A Comparison of two 4th-Order Numerical Ordinary Differential Equation Methods Applied to the Rabinovich-Fabrikant Equations

Clyde Meador
Marshall University
Email: meador16@marshall.edu

Advisor: Scott Sarra
Marshall University
Email: sarra@marshall.edu

November 29, 2009

Abstract

The Rabinovich-Fabrikant system is a chaotic system of nonlinear ordinary differential equations in three dimensions. Using the Local Iterative Linearization method and a Runge-Kutta method (both of fourth order and identical step-size) phase plots are generated and compared. Issues concerning the numerical approximation of chaotic systems are explored.

1 Introduction

The Rabinovich-Fabrikant system [1] (hereafter, the RF system) is a chaotic dynamical system of three ordinary differential equations (ODEs) in three variables and two constant parameters, as follows:

$$\begin{aligned} x' &= y(z-1+y^2)+ax \\ y' &= x(3z+1-x^2)+ay \\ z' &= -2z(b+xy) \end{aligned}$$

where the constant parameters a, $b > 0$. The RF system models self-modulation of waves in nonequilibrium media (specifically, dissipative media). For more information on the physics of the RF system, consult [1]. Later in the numerical experiments we will see that the set of parameters (a, b) has dramatic effects on the behavior of the system. This system is lesser known than others such as the Rössler and Lorenz systems [2, 3], but there is numerical research in the literature for the RF system. Additionally, statistical evidence that the RF system is chaotic was published by Luo *et al* in [4]. Much published numerical research of the RF system uses an algorithm called Local Iterative Linearization (LIL) for numerical integration of the system [4, 5, 6]. In this paper, 4th order LIL and Runge-Kutta (RK) methods of identical step-size are implemented in the MATLAB computing environment.

1.1 Chaos and other concepts

Some of the terms and concepts used in this paper may be unfamiliar to those who have not yet taken courses in numerical analysis or differential equations, so a brief overview of some concepts in the numerical study of chaos follows.

A *dynamical system* is a system whose state evolves according to a fixed, deterministic rule. A *nonlinear* equation has powers and functions of x other than x^1 and x^0. *Chaos* receives a working definition from Devaney [7] in three parts:
1) The system has the famous property of *sensitive dependence on initial conditions*, that is: there exists a $\delta > 0$ such that if you pick an point x_0 and a neighborhood around x_0, there is a point x_1 in that neighborhood such that the orbits of x_0 and x_1 will eventually separate by δ.
2) The system is *topologically mixing* - if you pick two open subsets of the domain, the orbit of one set will eventually intersect the other.
3) The system has *dense periodic orbits* - for any point in the domain and any neighborhood N of that point, there is at least one point from a periodic orbit in N.

2 Numerical methods for solving $x' = f(x)$

Numerical methods for differential equations are used to generate close approximations to the exact solution of problems which are difficult or impossible to solve analytically. They are essential to examine behavior of chaotic systems. Next we introduce Euler's method in order to later illustrate some key concepts. Afterward we describe the LIL and RK methods.

2.1 Euler's method

Euler's method is the following:

$$x_{n+1} = x_n + hf(x_n) \quad n = 0, 1, 2, ... \tag{1}$$

where h is the timestep.

Beginning with an initial condition x_0 at time t_0, we move forward repeatedly by timestep h. Euler's method is based on the Taylor expansion of x_{n+1}, as shown below:

$$x(t_{n+1}) = x(t_n + h) = x(t_n) + hx'(t_n) + \frac{1}{2}h^2x''(t_n) + \mathcal{O}(h^3) \tag{2}$$

where $\mathcal{O}(h^3)$ indicates the remainder of the series expansion. As h approaches zero, $\mathcal{O}(h^3)$ is no larger than Kh^3 for some fixed K.

Euler's method uses the first two terms of the Taylor expansion of x_{n+1}. Therefore, the truncation error of Euler's method is the difference between the numerical result and the value of the exact expansion, or:

$$\frac{1}{2}h^2y''(t_n) + \mathcal{O}(h^3). \tag{3}$$

2.2 Local Iterative Linearization

As described in [5], the 4th-order Local Iterative Linearization method is:

$$\begin{aligned} x_n &= 2x_{n-1} - \frac{8}{5}x_{n-2} + \frac{26}{35}x_{n-3} - \frac{1}{7}x_{n-4} + \\ &\quad \frac{h}{12600}(6463f(x_n) - 2092f(x_{n-1}) + 2298f(x_{n-2}) - 1132f(x_{n-3}) + 223f(x_{n-4})) \end{aligned}$$

where $n = 4, 5, 6, ...$ As LIL is a multistep method, four previous values of the vector x_i and the function evaluations $f_i = f(x_i)$ (for $i = 0, 1, 2, 3$) are necessary to begin implementation of LIL. Therefore, a 4th-order Runge-Kutta method (see next section) was used to generate the three results succeeding the initial conditions in order to begin using Local Iterative Linearization. This makes implementation and analysis of this method more difficult than other methods, as temporary use of another method is required. LIL is implemented as a *predictor-corrector* method. The evaluation of $f(x_n)$ is an extrapolation, which is then corrected by the other terms of the formula.

2.3 Runge-Kutta

"The most popular Runge-Kutta method" (RK4) is defined in [9] as follows:

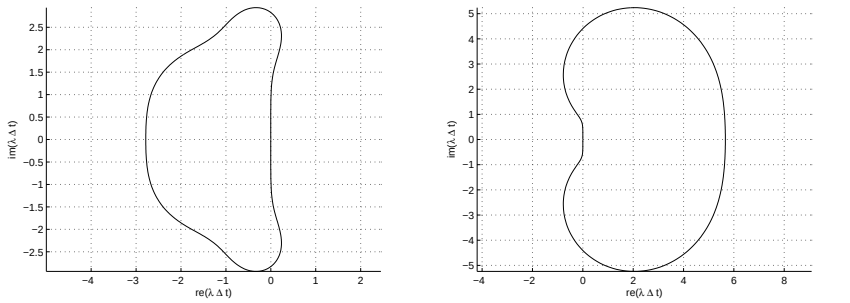
$$\begin{aligned} x_{n+1} - x_n &= \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ k_1 &= f(x_n) \\ k_2 &= f(x_n + \frac{1}{2}hk_1) \\ k_3 &= f(x_n + \frac{1}{2}hk_2) \\ k_4 &= f(x_n + hk_3) \end{aligned}$$

This method gives fourth-order accuracy with only four evaluations of f, whereas with Runge-Kutta methods that yield order higher than $\rho = 4$, the number of evaluations of f is greater than ρ. RK4 has the advantage of being a one step method: it does not require previous values, so one may go directly from t_n to t_{n+1} by taking multiple function evaluations within the step h.

3 Stability

The stability of a numerical method refers to the behavior and propagation of errors. A method is considered stable if the eigenvalues of the Jacobian matrix (which is constant for linear problems, but changes every step for nonlinear

problems) lie within the stability region of the method when scaled by h. Using MATLAB and the information in Danca [5] and Butcher [10], stability plots were generated for both numerical methods.



(a) 4th-Order Runge-Kutta (b) 4th-Order Local Iterative Linearization

Figure 1: Stability Regions

The 4th-order LIL method is implemented as a predictor-corrector method and therefore has explicit (bounded) stability. It was stated in [5] that the predictor-corrector implementation of this method has the same stability region as the implicit method (Figure 1(b).) However, numerical experiments indicate that this is not the case. A forthcoming paper will explore this in more detail and give a complete analysis. The explicit 4th-order Runge-Kutta method is stable for the area inside the plot in Figure 1(a). As can be seen in Figure 1 above, neither method is *absolutely stable* (including the entire left half-plane in its stability region) as defined in [10].

3.1 Examining stability

Linear stability is a strong characteristic of numerical methods, but when studying nonlinear systems, things are seldom so neat. Iserles [11] mentions a method for attempting to translate "linear theory to a nonlinear setting," while cautioning that such is better than nothing, but worse than embracing nonlinear properties from the outset. Nevertheless, this has been adopted here for illustrative purposes. A script was written in MATLAB to utilize the Runge-Kutta method used throughout this paper, and plot the eigenvalues of the Jacobian matrix of the RF system, scale them by the stepsize, and see whether they lie within the linear stability region.

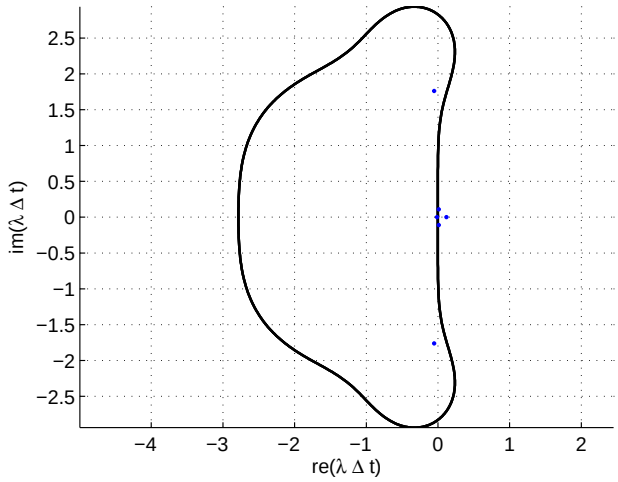


Figure 2: Eigenvalues and stability region, using RK4 for $a = 0.1, b = 0.05, t = 5$

To illustrate what happens when methods become unstable, we selected the parameter set $a = 0.1, b = 0.05$, which is (in our experience) stable for a step size of $h = 0.001$. The step size was raised to $h = 0.1$ to better illustrate instability. In Figure 2, at $t = 5$, one can notice scaled eigenvalues outside the stability region (remember: since RK4 is explicit, the stability region is *inside* the curve.)

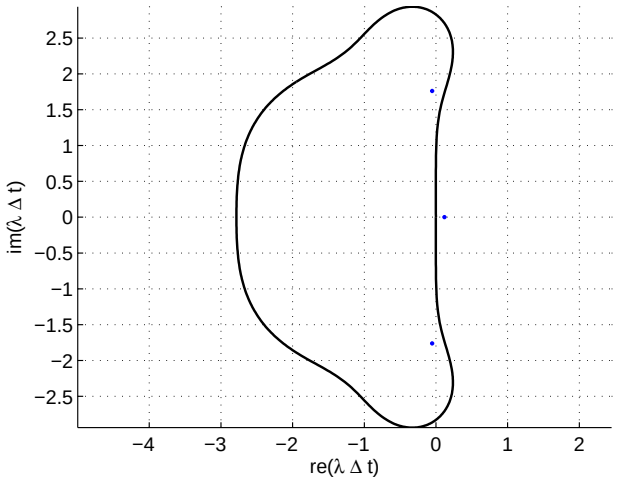


Figure 3: Eigenvalues and stability region, using RK4 for $a = 0.1, b = 0.05, t = 83$

Time $t = 83$ is the last time value for which our experimental setup could plot the rapidly-growing numerical result. The eigenvalues of the Jacobian matrix of the RF system had remained outside the stability region, and at this point the method became unstable and the numerical solution grew without bound (Figure 4.) The numerical errors could be growing without bound in this case. It is also possible that the method is accurately modeling an unbounded analytic solution. However, based on the eigenvalues exiting the stability region, we believe the former to be more likely. It is also interesting to note here that the eigenvalues in these two figures are practically indistinguishable, despite the large time difference. However, close examination of Fig. 2 shows eigenvalues not present in Fig. 3 which are very close to the boundary of the stability region.

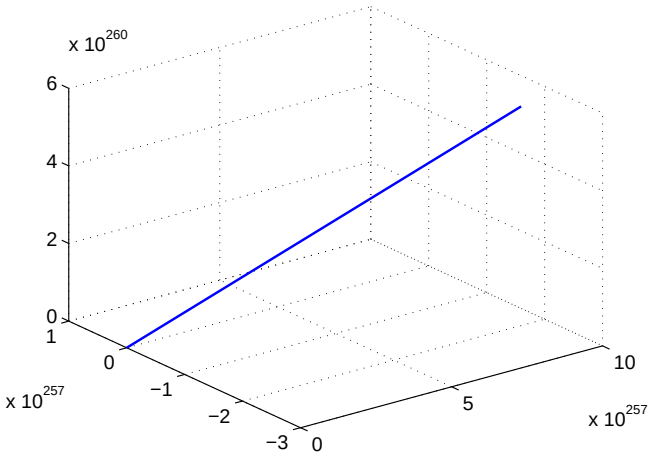


Figure 4: 3D phase plot of the RF equations using RK4 for $a = 0.1, b = 0.05, h = 0.1$

In Figure 4 is a phase plot of the system, illustrating the results of such unstable behavior. The numerical result grows without bound. If the corresponding analytic solution is bounded, numerical integration no longer approximates the behavior of the system.

4 Accuracy

When one discusses the *order of accuracy* (or more simply, order) ρ of a numerical method, we are referring to the truncation error

$$Er = \mathcal{O}(h^{\rho+1}). \tag{4}$$

That is, the error is proportional to the stepsize h, taken to the power $\rho + 1$. Euler's method (2,) a popular first-order numerical method, serves as a perfect example to illustrate core properties of the order of a method.

The increasing powers of h in 3 indicate that $\mathcal{O}(h^3)$ will have smaller bearing on the error than the $\frac{1}{2}h^2y''(t_n)$ term. Thus, we consider the error as being proportional to h^2, and so the order of Euler's method is 1.

A *convergence plot* can be used to numerically verify the theoretical order of accuracy. by matching a numerical method against a problem with an analytical solution. The absolute error at each timestep is computed and plotted (with both axes on logarithmic scales) versus the stepsize h. Since the error $\approx Ch^p$,

$$log(error) \approx log(Ch^p) \tag{5}$$

$$log(Ch^p) = log(C) + p(log(h)) \tag{6}$$

which gives us the slope-intercept form of a line $y = mx+b$, as $m = \rho$, $x = log(h)$, and $b = log(C)$.

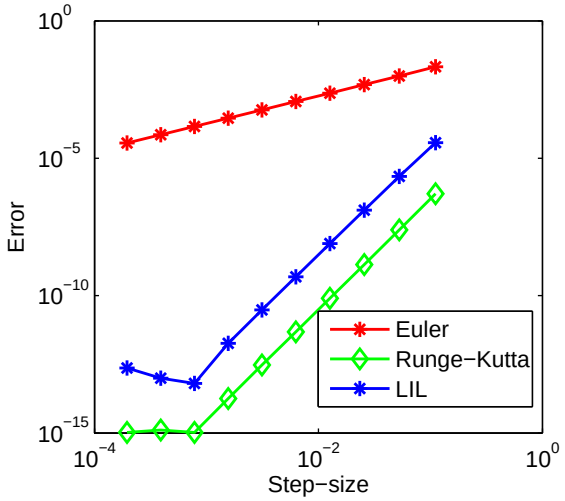


Figure 5: Error plot for the problem $y' = -y, y(0) = 1, t_0 = 0, t_{end} = 1, h = 0.001$

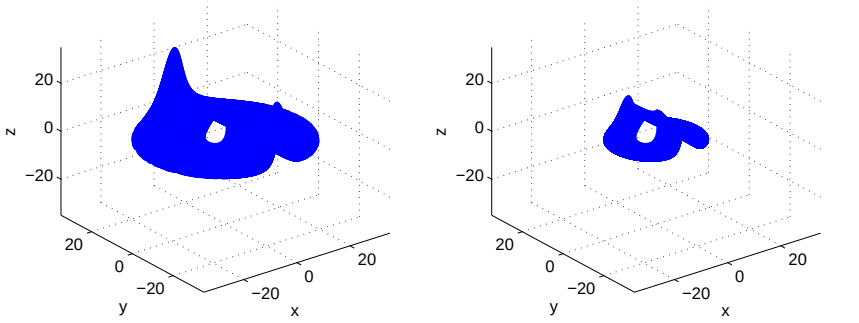
In Figure 5 the error increases as the step size h increases. As expected, the Euler's method plot has a slope of approximately 1, while the 4th order Runge-Kutta and LIL methods have error plots with slope 4.

5 Numerical experimentation

In the following experiments, we will use Runge-Kutta and LIL methods to solve the Rabinovich-Fabrikant equations for various parameters (a, b) and obtain phase portraits, which we will use to discuss the behavior of the system.

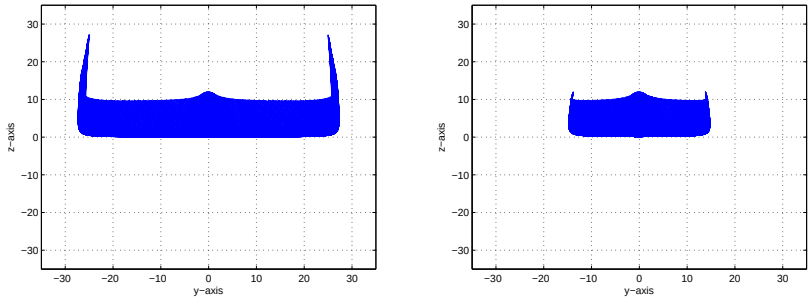
5.1 Region differences

For certain parameter sets, the phase portraits show qualitative similarities but marked region differences. The set $(a, b) = (0.1, 0.05)$ (Figures 6, 7), for example, shows a clear difference in scale, with the 4th-order Runge-Kutta result occupying a much larger space than the 4th-order LIL result.



(a) 4th-Order Runge-Kutta (b) 4th-Order Local Iterative Linearization

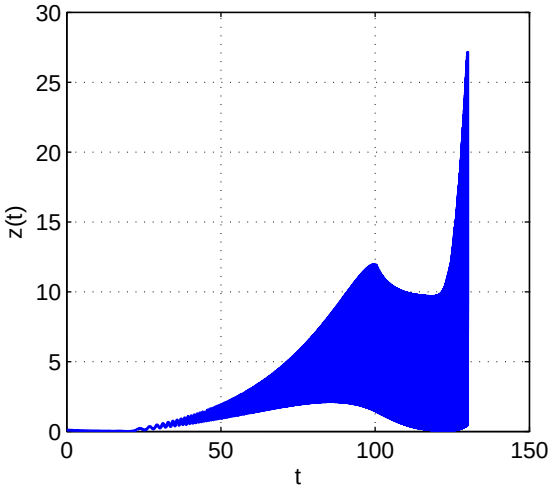
Figure 6: Phase plots for a=0.1, b=0.05, dt = 0.001



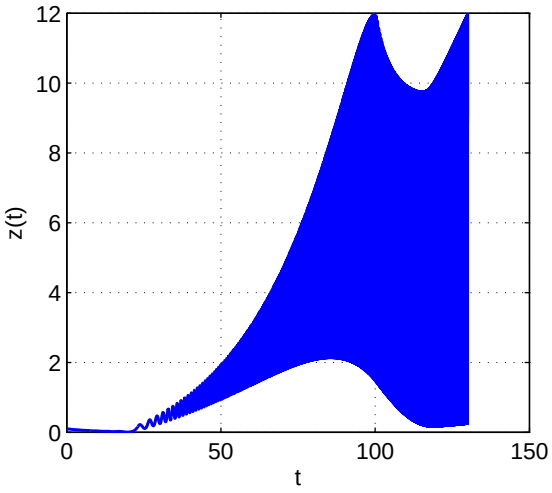
(a) 4th-Order Runge-Kutta (b) 4th-Order Local Iterative Linearization

Figure 7: Phase plots projected onto the yz plane for a=0.1, b=0.05, dt = 0.001

The figures were plotted from $t_0 = 0$ to $t_{End} = 130$, with step size $h = 0.001$ for both methods. Even the single variable vs. t plots are qualitatively very similar with the exception of z vs. t, (Figure 8).



(a) 4th-Order Runge-Kutta



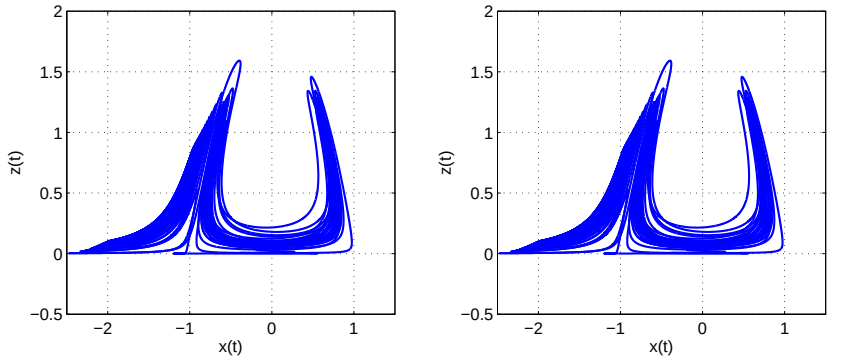
(b) 4th-Order Local Iterative Linearization

Figure 8: $z(t)$ versus t plot for $a = 0.1, b = 0.05$

These plots are very similar until late in the progression of t, when the z calculated with a 4th-order Runge-Kutta method rapidly moves from $z \leq 10$ (at about $t = 80$) to $25 \leq z$ (around $t = 125$.) The 4th order Local Iterative Linearization result, by comparison, does not exceed $z \leq 12$.

5.2 Agreement between methods

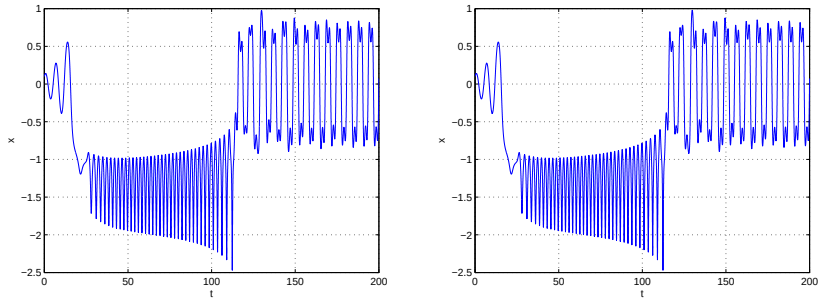
While qualitative agreements with regional differences were encountered in our numerical experiments, a more common result was complete agreement between methods, as seen with parameter settings $(a, b) = (0.1, 0.14)$ (see Figure 9)



(a) 4th-Order Runge-Kutta (b) 4th-Order Local Iterative Linearization

Figure 9: 2D phase plots (xz plane) for $a = 0.1, b = 0.14$

For these parameters, the numerical solutions were plotted from $t_0 = 0$ to $t_{End} = 200$, with step size $h = 0.001$ for both methods. Single-variable versus t plots in this case are also indistinguishable (for an example, see Figure 10 below.)



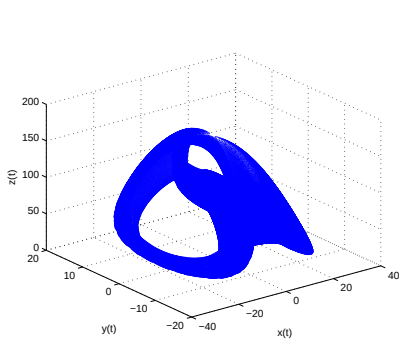
(a) LIL (b) RK4

Figure 10: Single variable plots x versus t for $a = 0.1, b = 0.14$

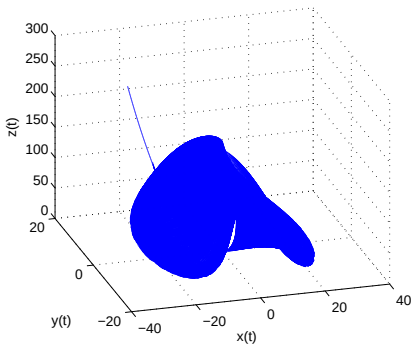
Figure 10 also clearly identifies transient behavior before the "saddle" attractor manifests (as named in [4]. However, both methods show identical phase

plots, and the conic region which may be an artifact is clearly noticeable in the x versus. t plot between $t = 25$ and $t = 125$.

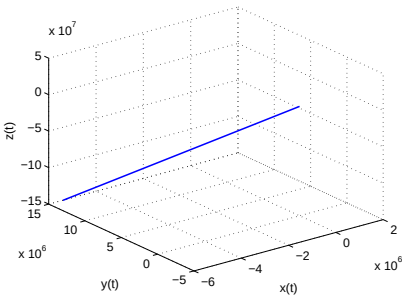
In [5] it was reported that for parameter settings $a = 0.3, b = 0.1$ the LIL method functions while the Runge-Kutta method does not. Our experimental results differ only slightly from these findings. The 4th-order LIL result agrees perfectly with the 3rd-order result presented in [5] but the Runge-Kutta plots (Fig 11)indicate the possibility of transient chaos in the system for these parameter values.



(a) RK4, $t = 50$



(b) RK4, $t = 60$



(c) RK4, $t = 70$

Figure 11: RK4 results for $a = 0.3, b = 0.1$

The Runge-Kutta results here present a very different seeming-attractor for low time values, but the eventual behavior is unbounded. This evidence of transient chaos does not agree with the LIL result (an attractor) but with the difficulty inherent in numerically studying the RF equations, it is impossible to tell which (if either) of the methods yield correct results in this case.

5.3 Other methods

In the senior thesis which was the seed for this paper, all experiments were also carried out using MATLAB's ode45 solver, which uses the highly-efficient Dormand-Prince Runge-Kutta method [14]. This method is easier to program and rarely fails in the sense of results growing boundlessly. However, with the inherent difficulty of studying a chaotic system of equations, we felt it were better to avoid the additional complication of the adaptive step-size. The inclusion of changing step-size presents yet another possible explanation for different results between methods. In our experience, ode45 results very often differ from those of RK4 and LIL4. This is not to say that ode45 is without merit for this problem - merely that it was excluded from this paper alone to avoid complicating the interpretation of results. Future numerical research of this system would benefit from the use of different methods such as Dormand-Prince and Fehlberg adaptive Runge-Kutta Methods. Study of an instance of failure of an adaptive method was presented in [15] and may help illustrate potential pitfalls in the use of these methods.

6 Conclusions

The Rabinovich-Fabrikant equations have a relative lack of published research when compared to more famous systems, and there is far more numerical experimentation in the literature than analytical results. It has been indicated that numerical methods are statistically likely (to varying degrees as indicated in [4]) to be modeling a chaotic system.

Additionally, it has been stated in the literature that Runge-Kutta methods are unsuitable for this system [4]. However, our numerical experience indicates that both Local Iterative Linearization methods and Runge-Kutta methods can become unstable for certain parameter settings (in our results, particularly $(a, b) = (3.5, 0.1)$ or $(0.2, -1)$. Furthermore, the implicit Local Iterative Linearization method of 4th-order is not absolutely stable, and the implementation as a predictor-corrector method yields similar results to the Runge-Kutta implementation while requiring longer, more intensive computations.

There are still many avenues for numerical experimentation with the Rabinovich-Fabrikant equations. Extended precision floating point arithmetic could be used to explore their sensitive dependence on initial conditions. As well, higher order methods and absolutely stable methods could be used to reexamine phase portraits, especially for areas where 4th-order methods (and lower, as 3rd-order LIL has been a prevalent method in literature) disagree. Newer developments such as shadowing [12] and defect control [13] bear application to this system in order to gain confidence in the validity of numerical methods for chaotic dynamical systems.

Appendix: MATLAB Code

We have included MATLAB code (compatible with version 2008a) which implement the Runge-Kutta and Local Iterative methods used in this paper. These are a suitable starting point for those wishing to expand on this research. Most commenting has been omitted for brevity.

LIL4

The script in listing 1 performs the numerical integration only (for one step) and so must be repeatedly utilized by a driver.

Listing 1: Fourth-order LIL method

```
% Local Iterative Linearization (LIL), m=4

function v = LIL4(vm1,vm2,vm3,vm4,t,dt,F)
    fk = feval(F,4*vm1-6*vm2+4*vm3-vm4);
   fm1 = feval(F,vm1);
   fm2 = feval(F,vm2);
   fm3 = feval(F,vm3);
   fm4 = feval(F,vm4);
   h = (dt/12600);
   uk = (h*(6463*fk-2092*fm1+2298*fm2-1132*fm3+223*fm4));
     v = 2*vm1-(8/5)*vm2+(26/35)*vm3-(1/7)*vm4 + uk;
```

Listing 2 is a driver which declares all initial conditions and startup values before using rk4 to find values for 3 steps after the initial conditions, providing the 4 startup values to begin using LIL4 (Listing 1). The parameters (a, b) in the Rabinovich-Fabrikant equations, are included as input parameters for the function. Script for a 3D phase plot of the results is included.

Listing 2: Fourth-order LIL method driver

```
function LIL4Driver(a,b)

  t0 = 0; tEnd = 200;
  x0 = 0.1;
  y0 = -0.1;
  z0 = 0.1;
  dt = 0.001;
  T = t0:dt:tEnd;
  v = zeros(3,length(T));
  v(:,1) = [x0; y0; z0];

  t = t0; %rk4 develops initial values
  v(:,2) = rk4(v(:,1),t,dt,@F);
  t = t+dt;
```

```
  v(:,3) = rk4(v(:,2),t,dt,@F);
  t = t+dt;
  v(:,4) = rk4(v(:,3),t,dt,@F);
  t = t+dt;

  k = 4; %counter

 while t<tEnd %perform LIL
 v(:,k+1)=LIL4(v(:,k),v(:,k-1),v(:,k-2),v(:,k-3),t,dt,@F);
 t = t + dt;
 k = k + 1;
 end

% 3D plot
plot3(v(1,:),v(2,:),v(3,:))
grid on
xlabel 'x(t)', ylabel 'y(t)', zlabel 'z(t)'

%RF system
  function f = F(x,t)
     f = [x(2)*( x(3) - 1 + x(1)^2 ) + a*x(1);
         x(1)*( 3*x(3) + 1 - x(1)^2 ) + a*x(2);
         -2*x(3)*( b + x(1)*x(2) ) ];
  end
end
```

Runge-Kutta

Listing 3 implements the RK4 method described in the Numerical Methods section. Again, it must be called by a driver.

Listing 3: Fourth-order Runge-Kutta method

```
function v = rk4(V,t,k,F)

   s1 = feval(F,V,t);
   s2 = feval(F,V + k*s1/2,t+k/2);
   s3 = feval(F,V + k*s2/2,t+k/2);
   s4 = feval(F,V + k*s3,t+k);

   v = V + k*(s1 + 2*s2 + 2*s3 + s4)/6;
```

The RK4 driver in Listing 4 uses a set of initial values declared in the code with (a, b) declared as input parameters for the function to numerically integrate the RF equations.

Listing 4: Fourth-order Runge-Kutta method driver

```
function rk4driver(a,b)

  t0 = 0;
  dt = 0.001;
  tEnd = 200;
  x0 = 0.1;
  y0 = -0.1;
  z0 = 0.1;
  init = [x0; y0; z0];
  T = t0:dt:tEnd;
  v = zeros(3,length(T));
  v(:,1) = [x0; y0; z0];
  t = t0;
  k = 1;

  while t<tEnd
    v(:,k+1) = rk4(v(:,k),t,dt,@F);
    t = t+dt;
    k = k+1;
  end

% 3D plot
plot3(v(1,:),v(2,:),v(3,:))
grid on, xlabel 'x(t)', ylabel 'y(t)', zlabel 'z(t)'

% RF system
function f = F(x,t)
f = [x(2)*(x(3)-1+x(1)^2)+a*x(1);
    x(1)*(3*x(3)+ 1-x(1)^2)+a*x(2);
    -2*x(3)*(b+x(1)*x(2))];
end
end
```

References

[1] Rabinovich, M.I., A.L. Fabrikant. Stochastic self-modulation of waves in nonequilibrium media. *Zh. Eksp. Teor. Fiz. (Sov),* **77** (1977) 617-629.

[2] Rössler, O.E. An equation for continuous chaos. *Phys. Lett. A,* **57**(5), (1976) 397-398.

[3] Lorenz, E.N. Deterministic nonperiodic flow. *J. Atmospheric Sci.* **20** (1963) 130-141

[4] Luo, Xiaodong, Michael Small, M.F. Danca, Guanrong Chen. On a dynamical system with multiple chaotic attractors. *Int. J. Bif. Chaos.,* **17, 9** (2007) 3235-3251

[5] Danca, M.F. A multistep algorithm for ODEs. *Dyn. Cont. Disc. Imp. Sys.,* **13** (2006) 803-821

[6] Danca, M.F., Guanrong Chen. Bifurcation and chaos in a complex model of dissipative medium. *Int. J. Bif. Chaos,* **14,10** (2004) 3409-3447.

[7] Devaney, Robert L. *An Introduction to Chaotic Dynamical Systems* (2003), Westview.

[8] Strogatz, Steven H. *Nonlinear dynamics and chaos* (1994), Westview.

[9] Lambert, J.D., *Computational Methods In Ordinary Differential Equations* (1973), John Wiley and Sons.

[10] Butcher, J.C. *Numerical Methods for Ordinary Differential Equations* (2008), John Wiley and Sons.

[11] Iserles, Arieh. *A First Course in the Numerical Analysis of Differential Equations* (1996), Cambridge University Press.

[12] Hayes, Wayne B., Kenneth R. Jackson. Rigorous shadowing of numerical solutions of ordinary differential equations by containment. *SIAM. J. Numer. Anal.,* **41, 5** (2003) 1948-1973

[13] Enright, W.H., Wayne B. Hayes. Robust and reliable defect control for Runge-Kutta methods. *ACM Trans. Math. Software,* **33, 1** (2007)

[14] Dormand, J.R. and P.J. Prince. A family of embedded Runge-Kutta formulae. *J. Comp. App. Math.* **6** (1980) 19-26.

[15] Skufca, J.D. Analysis Still Matters, A Surprising Instance of Failure of Runge-Kutta-Felberg of ODE Solvers. *SIAM Review,* **46** (2004) 729-737