a time scale separation into two and three distinct time scales.

2 The multirate method for linear systems

A multirate method for integrating ordinary differential equations is one in which different equations are integrated by using different step sizes. The multirate method combines the robustness of a variable step method with independent step size capabilities. The principle of the multirate method is the integration of each variable with a step length which is necessary and sufficient for the required accuracy. Although multirate methods are conceptually simple, there are still many problems and open questions regarding their theory, formula, and implementation. This section discusses the use of multirate methods for solving a linear system with n ordinary differential equations.

2.1 The multirate method for three time scales

Consider a 3×3 linear time-invariant system of differential equations:

$$\dot{y}_1 = a_{11}y_1 + a_{12}y_2 + a_{13}y_3 \quad (1)$$
$$\dot{y}_2 = a_{21}y_1 + a_{22}y_2 + a_{23}y_3 \quad (2)$$
$$\dot{y}_3 = a_{31}y_1 + a_{32}y_2 + a_{33}y_3 \quad (3)$$

where $a_{ij} \in R$ for $1 \leq i \leq 3$ and $1 \leq j \leq 3$. Applying the 2nd order trapezoidal integration algorithm to discretize the system, and using the same step size h for all three functions, the following relationship is obtained:

$$M_1 y(t+h) = M_2 y(t) \quad (4)$$
$$y(t+h) = M_1^{-1} M_2 y(t) \quad (5)$$
$$y(t+h) = M y(t) \quad (6)$$

where

$$y(t+h) = [y_1(t+h) \;\; y_2(t+h) \;\; y_3(t+h)]^T$$
$$y(t) = [y_1(t) \;\; y_2(t) \;\; y_3(t)]^T$$

and

$$M = \begin{bmatrix} 1 & -\frac{ha_{12}}{2-ha_{11}} & -\frac{ha_{13}}{2-ha_{11}} \\ -\frac{ha_{21}}{2-ha_{22}} & 1 & -\frac{ha_{23}}{2-ha_{22}} \\ -\frac{ha_{31}}{2-ha_{33}} & -\frac{ha_{32}}{2-ha_{33}} & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \frac{2+ha_{11}}{2-ha_{11}} & \frac{ha_{12}}{2-ha_{11}} & \frac{ha_{13}}{2-ha_{11}} \\ \frac{ha_{21}}{2-ha_{22}} & \frac{2+ha_{22}}{2-ha_{22}} & \frac{ha_{23}}{2-ha_{22}} \\ \frac{ha_{31}}{2-ha_{33}} & \frac{ha_{32}}{2-ha_{33}} & \frac{2+ha_{33}}{2-ha_{33}} \end{bmatrix} = M_1^{-1} M_2 \quad (7)$$

The matrix M may be used to analyze the numerical stability of the method as well as the computational complexity. For example, any integration method is guaranteed to be numerically stable if the eigenvalues of the matrix M lie strictly within the unit circle in the complex plane. In addition, the computational effort required to compute the matrix M is directly related to the computational effort required to solve the linear system of equations for each time step.

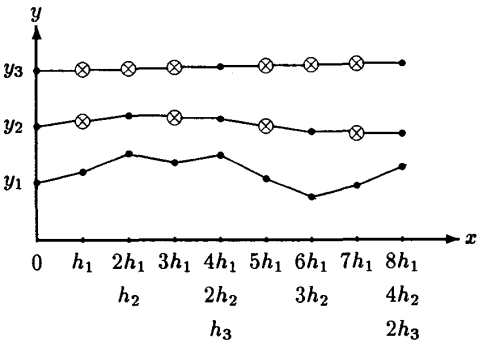

Figure 1: Three time scale example

Now consider three linear functions which may be discretized with three different step sizes. For simplicity, let $C_i!$ be defined as

$$C_i! = C_i \cdot C_{i-1} \cdot C_{i-2} \cdot \ldots \cdot C_2 \cdot C_1$$

for $1 \leq i \leq n$ and $C_i \triangleq \frac{h_i}{h_{i-1}}$. Note that this expression is different from the mathematical expression "$C!$". Also note that $C_1 = 1$. Without loss of generality, it can be assumed that

$$h_1 \leq h_2 \leq h_3 \leq \ldots \leq h_{n-1} \leq h_n$$

and

$$h_2 = C_2 h_1 = C_2 C_1 h_1 = C_2! h_1$$
$$h_3 = C_3 h_2 = C_3 C_2 C_1 h_1 = C_3! h_1$$
$$\vdots$$
$$h_n = C_n h_{n-1} = C_n C_{n-1} \ldots C_2 C_1 h_1 = C_n! h_1$$

The three time scale case is illustrated in Figure 1, where $y_1(t)$ is the fastest varying state and $y_3(t)$ is the slowest varying state. Note that $h_2 = 2h_1$ and $h_3 = 2h_2 = 4h_1$, thus $C_2 = 2, C_3 = 2$, and $C_3! = 4$.

Consider the calculation of the system states at time $t = 4h_1 = 2h_2 = h_3$:

$$\begin{aligned} y_3(t+h_3) = \; & y_3(t) \\ & + \frac{h_3}{2}\left[a_{31}\left(y_1(t+4h_1) + y_1(t)\right)\right. \\ & + a_{32}\left(y_2(t+2h_2) + y_2(t)\right) \\ & \left. + a_{33}\left(y_3(t+h_3) + y_3(t)\right)\right] \end{aligned} \quad (8)$$

$$\begin{aligned} y_2(t+2h_2) = \; & y_2(t+h_2) \\ & + \frac{h_2}{2}\left[a_{21}\left(y_1(t+4h_1) + y_1(t+h_2)\right)\right. \\ & + a_{22}\left(y_2(t+2h_2) + y_2(t+h_2)\right) \\ & \left. + a_{23}\left(y_3(t+h_3) + y_3(t+h_2)\right)\right] \end{aligned} \quad (9)$$

$$\begin{aligned} y_1(t+4h_1) = \; & y_1(t+3h_1) \\ & + \frac{h_1}{2}\left[a_{11}\left(\dot{y}_1(t+4h_1) + y_1(t+3h_1)\right)\right. \\ & + a_{12}\left(y_2(t+2h_2) + y_2(t+3h_1)\right) \\ & \left. + a_{13}\left(y_3(t+h_3) + y_3(t+3h_1)\right)\right] \end{aligned} \quad (10)$$

Note that each variable is integrated with the step size which is appropriate for its time response. Note also that not all states are available at the desired time (those marked by $\otimes$ in Figure 1) and must be approximated. The simplest approximation is a linear interpolation between calculated values. Thus

$$\hat{y}_2(t+kh_1) = \frac{k}{C_2}y_2(t+h_2)+\left(1-\frac{k}{C_2}\right)y_2(t) \qquad 1\le k\le C_2$$

and

$$\hat{y}_3(t+kh_1) = \frac{k}{C_3!}y_3(t+h_3)+\left(1-\frac{k}{C_3!}\right)y_3(t) \qquad 1\le k\le C_3$$

After repeated interpolations and substitutions, the following expression for $y_1(t+h_3)$ may be obtained:

$$\begin{aligned} y_1(t+h_3) &= \beta_1^{C_3!}y_1(t)+\frac{h_1a_{12}}{2-h_1a_{11}}\sum_{j=1}^{C_3}\sum_{k=1}^{C_2}\beta_1^{C_2(C_3-j)}\beta_1^{C_2-k}\cdot \\ &\left[\left(2-\frac{2k-1}{C_2}\right)y_2(t+(j-1)h_2)+\left(\frac{2k-1}{C_2}\right)y_2(t+jh_2)\right] \\ &+\frac{h_1a_{13}}{2-h_1a_{11}}\sum_{k=1}^{C_3!}\beta_1^{C_3!-k}\cdot \\ &\left[\left(2-\frac{2k-1}{C_3!}\right)y_3(t)+\left(\frac{2k-1}{C_3!}\right)y_3(t+h_3)\right] \end{aligned} \tag{11}$$

where

$$\beta_i=\frac{2+h_ia_{ii}}{2-h_ia_{ii}}$$

Similarly, expressions for $y_2(t+h_3)$ and $y_3(t+h_3)$ can be found:

$$\begin{aligned} y_2(t+h_3) &= \beta_2^{C_3}y_2(t) \\ &+\frac{h_2a_{21}}{2-h_2a_{22}}\sum_{k=1}^{C_3}\beta_2^{C_3-k}[y_1(t+(k-1)h_2)+y_1(t+kh_2)] \\ &+\frac{h_2a_{23}}{2-h_2a_{22}}\sum_{k=1}^{C_3}\beta_2^{C_3-k}\cdot \\ &\left[\left(2-\frac{2k-1}{C_3}\right)y_3(t)+\left(\frac{2k-1}{C_3}\right)y_3(t+h_3)\right] \end{aligned} \tag{12}$$

and

$$\begin{aligned} y_3(t+h_3) &= \beta_3y_3(t)+\frac{h_3a_{31}}{2-h_3a_{33}}[y_1(t)+y_1(t+h_3)] \\ &+\frac{h_3a_{33}}{2-h_3a_{33}}[y_2(t)+y_2(t+h_3)] \end{aligned} \tag{13}$$

Note that since the trapezoidal method is an implicit method, there is an implicit dependence on the state variables at both previous and current time steps in addition to the dependence introduced by the interpolation. In the linear case, it is possible to apply a reduction process to find a closed form for the matrix M (introduced in equation (6)) which will explicitly relate $[y_1(t+4h_1)\ y_2(t+2h_2)\ y_3(t+h_3)]^T$ to $[y_1(t)\ y_2(t)\ y_3(t)]^T$. This process will be discussed in the next section for a generalized linear system of n distinct time scales.

2.2 The multirate method for n time scales

Consider a system of n linear functions:

$$\dot{y}_1 = a_{11}y_1+a_{12}y_2+\ldots+a_{1n}y_n \tag{14}$$

$$\dot{y}_2 = a_{21}y_1+a_{22}y_2+\ldots+a_{2n}y_n \tag{15}$$

$$\vdots$$

$$\dot{y}_n = a_{n1}y_1+a_{n2}y_2+\ldots+a_{nn}y_n \tag{16}$$

where where $a_{ij}\in R$ for $1\le i\le n$ and $1\le j\le n$. Following the same approach discussed for the three time scale system, the following expression may be obtained for any $1\le i\le n$:

$$\begin{aligned} y_i(t+h_n) &= \beta_i^{\frac{C_n!}{C_i!}}y_i(t) \\ &+\sum_{m=1}^{i-1}\sum_{k=1}^{\frac{C_n!}{C_i!}}\beta_i^{\frac{C_n!}{C_i!}-k}\frac{h_ia_{im}}{2-h_ia_{ii}}\cdot \\ &[y_m(t+(k-1)h_i)+y_m(t+kh_i)] \\ &+\sum_{m=i+1}^{n}\sum_{j=1}^{\frac{C_n!}{C_m!}}\sum_{k=1}^{\frac{C_m!}{C_i!}}\left\{\beta_i^{\frac{C_m!}{C_i!}\left(\frac{C_n!}{C_m!}-j\right)}\beta_i^{\frac{C_m!}{C_i!}-k}\frac{h_ia_{im}}{2-h_ia_{ii}}\right. \\ &\left[\left(2-\frac{C_i!(2k-1)}{C_m!}\right)y_m(t+(j-1)h_m)\right. \\ &\left.\left.+\left(\frac{C_i!(2k-1)}{C_m!}\right)y_m(t+jh_m)\right]\right\} \end{aligned} \tag{17}$$

As in the previous discussion, it is possible to apply a reduction process to find a closed form for the matrix M which relates $y(t+h)=My(t)$.

The form of the matrix M is important for a variety of reasons. Firstly, the numerical stability of any integration method can be ascertained by computing the eigenvalues of M. This aspect of the matrix M is not discussed in this paper, but instead the matrix M will be used to estimate the potential savings gained from the multirate method. Note from the discussion of Section 2.1 that at any point in time, only a portion of the entire system is calculated at any given time. Thus rather than solving an $n\times n$ system at each step (requiring on the order of n^3 multiplications and divisions), considerably less computation is involved. Consider once again the three time scale example of Figure 1. A good estimate of the computational burden may be obtained by calculating the multiplications and divisions required to solve for the matrix M, recalling that $M=M_1^{-1}M_2$. The number of multiplications and divisions is $\frac{n^3}{3}+n^2-\frac{n}{3}$ for the LU factorization of M_1 and n^2 for the multiplication of $M_2y(t)$ for a total of $\frac{n^3}{3}+2n^2-\frac{n}{3}$. Therefore, the usual integration approach would require $8\cdot(\frac{3^3}{3}+2\cdot3^2-\frac{3}{3})=208$ multiplications and divisions. In the multirate method, the solution of $M_2y(t)$ requires $i\cdot(2n-i)$ multiplications. For example, to solve for $y_1(t+h_1)$, M_2 is a 1×5 matrix, since the right-hand-side vector would contain $[x_1(t)\ x_2(t)\ x_3(t)\ x_2(t+h_2)\ x_3(t+h_3)]^T$. In the example system, the number of multiplications and divisions would be $4\cdot(1+1\cdot5)+2\cdot(\frac{2^3}{3}+2^2-\frac{2}{3}+4\cdot2)+2\cdot(\frac{3^3}{3}+3^2-\frac{3}{3}+3\cdot3)=104$ for a computational speedup of 2:1. The estimated speed-up could be further increased if a larger step size ratio were possible, or if the ratio of the "fast" states to "slow" and "medium" states were small. This

argument can be extended to an n-dimensional system with n states, each possibly varying at a different rate. Once again assuming the computational expense of solving an $n_i \times n_i$ system to be $\sim n_i^3$ and neglecting interpolation costs, then to solve a system for N times steps where $N = C_i!$ takes:

$$\begin{array}{lll} \text{I:} & N\left(\frac{n^3}{3}+2n^2-\frac{n}{3}\right) & \text{constant step size } h \\ \text{II:} & \left(\frac{n^3}{3}+2n^2-\frac{n}{3}\right)+N\sum_{i=1}^{n-1} & \text{multirate method} \\ & \left(\frac{1}{C_i!}-\frac{1}{C_{i+1}!}\right)\left[\frac{i^3}{3}+2ni-\frac{i}{3}\right] & \text{with } h_1 = h \end{array}$$

For example, suppose $n = 400$, and $C_i = 10$ for $i = 100, 200, 300$ and $C_i = 1$ otherwise. This separates the 400 states into 4 distinct time separations with 100 states in each time frame. Recall that $N = C_{400!} = 1000$, then the computation burden may be estimated for the usual method (I) and the multirate method (II):

I: 21653×10^6

II: 717×10^6

for a speed up of approximately 30:1. Note that this measure is an estimate of the speedup and actual obtained savings may be different due to sparsity savings. This estimate is based on $\sim n^3$ for LU factorization, which may be decreased to $\sim n^2$ if sparsity is exploited.

The speedups calculated above may seem rather insignificant given the time-scale separation between the states, but what one needs to recall, is that the accuracy of the solution is being preserved by the interpolation process, and that the multirate method has an accuracy on the same order as the original integration method. This aspect of the multirate method will be discussed in further detail in the following section.

3 The multirate method for nonlinear systems

In this section, the multirate method, which was introduced for linear systems in the previous section, will be extended to the nonlinear case. Indeed, all practical power systems are nonlinear, thus a discussion of the application of the multirate method to nonlinear systems is in order. The basic approach for applying the multirate method to a nonlinear system is the same as with a linear system.

Consider a system of n nonlinear differential equations:

$$\dot{y}_1 = f_1(y_1, y_2, \ldots, y_n, t) \quad (18)$$

$$\dot{y}_2 = f_2(y_1, y_2, \ldots, y_n, t) \quad (19)$$

$$\vdots$$

$$\dot{y}_n = f_n(y_1, y_2, \ldots, y_n, t) \quad (20)$$

Each state may be integrated with a different step size h_i, where $h_1 \leq h_i \leq h_n$. With the linear system it was possible to find a closed form of the matrix M which related $y(t + h_n)$ to $y(t)$ directly, from which stability and computational information can be derived. This was possible since the interpolative relationship between variables could be known beforehand, due to the time invariant nature of the coefficients of the matrix A. In the nonlinear case, in order to obtain an interpolative approximation to the slow variables at the time points of interest, the final point of the slow variable must be first predicted and then the mid-points may be interpolated. These points will be iteratively updated via a nonlinear solution process, such as a Newton-Raphson iteration. The more accurate the prediction and interpolation algorithm, the fewer number of iterations are required to achieve the desired accuracy. Thus the computational burden of a nonlinear system will be greater than for a linear system of the same dimension. The speed-up ratio predicted by the linear analysis in Section 2.2 should still provide a rough estimate for the nonlinear system however, since both the constant step size case and the multirate case will require approximately the same number of Newton-Raphson iterations per time step if the prediction/interpolation steps are accurate enough.

Discerning the stability of the multirate method in the nonlinear case becomes quite complex. It is not nearly as straightforward as computing the eigenvalues of the matrix M, as in the linear case. An exhaustive discussion of the stability criteria is not appropriate for this paper. However, it will be discussed in context with the following power system example.

4 An Illustrative Example

The multirate method can be well illustrated with a small synchronous machine model, which is given in [7]

$$\frac{1}{\omega_s}\frac{d\Psi_d}{dt} = -\frac{R_a}{L'_d}\Psi_d + \frac{R_a}{L'_d}E'_q + \frac{\omega}{\omega_s}\Psi_q + V\sin\delta \quad (21)$$

$$\frac{1}{\omega_s}\frac{d\Psi_q}{dt} = -\frac{R_a}{L'_d}\Psi_q - \frac{R_a}{L'_d}E'_d - \frac{\omega}{\omega_s}\Psi_d + V\cos\delta \quad (22)$$

$$T'_{q0}\frac{dE'_d}{dt} = -\frac{L_q}{L'_q}E'_d - \frac{(L_q - L'_q)}{L'_q}\Psi_q \quad (23)$$

$$T'_{d0}\frac{dE'_q}{dt} = -\frac{L_d}{L'_d}E'_q - \frac{(L_d - L'_d)}{L'_d}\Psi_d + E_{xc} \quad (24)$$

$$\frac{d\delta}{dt} = \omega - \omega_s \quad (25)$$

$$\frac{2H}{\omega_s}\frac{d\omega}{dt} = T_m + \left(\frac{1}{L'_q} - \frac{1}{L'_d}\right)\Psi_d\Psi_q + \frac{1}{L'_q}\Psi_d E'_d + \frac{1}{L'_d}\Psi_q E'_q \quad (26)$$

The model contains the two stator/network flux linkages Ψ_d and Ψ_q, the voltage proportional to the field flux linkage E'_q, the voltage proportional to the damper winding flux linkage E'_d, and the electromechanical pair δ and ω. The data for this example is given in Table 1[7]. The applied disturbance is a reduction of the infinite bus voltage V from 1.0 to 0.8pu at time t=0.2 sec.

Table 1: Example Data

$R_a = 0$	$w_s = 377$ rad/sec	V=1.0 pu
$L_d = 0.9$ pu	L'_d=0.2 pu	$T'_{d0} = 5.00$ sec
$L_q = 0.7$ pu	L'_q=0.2 pu	$T'_{q0} = 0.13$ sec
$H = 5.0$ sec	$E_{xc} = 0.8$pu	$\delta_0 = 30°$

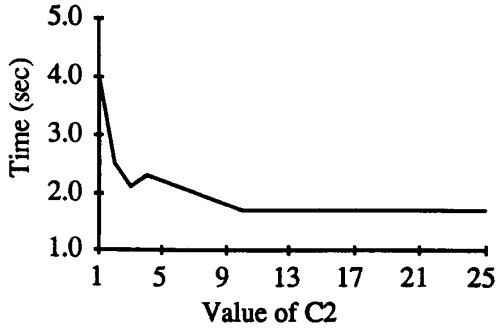

Figure 2: Computation time vs. C_2

The flux linkage variables Ψ_d and Ψ_q both exhibit highly oscillatory, negligibly damped responses with a frequency close to 60Hz. These are the "fast" variables. In order to effectively capture the dynamics of these oscillatory variables, the integration step size must be small enough to accurately reproduce the shape of the sinusoid. A sinusoid waveform can be nominally reconstructed from 8 points per cycle, but 16 points per cycle is preferable. For this reason, in this example, the smallest integration step size is chosen to be

$$h = \frac{(\text{frequency})^{-1}}{16} = \frac{\frac{1}{60}}{16} \cong 0.001s$$

If a variable step integration method were used to simulate this system, all the system variables would be discretized using the same time step $h = 0.001$ seconds. This time step could not be increased for better computational efficiency, because the oscillations are only negligibly damped, and thus would not decay in the time frame of interest, thus all advantages to using a variable step method are lost. The multirate method, however, is well suited for this type of problem. This system has a well-defined separation of time responses.

4.1 Two time scales

In the first example, a two-time scale separation will be considered, that is, the variables $[E'_d, E'_q, \delta, \omega]$ will be "slow" variables compared to $[\Psi_d, \Psi_q]$, and will be integrated with a step size $h_2 = C_2 h_1$, where $1 \leq C_2$. The results of this comparison are summarized in Table 2 which gives the maximum percent error over the simulation interval for each variable using the multirate method as compared to a constant step size method with $h_1 = 0.001s$. The computation time required as a function of C_2 is shown in Figure 2.

The relationship of time vs C_2 is not unexpected. When $C_2 = 1$, the computational burden will be dominated by the solution of the full 6×6 system. As C_2 increases, the dominance will shift to the 2×2 fast system, until the point where the infrequent computation of the full system is a small portion of the overall computation. The slight increase in computation time at $C_2 = 4$ is due to the increase of required Newton-Raphson iterations to achieve the required convergence accuracy. Far better speed-ups would be expected from a larger example where sparsity could be exploited or where the ratio of fast to total variables is smaller than one-third, as in this example.

Table 2: C_2 vs. Percent Error and Computation Time

C_2	Ψ_d	Ψ_q	E'_d	E'_q	δ	ω	cpu
1							4.0
2	0.34	0.64	0.48	0.01	0.06	0.00	2.5
3	0.80	1.55	1.05	0.03	0.17	0.00	2.1
4	1.27	2.49	1.92	0.05	0.32	0.01	2.3
5	1.58	3.23	2.80	0.08	0.52	0.02	2.2
6	1.69	3.52	3.77	0.11	0.77	0.03	2.1
7	1.49	3.15	4.66	0.15	1.08	0.03	2.0
8	0.87	3.20	5.29	0.19	1.47	0.04	1.9
9	1.11	3.98	6.39	0.25	1.96	0.05	1.8
10	2.87	5.04	8.54	0.33	3.59	0.06	1.7
11	4.91	8.23	11.53	0.45	3.43	0.07	1.7
12	7.04	12.23	15.25	0.61	4.57	0.12	1.7
13	8.94	16.34	20.62	0.83	6.39	0.16	
14	10.34	21.09	28.50	1.15	9.45	0.22	
15	11.42	31.91	43.13	1.66	15.60	0.32	
16	20.52	88.97	76.65	2.61	44.48	0.70	
17	47.42	136.50	48.24	5.21	68.24	0.44	
18	14.60	57.01	63.58	1.97	27.55	0.38	
19	10.40	29.87	39.93	1.39	13.11	0.26	
20	7.61	18.50	27.07	0.96	7.93	0.18	
21	5.21	11.86	18.47	0.67	5.26	0.13	
22	3.04	7.30	12.55	0.43	3.13	0.09	
23	1.32	3.41	8.26	0.24	1.49	0.06	
24	0.17	0.53	5.91	0.09	0.29	0.04	

In addition, this example illustrates several interesting points. Note that the solution error increases slowly until $C_2 = 10$ and then increases rapidly until the maximum percentage error is reached when $C_2 = 17$. At this point, there is approximately a 68% error in δ, a 48% error in Ed', and a 136% error in Ψ_q. However, if C_2 is increased to 24, the errors decrease to a point comparable to a small value of C_2 again. The waveforms of $\delta(t)$ for $C_2 = 1, C_2 = 8, C_2 = 17$, and $C_2 = 24$ are shown in Figures 3-6 respectively. The large error in $C_2 = 17$ can be attributed to the "sampling rate" of the fast subsystem by the slow subsystem. Recall that the slow subsystem is only integrated once every C_2 time steps. Since the fast subsystem is nearly oscillatory with periodicity $\frac{1}{60Hz} = 0.0167$ seconds, a time step of $C_2 = 17$ will sample the sinusoid at approximately the same point in each cycle, thus the fast subsystem will appear to be nearly constant, instead of rapidly varying. This poorly chosen sampling rate leads to the erroneous results.

4.2 Three time scales

In the two time scale case, for low values of C_2 $(1 \leq C_2 \leq 10)$ note that the variable E'_d has the largest error of all variables. This is due to the designation that E'_d is a "slow" variable, when in fact it may be considered a "medium" variable, not quite fast, but requiring more frequent updating than the slow variables. This necessitates the introduction of a third time scale. A selection of three time scale results are presented in Ta-

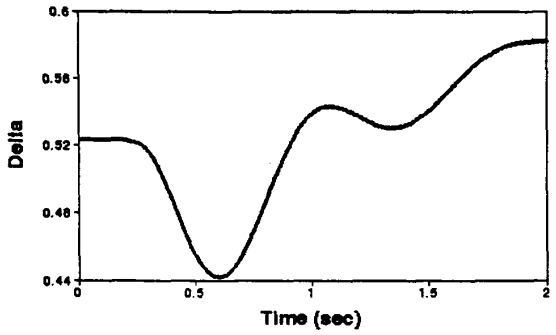

Figure 3: Standard output waveform for δ ($C_2 = 1$)

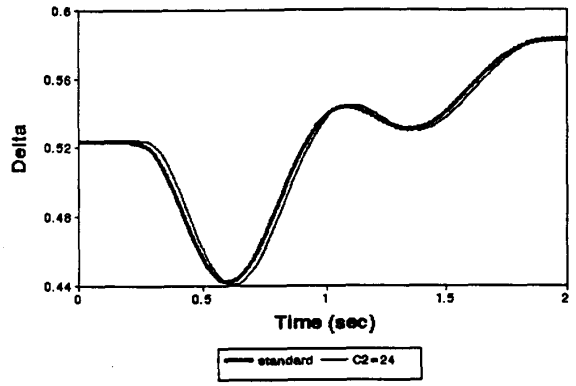

Figure 6: Output waveform for δ ($C_2 = 24$)

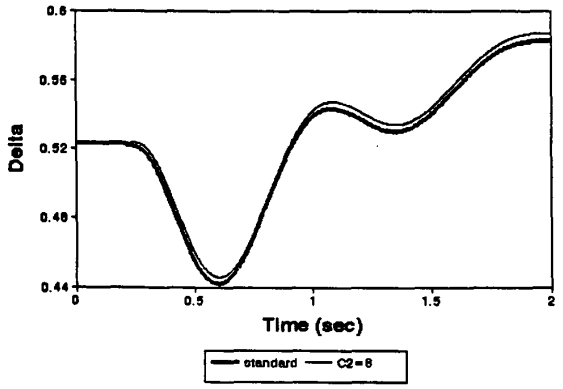

Figure 4: Output waveform for δ ($C_2 = 8$)

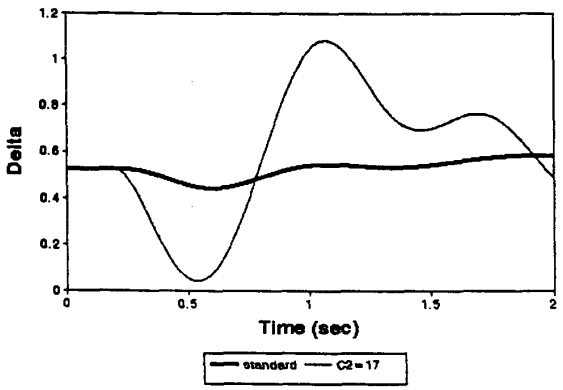

Figure 5: Output waveform for δ ($C_2 = 17$)

Table 3: C_2 & C_3 vs. Percent Error and CPU Time

C_2	C_3	Ψ_d	Ψ_q	E'_d	E'_q	δ	ω	cpu
1	1							4.0
2	2	1.30	2.31	0.71	0.03	0.36	0.01	2.3
2	3	1.69	3.00	1.18	0.05	0.88	0.03	2.3
2	4	0.75	3.39	1.86	0.08	1.66	0.04	2.1
2	5	1.50	5.91	3.14	0.15	3.07	0.07	2.0
2	6	6.46	10.37	5.19	0.26	5.45	0.13	1.9
2	7	9.99	20.93	10.10	0.52	11.04	0.25	1.9
2	8	30.19	125.34	51.49	2.69	63.70	0.92	
2	9	19.31	98.51	42.66	1.19	47.38	0.62	
2	10	6.64	21.12	8.42	0.43	10.08	0.23	
3	2	1.67	3.09	1.61	0.06	0.86	0.03	2.2
3	3	0.98	4.67	2.95	0.11	2.35	0.06	1.9
3	4	6.47	10.30	5.65	0.26	5.44	0.13	1.8
3	5	11.60	37.80	17.31	0.88	19.17	0.38	
3	6	19.32	98.75	42.55	1.20	47.49	0.62	
3	7	4.39	12:86	5.51	0.28	6.02	0.15	
4	2	0.73	3.34	2.98	0.09	1.62	0.04	1.9
4	3	6.50	10.20	6.33	0.26	5.42	0.13	1.8
4	4	30.28	125.66	52.38	2.71	63.87	0.92	
4	5	6.68	21.33	0.28	0.43	10.17	0.23	
5	2	2.56	5.77	4.89	0.14	2.99	0.07	1.8
5	3	11.65	37.89	18.71	0.88	19.2	0.38	1.8
5	4	6.71	21.48	9.84	0.43	10.24	0.23	

ble 3. Recall that the medium step size $h_2 = C_2 h_1$ and the slow step size $h_3 = C_3 C_2 h_1 = C_3! h_1$. Consider the two time scale example for $C_2 = 8$, where the variables $[E'_d, E'_q, \delta, \omega]$ are only integrated every $8h_1$. The error for E'_d is 5.3%, the error for Ψ_q is 3.2%, and the δ error is 1.4%. In the three time scale example $C_2 = 2, C_3 = 4$, thus $h_2 = 2h_1$, and $h_3 = 4h_2 = 8h_1$, and the variables $[E'_q, \delta, \omega]$ are only integrated every $8h_1$, whereas E'_d is integrated every $2h_1$. In this case the error in E'_d is reduced to 1.8%, while the errors in Ψ_q and δ remain fairly constant. Note however, that the computational time is increased slightly from 1.9 CPU to 2.1 CPU, thus trading accuracy for efficiency. Even in the case where $C_2 = 4$ and $C_3 = 2 (h_2 = 4h_1, h_3 = 8h_1)$, the error in E'_d is still reduced to about 3% with all other errors remaining constant. In this case, the computational time is 1.9 CPU, the same as the two time scale case. These results are intuitive; the more frequently a "medium" variable is calculated, the greater the accuracy, but often at the expense of computational efficiency.

5 Conclusions

In this paper, the multirate method was discussed in context with both linear and nonlinear systems. The results obtained from the small synchronous machine example for both the two and the three time scale example indicate that the multirate method holds great potential for being an efficient method for power system dynamic simulation. This is especially true in the case where a power system contains a small proportion of "fast" devices, such as DC lines, induction machines, or FACTs devices. The multirate method is extremely well suited for this type of system analysis.

Further study is underway on this method to better predict when the method may not give accurate results (as in the example when $C_2 = 17$). Also under consideration is the inclusion of algebraic constraints and better approximation strategies to increase the accuracy of the slow variable interpolation.

6 Acknowledgements

The support of this project by NSF (grant ECS-9257208) is gratefully acknowledged.

References

[1] J. Deuse and M. Stubbe, "Dynamic simulation of voltage collapse," to be published in *IEEE Transactions on Power Systems,* paper # 92 SM 396- 2 PWRS.

[2] A. Kurita, et. al. "Multiple time-scale power system dynamic simulation," *IEEE Transactions on Power Systems,*, vol. 8, no. 1, pp. 216-223, February 1993.

[3] C. W. Gear, "Multirate Methods for Ordinary Differential Equations," Department of Computer Science, University of Illinois at Urbana-Champaign, Report UIUCDCS-F-74-880, September 1974.

[4] C. W. Gear "Automatic Multirate Methods for Ordinary Differential Equations," Department of Computer Science, University of Illinois at Urbana-Champaign, Report UIUCDCS-R-80-1000, January 1980.

[5] M. L. Crow, "Computational algorithms for the multirate simulation of electric power system dynamics," EPRI Report #TR-101795. *Proceedings: EPRI/NSF Workshop on Application of Advanced Mathematics Applied to Power Systems.* pp. 2-25 – 2-28, April 1993.

[6] M. L. Crow and M. Ilić, "The parallel implementation of the waveform relaxation method for transient stability simulations," *IEEE Transactions on Power Systems,* vol. PWRS-5, no. 3, pp. 922-932, August 1990.

[7] P. W. Sauer, S. Ahmed-Zaid, and P. V. Kokotovic, "An integral manifold approach to reduced order modeling of synchronous machines," *IEEE Transactions on Power Systems,* vol. 3, no. 1, pp. 17-23, February 1988.

M. L. Crow (S'83, M'90) was born in Marion, Indiana. She received her B.S.E. in electrical engineering from the University of Michigan in 1985, her M.S. in electrical engineering from the University of Illinois in 1986, and her Ph.D. in electrical engineering in the area of Power and Energy Systems from the University of Illinois in 1989. She spent a year as an assistant professor at Arizona State University in the Electrical Engineering Department, before joining the University of Missouri–Rolla in 1991. She received an NSF Research Initiation award in 1991 and the National Science Foundation Young Investigator (NYI) award in 1992. Her primary area of research interest is computational algorithms, simulation, and analysis of power system dynamics. She is a member of the IEEE PES System Dynamic Performance Subcommittee and the IEEE PES Task Force on Software Engineering.

James G. Chen was born in Shanghai, China. He received the B.E. from Tong Ji University, China in 1982 and his M.S. from Central Missouri State University in 1991. From 1982 to 1989, he was an engineer at the Shanghai Institute of Process Automation Instrumentation. In 1987, he received the Scientific Advancement award from the Shanghai Municipal Government and the Award of Distinguishment from the State Department of PRC in 1988. He is currently a Ph.D. student in electrical engineering at the University of Missouri-Rolla. His area of interest is computational algorithms and power system stability. He is a member of the American Mathematical Society.