1994

Nonlinear system identification

Edward H. Ziegler

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Nonlinear System Identification

by:
Edward H. Ziegler IV

A Thesis Submitted
in Partial Fulfillment
of the Requirements
for the Degree of
MASTER OF SCIENCE
in
Mechanical Engineering

Approved by:

Prof. ______________________________________
Dr. Joseph Török (Thesis Advisor)

Prof. ______________________________________
Dr. P. Venkataraman

Prof. ______________________________________
Dr. Michael Kotlarchyk

Prof. ______________________________________
Dr. Charles Haines (Department Head)

Department of Mechanical Engineering
College of Engineering
Rochester Institute of Technology
February 1994
Permission Grant

I, Edward H. Ziegler IV, hereby grant permission to the Wallace Memorial Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. This permission is extended as long as no reproduction will be used for commercial use or profit.

Author's Signature: ______________________

Date: 2 - 8 - 73
Abstract

The prediction of a single observable time series has been achieved with reasonable accuracy and duration for the nonlinear systems developed by Rossler and Lorenz. Based on Takens' Delay-vector Space, an artificial system has been generated using a polynomial least squares technique that includes all possible fifth order combinations of the vectors in the delay space. Furthermore, an optimum shift value has been shown to exist, such that any deviation decreases the accuracy and stability of the prediction. Additionally, an augmented form of the autocorrelation function, similar to the delay vector expansion, has been investigated. The first inflection of this correlation, typically in the dimension of the system, tends to coincide with the optimum shift value required for the best prediction. This method has also been utilized in conjunction with the Grassberger-Procaccia Distance correlation function to accurately determine the fractal dimension of the systems being investigated.
Acknowledgments

I would like to thank everyone who made this thesis possible, especially:

Dr. Joseph Török for his inspiration and enthusiasm in the area of nonlinear system dynamics.

Dr. Venkataraman and Dr. Kotlarchyk for their time and knowledge contributed to this thesis.

Dr. Charles Haines and Dr. Paul Petersen for their guidance and leadership which make this engineering program so successful.

Manu Mathew, William Robertson, and Paul Danckaert for their assistance and friendship throughout this endeavor.

My parents, and the rest of my family, for making it possible for me to achieve my aspirations.

Most of all, Molly McAllister for her patience, assistance, and kindness. May the rest of our lives be happily chaotic.
List of Symbols

\begin{itemize}
  \item \(a,b,c,F,G\) Parameters for state equations.
  \item \(a_0-a_p, A\) Polynomial coefficients and array.
  \item \(A_x\) Autocorrelation value.
  \item \(D_c\) Distance correlation function.
  \item \(e_i\) Local error between actual values and fitted approximations.
  \item \(i,j,k\) Array and vector indices.
  \item \(k_1 - k_4\) Temporary variable for Runge-Kutta Method.
  \item \(l, m\) Length of Pendulum, Mass of pendulum.
  \item \(M_c\) Multicorrelation value.
  \item \(N\) Total number of points in a vector.
  \item \(O\) Order of magnitude function.
  \item \(p\) Arbitrary maximum order of a polynomial.
  \item \(q_1, Q\) Generic state variable vector and vector space.
  \item \(r\) Radii increments.
  \item \(S\) Squared error summation function.
  \item \(t\) Time variable.
  \item \(T, T^\top\) Time Period for periodic functions, Matrix transpose.
  \item \(x(t), X\) State variable vector and space. Sampled data vector.
  \item \(x\) Independent variable or vector. Henon mapping variable.
  \item \(X_T\) Delayed vector space, containing shifted values of \(x(t)\).
  \item \(x_p\) Polynomial expansion matrix of \(x\) vector.
  \item \(X_p\) Polynomial expansion matrix of \(X\) vector space.
  \item \(y\) Dependent variable or vector. Henon mapping variable.
  \item \(\Delta t\) Sampling rate.
  \item \(\alpha\) Magnitude of the forcing function for the pendulum.
  \item \(\zeta\) The damping factor.
  \item \(\gamma\) Fractal dimension of the system.
  \item \(\eta\) Degree of Freedom for a system.
  \item \(\Theta\) Heavyside function.
  \item \(\mu\) One less than the degree of freedom, \(\eta-1..\)
  \item \(\nu\) Index delay, shift value.
  \item \(\tau\) Time delay, shift value.
  \item \(\omega_n\) Natural frequency.
  \item \(\theta, \dot{\theta}, \ddot{\theta}\) Angular position, velocity, and acceleration for the pendulum.
\end{itemize}
# Table of Contents

Abstract ................................................................................................................................. i  

Acknowledgments ................................................................................................................ ii  

List of Symbols ..................................................................................................................... iii  

1.0 Introduction ..................................................................................................................... 1  

1.1 Phase Space and Autonomous Systems ........................................................................... 4  

1.2 Strange Attractors ............................................................................................................ 10  

1.2.1 Henon Mapping ............................................................................................................. 13  

1.2.2 Rossler Attractor .......................................................................................................... 14  

1.2.3 Lorenz Attractor .......................................................................................................... 17  

1.2.4 Lorenz-2 Attractor ...................................................................................................... 20  

1.3 Runge-Kutta Integration ................................................................................................. 23  

1.4 Central Difference Method ............................................................................................. 24  

2.0 Determination of an Optimum Shift, \( \tau \) .................................................................... 25  

2.1 Autocorrelation Function ............................................................................................... 27  

2.2 Multicorrelation Function .............................................................................................. 35  

2.3 Conclusions and Recommendations for the Optimum Time Shift ................................. 40  

3.0 Determining Dimensionality ......................................................................................... 41  

3.1 Fractal Dimension .......................................................................................................... 43  

3.2 Dimensional Results ....................................................................................................... 46  

3.3 Conclusions and Recommendations for Dimension ....................................................... 52  

4.0 Prediction ....................................................................................................................... 53  

4.1 Least Square Approximation .......................................................................................... 55  

4.2 Multiple Variables Least Squares .................................................................................. 57  

4.3 Prediction Results .......................................................................................................... 61  

4.4 Conclusions and Recommendations for Prediction of Nonlinear Systems ..................... 71  

Appendix I : Matlab Programs .............................................................................................. 72  

References ............................................................................................................................ 104
1.0 Introduction

*If the doors of perception were cleansed,*

*everything would appear as it really is,*

*infinite.*

- William Blake

The purpose of this investigation is to serve as a basis for the study of nonlinear system dynamics. More specifically, it investigates the possibility of developing an artificial system of equations, similar to state equations, that can be used to predict the original data, with reasonable accuracy and duration\(^1\).

Since the complexity of a system is frequently greater than the amount of known information, it becomes necessary to extract the maximum amount of information from the data available\(^2\). The most general and useful case arises when only a single time-series data set is available, requiring all other information to be derived from this vector and time increment. Due to the interrelation inherent in nonlinear equations, the trajectory of each

\(^{**}\) The evaluation of the delayed vector coordinates of the Rossler Attractor, based on the first complete run of the fitting procedure.
variable is dependent on all the other variables; consequently, all the information required to reconstruct a given variable is contained within itself. The most accepted method to extract this information is based on Floris Takens' theory of delay-vector space, $X\tau$, which is generated by the following transpose:

$$X\tau = \begin{pmatrix} x(t) \\ x(t + \tau) \\ \vdots \\ x(t + \mu\tau) \end{pmatrix}^T,$$

where $\tau$ is the time shift, or delay value, and $\eta$ is the degree of freedom of the system. Takens proved that the delayed-vector space is topologically equivalent to the phase portrait of the system. Based on this theory, an artificial system of equations may be developed, such that:

$$\dot{X}\tau = f(X\tau)$$

which resembles an ordinary system of state equations, except that the additional state variables are substituted by shifted values of the one data set.

The development of this vector space into a set of equations will be broken down into three main parts. First a shifting value that provides the maximum amount of information about the time series must be established. The standard method of determining this value, based on the autocorrelation function, and a modification of this technique, mutlicorrelation, are examined in Section 2, which also includes examples of the topological equivalence. Once an appropriate shifting value has been evaluated, the degrees of freedom for the system, equivalent to the number of equations required, can then be determined. Based on the topological properties of the delayed-vector space, the fractal dimension, $\gamma$, of the system can be calculated as the delay-vector space is expanded. The evaluation of this dimension is based on the distance correlation function introduced by Grassberger-Proccacia and is investigated in Section 3. Finally, the delayed-vector space can be constructed and then fitted against its numerically calculated derivative to generate an artificial system of equations.
which may be used to predict the system. The generalized least-squares algorithm used for
this procedure is developed and evaluated in Section 4, focusing on the possible dependence
on an optimum shifting value\(^1\). Conclusions and recommendations for future studies are
included at the end of each section.

This procedure is useful when only limited information about a system is known or
available, which is a frequent occurrence when investigating natural phenomena\(^9\). It may also
be used to render additional insight into existing nonlinear mechanical and electrical systems.
An example of this may be inferred from the pendulum, which is used to introduce common
terminology and properties of nonlinear systems in Sections 1.1 and 1.2, such as: phase space,
autonomous systems, and nonlinearity\(^10\). Also in Section 1.2, the systems being investigated
throughout this study are introduced, including: the Henon Mapping, and the Rossler, Lorenz,
and Lorenz-2 Attractors. Additionally, the numerical technique required to integrate these
equations and calculate necessary derivatives are included in Sections 1.3 and 1.4,
respectively\(^11\). All programs are listed sequentially, with respect to the text in which they are
referred, in Appendix I\(^12\).
1.1 Phase Space and Autonomous Systems

To begin our explanation of autonomous systems and phase space, consider the pendulum, often thought to be one of the simplest dynamical systems\(^{[10]}\). Depicted below is the standard representation of a simple pendulum, with center of mass, \(m\), located at a fixed length, \(l\), from its pivot point,

\[ \begin{align*}
\text{ml} & \ddot{\theta} + c\dot{\theta} + mg \sin \theta = \alpha \cos(\omega_f t)
\end{align*} \]

Reducing to nondimensional form, the equation becomes:

\[ \ddot{\theta} + 2\zeta\omega_n \dot{\theta} + \omega_n^2 \sin \theta = A \cos(\omega_f t) \]

where, \(\zeta\) is the damping factor and \(\omega_n\) is the natural frequency\(^{[13]}\). This system closely resembles a generic, second order system, which is easily achieved using the small angle approximation, \(\sin \theta \approx \theta\). Substituting variables \(x\) for \(\theta\), results in the familiar equation for a simple harmonic oscillator:

\[ \ddot{x} + 2\zeta\omega_n \dot{x} + \omega_n^2 x = A \cos(\omega_f t) \]

This linearization is generally assumed so that standard analytical techniques may be used.
The simplest case of this system is its free response with no damping, so that $A = 0$ and $\zeta = 0$, reducing the EOM to:

$$\ddot{x} + \omega_n^2 x = 0$$

For the typical initial conditions of zero displacement and unitary velocity, $x(0) = x_0 = 0$ and $\dot{x}(0) = \dot{x}_0 = 1$, the time series solution is readily shown to be:

$$x = \sin(\omega_n t)$$
$$\dot{x} = \omega_n \cos(\omega_n t)$$

The familiar sine and cosine time histories are shown in Figure 1.1.1A. In comparison, the phase space representation is shown in Figure 1.1.1B, which is simply the velocity verses the

**Figure 1.1.1:**
A) Time history of the free response of a second order, undamped system, position (top) and velocity (bottom).
B) The corresponding phase plane representation, $\omega_n = 1$. 

position, or more generally the derivatives plotted against the state variable. In this case and throughout the text, a two dimensional phase space, or phase plane, will be used. One benefit of using phase space is that, as long as the solution is bounded, an infinite time history may be easily represented. Furthermore, all information is contained geometrically.

The next simplest case is the free response of an underdamped system, $A=0$ and $0 < \zeta < 1$. Using the same initial conditions, $x_0 = 0$ and $x_0 = 1$, results in the following time series solutions:

$$ x = \frac{1}{\omega_d} e^{-\omega_d t} \sin(\omega_d t), \quad \text{where} \quad \omega_d = \omega_0 \sqrt{1-\zeta^2} $$

$$ \dot{x} = \frac{1}{\omega_d} e^{-\omega_d t} \left[ \omega_d \cos(\omega_d t) - \omega_0 \zeta \sin(\omega_d t) \right], $$

where $\omega_d$ is the resulting damped frequency of the system. Figure 1.1.2A and 1.1.2B depict the time history and phase portrait of the exponentially decaying oscillations, respectively. Note that for any reasonable initial condition, within the basin of attraction, this system will spiral to its steady state. Basins of attraction include all initial conditions whose trajectories will eventually converge to the attractor. In this case the steady state corresponds to the origin of the phase plane, which is accordingly called a point attractor.\(^{10}\)

This system may be alternatively expressed by the use of state equations, which is achieved by the simple substitution of variables $y = \dot{x}$, and appropriate changes in the initial conditions, $x_0 = x_0$ and $y_0 = \dot{x}_0$. The equation of motion becomes:

$$ \ddot{y} + 2\zeta \omega_0 y + \omega_0^2 x = A \cos(\omega_f t) $$

After an additional substition for the forcing term, $\phi = \omega_f t$, the following three state equations can be generated:

$$ \dot{x} = y $$

$$ \dot{y} = -\omega_0^2 x - 2\zeta \omega_0 y + A \cos \phi $$

$$ \dot{\phi} = \omega_f $$

This representation of the system is now considered to be autonomous, since there is no longer a dependence on time. Solutions to these equations are easily iterated from the initial
values using numerical integration techniques, such as the fourth-order Runge-Kutte method, which is the topic of section 1.3\textsuperscript{(11)}. Based on these equations, consider the harmonic response of a significantly underdamped second order system. The brief, initial transient portion and steady state response are depicted as a time history and phase portrait in Figures 1.1.3A and B, respectively. In the phase plane, the initial conditions interior and exterior are attracted to the elliptical steady state, known as a limit cycle or orbital attractor\textsuperscript{(10)}. 

Figure 1.1.2:
A) Time history of the free response of an underdamped, second order system, \(\zeta = 0.25\) and \(\omega_n = 1\), position (top) and velocity (bottom).

B) --- Equivalent phase space representation, --- additional initial condition to demonstrate point attraction.
Figure 1.1.3:

A) Time history of the harmonic response of an underdamped, second order system,
\( \zeta = 0.25, \omega_n = 1, \omega_f = 2/3 \text{ rad./sec}, \) and \( A = 0.9. \)

B) The equivalent phase plane system depicting a limit cycle and/or orbital attractor.

Compare the phase portraits of linearized model of the pendulum to the original, nondimensional EOM which may be rewritten as the following autonomous state equations:

\[
\begin{align*}
\dot{\theta} &= \omega \\
\dot{\omega} &= -\omega^2 \sin \theta - 2\zeta \omega \omega + A \cos \phi \\
\dot{\phi} &= \omega_f
\end{align*}
\]

Due to the retention of the nonlinear sine term, the true dynamics of the system are more accurately described\(^{[10]}\). An example of a point attractor and limit cycle, equivalent to figures
1.1.2B and 1.1.3B, are shown in figures 1.1.4 and 1.1.5, respectively\textsuperscript{10}. While the behavior is similar, the trajectories are slightly irregular, reflecting the nonlinearity of the system. The next section, Strange Attractors, further examines several properties and behaviors related to nonlinear systems.

**Figure 1.1.4:**
The actual free response of an underdamped pendulum. Depiction of point attraction, for parameters: \( \omega_n = 1, \ \zeta = 0.25, \ \omega_f = 2/3, \) and \( A = 0.9. \)

**Figure 1.1.5:**
The actual harmonic response of a pendulum. Depiction of a limit cycle and/or orbital attractor, for parameters: \( \omega_n = 1, \ \zeta = 0.25, \ \omega_f = 2/3, \) and \( A = 0.9. \)
1.2 Strange Attractors

"Strange attractor" is a descriptive term for the patterns generated in phase space by nonlinear dynamic equations, which can not be simply described using standard geometric terms, such as point, orbit, as discussed in the previous section\cite{9}. There is, however, a definite structure or pattern that a trajectory is attracted to and will stay within, given initial conditions within the basin of attraction\cite{14,15}. The purpose of this section is to acquaint those who are unfamiliar with several characteristics of chaotic systems, such as, non-linearity, parameter dependence, and sensitivity to initial conditions\cite{14,15}. Sub-sections 1.2.1 - 4 introduce the attractors used throughout this study, including: the Henon Mapping, Rossler's Attractor, Lorenz's Attractor, and a second variation of a Lorenz Attractor, respectively\cite{14,16}.

Nonlinearity refers to the coupling of variables in a system of equations that can not be expressed as a proportional relationship\cite{10}. Examples of nonlinear terms include: non-unitary and non-zero exponents, trigonometric functions, logarithmic functions, and especially the product of two or more variables. Additionally, unlike steady state solutions of linear differential equations, the solutions of nonlinear systems can not always be expressed by the linear superposition of periodic functions\cite{10}. However, it is the parameters of the equations which determine whether or not the system is periodic, quasi-periodic, or chaotic\cite{10,14}. Recall from the previous section the state equations for the pendulum:

\[
\begin{align*}
\dot{\theta} &= \omega \\
\dot{\omega} &= -\omega^2 \sin \theta - 2\zeta \omega \omega + A \cos \phi \\
\dot{\phi} &= \omega_f 
\end{align*}
\]

As shown in Figure 1.1.5, and based on common experience, the pendulum is usually considered to behave periodically. However, if the parameters are set to specific values, quasi-periodic and even chaotic behavior can theoretically be generated. One such chaotic interval occurs for values of the parameters of \( \omega_n = 1, \zeta = 0.25, \omega_f = 2/3, \) and \( 0.5 \leq A \leq 1.5, \) in which the behavior ranges through all varieties of behavior\cite{10}. Figures 1.2.1 and 1.2.2 show the quasi-period 2, for \( A = 1.07, \) and chaotic behavior, for \( A = 1.15, \) respectively\cite{10}. 

10
Figure 1.2.1:
Double periodic behavior for a forced pendulum with parameters:
\( \omega_n = 1, \zeta = 0.25, \)
\( \omega_f = 2/3, \) and
\( A = 1.07 \)

Figure 1.2.2:
Chaotic behavior for a forced pendulum with parameters:
\( \omega_n = 1, \zeta = 0.25, \)
\( \omega_f = 2/3, \) and
\( A = 1.15 \)

Originally noticed by Lorenz in 1961, sensitivity to initial conditions is one of the most important characteristics of chaotic systems\(^{[15]}\). It is the tendency for two trajectories that have almost identical initial conditions, to become radically different with the progression of time, as to have no relation. Consequently, the prediction of a nonlinear system is difficult and may only be achieved, with reasonable accuracy, for a limited number of iterations. To further emphasize the sensitivity of chaotic systems, Figure 1.2.3A shows two time histories of the
first position vector of identical Lorenz equations, with the exact same initial conditions, see section 1.2.3 for the equations used. The divergence here is caused by slightly different time steps used in the integration, causing small errors to be propagated in each iteration, until the results no longer have any correlation. Figure 1.2.3B shows the subsequent error between the two position vectors. This is similar to the error plots used in section 4 to demonstrate the differences between the actual velocities and the predicted ones.

**Figure 1.2.3:**
A) Two time portraits for the Lorenz Attractor, with identical initial conditions, using two different time increments for integration.
   --- $\Delta t = 0.002$
   and
   $\cdots \Delta t = 0.005$.
B) The subsequent error between the two position vectors, similar to the error plots used in section 4.
1.2.1 The Henon Mapping

Developed by Michel Henon as a simplified model for the dynamics of the Lorenz system, the Henon mapping is one of the simplest sets of equations that depicts nonlinear behavior\(^9\). The original difference equations used by Henon are as follows:

\[
x_{i+1} = 1 - 1.4x_i^2 + y_i \\
y_{i+1} = 0.3x_i
\]

Alternative parameters may be chosen that will also generate chaotic behavior. One of the key characteristics of this mapping is that it covers the entire attractor in relatively few, simple iterations, which is important when working with slower computers. Figure 1.2.4 shows a Henon mapping containing 10,000 points. This was generated using the Matlab program \textit{henon.m}, which runs independently given an initial condition \([x_0, y_0]\) and the number of additional points desired. The origin makes a good initial condition, although the first several points are obviously not on the attractor and are consequently considered transient. Common practice is to discard the first hundred or so points.

\textbf{Figure 1.2.4:}
Henon Mapping, 10,000 points.
1.2.2 The Rossler Attractor

Another simplification of Lorenz's equations is Otto E. Rossler's attractor. It is an artificial system created specifically to represent the stretch-and-fold characteristic of chaotic systems, reducing the nonlinearity to its most fundamental, a single coupling between the first and third variables\(^9\). It may be represented by the following equations:

\[
\begin{align*}
\dot{q}_1 &= -q_2 - q_3 \\
\dot{q}_2 &= q_1 + a \cdot q_2 \\
\dot{q}_3 &= b + q_1 \cdot q_3 - c \cdot q_3
\end{align*}
\]

where \(a=0.2\), \(b=0.2\), and \(c=4.6\) are the values used in our investigations.

The stretch-and-fold characteristic is clearly demonstrated in the plot of the first and second state variables shown in Figure 1.2.5. The three position vectors, with equal time steps, may be generated using the Matlab program \(\text{rossler.m}\) in conjunction with \(\text{rk4.m}\), from which the corresponding velocities may be calculated using \(\text{rosvel.m}\). Figure 1.2.6 depicts the time history of the three position vectors, with the corresponding phase planes shown in Figures 1.2.7 - 9.

**Figure 1.2.5:**
Typical trajectory for the first and second position vectors of the Rossler Attractor.
Figure 1.2.6: Time portrait of Rossler Attractor, time step, $\Delta t = 0.02$, and initial conditions, $q_0 \equiv [ -5.7168, -2.8577, 0.0190 ]^T$.

Figure 1.2.7: Phase portrait of the first position and velocity vectors of the Rossler Attractor. Base Vector used for analysis.
Figure 1.2.8: Phase portrait of the second position and velocity vectors of the Rossler Attractor.

Figure 1.2.9: Phase portrait of the third position and velocity vectors of the Rossler Attractor.
1.2.3 Lorenz Attractor

Running a simplified computer weather simulation in 1961, Edward Lorenz first noticed the property of sensitive dependence on initial conditions, better known as the butterfly effect\cite{15}. Based on these results, Lorenz looked for a simpler system which would still produce this complex behavior. This was achieved in the following equations, intended to model convection:

\[
\begin{align*}
\dot{q}_1 &= -a \cdot q_1 + a \cdot q_2 \\
\dot{q}_2 &= c \cdot q_1 - q_2 - q_1 \cdot q_3 \\
\dot{q}_3 &= q_1 \cdot q_2 - b \cdot q_3
\end{align*}
\]

where \(a=10\), \(b=8/3\), \(c=28\) are the parameters originally chosen by Lorenz and used throughout this study\cite{9}.

Like the Rossler Attractor, the position vectors and corresponding velocities may be generated using the Matlab programs \textit{lorenz.m}, \textit{rk4.m}, and \textit{lorvel.m}. The well known first and third position trajectory, resembling butterfly wings, is shown in Figure 1.2.10. The time histories of the three position vectors are depicted in Figure 1.2.11, with the corresponding phase planes shown in Figures 1.2.12 - 1.2.14.

\textbf{Figure 1.2.10:}
Lorenz's butterfly, typical trajectory for the first and third position vectors, 10,000 points.
Figure 1.2.11: Time portraits of Lorenz Attractor, time step, $\Delta t = 0.005$, and initial conditions, $q_0 \equiv [5.0441, -2.2363, 31.6983]^T$.

Figure 1.2.12: Phase portrait of the first position and velocity vectors of the Lorenz Attractor. Base vector used for analysis.
Figure 1.2.13: Phase portrait of the second position and velocity vectors of the Lorenz Attractor.

Figure 1.2.14: Phase portrait of the third position and velocity vectors of the Lorenz Attractor.
1.2.4 Lorenz-2

In order to verify the methods developed using the Rossler and the first Lorenz Attractors, a third dynamic system was sought. This lead to the following equations based on Lorenz's 1963 article in *Atmospheric Science*[^16].

\[
\begin{align*}
\dot{q}_1 &= -a \cdot (q_1 - F) - q_2^2 - q_3^2 \\
\dot{q}_2 &= q_1 \cdot q_2 - b \cdot q_1 \cdot q_3 - q_2 + G \\
\dot{q}_3 &= b \cdot q_1 \cdot q_2 + q_1 \cdot q_3 - q_3
\end{align*}
\]

where \(a=0.25\), \(b=4\), \(F=8\), and \(G=1\) are the parameters used in this study.

Although this system is likely to pre-date the attractor described in the previous section, it will be referred to as the Lorenz-2 Attractor, while the more familiar set of equations will be addressed as just the Lorenz Attractor. The position vectors and velocities may be generated using the Matlab programs *lorenz2.m* in conjunction with *rk4.m*, and *lorvel2.m*, respectively.

Figure 1.2.15 shows the clearest two dimensional representation of the attractor, the second and third position trajectory. Figure 1.2.16 depicts the time histories of the three position vectors, with the corresponding phase planes shown in Figures 1.2.17 - 19.

**Figure 1.2.15:**
Typical trajectory of the second and third position vectors of the Lorenz-2 Attractor, 10,000 points.
Figure 1.2.16: Time portrait of Lorenz-2 Attractor, time step, $\Delta t = 0.005$, and initial conditions, $q_0 = \begin{bmatrix} 0.8489 \\ -0.8378 \\ -1.6962 \end{bmatrix}$.

Figure 1.2.17: Phase portrait of the first position and velocity vectors of the Lorenz-2 Attractor.
Figure 1.2.18: Phase portrait of the second position and velocity vectors of the Lorenz-2 Attractor.

Figure 1.2.19: Phase portrait of the third position and velocity vectors of the Lorenz-2 Attractor. Base vector used for analysis.
1.3 Runge-Kutta Integration Method

In order to generate solutions for the nonlinear state equations of the pendulum, the Rossler Attractor and the two Lorenz Attractors, see the previous section, a numerical integration method is required. The Matlab package includes two variable step integration methods: ODE23, which uses second and third order Runge-Kutta methods; and ODE45, which uses fourth and fifth order methods\[^{12}\]. Although the optimization increases the speed and accuracy of the integration, it also results in varying time increments, which does not typically reflect sampled data. Additionally, the vector can not be differentiated using ordinary fixed interval methods. Consequently, a simpler algorithm, using the ordinary fourth-order Runge-Kutta method, has been created, \textit{rk4.m}, which implements the following equations:

\[ X_{i+1} = X_i + \frac{1}{6} (k_1 + 2k_2 + 2k_3 + k_4), \]
\[ k_1 = \Delta t \cdot f(X_i, t_i), \]
\[ k_2 = \Delta t \cdot f(X_i + \frac{1}{2} k_1, t_i + \frac{1}{2} \Delta t), \]
\[ k_3 = \Delta t \cdot f(X_i + \frac{1}{2} k_2, t_i + \frac{1}{2} \Delta t), \]
\[ k_4 = \Delta t \cdot f(X_i + k_3, t_i + \Delta t), \]

where \( X \) represents the state vectors, and \( t \) represents the corresponding time vector with increments of \( \Delta t \). The function, \( f \), represents the state equations involving \( X \) and \( t \), as in the pendulum example in section 1.1. The integration programs all require the input function to be written as a separate M-file, which may then be called using single quotes around the name, for example 'rossler'. Additional input arguments include: initial time, \( t_i \); time step, \( \Delta t \); final time, \( t_f \); and the initial state variable, \( q_0 \). In contrast to the intrinsic functions ODE23 and ODE45, \textit{rk4.m} outputs the state vectors first and time vector second. The state equation programs may be easily modified if desired. Another application of this Runge-Kutta method is \textit{rkpoly.m}, which is used in conjunction with \textit{nlpoly.m} to solve the coefficient matrix generated to fit the data, using \textit{nlfit.m}. This application is discussed in further detail in section 4.3.
1.4 Central Difference Method for Differentiation

The opposite of numerical integration, numerical differentiation, is equally important to this investigation. Since Matlab does not have a specific differentiation procedure, cent2.m, cent4.m, and cent6.m have been created using the following central difference methods, respectively:

Second-order central difference method

\[ \dot{x}_i = \frac{x_{i+1} - x_{i-1}}{2\Delta t} + O(\Delta t^2) \]

Fourth-order central difference method

\[ \dot{x}_i = \frac{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}}{12\Delta t} + O(\Delta t^4) \]

Sixth-order central difference method

\[ \dot{x}_i = \frac{x_{i+3} - 9x_{i+2} + 45x_{i+1} - 45x_{i-1} + 9x_{i-2} - x_{i-3}}{60\Delta t} + O(\Delta t^6) \]

where \( \dot{x}_i \) is the derivative at point \( x_i \), and \( \Delta t \) is the sampling increment\(^{[11]} \). The order of the central difference refers to both its accuracy and the number of unusable points. For example, in a sixth-order central difference, the error due to truncation is on the order of magnitude, \( O \), of the time step raised to the sixth power. Additionally, the derivatives for the first and last three points are not available using this technique, and consequently lost.

In its present form, this study uses the central difference method for two applications. The second-order central difference is used to find the first inflection point of the correlation function, see section 2.2, where the value of the derivative is less important than the number of discarded points. The more accurate, sixth-order method has been formulated to calculate the derivatives that will be used to fit the delayed space equations. This is necessary to minimize the accumulation of error resulting from the series of numerical approximations being performed, see section 4.3 for the actual procedure.
2.0 Determination of an optimal shift, $\tau$.

*We do what we can,*  
*and then make a theory to prove*  
*our performance the best.*  

- Emerson

The shifting time, $\tau$, is fundamental to the evaluation of the system dynamics based on Takens' theory, described in the introduction, Section 1.0\textsuperscript{[4]}. This value is essential to evaluation of the fractal dimension of the system, covered in Section 3, and ultimately determines the stability of the curve fitting procedure discussed in Section 4. Consequently, it is of primary importance to establish an accurate method of determining this value. There are a considerable number of articles devoted to the evaluation of this quantity; however, a definitive method for finding the optimum value, $\tau_{\text{opt}}$, has yet to be established\textsuperscript{[17]}. The most common practice is based on the discrete, non-periodic autocorrelation function, which is

** The Multicorrelations of the Rossler Attractor for dimensions two through five, used to evaluate an approximate optimum shifting value.
examined in Section 2.1. A variation of this method, multicorrelation, is introduced in Section 2.2[7].

Since only discrete systems are being evaluated, the optimum time shift, $\tau_{opt}$, must be approximated as an index shift, $v_{opt}$, where:

$$\tau \equiv v \cdot \Delta t$$

The values considered to be optimum in this study have been established through a trial and error procedure, bracketing the index which resulted in the best curve fit. This process is investigated in detail in Section 4. The approximate optimum values for the systems being investigated are listed in the Table 1.0.1, below. These values are averaged from several curve fittings and are recommended as bench marks, not absolutes, for the local optimum shift. Optimum values tend to vary slightly due to the local dependence of the section of the attractor being investigated.

<table>
<thead>
<tr>
<th>Estimated Optimum Time Shift, $\tau$</th>
<th>Rossler</th>
<th>Lorenz</th>
<th>Lorenz-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.46 sec.</td>
<td>0.16 sec.</td>
<td>0.17 sec.</td>
<td></td>
</tr>
<tr>
<td>$\Delta t = 0.02, v = 23$</td>
<td>$\Delta t = 0.005, v = 32$</td>
<td>$\Delta t = 0.005, v = 34$</td>
<td></td>
</tr>
<tr>
<td>$\Delta t = 0.005, v = 92$</td>
<td>$\Delta t = 0.002, v = 80$</td>
<td>$\Delta t = 0.002, v = 85$</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.0.1: General optimum shifting values for the Rossler, Lorenz and Lorenz-2 Attractors.
2.1 Autocorrelation

The autocorrelation function is typically used to provide information about random functions\textsuperscript{[13]}. Specifically, it evaluates the dependence of the function value at one time, \( t \), on the function value at another time, \( t+\tau \). In its most familiar form, for continuous, periodic functions, the autocorrelation is expressed as:

\[
A_c(\tau) = \frac{1}{T} \int_{0}^{T} x(t) \cdot x(t + \tau) dt, \quad 0 \leq \tau \leq T
\]

where \( T \) represents the period of the function and \( \tau \) is the time shift\textsuperscript{[13]}. Qualitatively, this correlation gives a maximum value when \( x(t) \) and \( x(t+\tau) \) are identical, which occurs when the shifting value is equal to zero and at multiples of the period, \( nT \). Another point of interest is when the autocorrelation goes to zero. At this value the two functions are theoretically orthogonal, which was originally expected to be the shift that would yield the maximum amount of information about the system.

When only numerical data is available, the discrete, periodic form of the equation may be used, which is written:

\[
A_c(j) = \frac{1}{N} \sum_{i=1}^{N} x_i \cdot x_{i+j}, \quad j = 0, 1, 2, \ldots, N/2
\]

where \( N \) is the number of points in one period and \( j \) is the index shift. Only the first half of the correlation needs to be performed, since the second half is simply its mirror image and does
not increase the amount of information about the system. To keep the total number of multiplications equal to N, use either additional points after the first period or roll over the original data set.

This equation may be generalized for discrete, non-periodic data sets, by considering the period to be infinitely long, consequently making N the total number of points. Since the data is now finite, appropriate adjustments must be made in the normalization, resulting in the following equation:

\[
A_e(j) = \frac{1}{N-j} \sum_{i=1}^{N-j} x_i \cdot x_{i+j}, \quad j=0,1,2,\ldots,N/2
\]

Due to the decreasing amount of information contained in the N-j length vectors, only the first half of the shifted values, N/2, are generally considered, and typically less may be of interest. This function is implemented by the Matlab program \textit{autocor.m}, which also evaluates the first local minimum of the autocorrelation, denoted as the index shift \( v_{\text{min}} \). It is this point which will be shown to be a possible property of the attractor, as opposed to the previously mentioned index shift causing the autocorrelation to go to zero, denoted as \( v_0 \).^7

Our investigation into time shifts begins with the Henon Mapping, due to its speedy generation and the relatively few points required to describe the attractor. The autocorrelations for the x and y vectors are shown in Figure 2.1.1. Aside from the difference in magnitudes, the correlations are identical. For both, the first shift, \( v = 1 \), causes the autocorrelation to cross zero and to be at its first local minimum; consequently, the difference between these two points can not yet be distinguished. However, several important properties may still be noted from this example. The delayed coordinate representation may be created by plotting the shifted values, \( x_{i+v} \), against the original vector. The results resemble a flipped version of the original mapping, adjusted by some constant, see Figure 2.1.2. This is expected since the y value is simply scaled from the previous x value. \(^7\) See Section 1.2.1 for the equations generating this mapping. More importantly, this property holds true for dynamic systems as well, which will be demonstrated for the Rossler and Lorenz systems. Lastly,
Figure 2.1.3 depicts the effect of choosing a larger shifting value that's autocorrelation is closer to zero, $v = 5$. The points begin to scatter and no longer maintain the form of the original attractor, demonstrating the affect of over-shifting and the dependency recent values\textsuperscript{[17]}. This condition worsens as the shifting value is increased.

Figure 2.1.1:
Autocorrelation of Henon Mapping, $x$ (top) & $y$ (bottom). The first index crosses the zero axis and is also the first local minimum, $v_0 = \nu_{\min} = 1$. 

Figure 2.1.2: 
Delayed coordinate representations of the Henon Mapping, $x$ (larger) and $y$ (smaller), $v = 1$. 


Figure 2.1.3:  
Delayed coordinate representation of the Henon Mapping, with an index shift of $v = 5$. The loss of the attractor form is the result over-shifting.

Next, consider the autocorrelation for the Rossler Attractor, which tends to be extremely consistent, regardless of the starting point or the number of points being evaluated. This may be demonstrated by running two correlations with differing initial conditions, one of twenty thousand points and the other for just two thousand. An example of this may be seen in Figure 2.1.4, which depicts almost identical values for $v_0$ and $v_{min}$. This consistency is likely due to the almost periodic behavior of the attractor, indicative of its simplicity, and is

Figure 2.1.4:  
Autocorrelation for two Rossler Attractors, with different initial conditions and $\Delta t = 0.02$.  
- 20,000 pts, IC$_1$  
  $v_0 = 74$, $v_{min} = 145$  
- 2,000 pts, IC$_2$  
  $v_0 = 74$, $v_{min} = 146$
not apt to apply to more complicated ones. As with the Henon Mapping, the Rossler Attractor may be depicted using delay coordinates. The orthogonal vectors created by using the shift \( v_0 \) are shown in Figure 2.1.5. While the construction is possibly recognizable as Rossler's, there is a slight folding which is indicative of over-shifting\(^{[17]}\). In contrast, Figure 2.1.6 depicts the delay coordinate representation of the optimal shifting value, \( v_{\text{opt}} \).

**Figure 2.1.5:**
Delay coordinate construction of a Rossler Attractor with \( \Delta t = 0.02 \) and \( v = v_0 = 74 \). Folding indicates over-shifting.

**Figure 2.1.6:**
Delay coordinate construction of the Rossler Attractor, with \( \Delta t = 0.02 \), using optimal shifting value, \( v_{\text{opt}} = 23 \).
In contrast to the Rossler Attractor, the autocorrelation for the Lorenz Attractor is extremely inconsistent when the initial conditions or the length of the section being evaluated are varied. This is demonstrated in Figure 2.1.7, where the correlations for four position vectors are compared, starting with two initial conditions and continuing for either ten or twenty thousand points. From this figure it is apparent that the four correlations do not cross the zero axis at the same point, clearly proving that \( v_0 \) is not a consistent characteristic of the system. Although the values of the first local minimums are not identical, they are much closer to being coincident, increasing the likelihood of this being a property of the attractor. Furthermore, the 20,000 point correlation, for the second initial condition, just misses having a minimum closer to that of the 10,000 point correlation. Figure 2.1.8 shows the over-shifting of the Lorenz Attractor using the smallest of the values obtained from these autocorrelations. This should be compared to the delayed coordinate representation of the optimum shift, depicted in Figure 2.1.9, which is considerably less than the first local minimum.

**Figure 2.1.7:**
Autocorrelations for four Lorenz Attractors of varying initial conditions and lengths, with \( \Delta t = 0.005 \).
- 20,000 pts, IC
  \( v_0 = 124, v_{\text{min}} = 157 \)
- 10,000 pts, IC
  \( v_0 = 117, v_{\text{min}} = 156 \)
- 20,000 pts, IC
  \( v_0 = 523, v_{\text{min}} = 195 \)
- 10,000 pts, IC
  \( v_0 = 569, v_{\text{min}} = 127 \)
Figure 2.1.8:
Delay coordinate construction of a Lorenz Attractor with $\Delta t = 0.005$ and $v = v_0 = 117$. Folding indicates over-shifting.

Figure 2.1.9:
Delay coordinate construction of the Lorenz Attractor, with $\Delta t = 0.005$, using optimal shifting value, $v_{\text{opt}} = 32$.

In their article *Prediction in Chaotic Nonlinear Systems*, Henry Abarbanel, et. al., suggest using 1/10 to 1/20 of the local minimum as the optimum shift, which they admit is a somewhat arbitrary method\[^7\]. Furthermore, this technique does not adequately approximate the optimum values obtained from the curve fitting for either Rossler or Lorenz systems. Consequently, a more definitive method for determining the optimum shift has been deemed necessary. Although seemingly trivial, it is important to note that the most pertinent
information about a point is contained locally, as the shift approaches zero, \( v \to 0 \). The loss of this quality is represented by the over-shifting depicted in the previous examples for Henon, Rossler, and Lorenz. The transition from this dependency on the most recent values, to the first local minimum of the autocorrelation, is the first inflection point. For the Lorenz Attractor, this inflection point occurs almost exactly at the optimum shifting value required by the curve fitting. Regretfully, the corresponding point for the Rossler Attractor is still over three times the optimum value. This discrepancy has led to the investigation of the inflection point for higher-order correlations, or Multicorrelation, which is the topic of the next section.
2.2 Multicorrelation

Multicorrelation generalizes the autocorrelation function by adding multipli-shifted vectors into the correlation, to some higher dimension, $\eta$, similar to the dimensional expansion of Takens' theory. This may be expressed by the following equation:

$$M_c(\eta, j) = \frac{1}{N-\mu j} \sum_{i=1}^{N-\mu j} x_i \cdot x_{i+\mu j} \cdot x_{i+2\mu j} \cdots x_{i+\mu j}, \quad j=0,1,2,\ldots,N/2$$

for simplicity $\mu=\eta-1$, one less then the dimension. The autocorrelation is equivalent to a two dimensional multicorrelation. This is the algorithm implemented by the Matlab programs *multicor.m*, which calculates the correlation for an individual dimension, and *mdimcor.m*, which calculates the correlations for two through some final dimension. These programs use a second-order central difference method to calculate the derivative and then find the first local maximum or minimum, corresponding to the first inflection point of the correlation, $v_{\inf}$. Additionally, the correlations are normalized by dividing by the first value, so that the graphical results may be more easily compared.

Typical 2-5 dimensional multicorrelations for the Rossler, Lorenz, and Lorenz-2 Attractors are depicted in Figures 2.2.1 - 3, respectively. As with the autocorrelation, the

**Figure 2.2.1:**

2-5 dimensional multicorrelation for the Rossler Attractor.

- 2, $v_{\inf} = 70$
- 3, $v_{\inf} = 22$
- 4, $v_{\inf} = 17$
- 5, $v_{\inf} = 10
multicorrelations for the Rossler Attractor are extremely regular and do not alter significantly, by changing either the length or initial condition. On the other hand, the correlations for Lorenz vary substantially with the section of the attractor being investigated. The Lorenz-2 is somewhere in between, not as orderly as Rossler, and not as irregular as the other Lorenz system. Due to the additional terms in the correlations, each consecutively higher dimension contains an increased number of maximums and minimums. Additionally, the Lorenz
Attractor shows an obvious odd and even bias. This is due to the structure of the system, having two basins of attraction. Although not as influential, this odd/even property of the multicorrelation is common to all attractors.

The first inflections, for these correlations and additional initial conditions and time steps, are listed in Table 2.2.1. For comparison, at the end of each row is the equivalent local

Table 2.2.1: First inflection points for dimensions 2-5 for various initial conditions and time steps, for the Rossler, Lorenz, and Lorenz-2 Attractors, including optimal shifting time resulting in the best curve fit for the beginning of that section.

<table>
<thead>
<tr>
<th>Attractor</th>
<th>Init. Cond.</th>
<th>No. of Points</th>
<th>Time Shift</th>
<th>Multicorrelation Dimension</th>
<th>Optimum Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rossler</td>
<td>IC₁</td>
<td>5000</td>
<td>0.020</td>
<td>70 22 17 10</td>
<td>~ 23</td>
</tr>
<tr>
<td></td>
<td>IC₂</td>
<td>5000</td>
<td>0.020</td>
<td>69 22 17 10</td>
<td>~ 23</td>
</tr>
<tr>
<td></td>
<td>IC₂</td>
<td>20000</td>
<td>0.005</td>
<td>277 84 69 40</td>
<td>~ 92</td>
</tr>
<tr>
<td></td>
<td>IC₃</td>
<td>20000</td>
<td>0.005</td>
<td>276 84 69 40</td>
<td>~ 91</td>
</tr>
<tr>
<td>Lorenz</td>
<td>IC₁</td>
<td>20000</td>
<td>0.005</td>
<td>33 22 10 12</td>
<td>~ 32</td>
</tr>
<tr>
<td></td>
<td>IC₂</td>
<td>20000</td>
<td>0.005</td>
<td>33 23 10 12</td>
<td>~ 32</td>
</tr>
<tr>
<td></td>
<td>IC₂</td>
<td>25000</td>
<td>0.002</td>
<td>82 52 24 27</td>
<td>~ 77</td>
</tr>
<tr>
<td></td>
<td>IC₃</td>
<td>25000</td>
<td>0.002</td>
<td>82 28 24 13</td>
<td>~ 80</td>
</tr>
<tr>
<td>Lorenz-2</td>
<td>IC₁</td>
<td>20000</td>
<td>0.005</td>
<td>53 34 16 13</td>
<td>~ 34</td>
</tr>
<tr>
<td></td>
<td>IC₂</td>
<td>20000</td>
<td>0.005</td>
<td>51 35 15 13</td>
<td>~ 35</td>
</tr>
<tr>
<td></td>
<td>IC₂</td>
<td>25000</td>
<td>0.002</td>
<td>123 85 37 31</td>
<td>~ 85</td>
</tr>
<tr>
<td></td>
<td>IC₃</td>
<td>25000</td>
<td>0.002</td>
<td>134 86 40 32</td>
<td>~ 85</td>
</tr>
</tbody>
</table>
optimum time shift, based on the best curve fit for the first 20% of points from the section being evaluated. The curve fitting process is covered in detail in section 4, with the results for selected sections shown in section 4.3. As mentioned in the previous section, the optimum shifting value for the Lorenz Attractor is approximately equal to the first inflection point for a two dimensional correlation, equivalent to the autocorrelation. For the Rossler and Lorenz-2 systems, the optimums are very close to the first inflection for a three dimensional correlation. This discrepancy is most likely due to the strong odd/even bias noted for the Lorenz Attractor, possibly causing the optimum shift to have a stronger affiliation for the even correlation. The possibility of fitting the Lorenz Attractors with only two delayed vectors was attempted and proved unobtainable using the current curve fitting procedure.

The deviation between the first inflection points and the optimum values is in part due to local affects, resulting from curve fitting a smaller segment of the attractor than was used in the correlation. Furthermore, the inflection points seems to be closer to the optimum shift when there is only limited information available for the fitting procedure. This may be inferred from the Rossler Attractor, which is the easiest to fit, yet the inflection point diverges from the optimum shift as the time step is decreased, effectively increasing the amount of information. The fit for the Lorenz system is not as stable as Rossler's and the inflection point remains fairly consistent with respect to the optimum. A completely stable fit for the Lorenz-2 was not obtained using the current fitting procedure. The values listed resulted in the longest and most stable fit. As the data becomes increasingly difficult to fit, owing to the nonlinearity or lack of information of the system, the inflection point becomes more crucial to the stability of the prediction. Consequently the importance of the inflection is directly related to the degree of nonlinearity of the system.

Another possible cause for the discrepancy of the inflection points, also based on the degree of nonlinearity, is that the exact inflection point would occur at the appropriate fractal dimension of the system. Since a fractal dimension is not achievable using this procedure, the inflection must consequently be inferred from the most applicable whole dimension. Since the
Rossler and Lorenz systems have fractal dimensions slightly over two, their inflection points are not as binding to the third dimensional correlation. Where as the more nonlinear Lorenz-2 Attractor is much more dependent on three equations. Again, it is the degree of nonlinearity that dictates the dependence of the optimum value on the inflection point.
2.3 Conclusion and Recommendations for the Optimum Time Shift

The standard method of estimating the shifting value based on the autocorrelation function has been expanded to include multipli-shifted vectors, similar to Takens' delay-vector expansion. The first inflection point of this correlation has been shown to be a fairly consistent method of determining the optimum shift required by the curve fitting procedure used in Section 4, based on a polynomial least-squares technique. For most cases the proper inflection occurs at the dimension equivalent to the degrees of freedom for the system. However, for the Lorenz system this point is found at the preceding dimension, possibly due to an odd/even bias caused by the multiple basins of attraction. Furthermore, this method has been shown to be more critical as the prediction becomes more difficult, which occurs when the sampling rate is decreased or as degree of non-linearity is increased.

Although this procedure works well for the low dimensional systems investigated here, its applicability to higher order systems, with varying degrees of nonlinearity, must still be established. Of equal or greater importance, the influence of noise on this method must also be determined, including its applicability to real data. Furthermore, a mathematical basis is still required to prove the validity of using the first inflection point of the multicorrelation as the optimum shift. It is also recommended that the procedure for finding the first inflection point be updated, to use a forward differencing method or equivalent process, so the first points are not discarded. Additionally, the effect of this method on the evaluation of the fractal dimension is investigated in the next section, Determining Dimensionality.
3.0 Determining Dimensionality

"The classification of the constituents of a chaos, nothing less here is essayed."

-Herman Melville, Moby Dick

Once a suitable time shift has been determined, it is then necessary to establish the number of independent variables and corresponding state equations that are required to fully describe the system, known as the degree of freedom, \( \eta \). While this quantity is not directly measurable, the complexity of the system may be gleaned in the form of its equivalent fractal dimension, \( \gamma \), which describes the interrelation of these variables\(^9\). Since equations can only exist in integer quantities, it can be shown that the degree of freedom may be obtained by rounding the dimensionality to the next higher whole value. For example, the Lorenz Attractor has a fractal dimension of approximately 2.06, and consequently requires three independent variables and state equations to generate this motion.

** The dimensional evaluation of the delayed vector coordinates of the Rossler Attractor, based on the first complete run of the fitting procedure.
Although there are numerous methods being investigated to establish this dimension, this study has chosen to use the most accepted method, the Grassberger-Procaccia distance correlation function\[^8\]. The basis of this theory is explained in section 3.1, and then applied to the systems being investigated in section 3.2. Table 3.2.1 lists the best obtained published values for the fractal dimensions that will be used as bench marks for our study\[^2,8,16,18\]. The approximation symbols denote that no exact numerical value has been specified by the publications.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Henon</th>
<th>Rossler</th>
<th>Lorenz</th>
<th>Lorenz-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.26</td>
<td>~2.02</td>
<td>2.06</td>
<td>~2.51</td>
</tr>
</tbody>
</table>

**Table 3.2.1:** Published system dimensions.
3.1 Fractal Dimensions

The numerical determination of dimension is most easily understood topologically, by considering the points of a system contained within a hypersphere of radius, \( r \), and some dimension, \( \eta \).

When the dimension of the hypersphere is smaller than the system's fractal dimension, \( \eta < \gamma \), the distances between the points are compressed, represented above by the space \( r^{\eta-1} \). This is illustrated by the two points connected by the dashed lines for which the distance is compressed as the space is reduced from \( \eta \) to \( \eta - 1 \). Oppositely, the distances will remain constant when the dimension of the hypersphere is equal to or greater than the system's, \( \eta \geq \gamma \). For example, the points on a plane are equally represented in two or three dimensional space. Hence, the dimension of a system converges as the embedding dimension of the hypersphere is increased above the dimension of the system.

To evaluate the fractal dimension, it can be shown that for sufficiently small radii, the number of points contained within a hypersphere of a given radius, \( D_c(r) \), increases
proportionally to the radius, which can be expressed by the relationship:

\[ D_c(r) \propto r^\gamma \]

or equivalently represented as the ratios:

\[ \frac{D_{c_1}}{D_{c_2}} = \frac{r_1^\gamma}{r_2^\gamma} \]

Taking the log of both sides and solving for the dimension results in the following equation:

\[ \gamma = \frac{\Delta \ln D_c}{\Delta \ln r} \]

Consequently, the dimension may be extrapolated from the slope of the double logarithmic plot of the number of points contained within a range of radii\(^8\). The use of dimensional embedding to determine the fractal dimension is readily applied to Takens' delayed-vector space, \(X_\tau\). As previously stated, the system's dimension will converge as the number of additionally shifted vector terms is increased.

One of the most accepted methods to perform this embedding is the distance correlation function, \(D_c(r)\), proposed by Grassberger and Procaccia, which calculates the distance between the \(i^{th}\) and \(j^{th}\) rows of a vector space, of length \(N\), and relates it to a given radius, \(r\), as follows:

\[ D_c(r, N, \eta) = \frac{2}{N(N-1)} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \Theta(r - \|X_{\tau_i} - X_{\tau_j}\|), \]

where \(X_{\tau_i}\) and \(X_{\tau_j}\) are rows of the delayed vector space expanded to dimension \(\eta\), see Section 1.0\(^{19}\). The heaviside function, \(\Theta\), is used to determine whether the distance is within the radius and depending on its argument, equals either one or zero, as follows:

\[ \Theta(y > 0) = 1 \]

\[ \Theta(y \leq 0) = 0 \]

The Euclidean Norm, represented by the double vertical bars, is used to calculate the distances, and may be expressed as:

\[ \|X_{\tau_i} - X_{\tau_j}\| = \sqrt{(X_{\tau_{i,1}} - X_{\tau_{j,1}})^2 + (X_{\tau_{i,2}} - X_{\tau_{j,2}})^2 + \cdots + (X_{\tau_{i,\mu}} - X_{\tau_{j,\mu}})^2} \]

where \(\mu\) is one less than the embedding dimension for notational convenience\(^7\).
Figure 3.1.1:
Typical dimensional embedding, showing the log-log plot of the distance correlation function vs. a range of radii.

Once this process has been repeated for a range of radii, a double logarithmic plot of the distance correlation values verses the radii may be generated. This is depicted in Figure 3.1.1 for the entire range of radii. Above a certain value, $R_u$, the slope tapers off until its asymptotic to zero, when all the points are contained in the hypersphere. At the opposite extreme, below $R_1$, there are not a sufficient number of points being evaluated to give an accurate correlation. Consequently, the almost linear portion between these values, the scaling region, is used to calculate the slope representing the system's dimension\(^{20}\).

The process of determining the dimension is implemented using the Matlab program *dimembed.m*. For each dimension, it performs the Grassberger-Procaccia distance correlation function and evaluates the number of points that fall within a specified scaling region. From this it then calculates the slope corresponding to the fractal dimension of the system and then plots the results. The procedure stops when embedding dimension is some preset value above the systems dimension, to allow for convergence. Each iteration also updates the maximum and minimum radii, which are then used for the next dimension. The application of this program is investigated in the next section, Dimensional Results.
3.2 Dimensional Results

The Grassberger-Procaccia distance correlation function is used exclusively throughout this study to determine the fractal dimensions of the systems. In an effort to obtain the most accurate results, various sections of the correlation have been investigated for use as the scaling region. Additionally, the affect of a constant shifting value is compared to the implementation of individual shifting values for each embedding dimension, evaluated by the multicorrelation function. Finally the influence of the sampling rate and size of the attractor section are evaluated generating the most reliable results. The convergence of the fractal dimension will be considered to be either the constant increase toward the known dimensional value, or subsequent oscillation about it, as the embedding space is expanded.

Our investigation into fractal dimensions begins again with the Henon Mapping, due to the relatively few number of points required to fully describe the system\(^2\). Figure 3.2.1 shows the dimension slowly converging to the desired value of 1.26. These results were obtained from a four thousand point Henon Mapping, embedded into dimensions three through nine. However, for another section of four thousand points, the results were just as

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure3.2.1.png}
\caption{Dimensional Embedding of a 4,000 point Henon Mapping, slowly converging toward a fractal dimension of 1.26.}
\end{figure}
likely to diverge. Smaller sections were originally processed resulting in a dimension of 1.21, which is a common value. Using DEC5000-125 series workstations, operating at six million instructions per second, these results required approximately four hours to obtain only inconclusive results. Due to the exponential nature of the distance correlation function, the anticipated twenty thousand points typically used to evaluate the other attractors would take approximately $5^2$ times as long, or approximately four days\textsuperscript{21}. Higher dimensional embeddings also require additional time due to the increased number computations in the distance computations resulting from the extra columns.

In an effort to achieve adequate results with minimum computation time, the distance correlations were run for only ten thousand points for the Rossler and Lorenz systems, requiring only about twenty-six hours each. Using moderate time steps and the optimum shifting values, the embedding dimension was significantly increased, by steps of three, up to twenty-one, in hopes that the fractal dimension would converge at higher dimensions. The results of these embeddings are shown in Figures 3.2.2 and 3.2.3, respectively. Regretfully, the Rossler Attractor only reached a dimension of 1.94 and the Lorenz Attractor completely diverged after the second embedding into six dimensions. In comparison to previous runs for

**Figure 3.2.2:**
Dimensional Embedding of a 10,000 point Rossler Attractor, failing to converge to a desired fractal dimension of 2.02, with $\Delta t = 0.02$ and a constant delay value, $\nu = 23$. 

<table>
<thead>
<tr>
<th>Dim.</th>
<th>Slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.75</td>
</tr>
<tr>
<td>6</td>
<td>1.86</td>
</tr>
<tr>
<td>9</td>
<td>1.85</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dim.</th>
<th>Slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1.86</td>
</tr>
<tr>
<td>15</td>
<td>1.89</td>
</tr>
<tr>
<td>18</td>
<td>1.92</td>
</tr>
<tr>
<td>21</td>
<td>1.94</td>
</tr>
</tbody>
</table>
Figure 3.2.3:
Dimensional Embedding of a 10,000 point Lorenz Attractor, diverging from a desired fractal dimension of 2.06, with $\Delta t = 0.005$ and a constant delay value, $\nu = 31$.

smaller sections of the attractors, the fractal dimension does not appear to be greatly affected by embedding spaces much greater than the dimension of the system. In these examples and the Henon Mapping, the distances between the embedding curves decreases relatively uniformly with increasing embedding dimension. This is due to the constant shifting values used for all the embeddings, which also seems to cause the fractal dimensions to increase or decrease relatively uniformly. In these examples the scaling region was based solely on approximately the middle third of the correlation, known as the Golden Section$^{[20]}$.

Based on these results and several other embeddings implementing a variety of ranges, it was decided that a section between 25% and 50% of the correlation tended to be the most consistent$^{[20]}$. Additionally, due to the nature of exponentials, the upper radius has been limited to be less than the log of one. Using this criteria, Figure 3.2.4 depicts one of the attempts to embed twenty thousand points of a Lorenz Attractor. In an effort of improve the accuracy of the points generated using the Runge-Kutta method, this run used a relatively small time step, $\Delta t = 0.001$. Although, the fractal dimensions did not reach the desired value, the results ended up fairly consistent, not rapidly diverging which frequently occurs for the Lorenz Attractor. This is in part due to the use of the multicorrelation function, implemented
by the Matlab program \textit{multicor.m}, which calculates a shifting value for each embedding dimension, see Section 2.2. This resulted in a decrease in the distances between the correlations, until the correlations almost overlap. This is due to the infection of the multicorrelation becoming increasingly small as the dimension is increased, until it is no longer recognizable. The shifting value then increases to the next higher inflection, which is represented by the subsequent large shift between the slopes of the distance correlation. This property is more readily visible in Figures 3.2.5 - 7, which show the distance correlations converging to the proper fractal dimensions for the Rossler, Lorenz, and Lorenz-2 systems, respectively. The correlations used the same scaling region criteria and shifting values, determined by the multicorrelation function, as the previous example. However, the twenty thousand points were based on sampling every fourth point from the original eighty thousand point attractors, generated with reasonable time steps\cite{16}. This effectively increased the quantity of information about the systems dynamics, without losing accuracy during the integration process. The fractal dimension is intended to embody the system's dynamics and can not be determined by simple dynamical closeness. Due to the composite time-step being

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\text{Dim.} & \text{Slope} \\
\hline
3 & 1.693 \\
5 & 1.688 \\
7 & 1.682 \\
9 & 1.680 \\
11 & 1.675 \\
13 & 1.673 \\
\hline
\end{tabular}
\end{table}

Figure 3.2.4: Dimensional Embedding of a 20,000 point Lorenz Attractor, diverging from a desired fractal dimension of 2.06, with $\Delta t = 0.001$ calculating the delay value at each iteration.
Figure 3.2.5:
Dimensional Embedding of a 20,000 point Rossler Attractor, converging to the desired fractal dimension of 2.027 (+0.034,-0.045), with $\Delta t = 0.08$ calculating the delay value at each iteration.

<table>
<thead>
<tr>
<th>Dim.</th>
<th>Slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.982</td>
</tr>
<tr>
<td>4</td>
<td>2.009</td>
</tr>
<tr>
<td>5</td>
<td>1.988</td>
</tr>
<tr>
<td>6</td>
<td>2.054</td>
</tr>
<tr>
<td>7</td>
<td>2.035</td>
</tr>
</tbody>
</table>

Figure 3.2.6:
Dimensional Embedding of a 20,000 point Lorenz Attractor, converging to the desired fractal dimension of 2.067 (+0.027,-0.040), with $\Delta t = 0.02$ calculating the delay value at each iteration.

<table>
<thead>
<tr>
<th>Dim.</th>
<th>Slope</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2.053</td>
</tr>
<tr>
<td>4</td>
<td>2.027</td>
</tr>
<tr>
<td>5</td>
<td>2.045</td>
</tr>
<tr>
<td>6</td>
<td>2.074</td>
</tr>
<tr>
<td>7</td>
<td>2.076</td>
</tr>
<tr>
<td>8</td>
<td>2.074</td>
</tr>
</tbody>
</table>

relatively large, the inflection point of the multicorrelation decreases very quickly, causing the convergence and subsequent jumps in the distance correlations to become more apparent. In conjunction with this shift change, the dimensional values also oscillate about the true fractal dimension.
Figure 3.2.7:
Dimensional Embedding of a 20,000 point Lorenz-2 Attractor, converging to the desired fractal dimension of 2.510 (±0.033, -0.082), with Δt = 0.04, calculating the delay value at each iteration.
3.3 Conclusion and Recommendations for Dimension

The fractal dimension for the Henon Mapping, and the Rossler, Lorenz, and Lorenz-2 Attractors have been accurately determined to within 3.3% error, by implementing the Grassberger-Procaccia distance correlation function. The best results were achieved using the scaling region within the second quarter of the of the correlation, with the additional condition of the radius being less than one. Furthermore, it has been established that the fractal dimension is dependent on the nearby distances generated by the system dynamics, as opposed to just dynamic closeness due to small time increments. Consequently, calculations for the dimension are more favorably affected by lengthening the sampling period than by increasing the sampling rate. The application of the multicorrelation function to determine a new shift for each embedding tends to normalize the dimension value obtained, so that the calculated values vacillates about the actual value. Another property of the multicorrelation function is that the value of the first inflection decreases as the dimension is increased, until it becomes inapplicable and the next inflection point is selected. This property may be used as an limiting feature for when the embedding dimension has been exceeded.

Although this method has proven to be adequate for the low dimensional system investigated in this study, there is still doubt as to its applicability and reliability for higher dimensional systems\textsuperscript{[9]}. Another convergence criteria, suggested by Peitgen et. al., is to embed up to twice the dimension of the system plus one, $2\gamma+1$, which becomes extensive for high dimensional systems. Due to the extensive numerical computations required by this algorithm and the limited application of matrix manipulation, another language, such as C, is highly recommended for this procedure. Reducing the number of computations may also be achieved by limiting the scaling region to only one logarithmic interval above the smallest stable radius\textsuperscript{[20]}. Lastly, since it is only necessary to extract the degrees of freedom of the system, not the exact dimension, an alternative method which does not require as extensive computations should be developed\textsuperscript{[22,23]}. 
4.0 Prediction of Nonlinear Systems

"Prediction is difficult, especially of the future."

- Niels Bohr

Having determined the degree of freedom, \( \eta \), for the system and a corresponding initial shifting value, \( v \), an artificial set of equations may now be developed, such that:

\[
\dot{X}_v = f(X_v)
\]

where \( \dot{X}_v \) is the numerical derivative of the discrete form of Takens' delayed-vector space, \( X_v \), which may be expressed as:

\[
X_v = \begin{pmatrix}
  x_1 & x_{1+v} & \cdots & x_{1+\mu v} \\
  x_2 & x_{2+v} & \cdots & x_{2+\mu v} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_i & x_{i+\mu v} & \cdots & x_{i+\mu v} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_N & x_{N+v} & \cdots & x_{N+\mu v}
\end{pmatrix}
\]

for notational simplicity, \( \mu = \eta - 1 \), one less than the dimension\(^1\). The continuous form of

** Prediction of the third position vector of the Lorenz-2 Attractor.
this equation, $X_t$, is introduced in Section 1.0. The remaining problem for this section is how to develop a set of dynamic equations capable of predicting the future values of the system, based solely on the numerical derivative as a function of the delayed-vector space.

This study has chosen to use a least-squares approach to fit a set of coefficients to a nonlinear polynomial containing all combinations of the delayed-vector terms, up to some final power. For completeness, a review of the development of least-squares approximation for a single variable is covered in Section 4.1. This method is then expanded in Section 4.2 to account for multiple variables, which also demonstrates the effectiveness of the algorithm for fitting typical multivariable cases. Finally the application to the delayed-vector space is investigated in section 4.3, which evaluates the affect of shifting values on the predictability of the fitting process.
4.1 Review of Least-Squares Approximation

One of the most common procedures for predicting data is the method of least-squares. In order to understand its application to Takens' delay-vector space, it is first necessary to have a general understanding of this fitting process. The objective is to find the optimum polynomial coefficient matrix, \( a = \{ a_0, a_1, a_2, \ldots, a_p \}^T \), that predicts the dependent variable, \( y \), for each independent value, \( x \), such that:

\[
y_i = f(x_i) = a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_p x_i^p
\]

where \( p \) is the order of the polynomial\(^{[11]}\). The local error, \( e_i \), depicted below, is calculated by subtracting the polynomial results from the actual values.

The squares function, \( S \), is defined as the sum of these local errors, squared, which is dependent on the coefficients of the polynomial as follows\(^{[11]}\):

\[
S(A^T) = \sum_{i=1}^{N} e_i^2 = \sum_{i=1}^{N} (y_i - a_0 - a_1 x_i - a_2 x_i^2 - \ldots - a_p x_i^p)^2
\]

Minimization of the error is achieved by setting the partial derivatives of this function equal to zero, for each polynomial coefficient, resulting in the following equations\(^{[11]}\):

\[
\frac{\partial S}{\partial a_0} = \sum_{i=1}^{N} -2 \cdot (y_i - a_0 - a_1 x_i - a_2 x_i^2 - \ldots - a_p x_i^p) = 0,
\]

\[
\frac{\partial S}{\partial a_1} = \sum_{i=1}^{N} -2 x_i \cdot (y_i - a_0 - a_1 x_i - a_2 x_i^2 - \ldots - a_p x_i^p) = 0,
\]

\[
\vdots
\]

\[
\frac{\partial S}{\partial a_p} = \sum_{i=1}^{N} -2 x_i^p \cdot (y_i - a_0 - a_1 x_i - a_2 x_i^2 - \ldots - a_p x_i^p) = 0
\]
Expansion of the summations and rearrangement of terms result in the following system of equations:

\[
\begin{align*}
\sum_{i=1}^{N} a_0 + \sum_{i=1}^{N} a_1 x_i + \sum_{i=1}^{N} a_2 x_i^2 + \ldots + \sum_{i=1}^{N} a_p x_i^p &= \sum_{i=1}^{N} y_i, \\
\sum_{i=1}^{N} x_i + \sum_{i=1}^{N} x_i^2 + \sum_{i=1}^{N} x_i^3 + \ldots + \sum_{i=1}^{N} x_i^{p+1} &= \sum_{i=1}^{N} x_i \cdot y_i, \\
\vdots \\
\sum_{i=1}^{N} x_i^p + \sum_{i=1}^{N} x_i^{p+1} + \sum_{i=1}^{N} x_i^{p+2} + \ldots + \sum_{i=1}^{N} x_i^{2p} &= \sum_{i=1}^{N} x_i^p \cdot y_i.
\end{align*}
\]

From this, the following matrix expression is readily developed:\(^{111}\):

\[
\begin{bmatrix}
N & \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \ldots & \sum_{i=1}^{N} x_i^p \\
\sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \sum_{i=1}^{N} x_i^3 & \ldots & \sum_{i=1}^{N} x_i^{p+1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\sum_{i=1}^{N} x_i^p & \sum_{i=1}^{N} x_i^{p+1} & \sum_{i=1}^{N} x_i^{p+2} & \ldots & \sum_{i=1}^{N} x_i^{2p}
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
\vdots \\
a_p
\end{bmatrix}
= \begin{bmatrix}
\sum_{i=1}^{N} y_i \\
\sum_{i=1}^{N} x_i \cdot y_i \\
\vdots \\
\sum_{i=1}^{N} x_i^p \cdot y_i
\end{bmatrix}
\]

These matrices are easily generated by the following expression:\(^{24}\):

\[
x_p^T x_p a = x_p^T y
\]

where,

\[
x_p = \begin{bmatrix}
1 & x_1 & x_1^2 & \ldots & x_1^p \\
1 & x_2 & x_2^2 & \ldots & x_2^p \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_N & x_N^2 & \ldots & x_N^p
\end{bmatrix}, \quad a = \begin{bmatrix}
a_0 \\
a_1 \\
a_2 \\
\vdots \\
a_p
\end{bmatrix}, \quad \text{and } y = \begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{bmatrix}
\]

The coefficient array can now be solved by multiplying each side by the inverse of \(x_p^T x_p\). The key feature is that each column of the matrix \(x_p\) is generated by raising the independent variable to the desired power. This property may also be applied for any function of interest, such as logarithmic or exponential. A number of such simple manipulations can be found in the Matlab program \textit{fit.m}, which was a cursory study of least-squares approximation for nonlinear regressions. More importantly, it lead to the concept of a generalized fitting algorithm described in the next section.
4.2 Multiple Dependent Variable Least-Squares

The purpose of this section is to establish a method for predicting the results, $y$, from a number of independent variables, $X$, such that:

$$ y_i = f(X_i) = f(x_{i,1}, x_{i,2}, \ldots, x_{i,\eta}) $$

where $\eta$ represents the degrees of freedom for the system. In order to simplify the algorithm, it is first necessary to include a column of ones before the independent vector space, $X$, to generate the matrix $\Psi$, as follows:

$$ \Psi = \begin{bmatrix}
1 & x_{1,1} & x_{1,2} & \cdots & x_{1,\eta} \\
1 & x_{2,1} & x_{2,2} & \cdots & x_{2,\eta} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_{N,1} & x_{N,2} & \cdots & x_{N,\eta}
\end{bmatrix} $$

where $N$ is the number of points in the data set. The multivariable polynomial matrix, $X_p$, is then generated by multiplying all possible $p$-order combinations of these columns. Due to the additional ones column, this is easily achieved using the following expression:

$$ X_{p_{i,j}} = \Psi_{i,k_1} \cdot \Psi_{i,k_2} \cdot \ldots \cdot \Psi_{i,k_p}, \quad \text{for} \quad k_1 = 1, \eta + 1, $$

$$ k_2 = k_1, \eta + 1, $$

$$ \vdots $$

$$ k_p = k_{p-1}, \eta + 1. $$

where the value of $j$ is incremented with each change of the $k$ indices. The resulting matrix, $X_p$, contains all combinations of the columns up to $p^{th}$-order. This matrix is comparable to the single variable polynomial matrix, $x_p$, developed in the previous section. This is easily shown by considering a one dimensional system, $\eta=1$; $X_p$ reduces simply to $x_p$. A more complex example, for 3 degrees of freedom, $\{x_1, x_2, x_3\}$, and $3^{rd}$-order combinations, results in the following twenty columns:

$$ 1^3 \quad 1^2 x_1 \quad 1^2 x_2 \quad 1^2 x_3 \quad 1x_1^2 \quad 1x_1 x_2 \quad 1x_1 x_3 \quad 1x_2^2 \quad 1x_2 x_3 \quad 1x_3^2 \quad \ldots \ldots $$

$$ \ldots \quad x_1^3 \quad x_1^2 x_2 \quad x_1^2 x_3 \quad x_1 x_2^2 \quad x_1 x_2 x_3 \quad x_1 x_3^2 \quad x_2^3 \quad x_2^2 x_3 \quad x_2 x_3^2 \quad x_3^3 $$
The least-squares technique may then be applied, as shown in the preceding section:

\[ X_p^T X_p a = X_p^T y \]

The coefficient array may then be solved using ordinary methods and will correspond to the columns generated in the polynomial matrix, like the previous example. Additional nonlinear regressions may also be added in a similar fashion, if desired. In its present form, the Matlab program \textit{nlfit.m} calculates the coefficients for a fifth order expansion, given any dependent vector, \( y \), and its vector space argument, \( X \).

The application of this method to system dynamics is a straightforward substitution of variables:

\[ X = Q = \{q_1 \ q_2 \ \cdots \ q_\eta\}, \text{ and } Y = \dot{Q} = \{\dot{q}_1 \ \dot{q}_2 \ \cdots \ \dot{q}_\eta\} \]

After appropriate expansion to polynomial matrix, \( Q_p \), each state equation,

\[ \dot{q}_j = f_j(q_1, q_2, \ldots, q_\eta), \]

is solved for individually, resulting in \( \eta \) coefficient vectors,

\[ A = \{a_1 \ a_2 \ \cdots \ a_\eta\} \]

The final form of the equation can then be expressed as:

\[ Q_p^T Q_p A = Q_p^T \dot{Q} \]

Since the derivatives are frequently unknown, a central difference procedure can be used to estimate these values, see Section 1.4. For improved accuracy, the sixth order method is recommended, which is implemented by the Matlab program \textit{cent6.m}; the first and last three points will be discarded due to this procedure. To test the accuracy of this procedure, the system may be reiterated, given an initial condition, using the Matlab program \textit{rkpoly.m}, which also requires \textit{nlpoly.m}. The corresponding velocities are generated using \textit{nlpvel.m}.

During the development of this procedure, various aspects were tested, including the order of the polynomial expansion and the accuracy of the central difference applied. When the actual velocities are used in the fitting process, the resulting polynomial coefficients are exact, for all orders of polynomial equal to or above those employed in the system's equations. For numerically approximated derivatives, the accuracy of the central differencing procedure
Figure 4.2.1:
Prediction of the first position vector of the Rossler Attractor, using the actual position vectors and derivatives estimated by a sixth order central difference method, --- Original --- Predicted.

Figure 4.2.2:
Prediction of the first position vector of the Lorenz Attractor, using the actual position vectors and derivatives estimated by a sixth order central difference method, --- Original --- Predicted.

dictates the predictability. Figures 4.2.1 and 4.2.2 depict the results of a typical prediction of the first position vectors for the Rossler and Lorenz Attractors, respectively. For these two cases, third order coefficient combinations have been implemented, in conjunction with a sixth order central difference method to approximate the derivatives. The resulting coefficients
come within a ten-thousandth of the actual values. However, this small difference causes the eventual divergence from the original values, due to the cumulative error propagation in the integration process.

The application to Takens' theory is similarly a straightforward substitution of variables. However, in this case only one set of values is required, \( x(t) \). From this data, the delayed-vector space, \( X_{\tau} \), is generated based on the predetermined values for the shift, \( \tau \), and dimension, \( \eta \). This array is then used in place of \( X \), while the derivatives are solved numerically and substituted for \( Y \), as follows:

\[
X = X_{\tau} = \begin{bmatrix} x(t) \\ x(t + \tau) \\ \vdots \\ x(t + \mu \tau) \end{bmatrix}^T, \quad \text{and} \quad Y = X_{\tau} = \begin{bmatrix} \dot{x}(t) \\ \dot{x}(t + \tau) \\ \vdots \\ \dot{x}(t + \mu \tau) \end{bmatrix}^T
\]

where \( \mu = \eta - 1 \), one less than the dimension. Again, after expansion to the desired order polynomial, \( X_{\tau_p} \), the least squares technique can be applied:

\[
X_{\tau_p}^T X_{\tau_p} A = X_{\tau_p}^T X_{\tau_p}
\]

The resulting coefficient matrix, \( A \), contains \( \eta \) coefficient vectors corresponding to the artificial state equations. This entire process is implemented by the Matlab program \( \text{recon.m} \), which also integrates the coefficients, using \( \text{rkpoly.m} \) and \( \text{nlpoly.m} \), as well as, calculates the corresponding derivatives, using \( \text{nlpvel.m} \). The integration process may be bypassed using \( \text{initcond.m} \), which effectively predicts the initial velocities for each existing delayed-vector coordinate. An example of this procedure is shown as the introductory figure of Section 1, which was the first attempt to predict the Rossler Attractor. The application of these methods to the Rossler, Lorenz, and Lorenz-2 systems is covered in the next section, Prediction Results.
4.3 Prediction Results

The polynomial Least Squares method developed in the previous section for Takens' Delay-vector Space will now be applied to the prediction of a single position vector for each of the three attractors evaluated in this study. The predicted systems are integrated from a single set of initial conditions using a fourth order Runge-Kutta method which implements the polynomial coefficient matrix. The predictability of the artificial system will be evaluated qualitatively based on graphical results. The primary assessment will be based on a plot of the error between the predicted velocities and the actual velocities, with additional comparisons between the predicted values and the actual system. Based on these methods, the possibility of an optimum shifting value will be shown to exist, such that increasing or decreasing the delay value detrimentally effects the predicted values, by increasing the local deviation and being less stable. Table 2.2.1 lists the local optimum shifting values resulting from the prediction of a variety of sections for each attractor. The systems will be investigated in order of increasing nonlinearity, Rossler, Lorenz and Lorenz-2, which will also be demonstrated to be the predominant criteria for predictability.

The first attempt to predict the Rossler Attractor is shown as the lead-in picture for Section 1, on page one. However, this is not an actual prediction, only the evaluation of the velocities resulting from the coefficient matrix applied to the original points, equivalent to calculating the initial condition for each point. Based on only third order polynomial combinations, this original prediction was not very exact, which is apparent in the oscillation of the predicted initial conditions about the real values. Consequently, the polynomial expansion has been increased to include fifth order terms, which proves adequate. Figure 4.3.1 depicts the prediction of a well distributed section of the Rossler Attractor, with an optimum shift of \( v_{opt} = 92 \). The corresponding error plot between the actual and predicted velocities is shown in Figure 4.3.2. The error due to optimum shift value is clearly bracketed by the errors from the next higher and lower shift values. Being the least nonlinear of the systems investigated, with a
Figure 4.3.1: Phase portrait of the first position and velocity vectors of the actual and predicted Rossler Attractor, $\Delta t = 0.005$.

Figure 4.3.2: Error between the actual and predicted velocity for the first velocity vectors of the Rossler Attractor, $\Delta t = 0.005$. 

62
Figure 4.3.3: Phase portrait of the first position and velocity vectors of the actual and predicted Rossler Attractor, $\Delta t = 0.005$, depicting unstable coefficient matrix.

Figure 4.3.4: Error between the actual and predicted velocity for the first velocity vectors of the Rossler Attractor, $\Delta t = 0.005$. 
fractal dimension of approximately 2.02, the Rossler Attractor is the most easily and best predicted. This is demonstrated by the fact that the shifting value determined using the multicorrelation, $v_{inf} = 84$, almost ten percent less than the optimum, remains stable, but is clearly not the best prediction.

The section of the attractor being fit does not always result in a stable coefficient matrix, as shown in Figures 4.2.3 and 4.3.4. Still, the error from the optimum shift is bracketed by the error due to next higher and lower shift values. In this case, the optimum shift is the only prediction that remains stable, although it does not accurately reflect the original data. This is likely due to the small range of the attractor that has been used in the fitting process. Consequently the segment chosen plays an important role in the eventual predictability of the system. Generally, a better and more stable prediction will result when more information is contained in the fitted section. Based on numerous predictions of various sections of the attractor, the optimum shift has been determined to typically occur at a delay of approximately, $\tau_{opt} \approx 0.46$ seconds. Due to the relatively regular behavior of the Rossler system, there are only insignificant local effects to cause deviation from this value.

Only slightly more nonlinear, with a fractal dimension of 2.06, the prediction of the Lorenz Attractor is significantly more effected by the shift value. In the search for a stable coefficient matrix for this system, the possibility of an optimum shifting value was first realized. Figure 4.3.5 shows the error plots for a range of shift values, in which the error values seem to inflect periodically about some minimum error. The corresponding shift value should generate the best prediction. After approximately 300 points, there is a clear separation between the errors with shift values above 32 and those below 29. A shift of 31 resulted in the best prediction. Figures 4.3.6 and 4.3.7 depict the prediction for each of these three shifting values. A simplified error plot for these values is shown in Figure 4.3.8, which clearly depicts the bracketing of the optimum value, as well as a significant reduction in error.
Figure 4.3.5: Error between the actual and predicted velocity for the first velocity vectors of the Lorenz Attractor, $\Delta t = 0.005$.

Figure 4.3.6: Phase portrait of the first position and velocity vectors of the actual and predicted Lorenz Attractor, $\Delta t = 0.005$. Depicts spiraling too tightly, $29 < \tau_{opt}$.
Figure 4.3.7: Phase portrait of the first position and velocity vectors of the actual and predicted Lorenz Attractor, $\Delta t = 0.005$. Depicts spiraling too loosely, $32 > \tau_{opt}$.

Figure 4.3.8: Error between the actual and predicted velocity for the first velocity vectors of the Lorenz Attractor, $\Delta t = 0.005$. 
Figure 4.3.9: Phase portrait of the first position and velocity vectors of the actual and predicted Lorenz Attractor, $\Delta t = 0.005$, depicting unstable coefficient matrix.

Figure 4.3.10: Error between the actual and predicted velocity for the first velocity vectors of the Lorenz Attractor, $\Delta t = 0.005$. 
Another visible property of the optimum shift value is its tendency to expand or contract in accordance to the actual system. This is evident in the equivalent trajectories of 500 points shown in Figure 4.3.5. For values less than the optimum, the prediction spirals too tightly. Conversely, shift values greater than the optimum spiral out too loosely. This can also be seen in the full picture, Figure 4.3.6. Stated more generally, values greater than the optimum move too quickly to the next values, while values less than the optimum adhere to previous points. A more dramatic case is shown in figures 4.3.9 and 4.3.10 where only the optimum shift remains stable, while the bracketing values quickly go off to infinity. This is not uncommon for the prediction the Lorenz, for which even the optimum shift does not always remain stable. This is most likely due to its slightly greater nonlinearity than Rossler's, which may also account for the variation of its optimum shift.

The effect of nonlinearity on predictability is further emphasized by the unobtainability of a completely stable coefficient matrix for the Lorenz-2 Attractor, which has a fractal dimension of 2.51. Consequently, the optimum shift value is based solely on the longest, relatively stable, prediction. Two examples are shown in Figures 4.3.11 and 4.3.13, with the corresponding plots of the error between the actual and predicted velocities shown in Figures 4.3.12 and 4.3.14, respectively. As with the previous attractors, the error due to the optimum shift is bracketed by the errors of their next higher and lower shift values. For the first case these values offer virtual identical results, although the next higher and lower values broke off to infinity much sooner. The second case stays stable significantly longer than its bracketing values, much like the previous examples for Rossler and Lorenz. Predictions using larger sampling rates were not quite as successful. Various attempts were made to improve the results, including increasing the order of the polynomial terms, increasing the length of the section being evaluated, and increasing the dimension space. Regretfully these all require extensive computational time and no conclusive results have been obtained at this time.
Figure 4.3.11: Phase portrait of the third position and velocity vectors of the actual and predicted Lorenz-2 Attractor, \( \Delta t = 0.002 \), depicting unstable coefficient matrix.

Figure 4.3.12: Error between the actual and predicted velocity for the third velocity vectors of the Lorenz-2 Attractor, \( \Delta t = 0.002 \).
Figure 4.3.13: Phase portrait of the third position and velocity vectors of the actual and predicted Lorenz-2 Attractor, $\Delta t = 0.002$, depicting unstable coefficient matrix.

Figure 4.3.14: Error between the actual and predicted velocity for the third velocity vectors of the Lorenz-2 Attractor, $\Delta t = 0.002$. 
4.4 Conclusions and Recommendations for Prediction of Nonlinear Systems

The predictions of the first position vectors for Rossler and Lorenz systems have been achieved with moderate success, using a Least-Squares approximation, implementing fifth order polynomial combinations. Furthermore, it has been demonstrated that an optimum shifting value is likely to exist, such that any deviation from this value reduces the effectiveness of the prediction. This is visible in the bracketing of the prediction error of optimum shift value by the errors of next higher and lower shifting values. Additionally, shift values greater than the optimum tend to drift toward future values, while values less than the optimum adhere to previous points. Based on the best predictions for numerous sections, optimum shift values of 0.46 and 0.16 seconds have been estimated for the Rossler and Lorenz Attractors, respectively. Deviation from the optimum shift for individual sections is most likely caused by local effects. Additionally, sections that cover a wide range of the attractor tend to increase the stability of the prediction. Although, a stable prediction of the Lorenz-2 system was not adequately achieved, the possibility of optimum shifting value still seems probable, resulting in a value of 0.17 seconds. Furthermore the predictability of a system clearly decreased with increasing nonlinearity.

Further research into the application of higher order polynomial terms will hopefully lead to a successful prediction of the Lorenz-2 Attractor. Other aspects of the fitting process that need additional investigation are the influence of segment length and role of the sampling rate. An improved forth-fifth order Runge-Kutta integration is also likely to increase the predictability of the systems. Most importantly, a quantitative method for evaluating the best prediction needs to be developed. This could then be used to evaluate the predictability of other initial conditions, the influence of noise, and experimental systems. Due to the rapidly expanding nature of the polynomial coefficient matrix, this method becomes computationally intensive for larger systems and stronger nonlinearity. Consequently, investigations into alternative methods are strongly recommended[1].
Appendix I - Matlab Programs

Pend.m ..................................................73
Pendlim.m .............................................73
Henon.m ...............................................74
Rossler.m .............................................75
Rosvel.m ..............................................75
Lorenz.m ...............................................76
Lorvel.m ...............................................76
Lorenz2.m ..............................................77
Lorvel2.m ..............................................77
Cent2.m ...............................................78
Cent4.m ...............................................78
Cent6.m ...............................................79
Rk4.m ..................................................80
Rkpoly.m ..............................................81
Autocor.m .............................................82
Multicor.m ...........................................83
Mdimcor.m ............................................84
Dimembed.m ..........................................86
Nlfit.m ................................................89
Nlpol.m ...............................................90
Nlpvel.m ..............................................91
Recon.m ...............................................92
Continue.m ..........................................93
Initcond.m ..........................................94
Delayed.m ............................................95
Fit.m ..................................................96
Expand.m .............................................102
Printhp.m ............................................103
Printps.m ............................................103
PEND.M

This is a MATLAB program to be called by RK4.M, ODE23, or ODE45 to evaluate the position vectors of a driven pendulum. Must be surrounded by single quotes, 'pend'. Use PENDLIM.M to limit the position vector between negative and positive PI. Use q(:,1) for the angular Position vector and q(:,2) for the angular Velocity vector.

[qdot]=pend(q,t)

Created by: Edward H. Ziegler, 11/93

function [qdot]=pend(q,t)

PARAMETERS
zeta=damping factor
A=magnitude of nondimensional force
wf=driving frequency

zeta=0.25;A=1.07, wf=2/3;

FUNCTIONS

qdot=[q(2),(-sin(q(1))-2*zeta*q(2)+A*cos(q(3))),wf];

PENDLIM.M

This is a MATLAB program to limit the angular position values, generated using PEND.M and RK4.M, to negative and positive PI. Use q(:,2) for the angular Position vector and q(:,1) for the angular Velocity vector.

[q]=pendlim(q)

Created by: Edward H. Ziegler, 11/93
function [q]=pendlim(q)
[m,n]=size(q);

for i=1:m
    if q(i,1) > pi
        q(i,1)=q(i,1)-2*pi;
    elseif q(i,1) < -pi
        q(i,1)=q(i,1)+2*pi;
    end
end

HENON.M

% This is a MATLAB program to determine the values of the HENON
% attractor. Two vectors of length "m" are generated given the
% initial conditions "x1" and "y1".
% [x,y]=henon(x1,y1,m)
% Created by: Edward H. Ziegler, 11/93

function [x,y]=henon(x1,y1,m);

% INITIALIZATION AND SIZING OF VECTORS
x=zeros(m,1);
y=zeros(m,1);
x(1)=x1;y(1)=y1;

% ITERATIVE MAPPING PROCEDURE
for k=1:m-1
    x(k+1)=y(k)+1-1.4*x(k).^2;
    y(k+1)=0.3*x(k);
end;
ROSSLER.M

% This is a MATLAB program to be called by RK4.M, ODE23, or ODE45
% to evaluate the position vectors of a ROSSLER attractor. Must
% be surrounded by single quotes, 'rossler'.

% [qprime]=rossler(q,t)

% Created by: Edward H. Ziegler, 11/93

function [qprime]=rossler(q,t)
% PARAMETERS
a=0.2;b=0.2;c=4.6;
% FUNCTIONS
qprime=[-q(3)-q(2),q(1)+a*q(2),b+q(3).*(q(1)-c)];

-------------------

ROSVEL.M

% This is a MATLAB program to evaluate the velocities of a
% ROSSLER attractor after the position vector is generated
% using the RK4.M and ROSSLER.M programs.

% [qdot]=rosvel(q)

% Created by: Edward H. Ziegler, 11/93

function [qdot]=rosvel(q)
% PARAMETERS
a=0.2;b=0.2;c=4.6;
% FUNCTIONS
qdot(:,1)=-q(:,3)-q(:,2);
qdot(:,2)= q(:,1)+a*q(:,2);
qdot(:,3)= b+q(:,3).*(q(:,1)-c);
LORENZ.M

% This is a MATLAB program to be called by RK4.M, ODE23, or Ode45
% to evaluate the position vectors of a LORENZ attractor. Must
% be surrounded by single quotes, 'lorenz'.
%
% [qdot]=lorenz(q,t)
%
% Created by: Edward H. Ziegler, 11/93

function [qdot]=lorenz(q,t)
% PARAMETERS
a=10;b=8/3;c=28;
% FUNCTION
qdot=[-a*q(1)+a*q(2),c*q(1)-q(2)-q(1)*q(3),-b*q(3)+q(1)*q(2)];

LORVEL.M

% This is a MATLAB program to evaluate the velocities of a
% LORENZ attractor after the position vector is generated
% using the RK4.M and LORENZ.M programs.
%
% [qdot]=lorvel(q)
%
% Created by: Edward H. Ziegler, 11/93

function [qdot]=lorvel(q)
% PARAMETERS
a=10;b=8/3;c=28;
% FUNCTIONS
qdot(:,1)=-a*q(:,1)+a*q(:,2);
qdot(:,2)=c*q(:,1)-q(:,2)-q(:,1).*q(:,3);
qdot(:,3)=-b*q(:,3)+q(:,1).*q(:,2);
LORENZ2.M

% This is a MATLAB program to be called by RK4.M, ODE23, or Ode45
% to evaluate the position vectors of a LORRENZ2 attractor. Must
% be surrounded by single quotes, 'lorenz2'.
%
% [qdot]=lorenz2(q,t)
%
% Created by: Edward H. Ziegler, 11/93

function [qdot]=lorenz2(q,t)
% PARAMETERS
a=0.25;b=4;F=8.0;G=1;
% FUNCTION
qdot=[-q(2)^2-q(3)^2-a*(q(1)-F),q(1)*q(2)-b*q(1)*q(3)-q(2)+G,...
    b*q(1)*q(2)+q(1)*q(3)-q(3)];

LORVEL2.M

% This is a MATLAB program to evaluate the velocities of a
% LORRENZ2 attractor after the position vector is generated
%
% [qdot]=lorvel2(q)
%
% Created by: Edward H. Ziegler, 11/93

function [qdot]=lorvel2(q)
% PARAMETERS
a=0.25;b=4;F=8.0;G=1;
% FUNCTION
qdot(:,1)=-q(:,2).^2-q(:,3).^2-a*(q(:,1)-F*ones(q(:,1))); 
qdot(:,2)= q(:,1).*q(:,2)-b*q(:,1).*q(:,3)-q(:,2)+G*ones(q(:,2));
qdot(:,3)= b*q(:,1).*q(:,2)+q(:,1).*q(:,3)-q(:,3);
% This is a Matlab program to calculate the first derivative, 
% "xdot", using a second order central difference method, where 
% "x" is the input vector with a time step of "dt". 
% Derivatives at the endpoints are not estimated. "xshort" is 
% the appropiatly shortened version of the original vector. 
% 
% [xshort,xdot]=cent2(x,dt) 

function [xshort,xdot]=cent2(x,dt); 

[NP,MP]=size(x); 
%disp(['Calculating the derivative to second order accuracy.']) 
xdot=zeros(NP-2,MP); 
xdot(l:NP-2,:)=(x(3:NP,:)-x(l:NP-2,:))/(2*dt); 
xshort=x(2:NP-1,:); 

CENT4.M

% This is a Matlab program to calculate the first derivative, 
% "xdot", using a fourth order central difference method, where 
% "x" is the input vector with a time step of "dt" 
% Derivatives at the endpoints are not estimated. "xshort" is 
% the appropiatly shortened version of the original vector. 
% 
% [xshort,xdot]=cent4(x,dt) 

function [xshort,xdot]=cent4(x,dt); 

[NP,MP]=size(x);
CENT4.M (cont.)

disp(['Calculating the derivative to fourth order accuracy.'])
xdot=zeros(NP-4,MP);
xdot(1:NP-4,:)=(-x(5:NP,:)+8*x(4:NP-1,:)-8*x(2:NP-3,:)+x(1:NP-4,:)) ./ (12*dt);
xshort=x(3:NP-2,:);

---

CENT6.M

% This is a Matlab program to calculate the first derivative, 
% "xdot", using a sixth order central difference method, where 
% "x" is the input vector with a time step of "dt". 
% Derivatives at the endpoints are not estimated. "xshort" is 
% the appropriately shortened version of the original vector. 
% 
% [xshort,xdot]=cent6(x,dt)

function [xshort,xdot]=cent6(x,dt);

[NP,MP]=size(x);
disp('Calculating the derivative to sixth order accuracy.'),
xdot=zeros(NP-6,MP);
xdot(1:NP-6,:)=x(7:NP,:)-9*x(6:NP-1,:)+45*x(5:NP-2,:)-45*x(3:NP-4,:)+...
                    9*x(2:NP-5,:)-x(1:NP-6,:))/(60*dt);
xshort=x(4:NP-3,:);
% This is a MATLAB program that uses a forth order Runge-Kutta technique to integrate a given function "f" at constant time increments "dt", from initial time "ti" to final time "tf", and initial condition "q0". The function must be surrounded by single quotes and may be called from a separate "M file".

% 
% [q,t]=rk4(f,ti,dt,tf,q0);
% 
% Created by: Edward H. Ziegler, 11/93

function [q,t]=rk4(f,ti,dt,tf,q0);

    t=0:dt:tf;
    nf=(tf-ti)/dt;
    wd=length(q0);
    q=NaN*ones(nf,wd);
    q(1,:)=q0;
    for n=1:1:nf
        k1 = dt*feval(f, q(n,:), t(n));
        k2 = dt*feval(f, q(n,:)+0.5*k1 , t(n) + 0.5*dt);
        k3 = dt*feval(f, q(n,:)+0.5*k2 , t(n) + 0.5*dt);
        k4 = dt*feval(f, q(n,:)+k3 , t(n) + dt);
        q(n+1,:)=q(n,:)+(1/6)*(k1 + 2*k2 + 2*k3 + k4);
    end;
This is a MATLAB program that uses a fourth order Runge-Kutta technique to integrate multivariable polynomial coefficient matrix "a", as generated by NLFIT.M, at constant time increments "dt", given an initial condition "q0", and the desired number of iterations "m"

Equivalent expansion algorithms must be used in NLFIT.M, NLPOLY.M, and NLPVEL.M.

% [q,t]=rkpoly(a,dt,q0,m);

% Created by: Edward H. Ziegler, 11/93

function [q,t]=rkpoly(a,dt,q0,m);
disp('Integrating using a fourth order Runge-Kutta method.')
f='nlpoly';
n=length(q0);
t=(0:dt:m*dt)';
q=NaN*ones(m,n);
q(1,:)=q0;
diff=0;
for n=1:1:m-1
    k1 = dt*feval(f, q(n,:), a);
    k2 = dt*feval(f, q(n,:)+0.5*k1 , a);
    k3 = dt*feval(f, q(n,:)+0.5*k2 , a);
    k4 = dt*feval(f, q(n,:) + k3 , a);

    q(n+1,:)=q(n,:)+(l/6)*(k1 + 2*k2 + 2*k3 + k4);
end;
% This is MATLAB program to calculate the first tenth of the
% auto-correlation values, "ac", for a vector "x" and evaluate
% the first local minimum, "Nu".
%
% [ac,Nu]=autocor(x)

% Created by: Edward H. Ziegler, 11/93

function [ac,Nu]=autocor(x)
disp('Calculating the Delay Step')

% Initialize constants
n=length(x);
n2=fix(n/10);

% Calculate auto-correlation for first tenth of values
ac=ones(1,n2);
for k=1:n2;
    ac(k)=sum(x(1:n-k+1).*x(k:n))/(n-k+1);
end;
plot(ac),grid,title('Auto-Correlation of First 10% of values');
pause(7);

% Find first local minimum
len=length(ac);
ac1=ac(1:len-1);
ac2=ac(2:len);
i=find(ac1-ac2<0);
Nu=ceil(i(1)-1);
disp(['The first local minimum occurs at ', num2str(Nu)])
This is MATLAB program to calculate the first tenth of the multi-correlation values "mc", and evaluate the first inflection point "Nu", given a vector "x", time step "dt", and a single dimension "ndim". Called by DIMEMBED.M.

```matlab
function [mc,Nu]=multicor(x,dt,ndim)

disp(['Calculating the delay step for ', num2str(ndim), ' dimensions.'])

% Initialize constants
len=length(x);
n2=fix(len/10);
mc=zeros(n2,1);

% Calculate multi-correlation values
for j=1:n2;
    n=len-(ndim-1)*j;
    X=x(1:n+1);
    for k=1:ndim-1
        X=X.*x(k*j:n+k*j);
    end
    mc(j)=sum(X)/(n+1);
end;
mc=mc./mc(1);

% Find the first inflection point.
[mcshort, mcdot]=cent2(mc,dt);
L2=length(mcdot);
i=zeros(L2,1);
if mcdot(1) >= mcdot(2)
    i=find(mcdot(1:L2-1) <= mcdot(2:L2));
else
```

83
MULTICOR.M (cont.)

i=find(mcdot(1:L2-1) >= mcdot(2:L2));
end
Nu=i(1) + 1;
disp(['The first inflection occurs at ', num2str(Nu),...
' for ', num2str(ndim), ' dimensions. '])

MDIMCOR.M

% This is MATLAB program to calculate the first tenth of the
% multi-correlation values "Mc", and evaluate the first inflection
% point "Nu", given a vector "x", time step "dt", and for
% dimensions two through "enddim".
% [Mc,Nu]=mdimcor(x,dt,enddim)
%
% Created by: Edward H. Ziegler, 11/93

function [Mc,Nu]=mdimcor(x,dt,enddim)

% Initialize constants
len=length(x);
n2=fix(len /10);
Mc=zeros(n2,enddim-1);
Nu=zeros(enddim-1,1);

% Increment dimension
for ndim=2:enddim
disp(['Calculating the optimum delay step for ',num2str(ndim),'
dimensions.'])

% Calculate Multi-correlation values
mc=zeros(n2,1);
for j=1:n2;
    n=len-(ndim-1)*j;

X=x(1:n+1);
for k=1:ndim-1
   X=X.*x(k:j:n+k*j);
end
mc(j)=sum(X)/(n+1);
end;
Mc(:,ndim-1)=mc./mc(1);
% Normalize multi-correlation values.

% Find the first inflection point.

[mcshort,mcdot]=cent2(mc,dt);
L2=length(mcdot);
i=zeros(L2,1);
if mcdot(1)>=mcdot(2)
i=find(mcdot(1:L2-1)<=mcdot(2:L2));
else
   i=find(mcdot(1:L2-1)>=mcdot(2:L2));
end
Nu(ndim-1)=i(1) + 1;
disp(['The first inflection occurs at ',num2str(Nu(ndim-1)),'
   for ',num2str(ndim),' dimensions. '])
end
This is a MATLAB program to evaluate the dimensional embedding of a time series data set "x" measured at time step "dt". The Multicorrelation function is used to determine the delay shift value "Nu". The distance correlation "N" is calculated using the Grassberger-Proccacia technique at given distances "D" (logarithmic values are actually calculated). These values are plotted against their linear, least-squared fitted values "Ns", for which the slopes are stored in "a".

```
function [D,N,Ns,a,Nu]=dimembed(x,dt)

% Reshape "x" into a column vector and initialize variables.
clock1=clock;
x=x(:);
pmax=13; maxdim=13; % "ndim" is the current dimension.
ndim=3; skip=2; s1=1; s2=0; c=0; % "skip" increases the dimension.
Rmin=log(1e-13); Rmax=log(1e+13); % "Rmin" and "Rmax" are the range
clg; % of distances being considered.

while s1 > (ndim-maxdim)
    if c~=1
        % Find shifting value "Nu" for each dimension and plot Multicorrelation.
        [Mc,nu]=multicor(x,ndim,dt);
        Nu=[Nu ; nu];
        subplot(211),plot(Mc);
    end
end

% Create delayed coordinate vector space "xd".
len=length(x)-(ndim-1)*Nu;
N2=len*(len-1)/2;
xd=ones(len,ndim);
for p=1:ndim;
    xd(1:len,p)=x((p-1)*Nu+1:(p-1)*Nu+len);
end
end
```
DEMEMBED.M (cont.)

% Generate radius increment and allocate Vector sizes.
Rl=0.75*Rmin+0.25*Rmax;       % Only the center section of
Ru=0.5*Rmin+0.5*Rmax;        % Rmin and Rmax is being used.
if Ru > 0
    Ru =0;
end

dr=(Ru-Rl)/(pmax-1);
Rl=(Rl:dr:Ru)';
N1=zeros(pmax,1);n=zeros(pmax,1);

% Reset rmin and rmax, check counter.
Rmax=log(1e-13);Rmin=log(1e+13);
if c<=0
    disp(['Calculating initial maximum and minimum radii for ...
         ,num2str(ndim),' dimensions.'])
    ndim=ndim-skip;
else
    disp(['Calculating Slope for a ','num2str(ndim),'' dimensional embedding.'])
end

% Calculate distances between k and k+j rows.
for k=1:len-1;
    k1=len-k;
    p=(k)*ones(1,k1);
    xn=xd(p,:)-xd(k+1:len,:);
    r= 0.5*log(sum((xn.^2)));

% Search for minimum and maximum radii to be used in next iteration.
rmax=max(r);
if rmax > Rmax
    Rmax=rmax;
end;

rmin=min(r);
if rmin < Rmin

87
Rmin=rmin;
    end;

if c > 0
    Count number of points in each distance increment.
    for p=1:pmax
        n(p)=sum(r <= R1(p));
        N1(p)=N1(p) + n(p);
    end;
end;
end;
end;
end;
if c > 0
    D(:,c)=R1;
    N(:,c)=log(N1 ./N2);
    Nu(c,1:2)=[ndim,Nu];
    clg;
%
    Calculate slope, construct lines, and Plot with distance correlation.
    a(c,:)=(D(:,c),ones(D(:,c)))\N(:,c))';
    Ns(:,c)=[D(:,c),ones(D(:,c))]*a(c,:);'
    subplot(212),plot(D(:,c),N(:,c),'or',D(:,c),Ns(:,c),'-b'),...
    pause(1);

    s2=s1;s1=a(c,1);
    disp(['The Slope for a ',num2str(ndim),', ')
    ' dimensional embedding is ',num2str(a(c,1))])
    disp([num2str(etime(clock,clock1)/60),' minutes have elapsed'])
    disp(['
    end
    c=c+1;
    ndim=ndim+skip;
    end
%
    Plot lines depicting all distance correlations and slopes.
    clg;
    plot(D,N,'or',D,Ns,'-b'),grid;
This is a MATLAB program to calculate the multivariable polynomial coefficient matrix "a", based on least-squared regression technique for multivariables, with fifth order combinations of variables. The algorithm generates a best fit for the state equation based on the vector space "x" and its derivative "xd", which may be calculated using CENT6.M, CENT4.M, or CENT2.M. The equation may be integrated using RKPOLY.M in conjunction with NLPOLY.M, with corresponding velocities calculated with NLPVEL.M. INITCOND.M may also be used to evaluate the how well the coefficient fits the original data. Equivalent expansion algorithms must be used in NLFIT.M, NLPOLY.M, and NLPVEL.M.

function [a]=nlfit(x,xd);

disp('Calculating least square fit with fifth order combinations.')
[m,n]=size(x); [p,q]=size(xd);
c=0; x1=ones(m,1);
x=[x1 x];
for k1=1:n+1
    for k2=k1:n+1
        for k3=k2:n+1
            for k4=k3:n+1
                for k5=k4:n+1
                    c=c+1;
                    X(:,c)=x(:,k1).*x(:,k2).*x(:,k3).*x(:,k4).*x(:,k5);
                end
            end
        end
    end
end
for k=1:n
    a(k,:)=(X\xd(:,k)).';
end
This is a MATLAB program expands the vector space "q" so it may be multiplied times the multivariable polynomial coefficient matrix "a", as generated by NLFIT.M. It is the equivalent of the multivariable polynomial state equation with parameters "a". Use NLPVEL.M to calculate the derivatives of the resulting vector space.

Equivalent expansion algorithms must be used in NLFIT.M, NLPOLY.M, and NLPVEL.M.

function [qd]=nlpoly(q,a)

n=length(q);
c=0;
q1=[1 q];
for k1=1:n+1
    for k2=k1:n+1
        for k3=k2:n+1
            for k4=k3:n+1
                for k5=k4:n+1
                    c=c+1;
                    Q(1,c)=q1(:,k1).*q1(:,k2).*q1(:,k3).*q1(:,k4).*q1(:,k5);
                end
            end
        end
    end
end
qd=a*Q';
qd=qd';
This is a MATLAB program to calculate the derivative vectors "qd" generated by the multivariable polynomial expansion technique, given the vector space "q", generated by NLPOLY.M in conjunction with RKPOLY.M, and coefficient matrix "a", as generated by NLFIT.M. Equivalent expansion algorithms must be used in NLFIT.M, NLPOLY.M, and NLPVEL.M.

[qd]=nlpvel(q,a);

Created by: Edward H. Ziegler, 11/93

function [qd]=nlpvel(q,a)

[m,n]=size(q);
c=0;
uno=ones(m,1);
q1=[uno q];

for k1=1:n+1
    for k2=k1:n+1
        for k3=k2:n+1
            for k4=k3:n+1
                for k5=k4:n+1
                    c=c+1;
                    Q(:,c)=q1(:,k1).*q1(:,k2).*q1(:,k3).*q1(:,k4).*q1(:,k5);
                end
            end
        end
    end
end

qd=a*Q';
qd=qd';
This is a MATLAB program to reconstruct a time series data set using the "delayed vector" space theory and multivariable least squared reduction method. First the coefficient matrix "coef" is determined, given a column vector "X", the time step "dt", a delay value "delay", and a dimension "dim". This coefficient matrix is then used to integrate the first delayed values to obtain a new set of vectors "q", for the desired number of iterations "points". The derivative "qd" is then calculated from these values.

This program DOES INTEGRATE the values of the delayed coordinates. Requires additional programs CENT6.M, NLFIT.M, and NLPVEL.M.

```
function [coef,q,qd]=recon(X,dt,delay,dim,points)

len=length(X);

% Generate delayed vector space
for k=1:dim
    Xd(:,k)=X(l+(k-l)*delay:len+(k-dim)*delay);
end

% Generate multivariable coefficient matrix.
[xs,xd]=cent6(Xd,dt);
[coef]=nlfit(xs,xd);

% Integrate values of the new system.
[q,t]=rkpoly(coef,dt,Xd(4,:),points);

% Calculate derivatives of the new system.
qd=nlpvel(q,coef);
```
CONTINUE.M

% This is a MATLAB program to continue the iteration of a non-linear
% system based on the multivariable polynomial expansion, RKPOLY.M,
% given the coefficient matrix "coef", the existing iterations of the
% multivariable array "x", the equivalent derivate array "xd", the
% time step "dt", and the number of additional iterations "points".
% The new multivariable array "q" and its deriative "qd" contain
% the original values.
% Requires additional programs RKPOLY.M, NLPOLY.M, and NLPVEL.M.
%
% [q,qd]=continue(coef,x,xd,dt,points)

% Created by: Edward H. Ziegler, 11/93

function [q,qd]=continue(coef,x,xd,dt,points)

[m,n]=size(x);
% Generate nonlinear equations to describe motion

[q,t]=rkpoly(coef,dt,x(m,:),points+1);
qd=nlpvel(q,coef);
q=[x;q(2:points+1,:)];
qd=[xd;qd(2:points+1,:)];
% This is a MATLAB program to evaluate the initial conditions generated
% by the "delayed vector" space theory and multivariable least squared
% reduction method. This determines the coefficient matrix "coef",
% given a column vector "x", the time step "dt", a delay value "delay",
% and a dimension "dim". Based on this coefficient matrix, the values of
% the derivative "Xd" is calculated for each delayed coordinate "X".
% This DOES NOT INTEGRATE the values of the delayed coordinates.
% Requires additional programs CENT6.M, NLFIT.M, and NLPVEL.M.
%
% [coef,X,Xd]=initcond(x,dt,delay,dim)
%
% Created by: Edward H. Ziegler, 11/93

function [coef,X,Xd]=initcond(x,dt,delay,dim)

len=length(x);

% Generate delayed vector space.

for k=1:dim
    X(:,k)=x(1+(k-1)*delay:len+(k-dim)*delay);
end

% Generate multivariable coefficient array.
[xs,xd]=cent6(X,dt);
[coef]=nlfit(xs,xd);

% Calculate values of resulting derivatives.
Xd=nlpvel(X,coef);
This is a Matlab program to create a "delayed vector" coordinate system, "xTau", given a vector, "x", the delay value, "delay", and the desired dimension, "dim".

[xTau]=delayed(x,delay,dim)

Created by: Edward H. Ziegler, 11/93

function [xTau]=delayed(x,delay,dim)

len=length(x)-dim*delay;
xTau=zeros(len,dim);
for k=1:dim
    xTau(:,k)=x((k-1)*delay+1:(k-1)*delay+len);
end
This is a MATLAB program to find the best fitting coefficients "A" for the 28 equations listed below. These fits are based on variations of least-squared regression method of the vectors "x" and "y". The goodness of fit is also evaluated by the value "R", which is ideally equal to 1.

1. \( y = a_0 + a_1x \) 
2. \( y = a_0 + a_1x + a_2x^2 \) 
3. \( y = a_0 + a_1x + a_2x^2 + a_3x^3 \) 
4. \( y = 1 / (a_0 + a_1x) \) 
5. \( y = 1 / (a_0 + a_1x + a_2x^2) \) 
6. \( y = 1 / (a_0 + a_1x + a_2x^2 + a_3x^3) \) 
7. \( y = a_0 + a_1/x \) 
8. \( y = a_0 + a_1/x + a_2x^2 \) 
9. \( y = a_0 + a_1/x + a_2x^2 + a_3x^3 \) 
10. \( y = 1 / (a_0 + a_1/x + a_2x^2 + a_3x^3) \) 
11. \( y = 1 / (a_0 + a_1/x + a_2x^2 + a_3x^3) \) 
12. \( y = 1 / (a_0 + a_1/x + a_2x^2 + a_3x^3) \) 
13. \( y = ax^b \) 
14. \( y = ab^x \) 
15. \( y = ab^{1/x} \) 
16. \( y = ax^{bx} \) 
17. \( y = ax^{b/x} \) 
18. \( y = a*exp(bx) \) 
19. \( y = a*exp(b/x) \) 
20. \( y = a + b*ln(x) \) 
21. \( y = 1 / (a + b*ln(x)) \) 
22. \( y = ab^x x^c \) 
23. \( y = ab^{1/x} x^c \) 
24. \( y = a*exp((x - b)^2/c) \) 
25. \( y = a*exp((ln(x) - b)^2/c) \) 
26. \( y = ax^b (1-x)^c \) 
27. \( y = a(x/b)^c \exp(x/b) \) 
28. \( y = 1 / (a(x + b)^2 + c) \)

\[ [A,R]=fit(x,y) \]

Created by: Edward H. Ziegler, 11/93

function \([A,R]=fit(x,y)\)

\( N=length(x); \) 
\( y=y(:); x=x(:); \) 
\( A=zeros(28,4); \) 
\( R=zeros(28,1); \) 
\( X=ones(N,4); \) 
for m=1:4
\( X(:,m+1)=x.^(m); \) 
end
% 1-3. \( y = a_0 + a_1x + a_2x^2 + a_3x^3 \)
for \( n=1:3 \)
\[
A(n,1:n+1) = (X(:,1:n+1)y)' ;
\]
\[
y_1 = X(:,1:n+1)*A(n,1:n+1) ;
\]
\[
r = \text{corrcoef}(y,y_1) ;
\]
\[
R(n) = r(1,2) ;
\]
end

% 4-6. \( y = 1/( a_0 + a_1x + a_2x^2 + a_3x^3 ) \)
\( y_t = 1/y ; \)
for \( n=1:3 \)
\[
A(n+3,1:n+1) = (X(:,1:n+1)y_t)' ;
\]
\[
y_1 = 1/(X(:,1:n+1)*A(n+3,1:n+1)) ;
\]
\[
r = \text{corrcoef}(y,y_1) ;
\]
\[
R(n+3) = r(1,2) ;
\]
end

% 7-9. \( y = a_0 + a_1/x + a_2x^2 + a_3x^3 \)
\( X = 1/X ; \)
for \( n=1:3 \)
\[
A(n+6,1:n+1) = (X(:,1:n+1)y)' ;
\]
\[
y_1 = X(:,1:n+1)*A(n+6,1:n+1) ;
\]
\[
r = \text{corrcoef}(y,y_1) ;
\]
\[
R(n+6) = r(1,2) ;
\]
end

% 10-12. \( y = 1/( a_0 + a_1/x + a_2x^2 + a_3x^3 ) \)
\( y_t = 1/y ; \)
for \( n=1:3 \)
\[
A(n+9,1:n+1) = (X(:,1:n+1)y_t)' ;
\]
\[
y_1 = 1/(X(:,1:n+1)*A(n+9,1:n+1)) ;
\]
\[
r = \text{corrcoef}(y,y_1) ;
\]
\[
R(n+9) = r(1,2) ;
\]
end

% 13. \( y = ax^b \)
FIT.M (cont.)

\[ yt = \log(y); xt = \log(x); \]
\[ p = \text{polyfit}(xt, yt, 1); \]
\[ A(13, 1) = \exp(p(2)); A(13, 2) = p(1); \]
\[ y1 = A(13, 1) \cdot x \cdot A(13, 2); \]
\[ r = \text{corrcoef}(y, y1); \]
\[ R(13) = r(1, 2); \]

\% 14. \[ y = ab^x \]
\[ yt = \log(y); \]
\[ p = \text{polyfit}(x, yt, 1); \]
\[ A(14, 1) = \exp(p(2)); A(14, 2) = \exp(p(1)); \]
\[ y1 = A(14, 1) \cdot A(14, 2) \cdot x; \]
\[ r = \text{corrcoef}(y, y1); \]
\[ R(14) = r(1, 2); \]

\% 15. \[ y = ab^{(1/x)} \]
\[ yt = \log(y); xt = 1/x; \]
\[ p = \text{polyfit}(xt, yt, 1); \]
\[ A(15, 1) = \exp(p(2)); A(15, 2) = \exp(p(1)); \]
\[ y1 = A(15, 1) \cdot A(15, 2) \cdot (1/x); \]
\[ r = \text{corrcoef}(y, y1); \]
\[ R(15) = r(1, 2); \]

\% 16. \[ y = ax^{bx} \]
\[ yt = \log(y); xt = x \cdot \log(x); \]
\[ p = \text{polyfit}(xt, yt, 1); \]
\[ A(16, 1) = \exp(p(2)); A(16, 2) = p(1); \]
\[ y1 = A(16, 1) \cdot x \cdot A(16, 2) \cdot x; \]
\[ r = \text{corrcoef}(y, y1); \]
\[ R(16) = r(1, 2); \]

\% 17. \[ y = ax^{b/x} \]
\[ yt = \log(y); xt = \log(x) \cdot x; \]
\[ p = \text{polyfit}(xt, yt, 1); \]
\[ A(17, 1) = \exp(p(2)); A(17, 2) = p(1); \]
\[ y1 = A(17, 1) \cdot x \cdot A(17, 2) \cdot x; \]
FIT.M (cont.)

```
r=corrcoef(y,y1);
R(17)=r(1,2);

% 18. y = a*exp(bx)
yt=log(y);
p=polyfit(x,yt,1);
A(18,1)=exp(p(2));A(18,2)=p(1);
y1=A(18,1).*exp(A(18,2).*x);
r=corrcoef(y,y1);
R(18)=r(1,2);

% 19. y = a*exp(b/x)
yt=log(y); xt=1 ./x;
p=polyfit(xt,yt,1);
A(19,1)=exp(p(2));A(19,2)=p(1);
y1=A(19,1).*exp(A(19,2) ./x);
r=corrcoef(y,y1);
R(19)=r(1,2);

% 20. y = a + b*ln(x)
xt=log(x);
p=polyfit(xt,y,1);
A(20,1)=p(2);A(20,2)=p(1);
y1=A(20,1)+A(20,2).*log(x);
r=corrcoef(y,y1);
R(20)=r(1,2);

% 21. y = l/(a + b*ln(x))
yt=1 ./y; xt=log(x);
p=polyfit(xt,yt,1);
A(21,1)=p(2);A(21,2)=p(1);
y1=1 ./(A(21,1)+A(21,2).*log(x));
r=corrcoef(y,y1);
R(21)=r(1,2);

% 22. y = ab^x *x^c
```
FIT.M (cont.)

Y = log(y);
X3 = ones(N,3);
X3(:,2) = x; X3(:,3) = log(x);
p = (X3\Y);'
A(22,1) = exp(p(1)); A(22,2) = exp(p(2)); A(22,3) = p(3);
y1 = A(22,1) * A(18,2).^x.*x.^A(22,3);
r = corrcoef(y,y1);
R(22) = r(1,2);

% 23. y = ab^(1/x) x^c
Y = log(y);
X3(:,1) = ones(N,1); X3(:,2) = 1 ./x; X3(:,3) = log(x);
p = (X3\Y);'
A(23,1) = exp(p(1)); A(23,2) = exp(p(2)); A(23,3) = p(3);
y1 = A(23,1) * A(23,2).^(1 ./x).*x.^A(23,3);
r = corrcoef(y,y1);
R(23) = r(1,2);

% 24. y = a*exp((x - b)^2/c)
yt = log(y);
p = polyfit(x,yt,2);
A(24,3) = 1 ./p(1);
A(24,2) = -p(2).*A(24,3) ./2;
A(24,1) = exp(p(3) - A(24,2).^2 ./A(24,3));
y1 = A(24,1) .* exp(((x - A(24,2)).^2) ./A(24,3));
r = corrcoef(y,y1);
R(24) = r(1,2);

% 25. y = a*exp((ln(x)-b)^2/c)
yt = log(y); xt = log(x);
p = polyfit(xt,yt,2);
A(25,3) = 1 ./p(1);
A(25,2) = -p(2).*A(25,3) ./2;
A(25,1) = exp(p(3) - A(25,2).^2 ./A(25,3));
y1 = A(25,1) .* exp(((log(x) - A(25,2)).^2) ./A(25,3));
r = corrcoef(y,y1);
FIT.M (cont.)

R(25)=r(1,2);

% 26. y= ax^b (1-x)^c
yt=log(y);
X3(:,1)=ones(N,1); X3(:,2)= log(x); X3(:,3)= log(1-x);
p=(X3\Y).';
A(26,1)= exp(p(1)); A(26,2)=p(2); A(26,3)=p(3);
y1=A(26,1).*x.*A(26,2).*ones(N,1)-x.*A(26,3);
while y1 ~= NaN
    r=corrcoef(y,y1);
    R(26)=r(1,2);
end

% 27. y= a(x/b)^c exp(x/b)
Y=log(y);
X3(:,1)=ones(N,1); X3(:,2)= x; X3(:,3)= log(x);
p=(X3\Y).';
A(27,2)= 1/p(2); A(27,3)= exp(p(3));
A(27,1)= p(1)+p(3)*log(1/p(2));
y1= A(27,1).*(x./A(27,2)).*A(27,3).*exp(x./A(27,2));
while y1 ~= NaN
    r=corrcoef(y,y1);
    R(27)=r(1,2);
end

% 28. y= 1/(a(x + b)^2 + c)
yt= 1 ./y;
p=polyfit(x,yt,2);
A(28,1)= p(1); A(28,2)= p(2)/(2*p(1));
A(28,3)= p(3) - A(28,1)*A(28,2).^2;
y1=1 ./((A(28,1).*x+A(28,2)).^2 + A(28,3));
r=corrcoef(y,y1);
R(28)=r(1,2);
This is a Matlab program to increase the number of points in a data set, using linear interpolation between existing pairs of points to create a new vector, approximately "Ratio" times the one input, "x".

% [tl,xl]=expand(t,x,Ratio);

% Created by Edward H. Ziegler, 11/93

function [tl,xl]=expand(t,x,Ratio);

[M,N]=size(x);
dt=sum(t(2:M,:)-t(1:M-1,:))/(M-1);

tl=(t(l,:):dt/Ratio:t(M,:))';

xl= zeros((M-l)*Ratio+l,N);
for k = 0:Ratio-1
    xl(1+k:Ratio*(M-1)+k,:)=k.*(x(2:M,:)-x(l:M-l,:))./Ratio...+x(1:M-1,:);
end
xl((M-l)*Ratio+l,:)=x(M,:);
PRINTHPG.M

function printhpg
    meta metatmp     % Put current plot into temporary metafile
    gpp metatmp -dhpgl  % Invoke GPP, creating device specific output
    lpr -Pxerox8810 metatmp.hpgl % Delete the temporary metafile
    rm metatmp.met % This is a MATLAB program to send a MATLAB plot to a standard
    rm metatmp.hpgl % XEROX laser plotter in a UNIX based system.

PRINTPS.M

function printps
    meta metatmp     % Put current plot into temporary metafile
    gpp metatmp /dps % Invoke GPP, creating device specific output
    delete metatmp.met,* % Delete the temporary metafile
    rename metatmp.ps,*
    rename * .ps [.post]* .ps
    return
end
References


