

# Chaos and ODEs Part 1c: Numerical Solution Methods

Dan Hughes  
November 2007

## Introduction

The purpose of these notes is to give specificity relative to the solution methods that I have used. When the phrase 'explicit Euler method' is used in subsequent posts, for example, it will mean the solution method so labeled in these notes. Instead I simply report which methods have been used for the calculations to be discussed in following posts. It is not the objective of these notes to discuss the development and theoretical properties of numerical solution methods relative to chaotic response in dynamical systems. Consistency and stability of the discrete approximations and well-posedness of the initial value problem (IVP) and existence, and uniqueness of the solutions of the continuous equations are not of primary interest in this report. Generally, well-posedness means that a solution exists, it is unique, and it depends continuously on the initial data. And consistency of the discrete approximations with the continuous equations plus stable solution methods for the systems of algebraic equations arising from the discretizations guarantees convergence of solutions of the equations to the solutions of the continuous equations. There are theorems, of course, but I'm not going to be doing that kind of stuff in these notes.

The focus will be on calculational demonstrations about convergence, or lack thereof, of the discrete approximations as the discrete step size is refined. The primary objective of the present report is an investigation into the convergence of the discrete approximations to the solutions of the continuous equations. By convergence I will mean that as the size of the discrete increment for the independent variable is reduced, the calculated dependent variables approach constant values for all values of the independent variable. Under these conditions the approximate solution is said to be the solution for the continuous equations.

There are an almost uncountable number of textbooks and review papers on all aspects of numerical solution methods for ODEs available. I have primarily used Ascher and Petzold [1998] and Iserles [1996]. Textbooks and review papers that focus on numerical methods and dynamical systems are also available; Stuart and Humphries [1996] and Thompson and Stewart [2002] are two examples. Specialized focus is available in Mickens [1994, 2001] and Adams and Kulisch [1993]. Many of these books and papers will discuss in detail existence, uniqueness, consistency, stability, and convergence.

In almost all of discussions of the numerical solution methods that follows, the size of the discrete increment for the independent variable will be represented by  $h$ . To save word-processing work the methods will be written out for a single scalar equation. Extension to systems of equations is straightforward and presented in the textbooks mentioned above in this section.

While the nomenclature used in this report is standard, it is sometimes loosely applied in papers in the literature. Many papers will state, for example, that the Euler, or Runge-Kutta method is used to integrate the equation systems. Such statements are incomplete as the discussions in the following paragraph will show. The numerical

solution methods are explicitly summarized here so that when a reference is made to a solution method, the reader will know exactly what is meant.

The numerical solution methods that I have used are summarized in the following paragraphs. I will try to give citations to specific text in Ascher and Petzold for each method that I summarize.

### Taylor Series Expansion

The basis of many discrete approximations to continuous equations is the Taylor series expansion of functions. A general form of the continuous equations for an initial value problem for a first-order ODE is

$$\frac{dW}{dt} = G(t, W) \quad (1.1)$$

with specified initial value

$$W(0) = W_0 \quad (1.2)$$

Numerical solutions produce a sequence of discrete points  $(t_i, W_i)$  where  $W_i$  is an approximation to the true solution  $W(t_i)$  and the continuous solution is  $W(t)$ . The sequence of discrete values for the independent variable can be either uniformly or non-uniformly distributed over the interval of interest for the independent variable, say  $[a, b]$ . In these notes the starting point for the interval is always  $a=0$ .

A Taylor series expansion of the function  $W$  about the time  $t_n$  is

$$W(t_{n+1}) = W(t_n) + h_n \left( \frac{dW}{dt} \right)_m + \frac{1}{2} h_n^2 \left( \frac{d^2W}{dt^2} \right)_m + \frac{1}{3!} h_n^3 \left( \frac{d^3W}{dt^3} \right)_m + \dots \quad (1.3)$$

The discrete approximations to Eq. (1.1) are developed using Eq. (1.3). These are summarized in the following paragraphs.

### **The Explicit Euler Method** [Ascher and Petzold, page 37]

The explicit Euler method, also known as the forward Euler method, is the most straightforward of all numerical integration methods. The method is obtained directly from the Taylor series expansion by dropping RHS terms that are of order  $h^2$  and higher, and substituting Eq. (1.1) for the derivative of  $W$ . This process gives an approximation for the dependent variable at the new-time level directly in terms of the value at the previous time level

$$W_{n+1} = W_n + hG(t_n, W_n) \quad (1.4)$$

where  $h$  is the size of the discrete increment for the independent variable. As noted in a previous post, almost all the equation systems that have been investigated are

autonomous systems and the right-hand-sides (RHSs) are not functions of the independent variable. The independent variable might wander in and out of the nomenclature in these notes. The series of approximations  $(t_i, W_i)$  is constructed by starting at the initial value and marching the steps to the ending value of interest.

The explicit Euler method probably has the least amount of computational work per increment of all methods. Because of the larger truncation error in the approximation, however, quite small step sizes are usually needed for the discrete approximation to approach the solution of the continuous equations. Thus, over all, more computational work might be needed in the course of a calculation.

I always write my own software when I use the explicit Euler method. A basic framework of input and output processing can be written to be reusable. Then only the coding that evaluates Eqs. (1.4) needs to be changed as the equation systems change.

### **The Implicit Euler Method**[Ascher and Petzold, page 51]

The implicit Euler method, also known as the backward Euler method is obtained by evaluating the functions  $G$  on the RHSs at the new-time level instead of at the old-time level. This procedure gives

$$W_{n+1} = W_n + hG(t_n, W_{n+1}) \quad (1.5)$$

Since the unknown value of the dependent at the new-time  $(n+1)$  level appears on both sides of the equations, an iterative method is needed to solve the system of non-linear algebraic equations at each discrete increment of the independent variable. The method thus requires significant amounts of additional computational work compared with the explicit Euler method. A statement that holds generally for all implicit methods compared with explicit methods. A larger step size for the discrete increment can be used with implicit methods compared with explicit methods and maintain stability of the solution procedures. On the other hand, however, a larger discrete increment usually means larger differences between the numerical solution and the solutions of the continuous equations. The stopping criterion used with the iterative solution procedures introduces an additional parameter into the overall scheme of things.

The SEULEX routines [available from this page, <http://www.unige.ch/~hairer/software.html>] were used as the implementation of the implicit Euler method for the calculations to be reported later. Local modifications were made so that the code executed with a constant step size for the independent variable. Other local modifications were made to accommodate output of auxiliary information associated with my investigations. There are very likely hundreds of versions of the implicit Euler method available both on the Web and as routines in computer-maths programs.

### **The Explicit Midpoint Method**[Ascher and Petzold, pp. 74-75]

The explicit midpoint method is obtained by evaluating the functions  $G$  at two points in the interval  $(t_n, t_{n+1})$ . The method is

$$\hat{W}_{n+1/2} = W_n + \frac{h}{2}G(t_n, W_n)$$

(1.6)

and

$$W_{n+1} = W_n + hG(t_{n+1/2}, \hat{W}_{n+1/2})$$

I think this method has been referred to as the double approximation forward method in the papers by Saltzman [1962], Lorenz [1963] and Young [1966]. It is the method used by Lorenz in the 1963 paper by Lorenz.

**Explicit Runge-Kutta Method** [Ascher and Petzold, pp. 74]

The explicit Runge-Kutta method is summarized as follows. The method requires the evaluation of the RHS functions at four values of the independent variable

$$W_{n+1} = W_n + \frac{1}{6}(k_0 + 2k_1 + 2k_2 + k_3)$$

where

$$k_0 = hG(W_n)$$

$$k_1 = hG(W_n + \frac{1}{2}k_0)$$

$$k_2 = hG(W_n + \frac{1}{2}k_1)$$

$$k_3 = hG(W_n + \frac{1}{2}k_2)$$

(1.7)

The method is fourth-order in the local truncation error. There are an almost uncountable number of off-the-shelf routines that implement Runge-Kutta methods. Many of these are general-purpose programs designed to handle wide ranges of IVPs and many are based on implicit Runge-Kutta methods.

For the numerical calculations in this report, a straightforward local implementation of the explicit method was developed. The explicit method requires less computational work than implicit methods. Additionally as the focus of this report is on convergence of the discrete approximations the capability to handle larger step sizes was not of significant importance.

**An Analytical Taylor Series Method** [Ascher and Petzold, pp. 73-74]

A method based on an analytical Taylor series expansion of the RHS functions is outlined Chapter 4 of Ascher and Petzold [see Eq. 4.1]. The method is illustrated by use of a scalar ODE  $y' = f(y)$ . The truncated Taylor series for the dependent variable is

$$W_{n+1} = W_n + h_n(G)_n + \frac{h^2}{2} \left( \frac{dG}{dt} \right)_n + \dots + \frac{h^p}{p!} \left( \frac{d^{p-1}G}{dt^{p-1}} \right)_n \quad (1.8)$$

The analytical derivatives of the RHS functions, with proper accounting of all implicit dependencies, are substituted into (1.8). I'm using a shorthand nomenclature here to save word-processing equations. Let me know if you need clarification. A system of two linear equations is a good one for practice

$$\begin{aligned} \frac{dX}{dt} &= a_{11}X + a_{12}Y \\ \frac{dY}{dt} &= a_{21}X + a_{22}Y \end{aligned} \quad (1.9)$$

The first derivatives of the RHSs are

$$\begin{aligned} \frac{dG_1}{dt} &= a_{11} \frac{dX}{dt} + a_{12} \frac{dY}{dt} \\ &= a_{11} [a_{11}X + a_{12}Y] + a_{12} [a_{21}X + a_{22}Y] \\ \text{and} \\ \frac{dG_2}{dt} &= a_{21} \frac{dX}{dt} + a_{22} \frac{dY}{dt} \\ &= a_{21} [a_{11}X + a_{12}Y] + a_{22} [a_{21}X + a_{22}Y] \\ \text{and so forth} \end{aligned} \quad (1.10)$$

I developed software to implement this method for some of the equation systems that I solved. The specific systems will be identified when I post the results of my calculations; I think it was the original Lorenz system of 1963 and one or two others. Various orders of the derivatives, up to the sixth-derivative ( $p=7$ ), of the RHS functions were derived analytically and coded into the numerical integration program. The program was written so that the user could specify the order of the integration to be used at runtime.

It is imperative that convergence be checked with the order fixed and the discrete step size refined at constant order. It is in error to check convergence by increasing the number of terms in the series at fixed step size. It is easily shown that generally the higher-order derivatives rapidly decrease in importance because of the factor of  $h^n$  in the numerator and the factorial in the denominator.

While the analytical work is straightforward it is also tedious; the derivatives become somewhat more complicated as the order increases. The general expression for the third derivative is given on page 74 of Ascher and Petzold. The RHS functions used in the present report are not functions of the independent variable so the analytical derivatives are a little easier to handle. Nonetheless, independent Verification of the analytical derivatives and implementation into computer routines is of vital importance.

Although the method is a straightforward extension of the basic Taylor's expansion that is the basis of many numerical solution methods, it does not seem to be much in use. I don't know why. Analytical derivatives of almost all functions can be obtained by use of the basic mechanisms of differential calculus. Additionally, symbolic algebra by use of computer software like Maple, Mathematica, and MatLab can also obtain expressions for the derivatives, and provide code for several computer languages at the same time. And finally, automatic differentiation of coded functions by use of other computer software is a well-developed part of numerical solution methods. This latter capability has been developed for applications to analytical sensitivity analyses. Whatever the case, the applications in this report seem to be among the few for which the method has been used.

It is important to note that this is not the expansion of the solution in a polynomial series of the independent variable. That method is also referred to as Taylor series expansion method. Additionally, it is also not the Adomian decomposition method which also is based on series of polynomials. I used the method outlined by Ascher and Petzold on pages 73 and 74.

### **Off-the-Shelf ODE Solver Software**

General-purpose ODE-solver software has long been a staple of pre-packaged maths softwares. Many are easily found in netlib and by use of a Google search; there are hundreds. Some of the off-the-shelf codes I used for some calculations include RKSUITE, VODE, and DVERK. Almost all the OTS software packages are general-purpose and so include many features that are not needed for the kind of investigations of interest in this report. Many, for example, are designed to efficiently handle systems of stiff ODEs and/or systems of DAEs. The calculational results that will be reported in these notes were done with the DVERK routines and these can be found at <http://www.cs.toronto.edu/NA/dverk.f.gz>. The routines were modified to run at a fixed step size and to output auxiliary information for sine equation systems.

The very first exploratory calculations that initiated this entire investigation were conducted with an OTS software package. These calculations indicated that results of the numerical integration were functions of many of the user-related parameters in the programs. The packages that have estimates of the global error, and if this option is activated for a calculation, some systems of ODEs that exhibit chaotic response cannot be successfully run to completion. Changes in the step size and changes to other user-specified options generally led to solution differences. Convergence of the numerical solutions also could not be demonstrated. I decided to turn to the less complex methods and codes described herein for the calculations of interest.

### **Numerical Recipes Routines**

The Numerical Recipes routines contain a number of Runge-Kutta methods from simple explicit methods to more complex and sophisticated approaches. A few exploratory calculations were done with these routines applied to the original Lorenz 1963 system. I have not developed solvers based on the NR routines because the initial calculations helped to identify the basic problem that convergence was not being attained.

### Finite Difference Approximations

In addition to the standard methods discussed above, some equation systems were solved by application of finite-difference approximations. These approximations are a straightforward correspondence to some of the standard ODE solution methods. In particular, the lower order explicit methods are exact analogs of the finite difference methods.

The non-standard finite difference folks sometimes use this method along with special denominator functions for the Mickens non-standard methodology [1994, 2001]. An example is given by Letellier et al. [2004]. And check the previous list of references for Part 0 of these notes.

Let's summarize this method by use of an example using the Rossler system. A fully explicit approximation evaluates all the terms on the RHSs of the equations at the old-time level and the derivative on the LHSs are approximated by

$$\frac{dW}{dt} = \frac{W_{n+1} - W_n}{h} \quad (1.11)$$

The Rossler system presented in the previous notes, for example, can be written

$$\begin{aligned} X_{n+1} &= X_n - hY_n + hZ_n \\ Y_{n+1} &= Y_n + ahY_n + hX_n \\ &\text{and} \\ Z_{n+1} &= Z_n - chZ_n + hX_nZ_n + hb \end{aligned} \quad (1.12)$$

And these equations are seen to be exactly the explicit Euler method.

For a fully-implicit approach all the terms on the RHSs are evaluated at the new-time level. This method then gives three equations to be solved for the three dependent variables at the new-time level

$$\begin{aligned} X_{n+1} + hY_{n+1} - hZ_{n+1} &= X_n \\ Y_{n+1} - hX_{n+1} - ahY_{n+1} &= Y_n \\ &\text{and} \\ Z_{n+1} - hX_{n+1}Z_{n+1} + chZ_{n+1} &= Z_n + hb \end{aligned} \quad (1.13)$$

As always, the initial conditions must also be specified in order to set a well-posed problem.

These equations can be solved by any number of methods. Analytical reduction to a single equation for the  $Z$  variable at the new-time level is theoretically possible. The other two dependent variables would be solved by back-substitution. An iterative

Newton-Raphson method using the analytical Jacobian for the system is another approach. The latter could utilize a computer routine in which the analytical solution of 3x3 equation systems by Thomas' method is coded. Numerical evaluation of the Jacobian and standard maths library routines for linear algebra processes is yet another approach.

Other approaches between fully implicit and fully explicit are equally possible. The dependent variables for each given equation that appear on the RHSs could be evaluated at the new-time level while the other dependent variables are evaluated at the old-time level. Whatever the case, each approach must be evaluated for stability and convergence, and each approach will impose limits on the size of the discrete increment in the independent variable that can be used and still provide stable numerical solutions.

The specific finite-difference approximations used in the calculations will be given when the results are discussed.

Finite-difference approximations have not been applied to all the equation systems used in this investigation. I am open to suggestions that use of finite-difference methods would change the conclusions of this work. Let me know which equation systems are of interest to you and specifications of the time levels and we'll crank out some numbers. I have the analytical solution for 3x3 systems coded so all we have to do is feed the function some input.

#### **Adomian Decomposition Method (ADM)**

The Adomian decomposition method [1986, 1988, and a very expensive book available from Amazon] is a somewhat recent development in solution methods for IVP ODEs, and other systems. I have listed several investigations of application of the method to Lorenz-like, and other, chaotic response systems; see Part 1a. I have used the ADM for some of the equation systems studied by Vadasz and Olek [1999, 2000a, 2000b, 2001]. I will not outline the method here and instead refer the reader to the literature sources cited in Part 1a.

An interesting aspect was discovered during these calculations as follows. As the number of terms in the series associated with the method increases they begin to behave as all high-order polynomials and the terms have alternating signs with large numerical coefficients. Additional details will be given when the solutions are discussed.

Again, the size of the terms rapidly decreases as the number of terms in the series increases.