EPA/600/R-10/182 | September 2010 | www.epa.gov/ord

United States
Environmental Protection
Agency

San Luis Basin Sustainability Metrics Project: A Methodology for Evaluating Regional Sustainability

Office of Research and Development
National Risk Management Research Laboratory

Front Cover Photo Credits

Top - Medano Creek; Photo M. Hopton; Intermittent flow of Medano Creek in the Great Sand Dunes National Park and Preserve, San Luis Valley, Colorado.

Bottom Left - solar panel close-up; Photo by A. Karunanithi; Solar panel at a solar farm under construction in the San Luis Valley.

Bottom Right - SLV Potato Harvest; Photo by Stephen Ausmus; Potatoes being harvested in the San Luis Valley of south-central Colorado. Rotating potatoes with cover crops provides many benefits, including nitrogen management, improved soil and water quality, and bigger potatoes and higher yields. http://www.ars.usda.

EPA/600/R-10/182
September 2010

San Luis Basin Sustainability Metrics Project: A Methodology for Evaluating Regional Sustainability

Edited by
Matthew T. Heberling
Matthew E. Hopton

Authors
Heriberto Cabezas
Daniel Campbell
Tarsha Eason
Ahjond S. Garmestani
Matthew T. Heberling
Matthew E. Hopton
Joshua Templeton
Denis White
Marie Zanowick

National Risk Management Research Laboratory
Office of Research and Development
U.S. Environmental Protection Agency

and

Richard T. Sparks
U.S. Department of Agriculture

Office of Research and Development
National Risk Management Research Laboratory

Notice

The views expressed herein are strictly the opinions of the authors and in no manner represent or reflect current or planned policy by the federal agencies. Mention of trade names or commercial products does not constitute endorsement or recommendation for use. The information and data presented in this product were obtained from sources that are believed to be reliable. However, in many cases the quality of the information or data was not documented by those sources; therefore, no claim is made regarding their quality.

6.0
Fisher Information and Order

6.1 Introduction

This chapter presents the theory, data, and methodology for using Fisher information (FI) to assess the dynamic order and overall stability of the San Luis Basin (SLB) regional system. In this chapter we provide background on the method, details of the numerical approach, and include a simple example demonstrating the computation method. Further, we delineate the data used to compute FI for the SLB, present results from the FI analysis and discuss the strengths and weaknesses of the approach.

Fisher information is a key method used in information theory and provides a means of monitoring a broad range of system variables to characterize the dynamic order of a system, to include its regimes, and identify regime shifts (Cabezas et al. 2003). Fisher information has not traditionally been used as a metric of sustainability, and its application to sustainability is, in fact, a relatively new concept. However, its usefulness as a measure of sustainability has been documented in the peer-reviewed scientific literature (Cabezas and Fath 2002, Cabezas et al. 2003, Fath et al. 2003, Karunanithi et al. 2008, Mayer et al. 2007) over a wide array of applications ranging from real and modeled ecosystems to climate.

We have used FI as a metric in the SLB project because it captures a critical aspect of the system that the other metrics do not. Well-functioning economic, ecological, and social systems may be necessary to preserve the self-organization and resilience of an environmental system and FI captures these aspects. Conceptually, the repeatability of observations denotes order in dynamic systems (Karunanithi et al. 2008). Therefore, a dynamic system is said to be well functioning and orderly when its behavior over time follows characteristic, regular, and nearly predictable patterns. FI captures organization and resilience by assessing the dynamic behavior of systems.

To illustrate this point, we offer three examples. First, consider that biological systems have elaborate mechanisms that allow them to maintain a specific degree of organization (Kauffman, 1993). The fact that an ecosystem can be identified as such is itself evidence that it has observable properties that, although variable, are still sufficiently stable to define the system. Indeed, the properties do vary, but they do so in a characteristically orderly and almost predictable manner. Further, as a system experiences a regime shift (i.e., loss of characteristic properties), the variation of key properties generally increase (Carpenter and Brock 2006) which causes a loss of order. Second, similar arguments can be made for markets that are left to operate freely. The observable properties of the markets follow regular patterns, which indicate order. Movement toward market equilibrium helps to predict how prices will change (Maurice and Phillips 1992). The existence of patterns in the markets (i.e., prices as rationing and signaling devices) allows consumers and producers to make decisions with the expectation of particular results (Mankiw 2009; Maurice and Phillips 1992). Finally, consider that the four seasons of the year constitute an orderly dynamic system because the seasons succeed each other in regular, although not exactly precise, timing year after year (Houghton 2002). Indeed, autumn may sometimes come late and climatic fluctuations can occur; however, the seasons follow a pattern.

In summary, ecosystems and social systems follow very complex but real patterns of change over time. Many processes in ecosystems cycle and change along relatively orderly patterns often driven by the daily and yearly cycles of light, tides and seasons along with longer term decadal, centurial, and other temporal and spatial variations. These cycles, although complex, are sufficiently regular to be studied and understood (Odum 1971). Markets likewise follow complex but orderly patterns and they are sufficiently regular to be studied and understood (Maurice and Phillips 1992). In brief, many systems, when functioning well, show regular characteristic behavior, which is an indication of dynamic order. Accordingly, order, or more specifically dynamic order, is an important and very fundamental indicator of the state or condition of the system (Mayer et al. 2007). It is for this reason that FI has been incorporated into the suite of metrics used to assess sustainability in the SLB project.

6.1.1 Capturing Dynamic Shifts with Fisher Information

When any system, including the aforementioned ones, undergoes a change from one characteristic pattern or set of behaviors to another, the change is generally termed a regime shift. Because no two regimes have the same observable patterns, a regime shift is typically accompanied by corresponding changes in dynamic order, which can be tracked by FI. To illustrate this point, consider the very simple case where a system is represented by one variable (e.g., temperature) which is being measured over time. Keeping in mind that the

system here is represented by its temperature only, if the temperature is the same (i.e., within measurement error) every time it is measured, then the system can be said to exist in one state only. This system has as much order as it can possibly have, and the FI associated with the temperature is at a maximum. On the other hand, if the temperature is different every time a measurement is made, even after accounting for measurement error, say the temperature is rising linearly with time, then the system can be said to exist in many different states and the FI is at a minimum. Hence, a system with a well-defined pattern (i.e., constant properties) has high dynamic order and high FI, and a system that has no defined patterns (i.e., constantly changing properties) has low order and low FI. Although these two examples represent extreme behavior, real systems typically function between these extremes.

Further exploring this idea, consider a situation where the aforementioned system is cooled, allowed to remain at a constant low temperature, within measurement error, for a period of time, and then heated to a constant temperature with larger variability, perhaps very small amounts of heat are added or removed randomly. In this case, when the temperature is decreasing, the FI will be very small or zero. Once it reaches a constant low temperature, the FI will be very high. As the system is heated and the temperature increases, the FI will again be very small or zero (during the change), and then increases to a relatively steady FI. However, the final "steady" FI value will be less than the original due to the higher variability in the temperature. The system described here experienced a dynamic regime shift, as there was a noted drop in FI between two stable regimes, one with higher order than the other. From these explorations, it is evident that FI tracks the order and stability of the system. Although we used simple examples to illustrate the very basic relationship between system dynamics and FI, as discussed later, the FI method has been adapted to assess the dynamic changes in complex systems described by multiple and disparate variables.

6.2 Methods

6.2.1 Theory

Fisher information was formally developed by statistician Ronald Fisher as the information obtainable from data when estimating the value of a parameter (Fath et al. 2003, Fisher 1922). It has since been adapted as a measure of dynamic order. Details on the derivation of FI from its original form are provided in Fath et al. (2003), Mayer et al. (2007), Karunanithi et al. (2008), and Appendix 6-A.1. From the derivation, we obtain a representation of FI (denoted here as I to preserve the historical context; Fisher 1922, Frieden, 1998, 2004, Karunanithi et al. 2008, Mayer et al. 2007) based upon the probability of a system p(s) being in a particular state (s):

$$I = \int \frac{ds}{p(s)} \left[\frac{dp(s)}{ds} \right]^2 \tag{6.1}$$

From this expression, an approach was developed that affords the ability to assess the dynamic order of real systems and is derived as follows. In order to minimize calculation errors from very small probability values, we replace p(s) in Equation 6.1 with its amplitude, which is defined, by $q^2(s) \equiv p(s)$, such that:

$$\frac{dp(s)}{ds} = 2q(s)\frac{dq(s)}{ds} \therefore \left(\frac{dp(s)}{ds} \right)^2 = 4q(s)^2 \left(\frac{dq(s)}{ds} \right)^2 \tag{6.2a}$$

Substituting Equation 6.2a into Equation 6.1, the expression becomes:

$$I = 4\int \left[\frac{dq(s)}{ds} \right]^2 ds \tag{6.2b}$$

Equation 6.2b is adapted for use with discrete data by using a summation to approximate the integral and replacing ds and dq(s) with $\Delta s = s_i - s_{i+1}$ and $\Delta q = q_i - q_{i+1}$, such that:

$$I \approx 4\sum_{i=1}^{m} \left[\frac{q_i - q_{i+1}}{s_i - s_{i+1}} \right]^2 (s_i - s_{i+1}) \tag{6.2c}$$

In Equation 6.2c, m is the number of states and s_i is merely an index denoting a particular state of the system (e.g., s_1 is state 1, s_2 is state 2, etc.). Accordingly, $s_i - s_{i+1} = 1$ and Fisher information is then:

$$I \approx 4\sum_{i=1}^{m} \left[q_i - q_{i+1} \right]^2 \tag{6.3}$$

Equation 6.3 is our working expression for computing the FI metric.

6.2.2 Calculation Methodology

The basis of computing FI is assessing changes in the probability of observing different states of the system through time. Therefore, we must obtain information on its condition (state) over time. Borrowing from standard statistical mechanics approaches, we define a system by n measurable variables (x_i) that characterize the system and its state at any point in time (Mayer et al. 2007). Note that the correlation structure of the variable time series is not critical, because our goal is to assess changes in dynamic order and not to develop a predictive model of system behavior. By describing a dynamic system in this way, it is said to have a trajectory

in a phase space representing all possible states of the system in n-dimensions and time. In the absence of measurement error, each point in phase space is defined by specifying a value for each of the variables at a point in time, pt_i: ($x_1(t_i)$, $x_2(t_i)$ $x_3(t_i)$…x_n (t_i)). However, because measurements of real system parameters contain error (i.e., uncertainty), the state s of the system is delineated by a region rather than a point. In other words, two points that differ from each other by less than the measurement error are indistinguishable and can be thought of as two measurements within the same state. Conversely, two points that differ from each other by more than the measurement error are, in fact, two legitimately different points and consequently, exist in two distinct states of the system. From this conceptual description, we have a foundation for understanding the process of characterizing a dynamic system and assessing changes in its state over time.

In order to capture the dynamic behavior of a system, once the variables characterizing its state have been gathered, the time period is divided into time windows. We achieve this by first defining a parameter (*hwin*) that denotes the size (in time steps) of each window and then determining the amount of overlap (in time steps) desired for each window (*winspace*). The parameters *hwin* and *winspace* denote the integration window parameters used to move through the time series data by creating a sequence of overlapping windows such that *hwin* > *winspace* as shown in Fig. 6.1. This convention affords the ability to compensate for changes in dynamic behavior that may extend beyond the boundary of each window. The probability densities p(s) and corresponding FI are computed within each of these windows. Whereas *hwin* is selected based upon the amount of data available, we have empirically found that it should be at least eight time steps. The next step in the procedure is "binning" each point within the time windows into discrete states of the system. Estimating the uncertainty of the variables is key to defining the states of the system.

As previously mentioned, inherent in any measurement is a level of uncertainty. Accordingly, we define a parameter Δx_i as the measurement uncertainty for each variable (x_i) such that, if the condition:

$$\left|x_i(t_i) - x_i(t_j)\right| \leq \Delta x_i \tag{6.4}$$

is true for all variables at time t_i and t_j then the two points are indistinguishable and subsequently, grouped in the same state (i.e., binned together). Because multiple variables are used to characterize a system and each variable has a distinct measure of uncertainty, a state is represented as an n-dimensional hyper-rectangle where each side is defined by an uncertainty (Δx_i) for each variable. Therefore, this Δx_i is the size of the states for the system.

Typically, knowledge of the measurement uncertainty of underlying state variables is unknown; therefore, we have developed approaches for estimating Δx_i. One tactic is to find a relatively stable time period within the system trajectory, calculate the variation in each variable in this period, and assume this to be the measurement uncertainty. This approach is implemented by calculating the standard deviation (SD) for each variable in the "stable" period and applying Chebyshev's inequality, which is defined by:

$$P(\left|X-\mu\right| < kSD) \geq \left(1-\frac{1}{k^2}\right) \tag{6.5}$$

and indicates that independent of the type or form of the probability distribution, "The proportion of the observations falling within k standard deviations of the [population] mean is at least 1-1/k²" (Lapin 1975: 58). Therefore, we define the *size of states* parameter (Δx_i) as a function of SD, such that $\Delta x_i = \pm kSD_i$. By setting k = 2, $\Delta x_i = \pm 2SD_i$ at least 75% of the data would occur within this level of uncertainty. This provides a lower bound for the probability of values being k standard deviations from the mean. However, the probability could be much higher (>95% at k=2) if, for instance, the data are normally distributed. Given that each variable has a unique measure of uncertainty, the *size of states* then is noted as a row vector $\Delta x_i = [k\mathrm{SD}_1, k\mathrm{SD}_2 \ldots . k\mathrm{SD}_n]$, where k is a scalar constant. Another method of defining the *size of states* is by locating a similar system that exhibits stability and using the variation within this system as a measure of uncertainty for the system under study.

Once the integration window parameters (*hwin* and *winspace*) and *size of states* (Δx_i) are determined, the data may be distributed (binned) into different states of the system. The binning process begins with the first point within the time window taken as the center of the first state. A hyper-rectangle with the side lengths defined by Δx_i for each variable is established around the point, such that all of the points falling within its boundaries are considered to be in the same state. Then, the next unbinned point in the window is assumed to be the center of a new state, a new hyper-rectangle is constructed around it, and all the points within its boundaries are binned into that state. The process continues until all of the points within the first window are binned and is repeated for the remaining time windows until all of the points in each window are binned into states of the system and the data are exhausted.

When measurement uncertainty of the underlying state variables is known and independent, the binning process alone would work well in defining states of the system. Unfortunately, data collected from public sources, as in this project, often do not report the uncertainty associated with their data. Accordingly, an additional step is implemented to mitigate the effect of data error by applying a tightening level parameter (Karunanithi et al. 2008). The tightening level (TL) adjusts the binning criteria such that a point can be declared to be within a given hyper-rectangle (particular state of the system) when at least a certain percentage of the variables meet the *size of states* criteria (Equation 6.4). The percentage itself is the tightening level. For example, if a system is characterized by 100 variables and 95 of the variables indicate that a particular point fits within the state being evaluated, then the two points would be binned together at a 95% tightening level. Therefore, points may be binned into a particular state of the system if the *size of states* criteria is true for all variables in particular time steps or the number of variables that satisfies the *size of states* criteria is greater than the product of the tightening level (TL) and the total number of variables (Fig. 6.2). When all of the points in a window are binned into states at a given tightening level, a probability distribution (p_i) is generated for each window using:

$$P_i = \frac{\#\ points\ in\ state}{total\ \#\ points\ in\ time\ window} \qquad (6.6)$$

Next the amplitude, q ($q_i = \sqrt{p_i}$) is calculated for each state and FI for the time window is computed using Equation 6.3. The computed FI value itself is assigned to the middle of each time window (see example in Fig. 6.1). Because there are no rigorous criteria for setting the tightening level, FI is computed for multiple tightening levels between strict and relaxed tightening. Rather than establishing an arbitrary lower bound for the tightening level, relaxed tightening is set as the lowest tightening level at which more than one state is observed in the window and FI is calculated by taking the arithmetic average of FI between strict tightening (TL=100%) and lower bound (relaxed tightening level). Note that this results in multiple p_i and FI values for each window (Karunanithi et al. 2008). The final FI reported in each window is again an average over the tightening levels from strict to relaxed tightening. Finally, in accordance with the Sustainable Regimes Hypothesis (see section 6.2.3), a regime is denoted as sustainable when the dynamic order does not change with time (i.e., $(d\langle FI\rangle/dt \approx 0)$ where <FI> indicates a mean FI value). The <FI> is calculated by computing the mean of neighboring FI values. This convention affords the ability to focus on trends in dynamic order and not fluctuations. As such, Equation 6.7 is used to compute a three-point mean, $<FI>_j$ for window j:

$$<FI>_j = \frac{1}{3}\left(FI_{j+1} + FI_j + FI_{j-1}\right)$$

$$j = w\text{-}1,\ w\text{-}3,\ w\text{-}5,\ w\text{-}7, \ldots \qquad (6.7)$$

where w is the year corresponding to the last window of the original FI computation. Based on our experience, the three-point mean tends to preserve trends and deemphasize short fluctuations. The $<FI>_j$ is computed in reverse order to ensure that the more recent period is not omitted when <FI> is evaluated. However, given that a true measurement of uncertainty is unknown for the metrics or underlying variables, we can visually inspect the trends, but we cannot test for statistical significance.

6.2.3 Interpreting Fisher Information

The Sustainable Regimes Hypothesis encompasses conceptual ideas governing the use and interpretation of FI as a metric for assessing sustainability (Cabezas and Fath 2002, Karunanithi et al. 2008). In summary, the hypothesis states that: 1) well functioning systems exist within an orderly dynamic regime with non-zero FI that does not change with time (i.e., $(d\langle FI\rangle/dt \approx 0)$; 2) steadily decreasing FI signifies a progressive loss of dynamic order and denotes a system that is becoming disorganized and losing functionality; 3) steadily increasing FI indicates that the system that is becoming more ordered (although, not necessarily more desirable by humans); and 4) a steep decrease in FI between two dynamic regimes denotes a regime shift (Karunanithi et al. 2008). As a note, both conditions of statement 1 must be true for a system to be considered sustainable. In other words, a completely disorganized system has no order over time; therefore, a system with <FI> ≈ 0 is not sustainable even if $d\langle FI\rangle/dt \approx 0$.

Important elements in regime shift detection are determining the occurrence, pervasiveness, and intensity of a shift (Karunanithi et al. 2008). The fourth statement of the Sustainable Regimes Hypothesis indicates that a regime shift has occurred when there is a significant drop in FI between two dynamic regimes. Once a regime shift has been detected, the intensity simply relates to the level of the drop in FI (e.g., a more severe shift has a steeper drop). The pervasiveness of the shift relates to the number of system variables affected by the shift and is characterized by varying the tightening level and noting the lowest tightening level at which a particular shift can be detected. Recall, that the tightening level relates to the number of variables that must meet the *size of states* criteria for binning. Accordingly, the more relaxed (lower) the tightening level at which the regime shift is recognized, the more pervasive the shift.

6.2.4 Computing Fisher Information: A Simple Example

Below a simple example is used to demonstrate the procedure for computing FI. From Section 6.2.2, the basic algorithm is as follows: (1) establish the size of the time windows (*hwin*), (2) determine the time increment that the window will be moved forward (*winspace*) to create overlapping windows, (3) set a tightening level (TL), (4) bin all of the points into states within each window, (5) compute the probability density for each state in each time window – the result at this point will be a sequence of probability densities p_i for each time window, (6) calculate FI from the q_i in each time window, (7) set a new tightening level, (8) repeat steps 4 through 7 until all the computations have been done from strict tightening (TL = 100%) to relaxed tightening (the lowest TL at which more than one state is present in the window), (9) compute an average FI for each window over the tightening levels from strict to relaxed tightening and (10) calculate <FI> (mean FI) values for the system.

For the sake of simplicity, we used two demographic variables (population and personal income) from the SLB data (Fig. 6.3). Because the goal of this exercise is simply to step through the computation algorithm, we used the guidelines for the integration window parameters (see section 6.2.2) and selected values for *hwin* and *winspace* that met the criteria. Recall that the parameters should be set such that *hwin* ≥ 8 and *winspace* < *hwin*. Accordingly, we set *hwin* to eight time steps (years) and *winspace* to three time steps to ensure we ended up with a manageable number of windows for this computation demonstration. To provide insight into how the integration window parameter settings may affect the computation result, we performed a sensitivity analysis to examine the impact changes in these parameters have on FI (Appendix 6-B). However, the sensitivity analysis is not a requirement for setting *hwin* and *winspace*. The only specific parameter guidelines are provided in section 6.2.2.

Following the basic algorithm already described, we used the *hwin* and *winspace* values to partition the data into over lapping windows (Table 6.1). Next, we set the tightening level (TL) at 100% to include all variables in the computation and then binned the points into states of system within each window (Fig. 6.2). The purpose of binning is to determine the state of the system over time and the basis of this process is grouping points (e.g., $pt_1=(x_1(t_1), x_2(t_1))$ within an established boundary of uncertainty. As explained in section 6.2.2, if the level of uncertainty for the variables under study is unknown, it may be estimated. In this case, it was estimated by assuming a measurement uncertainty by computing the standard deviation (SD) of each variable over the first five time steps. Each SD was then multiplied by two in accordance with Chebyshev's theorem (Lapin 1975: 58), such that the level of uncertainty (Δx_i) defined as $\pm kSD_i$, is a 1×2 vector: $\Delta x_i = [2\times SD_1, 2\times SD_2] = [86285.39, 1639.83]$. From Fig. 6.2 and Equation 6.4, if the tightening level multiplied by the total number of variables in the time series meet the *size of states* criteria (i.e., $|x_i(t_i) - x_i(t_j)| \leq \Delta x_i$) then the two points at t_i and t_j are binned in the same state. The differences were computed by subtracting the value of the variables at each time step starting with the first point in the window (Table 6.2). For example, the difference between variable x_1 at time =1 and time = 2, i.e.,$|x_1(t_2) - x_1(t_1)| = |314914-310352| = 4562$.

A "bulls-eye" (radar) plot of the differences provides a visual depiction of point binning (Fig. 6.4). The first binning pass starts with assessing the distance from pt_1, accordingly, pt_1 is the center of this figure with an absolute difference value of (0, 0) and the remaining absolute differences are plotted on the corresponding axis (e.g., 2-1 = ($|x_1(t_2) - x_1(t_1)|$, $|x_2(t_2) - x_2(t_1)|$) = (4562, 511)). The red and blue lines represent the boundary of uncertainty (i.e., *size of states* = Δx_1 and Δx_2) around the center for each variable. Points 1, 2, and 3 are binned into state 1, because pt_2 and pt_3 are within the boundary of uncertainty (i.e., less than Δx_i from pt_1). The process then moves to the next "un-binned" point (pt_4), establishes it as the center of state 2 and then bounds state 2 as Δx_i from pt_4, such that points 4-8 are binned into state 2. The binning procedure continues to the next window until all points have been binned into a state of the system. Once all the points have been binned, then probability densities are computed for each window using Equation 6.6. In window 1, there were three points binned into state 1 and five in state 2 resulting in a 37.5% chance (p(1)=3/8) that the system is in state 1 and a 62.5% chance that it is in state 2. This process was repeated for each window, resulting in probability densities for each window (Fig. 6.5). Next, the amplitude ($q(s) = \sqrt{p(s)}$) was computed for each state (Table 6.3). To compute FI for each time window, the gradients of the amplitude ($q_i - q_{i+1}$) were used as in Equation 6.3 (see Fig. 6.6). At this point, the tightening level may be decremented (i.e., TL = TL-1) and the computation steps repeated as articulated in section 6.2.2. This produces FI values for the time series for TL ranging from strict to relaxed tightening. An average FI result is calculated by taking the average of all FI values computed within TL range. For this exercise, we only computed FI at TL = 100% and then

used Equation 6.7 to calculate a three-point <FI> to smooth any fluctuations in the result and focus on the general trend of the metric (Fig. 6.7). Further details for using the code and graphical user interface (GUI) to compute FI are provided in Appendices 6-C and 6-D. The procedure for computing FI has been automated in MATLAB (Release 2009a; Mathworks, Inc.) so that only time series data that characterize the system, *size of states*, *hwin*, and *winspace* need to be provided by the user to perform the analysis. The code is provided in Appendix 6-E.

6.3 Computing FI for the San Luis Basin: Data and Sources

To assess dynamic order in the SLB, we needed data that characterized the state of the system. Accordingly, it was necessary to gather data that represent environmental, social, and economic aspects of the region. Because the variables used to compute the other metrics satisfied this criterion, data for this project were selected from the datasets used to calculate EFA, GNRP and EmA (Chapters 3, 4, and 5, respectively). However, any data that adequately characterize a system may be used to calculate FI. Further, we selected variables that contained data for the entire 26 years of the study (1980 - 2005). Each of the variables was assigned to one of six categories for computing FI and encompassed information on consumption (food and forest), environment, and demographic characteristics, as well as, the energy consumption, land use, and agricultural production aspects of the SLB (Table 6.4). Details on each variable are provided in the corresponding chapters (Chapters 3-5).

6.4 Calculation Methodology

Following the approach described in sections 6.2.2 and 6.2.4, as well as, the guidelines for computing FI using MATLAB (Appendices 6-C and 6-D), the GUI (Main_Fisher_data_proc.m) was used to compute both FI and <FI>. The file containing data for the 53 variables over 26 years, the time file which indicates the years examined (i.e., 1980, 1981, 1982,… 2005), and the file containing the *size of states* were selected. Following the guidelines in Section 6.2.2, the *size of states* for the SLB was determined using a small number of years where data were relatively stable. Thus, the *size of states* was defined by calculating the SD of the first five data points (years) for each variable and then (in accordance with Chebyshev's inequality) multiplying the result by 2 (k = 2).

Given the amount of data and the results of the sensitivity analysis (Appendix 6-B), the time window was set to be eight time steps (*hwin* = 8) and the window increment was set to one time step (*winspace* = 1). Thus, FI was integrated over an eight-year window that was moved in one-year increments. The FI value reported represents the FI computed over a specific period and is reported in the center of that window (i.e., FI for 1984 = 1980-1987, FI for 1985 = 1981-1988, FI for 1986 = 1982-1989, etc). The <FI> (for each window *j*) was calculated using Equation 6.7, such that the $\langle FI \rangle_j$ reported is a mean placed in the center of the three points used to calculate it (e.g., $\langle FI \rangle_{2001} = (FI_{2000} + FI_{2001} + FI_{2002})/3$). Further, in order to explore drivers potentially responsible for changes in dynamic order of the system overall, we compared FI of the overall system (included all 53 variables) to that of the variables grouped by categories (e.g., energy consumption). Spearman's rank correlation coefficient analysis was used to assess the relationship between the dynamic order of the system and the variables by category. The statistical significance level established was $P \leq 0.05$.

6.5 Results

FI was computed for the SLB over the 26 years of the study and ranged in value from 2.92 in the period centering 2001 to 4.37 in the period centering 1996 (Fig. 6.8). <FI> ranged from 3.22 in the period centering 1986 to 3.89 in the period centering 1995 (Fig. 6.9; Table 6.5). The system <FI> increased initially, peaked in the period centering 1995, and then decreased slightly thereafter.

The minimum <FI> in the system in 1986 corresponded (visually) to a minimum <FI> in the consumption (food and forest) and agricultural production categories (Fig. 6.10). The peak in the system <FI> in 1995 corresponds (visually) to a peak in the <FI> of the environmental and energy categories, as well as, agricultural production which increases up to 1992 and essentially remains steady until 1998 (Table 6.5; Figs. 6.9 and 6.10). Although the <FI> of consumption (food and forest) category remains relatively steady, the <FI> of the other categories exhibit larger changes over time.

Similar to the overall system, dynamic order of the agricultural production category rose from 1989 to 1992, was relatively steady until 1998, and decreased thereafter (Fig. 6.10). The opposite was true of the demographic variables as the <FI> peaked in 1989, decreased from 1989 to 1995 and increased thereafter. By the end of the study period energy, environment, and demographic categories exhibited an increasing trend (Table 6.5; Fig. 6.10). Further, Spearman's rank analysis indicated a significant, negative correlation between the overall system and the demographic category ($r = -0.88$, $df = 4$, $P = 0.033$).

6.6 Discussion

According to criteria in the Sustainable Regimes Hypothesis (Section 6.2.3), the results of this <FI> assessment indicated that although there were changes in <FI> over time, <FI> was relatively steady during the period and there was no indication of an overall system regime shift (i.e., no sharp drop between two regimes). The dynamic order of the system increased up to 1995 and exhibited a small decrease thereafter. Although the demographic, energy, and environmental categories showed an increase in dynamic order at the end of the 26 years, the overall system, consumption (food and forest), land use, and agricultural production categories indicated a decreasing trend in <FI> which may denote some movement away from sustainability (i.e., decreasing dynamic order). There was, however, no sharp drop in <FI> that could indicate a regime shift.

Spearman's rank analysis indicated that there is an inverse relationship between the demographic category and that of the overall system. Toward the goal of policy setting and decision making, variables within a category may be explored to uncover possible drivers of dynamic behavior. As an illustration, a closer look at the FI of the demographic category revealed that changes in dynamic order seem to correspond to changes in the underlying variables (e.g., population and personal income). Because FI is a measure of system order, increases in variability of the system variables generally result in decreasing FI. As a simple exploration to see if we could identify such variables, we computed the annual percent change in the demographic variables as a measure of variability. While population generally increased during the period examined, the percent change varied during the 26 years (e.g., +1.6% from 1982 to 1983 and +0.9% from 1983 to 1984 graphically is a decrease). Further, unlike the percent change in personal income over time, the percent change in population decreased initially, peaked in 1995, and decreased thereafter. A visual comparison of these patterns and demographic FI suggests, the variation in population (as described by percent change) appears to be inversely proportional to demographic FI (i.e., low variation in demographic data corresponds to high FI values). For that reason, changes in annual percent change of population seem to correspond to changes in dynamic order of the demographic category (Fig. 6.11). From this simplified approach, there appears to be a relationship; however, other techniques (e.g., sensitivity analysis) to explore the effect of changes in underlying variables on FI may help determine which variables drive changes in dynamic order.

In summary, there was no indication of a regime shift in the overall system. Although, the system exhibited small changes in dynamic order during the 26 years, we conclude the SLB is relatively stable with a slight indication of possible movement away from sustainability near the end of the period.

6.7 Strengths and Weaknesses

One of the key strengths of FI is the ability to collapse data from complex, multivariate systems into a fundamental metric that can be computed over time and used to evaluate the dynamic behavior of systems. This is important because the characterization of complex, integrated systems for sustainability (social, ecological, and economic systems) often requires a large number of disparate variables. Traditional approaches to assessing regime shifts and system dynamics (e.g., variance) have typically only been demonstrated on simple model systems and work must be done to determine whether these methods can be used to evaluate real, complex systems (Scheffer et al. 2009). However, the calculation of dynamic order using FI is insightful, theoretically sound, and provides a means of evaluating both model and real systems that are characterized by multiple, disparate variables.

As is the case for many methods used to analyze complex systems, the computation of FI has its challenges. Some of the challenges include establishing the integration window parameters (i.e., *hwin* and *winspace*), the need for determining the measurement error for each variable (used to set the *size of states*), and the availability and quality of data to characterize the system. The integration window parameters are used to traverse through the time series data and establish overlapping windows for the FI computations. These parameters must be carefully selected based on amount of data available and knowledge of the system. The *size of states* is a key parameter used for binning points into states of the system and is based on estimating the amount of error present in the data. Strategic recommendations for determining the *size of states* and the integration window parameters were provided in sections 6.2.2 and Appendix 6-B, respectively. Data availability and quality issues are discussed in Chapter 7.

Another challenge is that because PDFs are used to calculate FI, one FI value is provided in each time window and reported in the center of that period. Accordingly, there is a unique FI for each time window and not each time step. Hence, FI values tend to come several years behind the latest data point (e.g., the last data were for 2005, but the last FI value was for 2001). Moreover, according to the Sustainable Regimes

Hypothesis, in order to capture the trends (and not fluctuations) in the dynamic order, a <FI> is calculated. We therefore use a three-point <FI> to evaluate trends in dynamic order over time. Based on our experience with these calculations, three points seem to smooth the result, while maintaining the characteristic changes in dynamic order associated with trends.

Further, the methodology can identify if a system is either maintaining its current order (i.e., $(d\langle FI\rangle/dt \approx 0)$or losing dynamic order $(d\langle FI\rangle/dt < 0)$ and heading toward or already experiencing a regime shift (i.e., a sharp drop in <FI> between two stable regimes). The loss of dynamic order, and certainly a regime shift, indicates the system is moving away from sustainability. However, an increase in <FI> indicates the system is gaining order, yet does not necessarily mean the system is moving toward a more preferable state (i.e., sustainability) in terms of human wants and needs (Karunanithi et al. 2008). Hence, FI is a one-sided test of sustainability and care must be taken on interpreting changes in dynamic order with reference to human preferences.

Lastly, we acknowledge that the details of the conceptual underpinning of FI analysis can be quite abstract and, perhaps, difficult for many potential users to understand. However, we also believe that anyone with a good grounding in science or engineering can understand the fundamental concept sufficiently to apply the method and interpret the results appropriately. We note that practical and effective end users of methods and technologies in many professions are not necessarily experts in the theory behind the tool being used. For example, most physicians are not experts in Quantum Mechanics, yet they can effectively use Magnetic Resonance Imaging (MRI) and interpret the results. Use of MRI may involve a medical technologist who serves as a bridge between those who designed the MRI and the physicians that interpret and use the results. The point here is that it should be possible to make the benefits of otherwise abstract concepts (such as FI) available to non-expert end users through software and/or individuals that act as intermediaries. To help make the concept more accessible, we provided an example of the computation and interpretation of FI using a simple example of dynamic order (Section 6.2.4). Further, we provided the MATLAB code necessary to compute FI and stand-alone software with a GUI to simplify calculating FI. The GUI improves the usability of the metric by automating the computation process.

6.8 References

Anderson, M.J. 2005. Trimming the FAT out of Experimental Methods. OE (Optical Engineering) Magazine, September 2005, p. 29.

Cabezas, H., Fath, B.D. 2002. Towards a theory of sustainable systems. Fluid Phase Equilibria 3-14, 194-197.

Cabezas, H., Pawlowski, C.W., Mayer, A.L., Hoagland, N.T. 2003. Sustainability: ecological, economic, technological and systems perspectives. Clean Technologies and Environmental Policy 5, 176-180.

Cabezas, H., Pawlowski, C.W., Mayer, A.L., Hoagland, N.T. 2005. Simulated experiments with complex sustainable systems: ecology and technology. Resources, Conservation and Recycling 44, 279-291.

Carpenter, S.R., Brock, W.A. 2006. Rising Variance: A Leading Indicator of Ecological Transition. Ecology Letters 9-3, 311-318.

Fath, B., Cabezas, H., Pawlowski, C.W. 2003. Regime changes in ecological systems: an information theory approach. Journal of Theoretical Biology 222- 4, 517-530.

Fisher, R.A. 1922. On the mathematical foundations of theoretical statistics. Philosophical Transactions of the Royal Society of London 222, 309-368.

Frieden, B.R. 1998. Physics from Fisher Information: A Unification. Cambridge University Press, Cambridge, UK.

Frieden, B. R. 2004. Science from Fisher Information: A Unification. Cambridge University Press, Cambridge, UK.

Gatenby, R.A. (Eds), Exploratory Data Analysis Using Fisher Information. London: Springer-Verlag, pp. 217-244.

Grove, J.M. 1988. The Little Ice Age. Routlege, New York, New York, USA.

Houghton, J. 2002. The Physics of Atmospheres. Cambridge University Press, Cambridge, UK.

Karunanithi A.T., Cabezas, H., Frieden, B.R., Pawlowski, C.W. 2008. Detection and assessment of ecosystem regime shifts from Fisher information, Ecology & Society 13(1), 22.

Kauffman, S.A. 1993. The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, Oxford, UK.

Lapin, L. 1975. Statistics: Meaning and Method. Harcourt Brace Jovanovich, New York, New York, USA.

Mankiw, N.G. 2009. Principles of Economics (5th Ed.). South-Western, Mason, Ohio, USA.

Mathworks, Inc. 2009. MATLAB Software: The Language of Technical Computing version 7.8.0.347: Release R2009a, Natick, MA.

Maurice, S.C., Phillips, O.R. 1992. Economic Analysis: Theory and Application, 6th Ed. Irwin, Homewood, Illinois, USA.

Mayer, A.L., Pawlowski C.W., Cabezas, H. 2006. Fisher Information and dynamic regime changes in ecological systems. Ecological Modeling 195, 72-82.

Mayer, A.L., Pawlowski, C.W., Fath, B.D., Cabezas, H. 2007. Applications of Fisher information to the management of sustainable environmental systems, in: Frieden, B.R., and

Montgomery, D. C. 1997. Design and Analysis of Experiments. John Wiley and Sons, Inc., pp. 4.

Odum, E.P. 1971. Fundamentals of Ecology, 3rd Ed. W.B. Saunders Company, Philadelphia, Pennsylvania, USA.

Scheffer, M., Bascompte, J., Brock, W. A., Brovkin, V., Carpenter, S. R., Dakos, V., Held, H., van Nes, E. H., Rietkerk, M., Sugihara, G. 2009. Early-warning signals for critical transitions. Nature, 461, 53-59.

Table 6.1 – Data were divided into a sequence of overlapping time windows. The eighth window was removed because there were not enough data in the time series to place in it. Therefore, all data were accounted for within seven windows. t = time (i.e., 1980-2005), x_1 = Population and x_2 = Personal income (in thousands). Point 1 (pt_1) was defined by ($x_1(t_1)$, $x_2(t_1)$) and is highlighted as 310352, 38469.

$pt_1 = (x_1(t_1), x_2(t1)) =$ (310352, 38469)

	Window 1				Window 2				Window 3		
#	t	x_1	x_2	#	t	x_1	x_2	#	t	x_1	x_2
1	1980	310352	38469	4	1983	384256	40112	7	1986	431318	40804
2	1981	314914	38980	5	1984	400888	40490	8	1987	436775	41104
3	1982	318946	39467	6	1985	424601	40369	9	1988	448738	41289
4	1983	384256	40112	7	1986	431318	40804	10	1989	504014	40903
5	1984	400888	40490	8	1987	436775	41104	11	1990	551552	40682
6	1985	424601	40369	9	1988	448738	41289	12	1991	549641	41283
7	1986	431318	40804	10	1989	504014	40903	13	1992	562981	41120
8	1987	436775	41104	11	1990	551552	40682	14	1993	625715	41466

	Window 4				Window 5				Window 6		
	t	x_1	x_2		t	x_1	x_2		t	x_1	x_2
10	1989	504014	40903	13	1992	562981	41120	16	1995	715547	43793
11	1990	551552	40682	14	1993	625715	41466	17	1996	754872	44566
12	1991	549641	41283	15	1994	661317	42564	18	1997	781542	45289
13	1992	562981	41120	16	1995	715547	43793	19	1998	839246	45902
14	1993	625715	41466	17	1996	754872	44566	20	1999	892574	46377
15	1994	661317	42564	18	1997	781542	45289	21	2000	910329	47097
16	1995	715547	43793	19	1998	839246	45902	22	2001	967522	46907
17	1996	754872	44566	20	1999	892574	46377	23	2002	1042786	47404

	Window 7				Window 8		
	t	x_1	x_2		t	x_1	x_2
19	1998	839246	45902	22	2001	967522	46907
20	1999	892574	46377	23	2002	1042786	47404
21	2000	910329	47097	24	2003	1042145	47598
22	2001	967522	46907	25	2004	1048231	48207
23	2002	1042786	47404	26	2005	1097044	48101
24	2003	1042145	47598	0	0	0	0
25	2004	1048231	48207	0	0	0	0
26	2005	1097044	48101	0	0	0	0

Table 6.2 – Absolute difference of variables x_1 (population) and x_2 (personal income) in each window using the first point as the center. For example, the absolute difference of $x_1(t_1)$ and $x_1(t_2)$ (i.e., $|x_1(t_1) - x_1(t_2)|$ is 4562 as highlighted here in blue and the absolute difference of point 2 from point 1 (i.e., 2-1) is 4562 and 511).

	Window 1			Window 2			Window 3			Window 4	
points	**x_1**	**x_2**	**points**	**x_1**	**x_2**	**points**	**x_1**	**x_2**	**points**	**x_1**	**x2**
2-1	4562	511	5-4	16632	378	8-7	5457	300	11-10	47538	221
3-1	8594	998	6-4	40345	257	9-7	17420	485	12-10	45627	380
4-1	73904	1643	7-4	47062	692	10-7	72696	99	13-10	58967	217
5-1	90536	2021	8-4	52519	992	11-7	120234	122	14-10	121701	563
6-1	114249	1900	9-4	64482	1177	12-7	118323	479	15-10	157303	1661
7-1	120966	2335	10-4	119758	791	13-7	131663	316	16-10	211533	2890
8-1	126423	2635	11-4	167296	570	14-7	194397	662	17-10	250858	3663

	Window 5			Window 6			Window 7	
points	**x_1**	**x_2**	**points**	**x_1**	**x_2**	**points**	**x_1**	**x_2**
14-13	62734	346	17-16	39325	773	20-19	53328	475
15-13	98336	1444	18-16	65995	1496	21-19	71083	1195
16-13	152566	2673	19-16	123699	2109	22-19	128276	1005
17-13	191891	3446	20-16	177027	2584	23-19	203540	1502
18-13	218561	4169	21-16	194782	3304	24-19	202899	1696
19-13	276265	4782	22-16	251975	3114	25-19	208985	2305
20-13	329593	5257	23-16	327239	3611	26-19	257798	2199

Table 6.3 – Amplitude (q(s)) for each window. These values were computed by first counting the number of points in each state within each window, computing a probability for each state in each window (p(s)) and calculating the amplitude, $q(s) = \sqrt{p(s)}$ for each state in each window.

Window		q(s)			
		1	**2**	**3**	**4**
	1	0.6124	0.7906		
	2	0.8660	0.5000		
	3	0.7071	0.7071		
	4	0.7071	0.5000	0.5000	
	5	0.5000	0.5000	0.6124	0.3536
	6	0.6124	0.6124	0.5000	
	7	0.6124	0.7071	0.3536	

Table 6.4 – Variables used to calculate FI for the San Luis Basin. These variables were selected from data used to compute EFA, GNRP, and EmA. The data and their sources are discussed in Chapters 3, 4, and 5, respectively. The variables include demographic, energy, land use, food and forest consumption, environmental and agricultural production categories for the system.

Category	**Variable**
Demographic	
	Personal income (thousands of dollars)
	Population (persons)
Energy	
	Coal consumption (short tons)
	Natural Gas Consumption (billion cubic feet)
	Petroleum Consumption (thousand barrels)
	Hydro-electric Consumption (million kilowatt hours; kWh)
	Wood and waste Consumption (trillion British thermal units; BTU)
	Solar Energy Absorbed (Joules per year; J/yr)
	Rain Chemical Potential (J/yr)
	Rain Geopotential on the Land (J/yr)
	Snow Geopotential on Land (J/yr)
	Rain Geopotential as Runoff (J/yr)
	Snow Geopotential as Runoff (J/yr)
Land type	
	Built-up land (hectares; ha)
	Arable land (ha)
	Pasture (ha)
	Forest including deforestation (ha)
Food and forest consumption	
	bovine, buffalo (pounds per capita; lbs/ca)
	sheep, goat (lbs/ca)
	non-bovine (lbs/ca)
	milk (gal/ca)
	cheese (lbs/ca)
	eggs (lbs/ca)
	fish (lbs/ca)
	cereals (lbs/ca)
	wheat (lbs/ca)
	vegetables (lbs/ca)
	maize (lbs/ca)
	fruit (lbs/ca)
	roots and tubers (lbs/ca)
	pulses (lbs/ca)
	coffee and tea (lbs/ca)
	cocoa (lbs/ca)
	oil seed (lbs/ca)
	fats (lbs/ca)
	sweetener consumption (lbs/ca)
	forest harvest (million board feet, MBF)

Category	Variable
Environmental	
	mean precipitation (centimeters; cm)
	Change in ground water storage (acre-feet)
	Change in surface water storage (acre-feet)
	SLB CO_2 emissions
	Evapotranspiration (J/yr)
	Soil Erosion (tons/year)
Food production	
	bovine, buffalo production (kilograms; kg)
	cereal production (kg)
	wheat production (kg)
	animal feed production (kg)
	roots and tubers production (kg)
	Potatoes All (Planted) (unit:acre)
	Wheat Other Spring (Planted) (unit:acre)
	Barley All (Planted) (unit:acre)
	Hay Alfalfa (Dry) (Harvested) (unit:acre)
	Oats (unit:acre)

Table 6.5 – Three-point mean Fisher information (<FI>) for the overall system and the variables grouped by the categories defined in Table 6.4.

Year	System	Consumption	Demographic	Energy	Environment	Land	Production
1986	3.22	3.34	5.02	4.59	3.86	4.82	4.56
1989	3.51	3.93	6.04	4.03	3.39	5.58	4.79
1992	3.64	3.66	4.14	4.63	4.61	6.32	5.99
1995	3.89	4.13	2.77	4.63	4.85	5.12	5.94
1998	3.88	4.48	2.81	3.82	4.32	6.00	6.02
2001	3.58	4.28	3.32	3.93	4.42	4.73	5.03

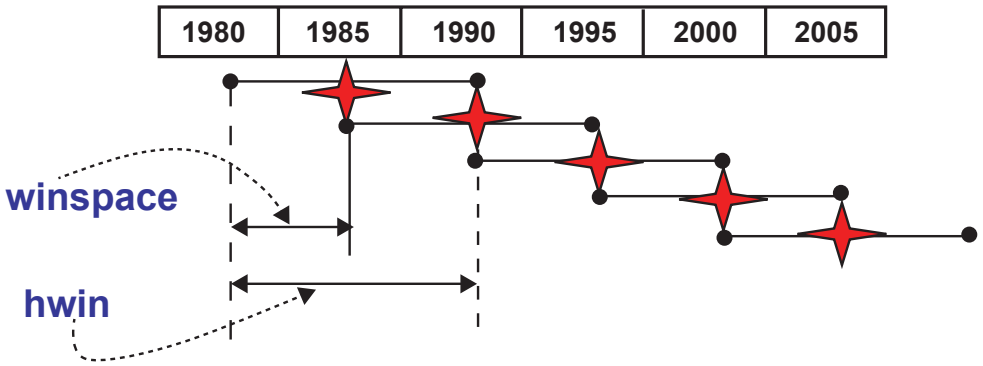

Figure 6.1 – Graphical representation of the sample settings for the Fisher information (FI) computation indicating the size of the integration window (*hwin*) and window increment (*winspace*) in time steps used to move through the data. In this example, *hwin* =10 and *winspace* = 5. The red star indicates the center point of the window denoting the placement of FI.

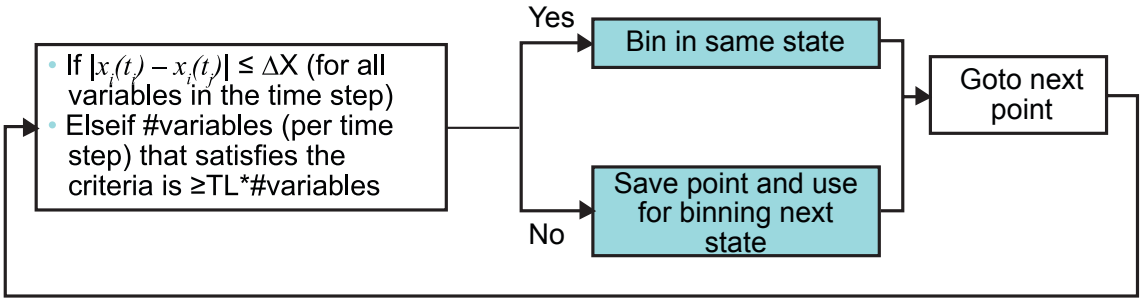

Figure 6.2 – Flow diagram of the binning algorithm in the Fisher information computation. According to Equation 6.4, if each variable $|x_i(t_i) - x_i(t_j)|$ is less than the *size of states* (Δx_i), then the two points at time *i* and *j* are indistinguishable and are grouped (binned) in the same state.

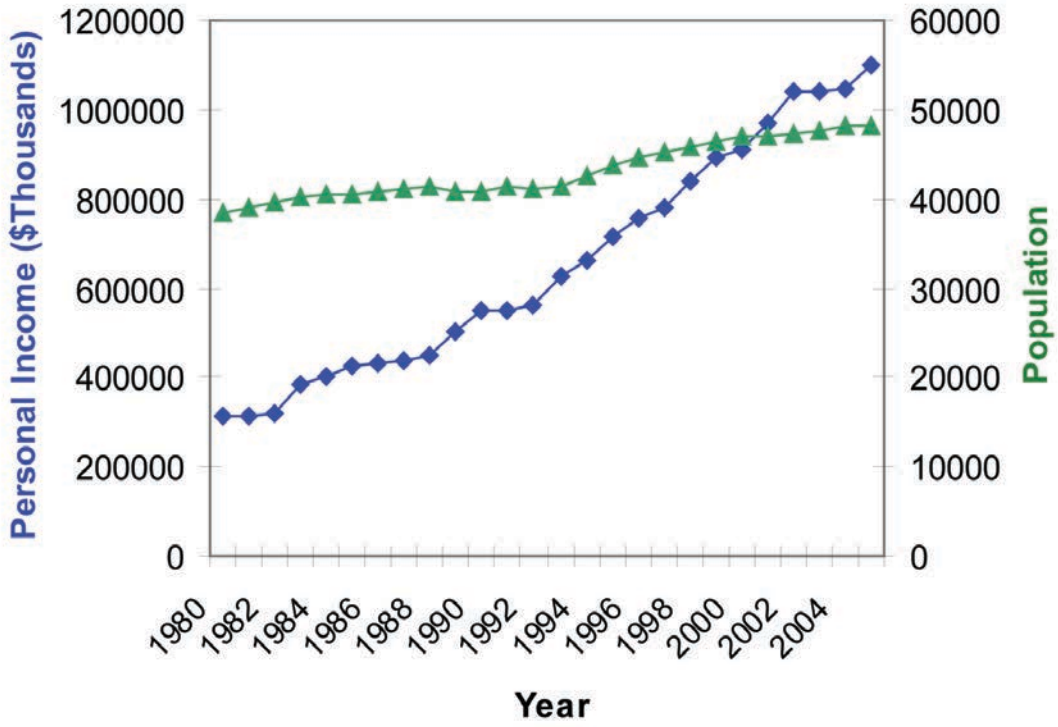

Figure 6.3 – Population (x_1, green triangles) and Personal Income (x_2, blue diamonds) values for the San Luis Basin from 1980 to 2005. These variables were used as data for the example FI computation exercise provided in Section 6.2.4.

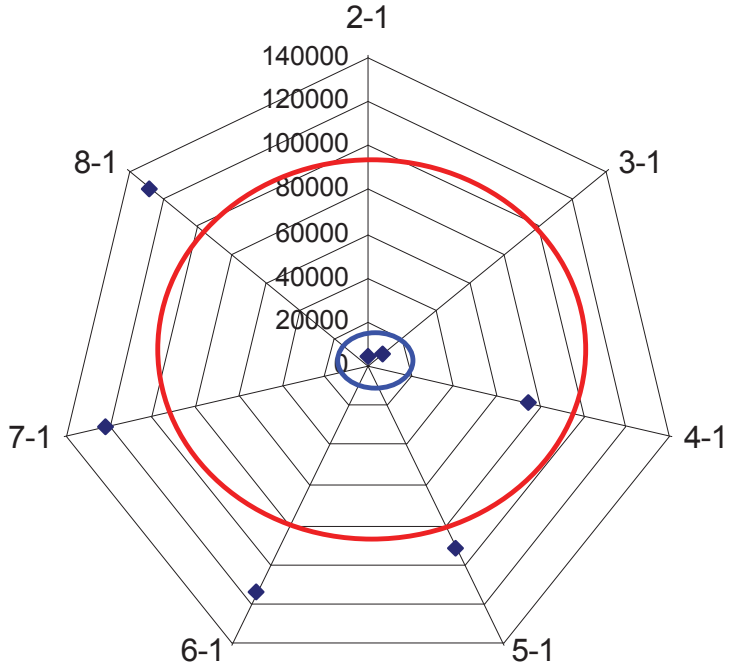

Figure 6.4 – Bulls-eye plot of binning points in state 1 for the simple example. Point 1 (pt_1: ($x_1(t_1)$, $x_2(t_1)$) = (310352, 38469)) is the center of this figure and the absolute difference from pt_1 are plotted on the corresponding axis, e.g., absolute difference 2-1 = (4562, 511). The values plotted are from Table 6.2. As described in Section 6.2.2, the level of uncertainty (i.e., *size of states* (Δx_i)) was computed for each variable such that in this plot Δx_1 and Δx_2 correspond to the red and blue line, respectively. The points are binned in a state when the difference is less than the level of uncertainty (i.e., *size of states*).

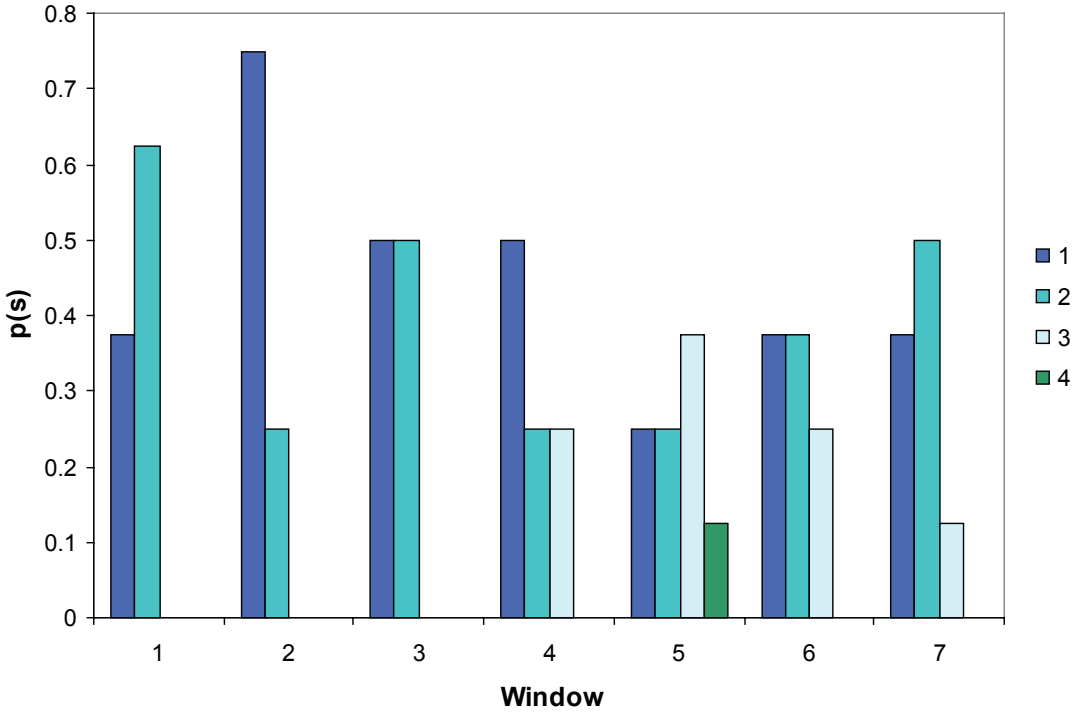

Figure 6.5 – Probability density (p(s)) for each window in the simple example (Section 6.2.4). These values were computed by first counting the number of points in each state within each window and then computing a probability for each state in each window. In window 3, four points were binned in both state 1 and state 2; therefore, the probability was 50% for each state.

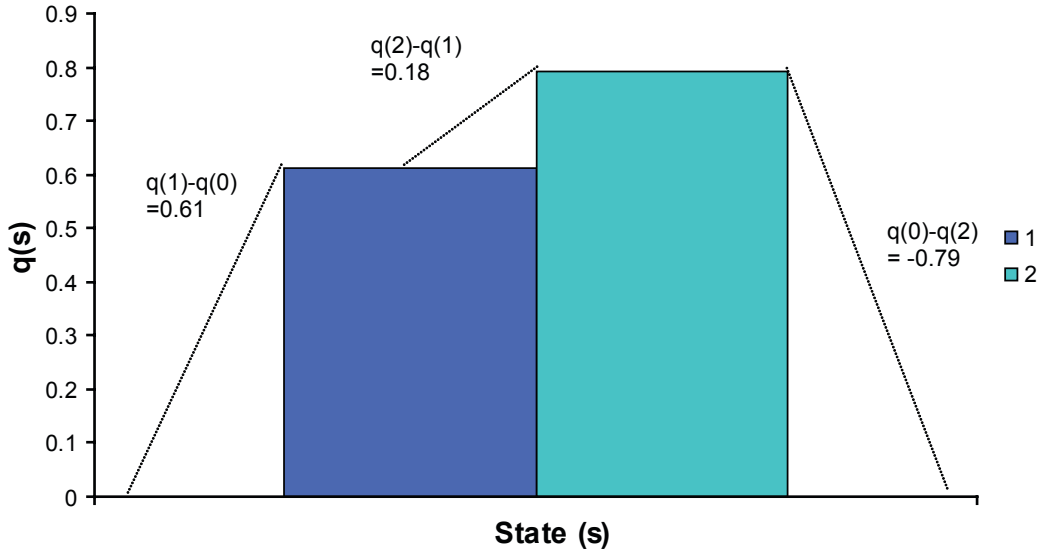

Figure 6.6 – Calculating gradients of the amplitude (q(s)) in window 1 of the simple example (Section 6.2.4). Gradients were used as a convention for mimicking a discrete density function and were computed as q_i-$q_{i+1,}$ where *i* is the state. These values used in Equation 6.3 to compute FI.

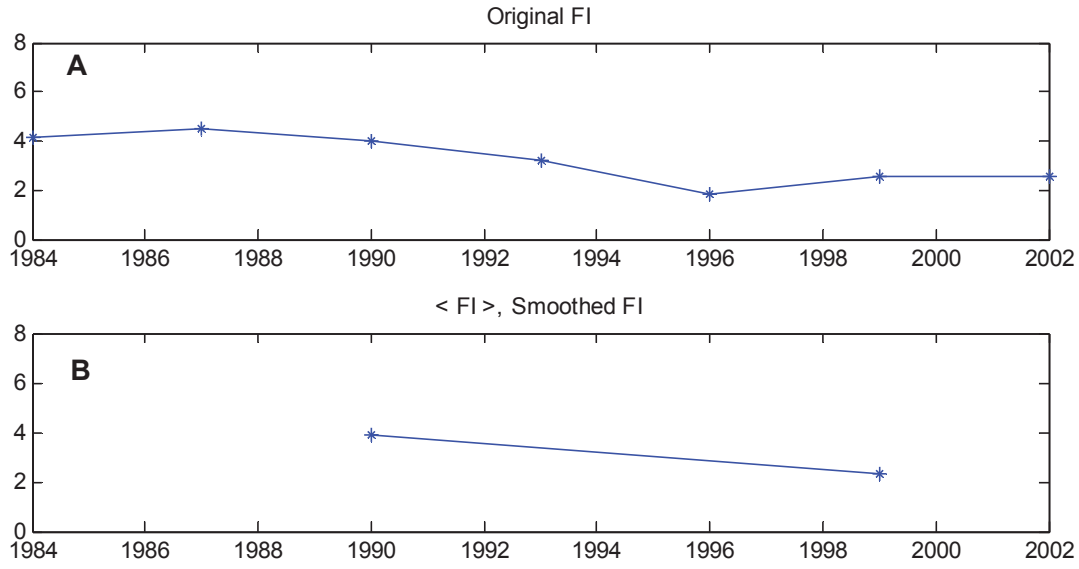

Figure 6.7 – Fisher information for each window from the simple example: (A) Fisher information (FI) and (B) Smoothed FI using three-point mean (<FI>), i.e., $<FI>_{1999} = (FI_{2002}+FI_{1999}+FI_{1996})/3$ and $<FI>_{1990} = (FI_{1993}+FI_{1990}+FI_{1987})/3$.

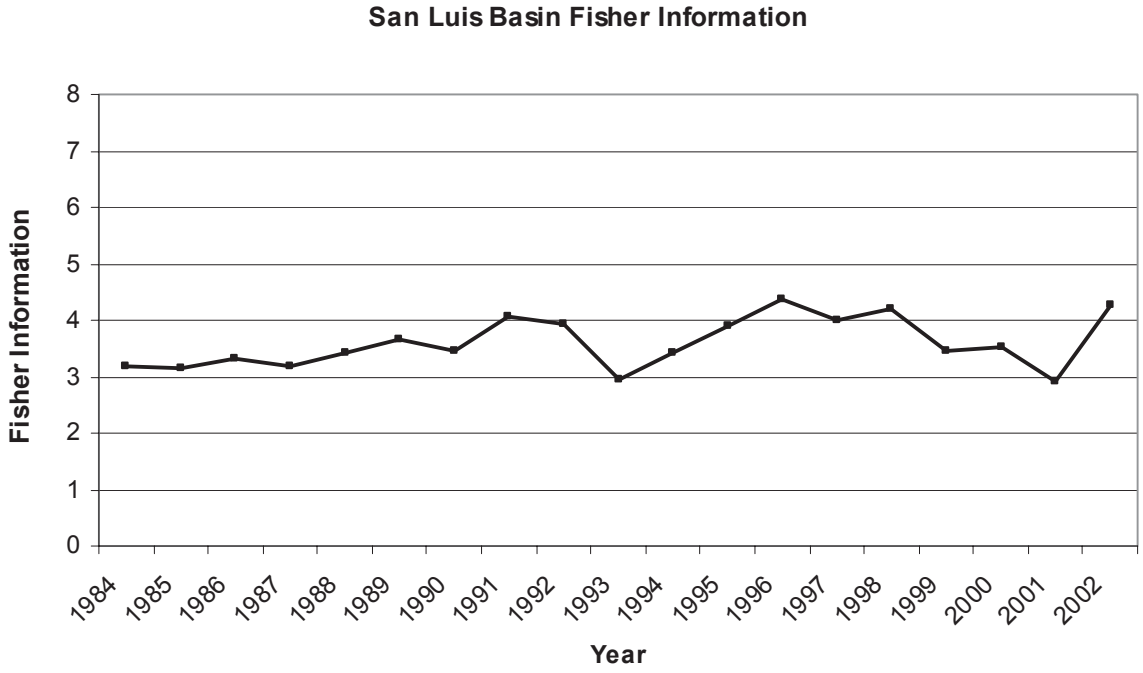

Figure 6.8 – Fisher information calculated for the San Luis Basin over a 26-year period (1980-2006). Each point represents the FI computed in each window and is reported in the center of that period (i.e., FI; 1984 = 1980-1987, 1985 = 1981-1988, 1986 = 1982-1989, etc.).

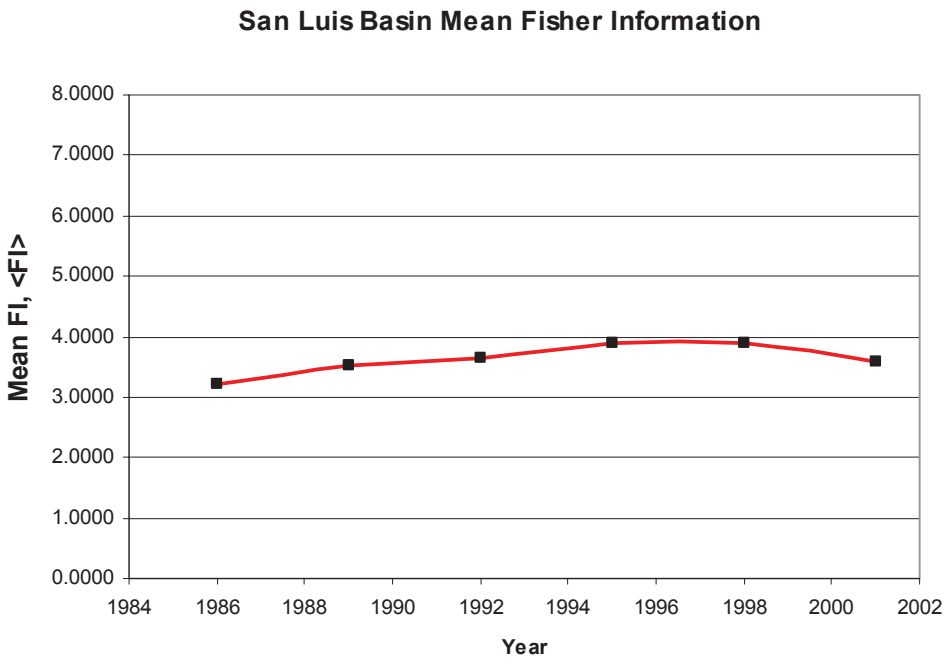

Figure 6.9 – Mean Fisher information (<FI>) for the San Luis Basin. <FI> was computed as a three-point average of the FI (Fig. 6.8) in order to smooth out short-term fluctuations and highlight trends, rather than fluctuations. The <FI> is reported in the center of the three points (i.e., <FI>; 2001 = 2002-2000, 1998 = 1999-1997, etc.).

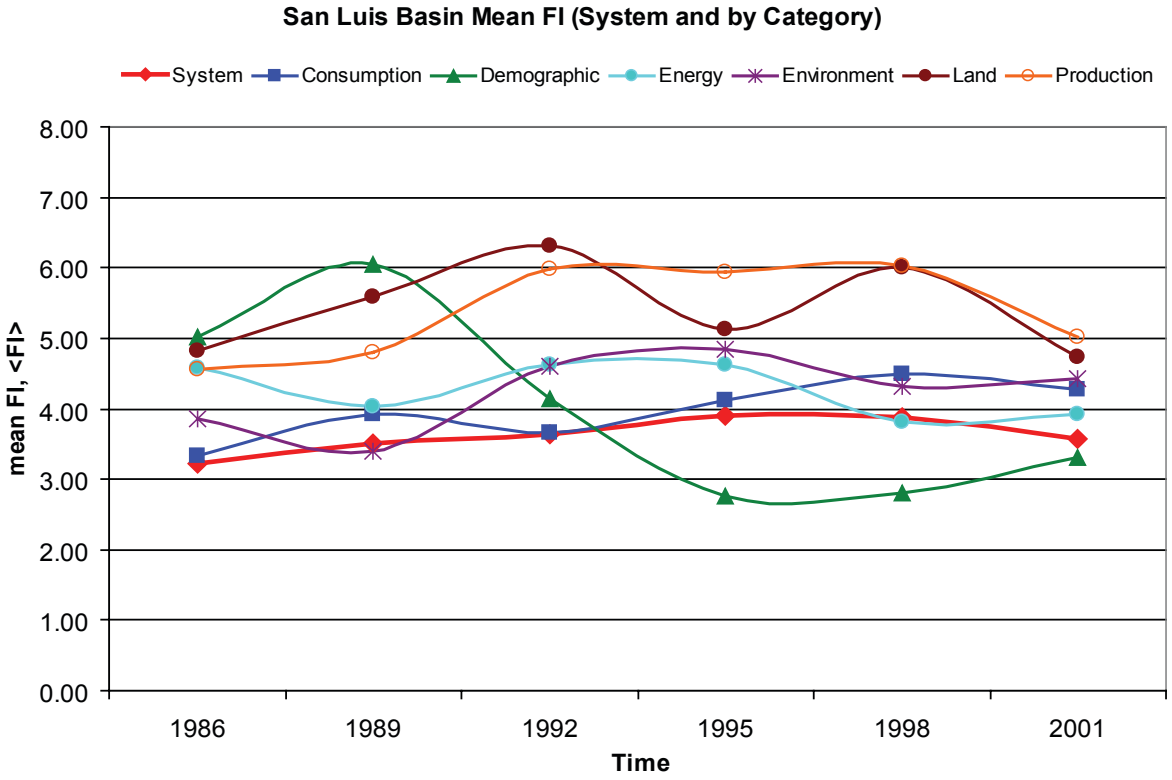

Figure 6.10 – Mean Fisher information of the San Luis Basin for the overall system and by category. Using this plot, we compared changes in system mean FI (<FI>) to the <FI> the variables grouped in categories. The <FI> is reported in the center of the three points (e.g., $\langle FI \rangle_{2001} = (FI_{2000} + FI_{2001} + FI_{2002})/3$).

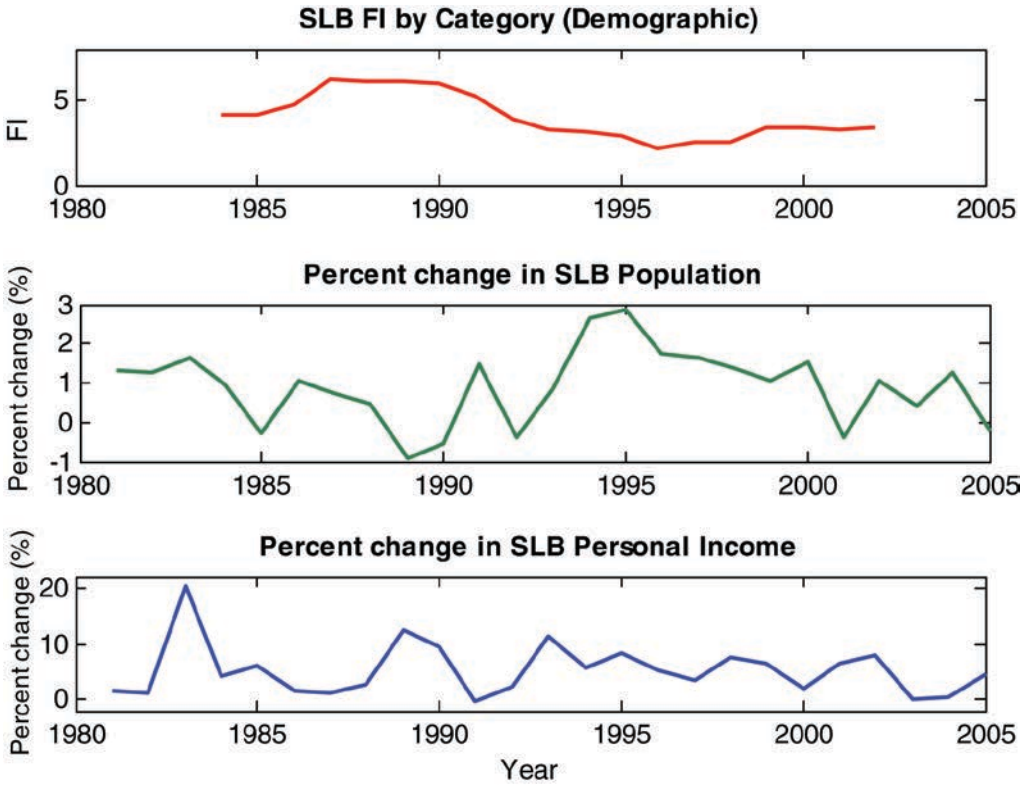

Figure 6.11 – Percent change in the San Luis Basin demographic variables. The percent change in population corresponded to the changes in dynamic order of the demographic category. The change in population decreased initially, peaked in 1995, and decreased thereafter which corresponds to the changes in dynamic order for the demographic category.

Appendix 6-A: Derivation Of Fisher Information

Fisher information was developed by the statistician Ronald Fisher as a statistical measure of information in data being used to fit a parameter (Fisher 1922). It is formally defined by Frieden (1998, 2004) as:

$$I(\theta) \equiv \int \frac{dy}{p(y|\theta)}\left[\frac{dp(y|\theta)}{d\theta}\right]^2 \quad (6\text{-A.1})$$

where, I is Fisher information and $p(y|\theta)$ is the probability density of observing a particular measured value of a variable y in the presence of a parameter θ. From this equation, I is a measure of the amount of information about θ that is obtainable from the measurement of y (we use I here to preserve the historical context; Fisher 1922, Frieden, 1998, 2004, Karunanithi et al. 2008, Mayer et al. 2007). For example, if y has no information about θ, then the derivative in Equation 6-A.1 is zero and I is zero as well. The parameter θ can represent many items. With that in mind, in order to transform this equation into a measure of order for complex dynamic systems, let the parameter θ be the mean $<y>$ of y over a particular period of time, T:

$$< y > \equiv \frac{1}{T}\int_0^T y(t)dt \quad (6\text{-A.2})$$

where T is the integration period for all observations of y. By substituting Equation 6-A.2 into (6-A.1), we have:

$$I(< y >) = \int \frac{dy}{p(y|< y >)}\left[\frac{dp(y|< y >)}{d < y >}\right]^2 \quad (6\text{-A.3})$$

Next, to represent the fluctuations in y around the mean $<y>$, we define a new variable s, such that $s \equiv y - <y>$. According to elementary statistics, systems where the variation around the mean is independent of the value of the mean are said to be shift-invariant (Frieden, 2004)

$$p(y|< y >) \equiv p(y - < y > |< y >) = p(s - < s >) = p_0(s) \quad (6.A.4a)$$

indicating that the probability distribution does not depend on the value of the mean. Using the results of Equation 6-A.4a and incorporating the chain rule (6-A.4b):

$$\frac{dp}{d < y >} = \frac{dp}{ds}\frac{ds}{d < y >} = \frac{dp}{ds} \quad (6\text{-A.4b})$$

where we have tacitly dropped the subscript "0" on $p(s)$, and Equation 6.A.1 becomes:

$$I = \int \frac{ds}{p(s)}\left[\frac{dp(s)}{ds}\right]^2 \quad (6\text{-A.5})$$

where $p(s)$ is the probability density finding a particular value of s. This is Equation 6.1 found in the main text of Chapter 6. Now let s be a state of the system in a Euclidean (linear) space defined by the observable variables of the system and time. That is, a particular state s is a region in a space, (i.e., linear phase space where the dimensions are the observable variables of the system and time). Then, $p(s)$ is the likelihood of observing a particular state, s.

From Equation 6-A.5, we note that I is proportional to dp / ds. In the context of order, systems can exist within two idealized extremes, perfect disorder and perfect order, as discussed in the main text within a different context. The perfect disorder case occurs when a system is unbiased toward any particular state. In other words, it has the same probability of being in one state as any other state of the system (s=1: n), i.e., p(s) = p(1) = p(2) =…p(n) and the probability density function (PDF) is flat (Fig. 6-A.1a) so that $dp / ds \rightarrow 0$. Accordingly, the system lacks order (which in some contexts can be thought of as predictability) and the resulting Fisher information approaches zero (i.e., $I \rightarrow 0$) (Fath et al. 2003). Perfect order occurs when repeated measurements of the system result in the same state over time. This more structured system has high order, is more predictable, and is biased toward a particular state or states. Accordingly, the PDF has a steep slope, $dp / ds \rightarrow \infty$ and Fisher information approaches infinity (i.e., $I \rightarrow \infty$) (Fig. 6-A.1b). However, real systems typically function between these two system extremes (e.g., Fig. 6-A.1c).

For completeness, we would like to point out that one reasonably elegant means of evaluating Equation 6-A.5 is by adapting a statistical mechanics approach and representing the system in its phase space. Here the space coordinates are again the measurable system variables, and the probability density for observing a particular state (s) is proportional to the time the system spends in state s, i.e., $p(s) \propto \Delta t(s)$. This method provides us with a resulting expression that is a function of the velocity ($R'(t)$) and acceleration ($R''(t)$) tangential to the system path in its phase space (Mayer et al. 2007). The expression used to compute the Fisher information under this theory is:

$$I = \frac{1}{T}\int_0^T \frac{(R''(t))^2}{(R'(t))^4}dt \quad (6\text{-A.6})$$

However, Equation 6.A-6 requires the evaluation of the first and second order derivatives tangential to the system path defined by the system variables (Cabezas et al. 2005, Fath et al. 2003, Mayer et al. 2006). Although this approach is appropriate for model systems for which smooth data are readily available, the challenge is that it

is extremely difficult to obtain high quality second order derivatives from the noisy, sparse data sets characteristic of real systems (Karunanithi et al. 2008). Accordingly, the numerical approach (derived and discussed in Chapter 6) was developed for calculating *I* to assess the dynamic order of real systems without the need to compute second derivatives. Further details of the analytical and numerical derivation of *I* can be found in Mayer et al. (2007) and Karunanithi et al. (2008).

References

Cabezas, H., Pawlowski, C.W., Mayer, A.L., Hoagland, N.T. 2005. Simulated experiments with complex sustainable systems: ecology and technology. Resources, Conservation and Recycling 44, 279-291.

Fath, B., Cabezas, H., Pawlowski C.W. 2003. Regime changes in ecological systems: an information theory approach. Journal of Theoretical Biology 222- 4, 517-530.

Fisher, R.A. 1922. On the mathematical foundations of theoretical statistics. Phil. Trans. Royal Society of London 222, 309-368.

Frieden, B.R. 1998. Physics from Fisher Information: A Unification, Cambridge University Press, Cambridge, UK.

Frieden, B. R. 2004. Science from Fisher Information: A Unification. Cambridge University Press, Cambridge, UK.

Karunanithi A.T., Cabezas, H., Frieden, B.R., Pawlowski, C.W. 2008. Detection and assessment of ecosystem regime shifts from Fisher information, Ecology & Society 13(1), 22.

Mayer, A.L., Pawlowski C.W., Cabezas, H. 2006. Fisher Information and dynamic regime changes in ecological systems. Ecological Modeling 195, 72-82.

Mayer, A.L., Pawlowski, C.W., Fath, B.D., Cabezas, H. 2007. Applications of Fisher information to the management of sustainable environmental systems, in:

Frieden, B.R., and Gatenby, R.A. (Eds.), Exploratory Data Analysis Using Fisher Information. London: Springer-Verlag, pp. 217-244.

Pawlowski, C.W., Cabezas, H. 2008. Identification of regime shifts in time series using neighborhood statistics. Ecological Complexity 5, 30-36

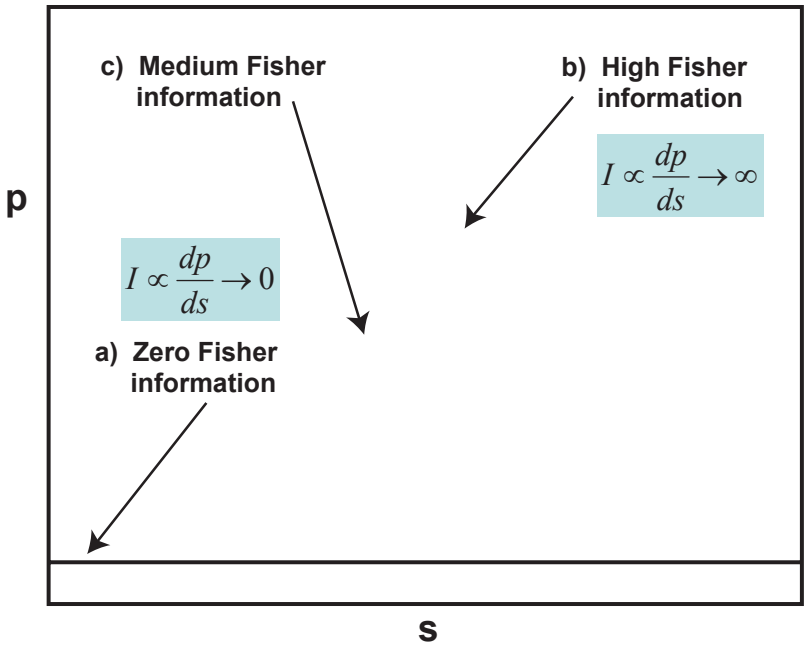

Figure 6-A.1 – Fisher information is proportional to the probability of system states (Pawlowski and Cabezas 2008). (a) Zero *I* results when a system has an equal probability of being in one state as any other state, resulting in a uniform probability density function (PDF), such that $dp / ds \to 0$ and $I \to 0$. (b) High *I* occurs when repeated measurements of the system result in the same state over time (i.e., the system is more predictable resulting in high predictability), a PDF with a steep slope, $dp / ds \to \infty$ and Fisher information approaching infinity (i.e., $I \to \infty$). (c) However, real systems typically function between these two extremes and exhibit medium *I*.

Appendix 6-B: Sensitivity Analysis On Integration Window Parameters

In section 6.2.4, we used a simple example to demonstrate the FI computation steps. The FI result was calculated using the integration window parameters selected in accordance with the basic criteria (section 6.2.2) and contained seven values (one for each time window) which were limited by the amount of data available, *hwin,* and *winspace* (Table 6.1). However, one of the challenges in computing FI is determining optimal settings for the integration window parameters. Accordingly, this appendix provides an exploration of the effect of changing the value of *hwin* and *winspace* on both the FI results and the resolution of results (number of FI results produced from a computation) for the data from the simple example. This exercise is not meant to cover every possibility; however, it is intended to (1) underscore the fact that care should be taken when selecting the integration window parameters and (2) provide key insights on the impact of the parameters on the FI result.

The basic guidelines for setting the integration window parameters are $hwin \geq 8$ and $winspace < hwin$ (section 6.2.2). For this exercise, we used the data from the simple example and examined four different values for *hwin* (ranging from 8 to 11 time steps) and *winspace* (ranging from 1 to 4 time steps). Graphing FI computed from these settings provide some sense of the effect changes in *winspace* and *hwin* on FI (Fig. 6-B.1). Each panel of Fig. 6-B.1 reflects the FI result with *hwin* constant and a different *winspace* for each computation. For example, panel A contains the FI results given *hwin* = 8 and *winspace* ranging from 1 to 4. The average value of FI (avgFI) and standard deviation (SD) of FI reported were computed from all the FI results in each panel as a summary statistic. For example, the avgFI and SD reported in panel A are the average value and standard deviation of FI given *hwin* = 8 and *winspace* from 1 to 4. In general, smaller *winspace* values resulted in more fluctuations in FI, yet, the overall impact of *winspace* on the value of FI was relatively minor. However, looking at avgFI for each panel, as *hwin* increased avgFI decreased. We performed similar analysis on the data by holding *winspace* constant and found that when assessing the variability of avgFI, the standard deviation is 0.11 as *winspace* changes and 0.44 as *hwin* changes. Therefore, it appears that the selection of *hwin* is the primary factor affecting FI. However, this only just begins to answer the question of the impact of the parameters on the FI result and provides minimal insight regarding optimal settings for *hwin* or *winspace* for this system.

One of the common approaches to assessing the impact of controllable (independent) variables on output responses includes evaluating the system one factor at a time (OFAT). Not only is this approach time consuming and costly, factor interactions are not considered (Anderson 2005, Montgomery 1997). In lieu of an OFAT approach, we opted to perform a sensitivity analysis by designing a factorial experiment using *hwin* and *winspace* as controllable factors and the average value of FI (avgFI) and resolution of FI results (Npts) as output response variables. A designed experiment affords the ability to determine critical factors by assessing the impact of controlled variable changes (and their interactions) on output responses. Using Design Expert 8 (StatEase, Inc., Minneapolis, MN) we created a design matrix for the 4^2 full factorial design showing the factor combinations and responses for each experiment (Table 6-B.1). An analysis of variance (at $\alpha = 0.05$) of avgFI revealed that both *hwin* ($F(3, 9) = 77.56$, $P \ll 0.05$) and *winspace* ($F(3, 9) = 4.64$, $P < 0.05$) are statistically significant factors (Table 6-B.2). However, note that the sum of squares (SS) and mean squares (MS) summarizes the amount of variability accounted for by each factor and random error (Montgomery 1997; NIST/SEMANTECH 2006). Accordingly, the results indicate that more of the variation in avgFI comes from changes in *hwin*. Similarly, we found that both factors have a statistically significant impact on the resolution of the FI results (number of FI results produced from a computation); however, *winspace* is dominant (Table 6-B.3). Further, there was a strong negative correlation ($r = -0.9934$, $P = 0.0066$, $df = 2$) between the avgFI and *hwin* (i.e., as *hwin* increases, avgFI decreases) and although *hwin* has little effect on the resolution of FI results, smaller *winspace* values produce a higher resolution of FI results (Figs. 6-B.2 and 6-B.3). From the analysis, it appears that *hwin* is the primary factor affecting the variability and amplitude of the avgFI result and *winspace* is the key parameter affecting the number of FI results produced from a computation (Npts). The challenge at this point is determining optimal settings for the integration window parameters such that the deviation in the avgFI result is minimized and the number of FI data points is maximized. In order to determine the optimal settings, we sought a solution to the multiple objective problem of (a) minimizing the standard deviation of FI, the standard error of the mean value of FI and standard error of the Npts and (b) maximizing Npts (Table 6-B.4). Using the optimization function within Design Expert 8 by StatEase, Inc., an optimal solution was found at *hwin* = 8 and *winspace* = 1 (Fig. 6-B.4). Using the solution from the sensitivity

analysis and numerical optimization, we calculated FI and the three-point mean FI result, <FI> (Fig. 6-B.5). These parameter settings were also used to compute FI for the SLB system.

References

Montgomery, D.C. 1997. Design and Analysis of Experiments. John Wiley and Sons, Inc., New York, NY.

NIST/SEMATECH. 2006. e-Handbook of Statistical Methods. Available online at http://www.itl.nist.gov/div898/handbook/toolaids/pff/1-eda.pdf. Last Accessed August 26, 2010.

Table 6-B.1 – Design matrix for the sensitivity analysis of the simple example data: Determining the impact of controllable factors, *hwin* and *winspace* on the response variables, average value of FI (avgFI) and the number of FI results (Npts) for computation.

	Factors		Response	
Run	***hwin***	***winspace***	**avgFI**	**Npts**
1	8	2	3.13511	10
2	9	2	2.74041	9
3	9	1	2.87719	18
4	11	1	2.22149	16
5	10	1	2.44636	17
6	11	2	2.13994	8
7	9	3	3.16116	6
8	8	1	3.18285	19
9	8	3	3.24848	7
10	10	2	2.3549	9
11	11	3	2.15498	6
12	9	4	2.90048	5
13	10	3	2.75425	6
14	10	4	2.44245	5
15	8	4	2.99323	5
16	11	4	2.12815	4

Table 6-B.2 – Analysis of Variance (ANOVA) for the average value of FI (avgFI). The results reveal that both *hwin* ($F(3, 9) = 77.56$, $P \ll 0.05$) and *winspace* ($F(3, 9) = 4.64$, $P < 0.05$) were significant factors affecting avgFI. Moreover, as an indicator of the amount of variability that is accounted for by factor effects (e.g., *hwin* or *winspace*) and random error (i.e., residual), the MS values reflect that more of the variability was determined by *hwin*.

Source	**Sum of Squares (SS)**	**df**	**Mean Square (MS)**	**F value**	**p-value Prob > F**	
Model	2.4199	6	0.4033	41.1021	< 0.0001	significant
A-hwin	2.2833	3	0.7611	77.5644	< 0.0001	significant
B-winspace	0.1366	3	0.0455	4.6397	0.0317	significant
Residual	0.0883	9	0.0098			
Cor Total	2.5082	15				

Table 6-B.3 – ANOVA for the resolution of Fisher information results (Npts). Like the ANOVA for avgFI, both *hwin* ($F(3,9) = 8.33$, $P < 0.05$) and *winspace* ($F(3,9) = 519$, $P << 0.05$) were statistically significant factors affecting Npts. However, the mean square (MS) values indicate that majority of the variability in Npts was due to *winspace*.

Source	Sum of Squares (SS)	df	Mean Square (MS)	F Value	p-value Prob > F	
Model	395.5	6	65.92	263.67	< 0.0001	significant
A-hwin	6.25	3	2.08	8.33	0.0058	significant
B-winspace	389.25	3	129.75	519	< 0.0001	significant
Residual	2.25	9	0.25			
Cor Total	397.75	15				

Table 6-B.4 – Criteria for determining the optimal value for *hwin* and *winspace* for the sample exercise. The goal of the numerical optimization was to minimize the deviation in avgFI and maximize Npts, subject to (a) minimizing the standard deviation of FI, the standard error of the mean value of FI and standard error of the Npts and (b) maximize Npts. The highest importance was placed on maximizing Npts and minimizing the standard error of avgFI and Npts. The optimization was performed in Design Expert 8 (StatEase, Inc.).

		Constraints		
Name	Goal	Lower Limit	Upper Limit	Importance
A:hwin	is in range	8	11	3
B:winspace	is in range	1	4	3
StdErr(mFI)	minimize	0.065521169	0.065521169	4
Npts	maximize	4	19	5
StdErr(Npts)	minimize	0.330718914	0.330718914	4
std mFI	minimize	0.421906393	1.178581952	3

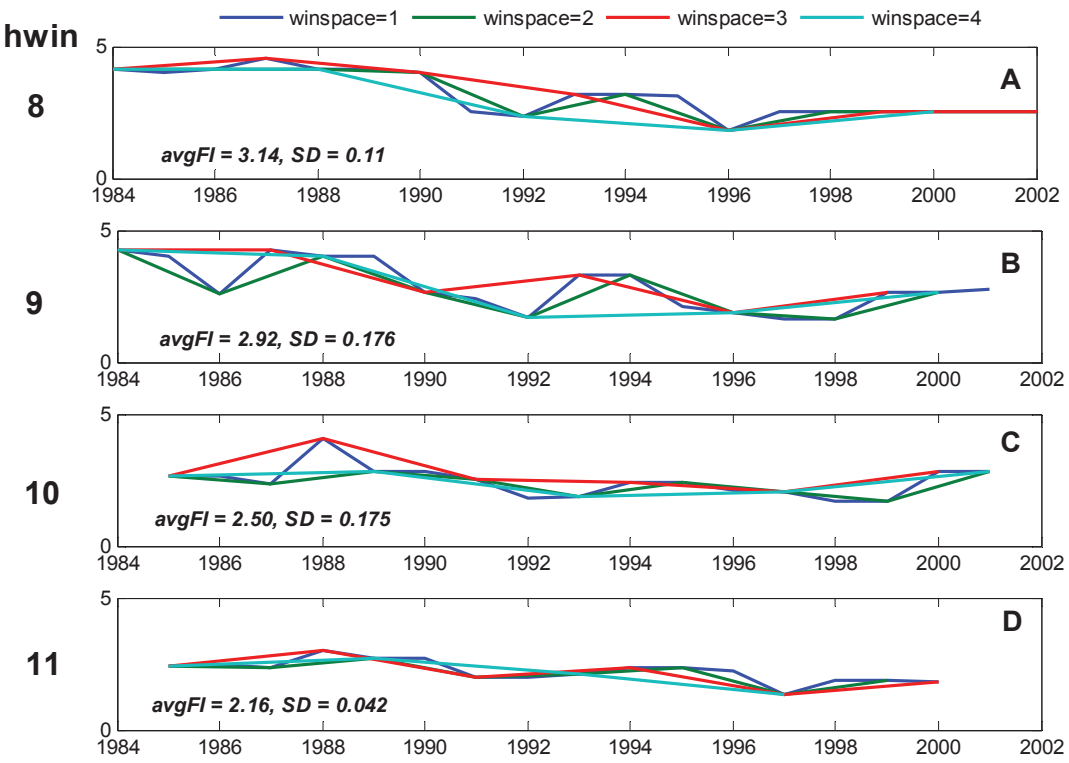

Figure 6-B.1 – The effect of changing *hwin* on the FI result. The value of *hwin* changes from panel to panel (A-D), while *winspace* varies within each panel. Therefore, each panel reflects the Fisher information result over time with *hwin* constant and a different *winspace* for each computation: (A) *hwin* =8, *winspace* =1 to 4, (B) *hwin* =9, *winspace* =1 to 4, (C) *hwin* =10, *winspace* =1 to 4, (D) *hwin* =10, *winspace* =1 to 4. The average value (avgFI) and standard deviation (SD) reported of FI in each panel were computed from all the FI results in each panel as a summary statistic. For example, avgFI and SD in panel A are the mean and standard deviation of the FI values given *hwin* = 8 and *winspace* = 1 to 4. Note that avgFI decreases as *hwin* increases and small *winspace* values result in more fluctuations (e.g., *winspace* =1 is represented by the blue line).

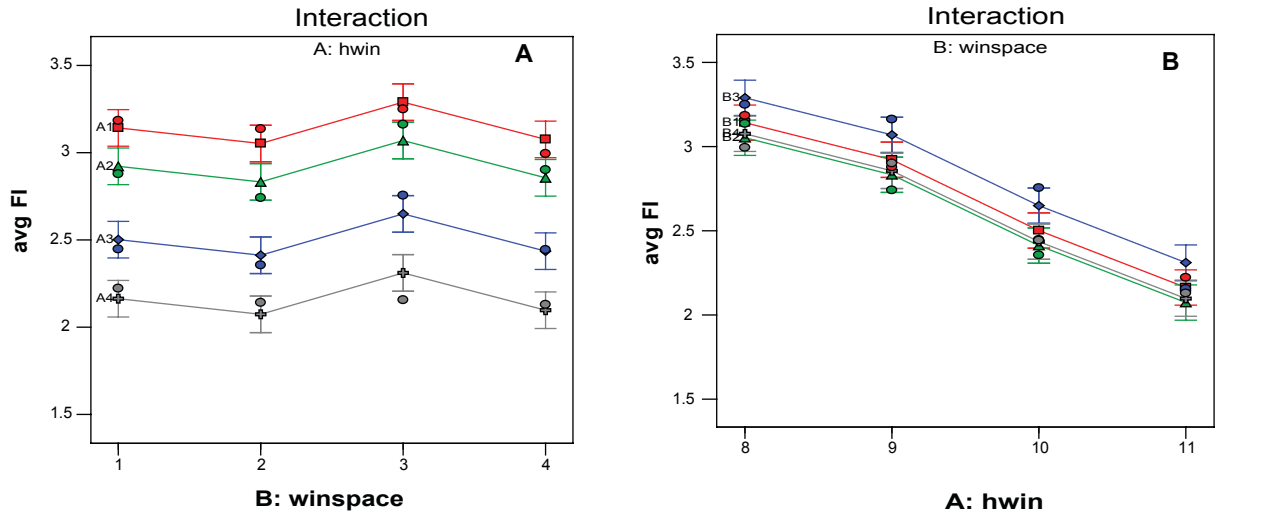

Figure 6-B.2 – Interaction plots for the average value of FI (avgFI). Plotting the value of avgFI as a function of *hwin* and *winspace* affords the ability to evaluate the impact of both parameters on avgFI. (A) Each *hwin* value is represented by a different color plot. With *winspace* varying along the x-axis, there is little variability in avgFI as *winspace* increases. Conversely, note that as *hwin* increases (e.g., A1: *hwin* = 8, A4: *hwin* = 11), the value of avgFI decreases. (B) The strong relationship between *hwin* and avgFI is also shown as *hwin* increases along the x-axis.

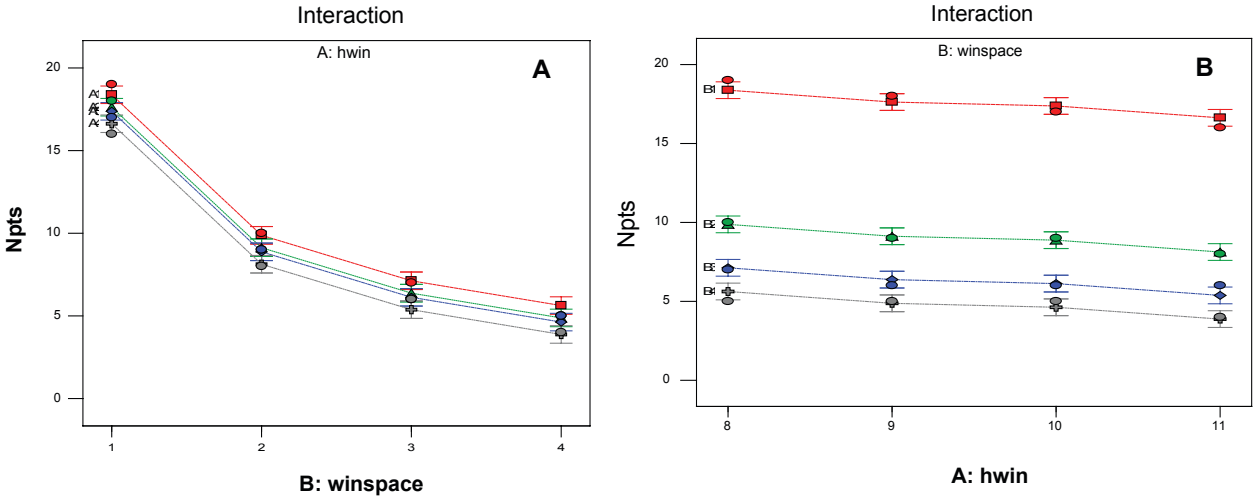

Figure 6-B.3 – Interaction plots for the number of FI results (Npts). Plotting the value of Npts as a function of *hwin* and *winspace* affords the ability to evaluate the impact of both parameters on Npts. (A) Each computation with a particular *hwin* value is represented by a different color plot. With *winspace* varying along the x-axis, there was a great deal of variability in Npts. (B) However, as *hwin* increases along the x-axis, there are minor changes in Npts. Further, there is a greater decrease in Npts as *winspace* goes from 1 to 2, than from 2 to 3 or 3 to 4.

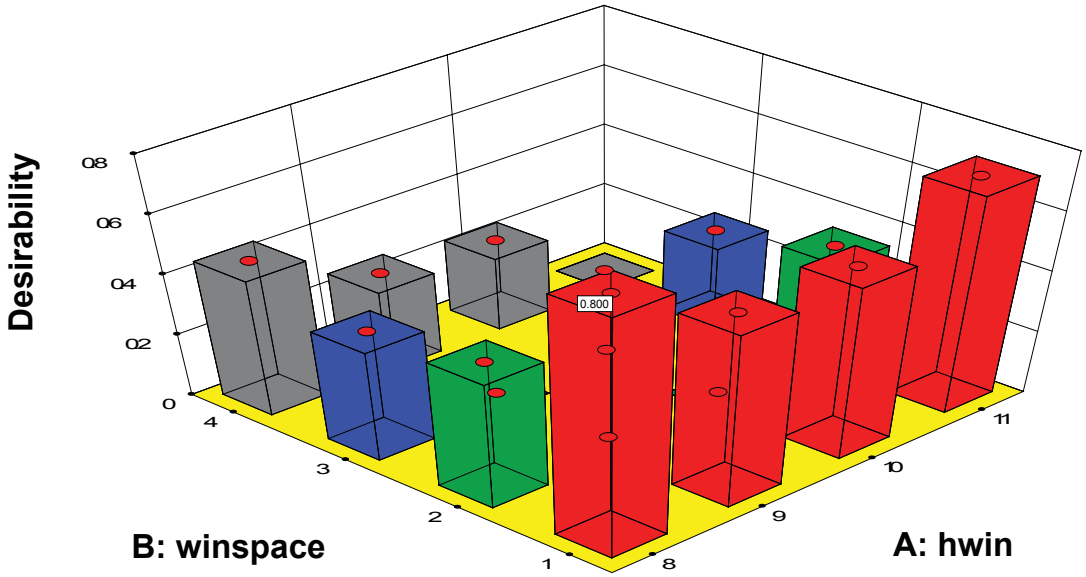

Figure 6-B.4 – Results of the numerical optimization of *hwin* and *winspace* using data from the simple example. Based on the optimization criteria of maximizing Npts and minimizing the deviation in avgFI, the optimal solution was found (using Design Expert 8) with integration window parameter settings of *hwin* = 8 and *winspace* = 1.

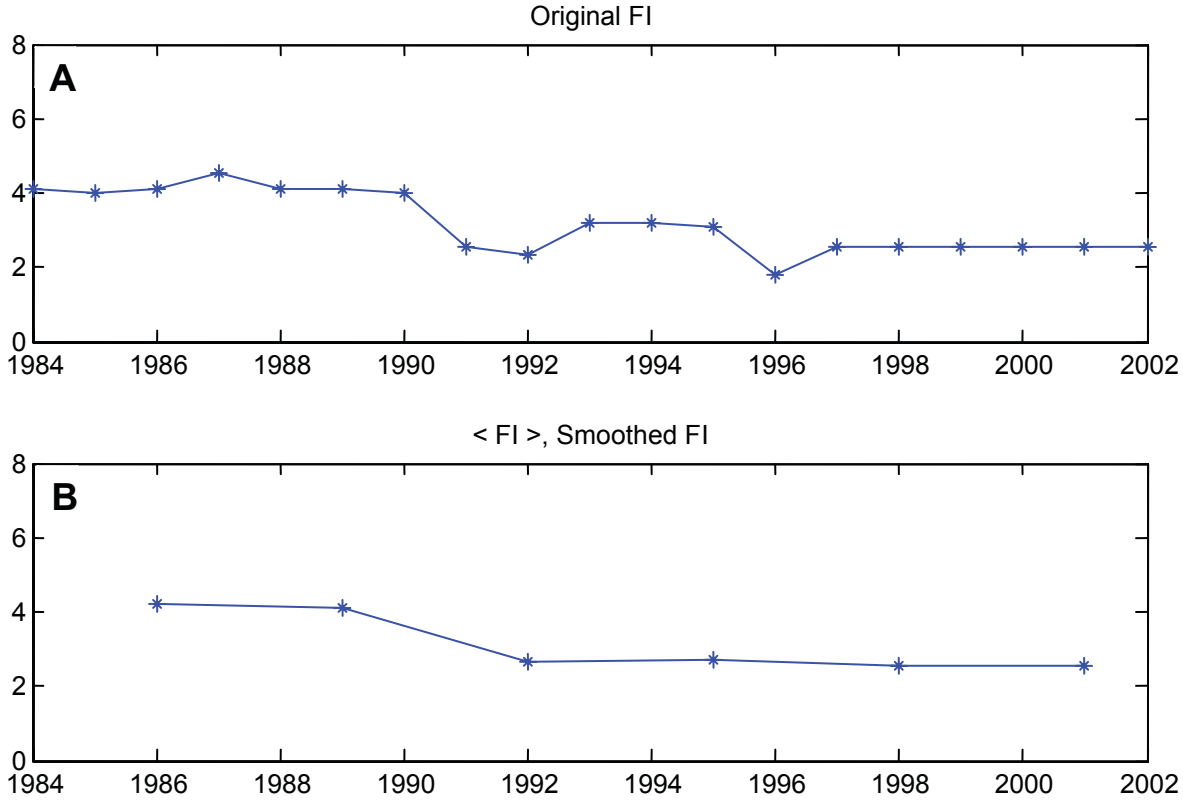

Figure 6-B.5 – Final Fisher information (FI) result from the simple example. (A) FI result from the optimal parameter settings (*hwin* = 8 and *winspace* =1). (B) Smoothed result: <FI>, three-point mean Fisher information.

Appendix 6-C: San Luis Basin Fisher Code User Manual – Utilizing MATLAB Files (Gfisher.m and CalcFishMean.m) to Calculate Fisher Information

Appendix 6-C.1: Using MATLAB Files to Calculate Fisher Information via Command Line: (GFisher.m, CalcAvgFish.m, and SmoothFI.m)

The primary MATLAB files developed for this project include GFisher m, CalcAvgFish m and Main_Fisher_data_proc.m (GUI) and AdvancedFishGUI.m (GUI). Other supporting files are Nfisherpdf.m, round2 m, CloseGUI m and SmoothFI.m. In this section, we present a general description for using MATLAB to execute the functions (e.g., GFisher m, CalcAvgFish.m) via command line to compute FI. Using the functions in this way requires some knowledge of MATLAB; however, this manual is not intended to be a MATLAB tutorial. Conversely, the GUIs (i.e., Main_Fisher_data_proc m and Advanced FishGUI m) are not command line functions and are the most user-friendly means of computing FI. They will be described in Appendix 6-C.2:

The file GFisher m contains the code developed in conjunction with the ongoing research within the US EPA's Sustainable Environments Branch for implementing the numerical approach to computing FI. Given that m=number of time steps (e.g., years) and n=number of characteristic variables, the command line for this code is:

[FI,midt_win] = GFisher (t,data,sost,hwin,winspace,TL,IP) where the required inputs into the function are:

t:	the time data for study of size m × 1
data:	the time series data of variables characterizing the state of the system of size m × n
sost:	a row vector of the *size of states* for each variable, size = 1× n
hwin:	the size of the time window
winspace:	the moving window increment
TL:	the tightening level (%)
IP:	a toggle on or off (1 or 0) for requesting an automatic plot of the FI result

The function outputs are:

FI:	Fisher information for each time window
midt_win:	Midpoint of each time window corresponding to the FI result

Prior to executing the code, the user must gather the input data as described above, to include: time series data of the variables that characterize the state of the system (data), the time period (e.g., 1980-2005) that the data represent time (t), the integration window parameters (*hwin* and *winspace*) and the *size of states* (sost) for each variable. Please refer to sections 6.2.2 and 6.2.4 for information on determining and gathering this information.

Once the data are compiled, they may be entered into MATLAB directly via command line or imported from a saved file of a supported type (e.g., csv, Excel). The data entry method is not critical when executing the command line functions; however, data must be in a particular format:

- The time series data of the variables characterizing the system (data) should be configured such that data for each variable are listed in one column with each row containing the variable values for a particular time step, producing a m× n matrix. For example, if there are 5 variables and 20 time steps. The data will be entered into a 20 × 5 matrix.
- The time data (time, t) should be formatted such that one time step value is entered in each row producing a m ×1 column vector with each time period corresponding to a datum point.
- The *size of states* (sost) data should be formatted such that there is a *size of states* value for each variable in each column, producing a 1 × n row vector (see 6.2.2 for computing *size of states*). For example, the 20 × 5 matrix mentioned above contains 5 variables and would have 5 *size of states* values, one for each variable (i.e., 1 × 5 row vector).

All of these inputs must be loaded into the workspace prior to executing the function. In other words, properly formatted data must be loaded into the workspace and parameter values must be defined in the MATLAB command window before entering the command.

While GFisher.m provides the FI result for one specified tightening level (e.g., 100% = 100), CalcAvgFish m calls GFisher.m to compute FI by incrementally changing the tightening level and calculating the average of the FI from the strict (TL = 100%) to relaxed tightening (see Section 6.2.2). The command line for this file is:

[AvgFish,PO_TL,mY]= CalcAvgFish (t,data,hwin,winspace,sost)

The function inputs are similar to GFisher and outputs are:

AvgFish: Average Fisher for each time window

PO_TL: Tightening Level the system is in perfect order. When there is a regime shift, it is an indication of pervasiveness

mY: Mean year for each time window

The SmoothFI.m file is used to compute the mean FI <FI>, a *m* point mean which smoothes the FI results to focus the analysis on trends and not fluctuations. The command line is: [FIm]=SmoothFI(m,t,FI), where *m* is the number of points to be used in calculating the average, t is the time vector and FI is the computed FI result from GFisher.m, CalcAvgFish m or Main_Fisher_data_proc.m GUI. The output is 2 columns containing the mean time and mean FI computed. In the SLB project, *m* was set to a default of three in order to provide a three-point mean FI.

Appendix 6-C.2: Graphical User Interfaces (GUI) for Calculating Fisher Information: (Main_Fisher_data_proc.m and AdvFishGUI.m)

To simplify the calculation of FI and enhance user-friendliness, a GUI was developed as an extension to the *San Luis Basin Sustainability Metrics Project* (Fig. 6-C.1). The GUI is an interface for data entry, executing code to compute FI, as well as, displaying and saving results from an FI analysis (Fig. 6-C.1). It is designed to simplify the procedure by allowing the user to select pre-formatted data files, alter pre-set parameter values and process data to calculate FI over numerous tightening levels (from strict to relaxed tightening). The computed FI is plotted in the GUI and the user is able to save the FI computations and plot for further use.

Although MATLAB can import multiple file types, the code for the GUI was setup to accept data specifically in Excel format. Accordingly, although the data files are created using the data formatting guidelines (see Appendix 6-C.1), there are a few caveats:

- The Excel file containing the variable time series data should be formatted as an m×n matrix with a descriptive name that ends in *data.xls* (e.g., *SLBdata.xls*).
- The time data should be saved in a separate Excel file whose descriptive name ends in time.xls (e.g., *SLBtime.xls*) and contains the m ×1 time data, where m is the number of time steps
- The *size of states* data should be saved in a separate Excel file whose descriptive name ends in *sost.xls* (e.g., *SLBsost.xls*) and contains the data in a 1× n row vector, where n is the number of variables.
- There should be no column headings in these files.

Navigating through the GUI, the [Add/Remove Files] menu tab contains controls that allow the user to select the data files for computing FI. The [Add Files] button is used to identify the location and add the pre-formatted files. Note that because there is one evaluation done at a time, only one time series, one *size of states,* and one time series data file (e.g., *SLBdata.xls*, *SLBtime.xls*, *SLBsost.xls*) should be loaded at a time. The [Delete Files] button may be used to remove unnecessary files. Once the data, time, and *size of state* files have been selected, the user inputs the *hwin* and *winspace* parameters, as well as, the number of points for the smoothing the FI results. The values for these parameters are preset to 8, 1, and 3 (respectively) for this project. The [START] button executes the function to perform the FI computation. Once this button is selected, all buttons and figures are disabled until the computation is complete and then a graph of the result (FI and smoothed FI (<FI>)) is displayed in the Plot Window. The [Save Plot] and [Save Calculations] buttons are used to save the FI plot and numerical result for further use. The [Save Calculations] button saves the FI values, date and time of calculation, as well as the smoothed mean FI results into an Excel file in the following format:

Row1 Col1: Date

Row2 Col1: Time (military)

Row3 Col1: Perfect order tightening level: PO_TL

Row4 Col2: Average FI results: [Year FI]

The smoothed <FI> results follow the average FI results

The [Reset] button allows the user to clear the figure and parameter settings and the [Help] button hyperlinks to a help file, which provides instructions on using the Main GUI. The [Advanced] button opens the Advanced Fisher GUI, which allows the user to assess the pervasiveness, and intensity of a regime shift, if one exists (see Section 6.2.2 and Fig. 6-C.2). Such an examination is pertinent only if a regime shift has been identified.

Given the same data and parameter settings that were established in the Main GUI (Main_Fisher_data_proc.m), clicking [Study] in the Advanced Fisher GUI executes the computation of the FI for multiple tightening levels and plots the result on two axes (axis1: TL=100%-60% and axis2: 50%-10%) along with the mean FI within the GUI interface. Through these plots, the user is able to study the pervasiveness and intensity of the regime shift as described in Section 6.2.2 and Karunanithi et al. (2008). Pervasiveness indicates the number of variables affected by the shift and is estimated by determining the lowest level in which the drop in FI is still present. This is based on finding the TL value at

which the system is perfectly ordered (i.e., FI = 8) for the entire period. The pervasiveness is calculated by subtracting the perfect order tightening level (POTL) from 100 and denotes the percentage of variables that are impacted by the shift. The intensity relates to the amplitude of the shift and is determined by how far the FI drops between two stable regimes. A system with a more intense shift would have a lower drop in FI before settling into a new dynamic regime.

The main benefit of the graphical interface is that it requires minimal user input and simplifies the evaluation process for FI computation. In conjunction with this project, MATLAB files for the GUIs were made into executable files using the MATLAB Compiler. This provides portability to locations where the MATLAB software is not readily available; thereby making the GUIs available for use on any desktop. However, these executable files are not modifiable.

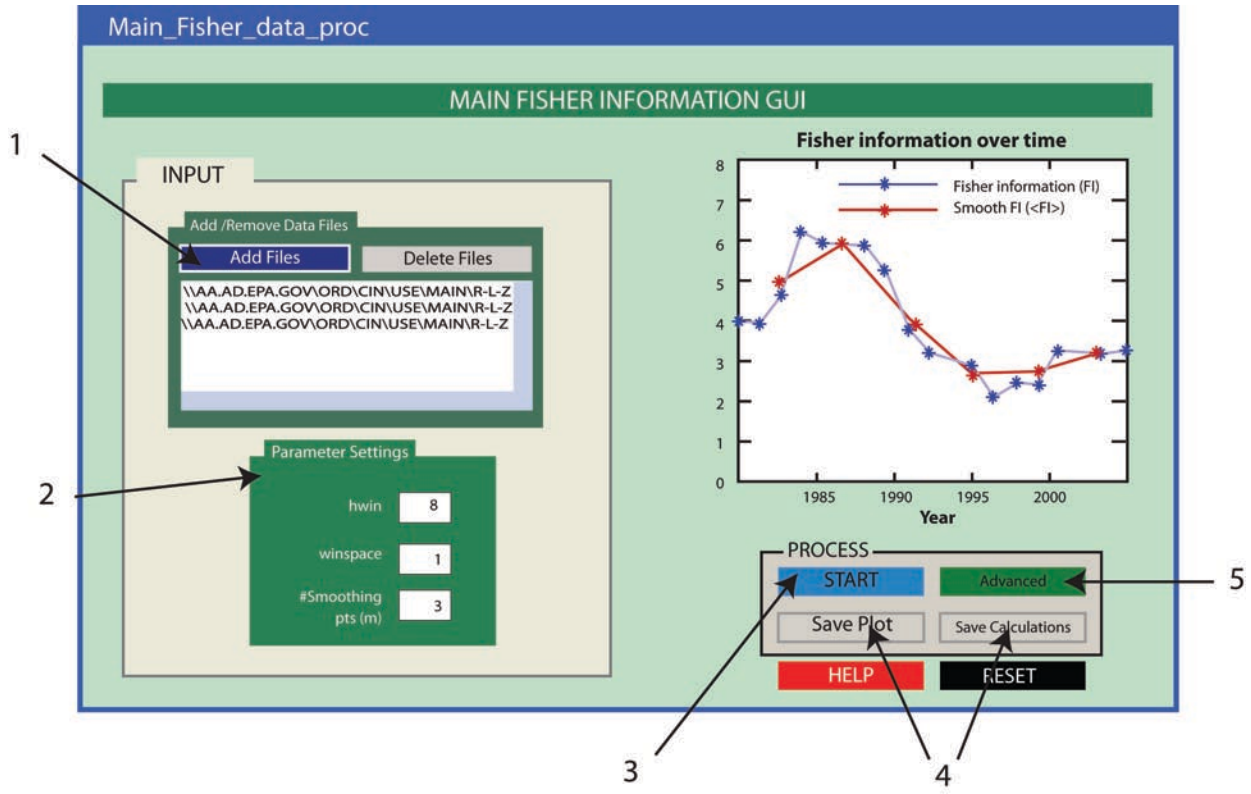

Figure 6-C.1 – GUI interface developed to compute FI from system data. The GUI is used for (1) importing pre-formatted data files, (2) choosing parameter settings (3) computing and plotting FI, (4) saving the FI results and (5) further study of regime shifts. The results plotted in this figure are from the simple example in Appendix 6-D.

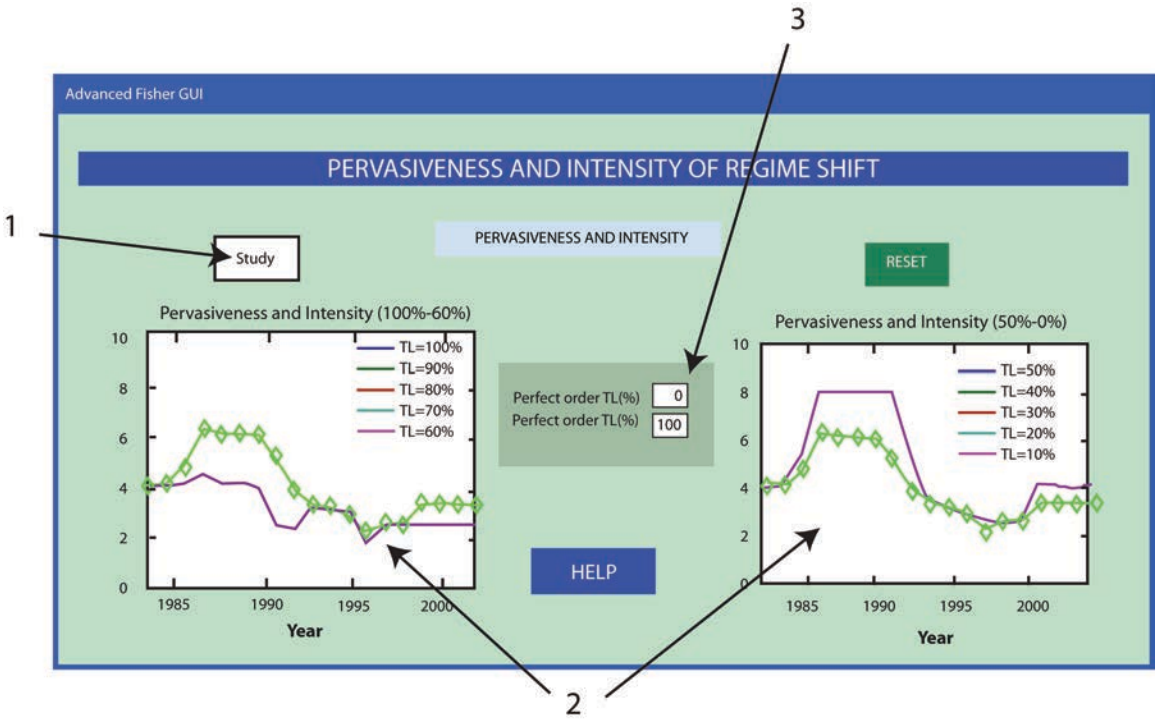

Figure 6-C.2 – The Advanced GUI was developed to study the pervasiveness and intensity of ecosystem regime shifts by (1) evaluating and (2) plotting the FI at various tightening levels. (3) The perfect order tightening level and pervasiveness are computed and shown in the text boxes. The results plotted are from the simple example as computed in Appendix 6-D. Using this simple example, note that the pervasiveness is 100% and there is one regime shift between two stable regimes.

Appendix 6-D: Using MATLAB to Compute FI for the Simple Example

In this section, we describe the steps involved in using the MATLAB GUI to analyze the data from the simple example. The first step was to save the time series, time, and *size of states* data into separate Excel files. In accordance with the guidelines in Appendix 6-C.2, the time series of the variables characterizing the system and the time data (i.e., 1980 to 2005) plotted in Fig. 6.3 were saved as *SLBExdata.xls* and *SLBExtime.xls* (Tables 6-D.1 and 6-D.2). The *size of states* for each variable as computed in section 6.2.4 (i.e., [86285.39 1639.83]) were saved in *SLBExsost.xls* (Table 6-D.3). Once the data files were set up, we opened the Main_data_proc GUI and clicked the [Add files] button to select the formatted data files (*SLBExtime.xls*, *SLBExdata.xls* and *SLBExsost.xls*) for computing FI. We left the default settings for the integration window parameters, *hwin* = 8, *winspace* = 1 and the number of smoothing points = 3 and pressed [START] to process the data and generate a plot of the FI and smoothed FI (<FI>) results (see Fig. 6-D.1). After the data were processed, the plot and FI calculations were saved using the [Save Plot] and [Save Calculations] buttons.

Although, the data used in this example do not fully characterize the sustainability aspects of the region (contains only two demographic variables), for the sake of this exercise, we assumed that these variables characterize the system under study. Looking at Fig. 6-C.1, we noted a drop in FI "bookended" by two stable regimes indicating the existence of a regime shift. Thus, in order to study the pervasiveness and intensity of the shift, we clicked the [Advanced] button to open the AdvancedFish GUI (Fig. 6-C.2). After clicking the [Study] button, the FI (green plot), as well as FI at various tightening levels from strict (TL = 100%) to completely relaxed tightening (TL=0%) are computed and plotted. In addition, the perfect order tightening level (POTL) and pervasiveness for the system were shown in the textboxes. Typically, the POTL may be observed graphically by noting the lowest tightening level at which the features of the shift (drop in FI) are still present. Note that there are only a few plots visible because there are only two variables in this study. At any rate, because the features of the shift remain intact for the plots, the POTL = 0 and pervasiveness = 100%, both variables are impacted by the shift.

Table 6-D.1 – Excel file containing the time series of the variables characterizing the system for the simple example (*SLBExdata.xls*). In this exercise, there are two variables (x_1 and x_2) such that each column contains the time series data for one variable. As described in Appendix 6-C.1, the data file is displayed as an m × n matrix with m = the number of time steps and n = the number of variables.

x_1	x_2
310352	38469
314914	38980
318946	39467
384256	40112
400888	40490
424601	40369
431318	40804
436775	41104
448738	41289
504014	40903
551552	40682
549641	41283
562981	41120
625715	41466
661317	42564
715547	43793
754872	44566
781542	45289
839246	45902
892574	46377
910329	47097
967522	46907
1042786	47404
1042145	47598
1048231	48207
1097044	48101

Table 6-D.2 – Excel file containing time data for simple example (*SLBExtime.xls*). In this exercise, the time period was from 1980 to 2005 and represents 26 time steps. Accordingly, as described in Appendix 6-C.1, the time file is displayed as an m × 1 matrix with m = the number of time steps.

Year
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005

Table 6-D.3 – Excel file containing the *size of state* (sost) data for simple example (*SLBExsost.xls*). In this exercise, there were two variables (x_1 and x_2) and the *size of states* was computed for each variable (Δx_1 and Δx_2). The values are exhibited as a 1 × n matrix with n = the number of variables.

x_1	x_2
86285.39	1639.83

Appendix 6-E: Fisher Information MATLAB Code

Code is available for download in the zipped file named "Appendix 6E - Fisher Information code.zip"

Gfisher.m

```
function [FI,midt_win,t_win]=GFisher(t,data,sost,hwin,winspace,TL,IP) %used with GUI
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%This program calculates FI as a measure of dynamic order.
%It is one of the products of an on-going effort within the US
%Environmental Protection Agency's Sustainable
%Environments Branch under the direction of Heriberto Cabezas.
%Key contributors to this effort include:
%Theory, methodology and direction from Heriberto Cabezas as adapted from
%Ronald Fisher's work in theoretical statistics(1922); Chris Palowski for
%the initial coding and framework: recurnumFI4 m (6/17/2004).
%Methodology enhancements and corresponding source code
%implementation by Arunprakash Karunanithi (3/13/2006) and additional
%modifications by Tarsha Eason (2008-2009).
%Further details on the methodology may be found in:
%Karunanithi, AT. Cabezas H, Frieden BR and Pawlowski CW. 2008. Detection
%and Assessment of ecosystem regime shifts from FI. Ecology
%& Society. 13(1): 22.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%This program calculates the FI from time series data.
%Pertinent parameters include:
%   FI          :       Fisher information
%   midt_win    :       mid point of each time window
%   t_win       :       time within each window
%   t           :       time data
%   data        :       Input data (matrix), size (#time steps, #variables)
%   sost        :       size of states
%   hwin        :       window size(number of points in each calculation)
%   winspace    :       number of time steps we move the window (<hwin)
%   TL          :       tightening Level
%   IP          :       Toggles the internal plots (IP=1==>on)
 %profile on
 FI=[];
t_win=[];
window=0;
for i=1:winspace:size(data,1) %start big loop to go through all the data
   %Revised Method: #pts=hwin
   lmin=min(i,size(data,1));
   lmax=min(i+hwin-1,size(data,1));
   NP=size(data(lmin:lmax,1),1);
if NP== hwin
   window=window+1;
   [pdf,neighbour]=Nfisherpdf(data,lmin,lmax,sost,TL/100); %calculate pdf for data
   %pdf
   %neighbour
   q=sqrt(pdf(:,1));  %convert to amplitude of the pdf
%modification
  counter=0;
  Q=0;
```

```
 neighbourQ=[];
 for j=1:size(q,1)
    if q(j,1)~=0
       counter=counter+1;
       Q(counter,1)=q(j,1);
       neighbourQ(counter,:)=neighbour(j,:);
     end
 end
%neighbourQ
%Q
%%%%%%%
%This portion of code re-arranges the states by assessing proximity.  As
%such, it calculates the Euclidean distance (zz) of the points in the state,
%finds the smallest distance and orders the q vector by the Euclidean
%distance.  This ensures that the states are indeed ordered by distance.
if size(neighbourQ,1)>2
minimumneighbourQ=0;
tempneighbourQ=0;
tempQ=0;
z=[];
for ii=1:(size(neighbourQ,1)-1)
   for jj=1:size(neighbourQ,1)
      if jj>ii
         z(ii,jj)=sqrt(sum(neighbourQ(ii,:)-neighbourQ(jj,:))^2);
      else
         z(ii,jj)=5000000000;
      end
   end
minimumneighbourQ=min(z(ii,:));
   for kk=2:size(neighbourQ,1)
      if kk>ii
         if z(ii,kk)==minimumneighbourQ
            tempneighbourQ=neighbourQ(ii+1,:);
            tempQ=Q(ii+1);
            neighbourQ(ii+1,:)=neighbourQ(kk,:);
            Q(ii+1)=Q(kk);
            neighbourQ(kk,:)=tempneighbourQ;
            Q(kk)=tempQ;
         end
      end
   end
end
end
 QQ=[0;Q(1:end);0]; %adding points (beginning and end) for edge gradients
 dq=diff(QQ)./1;  %calculating dqs
 %Calculates fisher for each window and places it in the middle of the %window
 if (i+(hwin-winspace))<=size(data,1)
   t_win(lmin:lmax,window)=[t(lmin:lmax)]; %time within each window
    FI(window)=[4*sum(dq.^2*1,1)]; %One fisher for each window
    midt_win(window)=ceil(mean(nonzeros(t_win(:,window))));
    %midt_win(window)= mean(nonzeros(t_win(:,window)));  %for ode calc
end
end
```

```
end
%FI
if IP==1  %plot outputs

subplot(2,1,1), plot(t,data) %plots data
%subplot(2,1,1), plot(data(:,1),data(:,2)) %plots data for ode
title('Time series data');
subplot(2,1,2),plot(midt_win,FI,'-b*')
title('Fisher information for each time window');
axis([min(t) max(t) 0 8]);
xlabel(,Time'); ylabel(,Fisher information');
end
```

NFisherpdf.m

```
function [pdf,neighbour]=NFisherpdf(data,lmin,lmax,sizeofstates,TL)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
%The methodolgy for calculating FI as a sustainability
%indicator as a measure of dynamic order has been an on-going interative
%effort within the US Environmental Protection Agency's Sustainable
%Environments Branch under the direction of Heriberto Cabezas.
%Key contributors to this effort include:
 %Theory, methodology and direction from Heriberto Cabezas as adapted from
%Ronald Fisher's work in theoretical statistics(1922); Chris Palowski for
%the conceptual framework and coding of initial Matlab program, recurnumFI4.m
%(6/17/2004); Methodology enhancements and corresponding source code
%implementation by Arunprakash Karunanithi (3/13/2006)  and additional
%functional and conceptual modifications by Tarsha Eason (2008-2009).
%Further details on the methodology can be found in: Karunanithi AT.
%Cabezas H, Frieden BR and Pawlowski CW. 2008. Detection and Assessment of
%ecosystem regime shifts from FI. Ecology & Society. 13(1): 22.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%
 %The purpose of this program is to bin the points into states by
%calculating the difference (dist) of each variable vector per timestep
%and then counting the variable vector if dist<= size of the state AND the
%percentage of variables in the vector that meets that criteria is over the
%tightening level (TL).  Further, the number of vectors that fit within a
%state are counted(rnum)and then used to calculate the pdf (rum/sum(rum)).

fp_tol=1e-12; %correcting for floating point significant digits
% Initialize the vector which will hold the counter for points in a state
   rnum=zeros(lmax-lmin+1,1);
%Initialize array of points that will be counted in line with lmin and lmax.
%This dummy variable is used to ensure that the rows are not double
%counted.
     pout=zeros(lmax-lmin+1,1);
   %
   % now, find recurrence points about each in window and record their
   % instances, ignoring points that have already been counted

   for j=lmin:lmax   %looping over the window in time steps
       j;
      count=0;
       if pout(abs(j-lmin)+1,1)==0  %no double counting variable vector
           pouttemp=zeros(size(pout));
           for k=lmin:lmax
               k;
            Mcount=0;
             for m=1:size(data,2)    %entire column length(#of variables)
                 m;

                  dist=abs(data(j,m)-data(k,m)); %for each variable 1 by 1
```

```
            %Dealing with floating point error
            r_dist=round2(dist,fp_tol); %rounds it to certain precision
            if r_dist<=(sizeofstates(1,m))

             Mcount=Mcount+1;
          end
          end

%Ensures that the number of variables that have been counted (meet
%the criteria is >= TL*total number of variables and this variable
%vector has not been previously counted

    if
Mcount>=TL*size(data,2))&(pout(abs(lmin-k)+1,1)==0)
              count=count+1;
              pouttemp(abs(lmin-k)+1,1)=1;
           end
        end
        rnum(abs(lmin-j)+1,1)=count; %number of points in state
        neighbour(abs(lmin-j)+1,:)=data(j,:);
      end
     pout=pout|pouttemp; %update no double counting using logical OR
    end
  %Calculating pdf
     if sum(rnum,1)==0
     pdf=rnum;
  else
     pdf=rnum/sum(rnum,1);
  end
size1=size(neighbour);
pdf;
```

CalcAvgFish.m

```
function [AvgFish,PO_TL,mY]= CalcMeanFish(t,data,hwin,winspace,sost)

%This code was developed to calculate the mean value of the Fisher
%information by computing FI over various tightening levels to find the
%point above which perfect order is experienced.  And then calculating a
%mean of the values for each time window.  The program returns:

%AvgFish:        Average Fisher information over TL from strict to relaxed
%PO_TL:          Tightening Level the system is in perfect order.  It is
%                connected to pervasiveness (1-PO_TL)
%mY:              Mean Year for each time window

% The code was developed by Tarsha Eason (2009)

%[AvgFish Y TL t_win]= Availdata(t,data,hwin,winspace,sost)
TL=[]; FisherMat=[];
profile on
display('Working....');
for i=1:100
   Tol=100-(i-1);

[FI,midt_win]=GFisher(t,data,sost,hwin,winspace,Tol,0);

FMat(:,i)=FI;
n=size(FI,2);  %# of Fisher information Calculations

%Calculates average FI for each time window by seeking the tightening
%level at which all resulting fisher calculation for each widow is at its
%max = 8, perfect order.  The average FI is calculated
%from the FisherMat which stores all of the Fisher results from each time
%window less the windows where all of the Fisher results are 8.
   if sum(FMat(:,i))<8*n %Tightening level with all Fishers = 8, perfect order
   % display('Yes');
    FisherMat(:,i)=FMat(:,i);
    TL=Tol; %Final value is lowest tightening level before all Fisher results are 8 (max FI)

  %else
   %    display('End TL Calcs');
  end
end %End Tol loop

if isempty(TL)==1
display('The system is in perfect order for this size of states at all tightening levels.  Please feel free to make an
adjustment accordingly!!');
f = warndlg('The system is in perfect order for this size of states at all tightening levels.  Please feel free to make an
adjustment accordingly!!', 'Warning Dialog');
 PO_TL=100;
else
PO_TL=TL-1;
end
mY=midt_win;
```

```
%Provides an ouput when the system is competely orderly at all tightening
%levels.  This results in no collection of Fisher values before the system
%is a maximum level.  Accordingly, FisherMat is empty indicates either true
%perfect order or a need to adjust the size of states (smaller)
 if isempty(FisherMat)==1;
    AvgFish=8*ones(size(FI));
else
 %Calculates mean FI for each time window
AvgFish=mean(FisherMat');
  end

%Provides an ouput when the system is competely orderly at all tighten
%levels.  Indicates either true perfect order or a need to adjust the size
%of states (smaller)

%Transposing (aesthetic)
AvgFish=AvgFish';
mY=midt_win';

%%%%%Simple Plots%%%%%%%%%%%%%%%%

subplot(2,1,1), plot(t,data) %plots data
%subplot(2,1,1), plot(data(:,1),data(:,2)) %plots data for ode
title('Time series data');

subplot(2,1,2), plot(mY,AvgFish)

axis([mY(1) mY(end) 0 9])
title('Fisher information over time');
xlabel('Time'); ylabel('Fisher information');
```

SmoothFI.m

```
function [FIm]=SmoothFI(n,t,FI,plot)

% Computes <FI>, a "m" point mean of FI smoothing results to focus on trends
% and not fluctuations.

% The code was developed by Tarsha Eason (2010)

data=[t FI];
A=sortrows(data,-1); FIs=A(:,2);ts=A(:,1);

window=0; I=[];T=[];A=[];B=[]; FIm=[];
for i=1 n:size(FI,1) %loop through FIs
   i

   %Revised Method: #pts=hwin
   lmin=min(i,size(FIs,1));
   lmax=min(i+n-1,size(FIs,1));
   NP=size(FIs(lmin:lmax,1),1);

   if NP== n
   window=window+1;

I(window)= mean(FIs(lmin:lmax));       %<FI>
T(window)=ceil(mean(ts(lmin:lmax)));
end
end

B=[T' I'];

FIm=sortrows(B,1);

if plot==1

subplot(2,1,1),plot(t,FI,'-*');  %plot original FI
title('Original FI'); axis([t(1) t(end) 0 8]);
subplot(2,1,2),plot(FIm(:,1),FIm(:,2),'-*');   %plot <I>
title('Smoothed FI(< FI >)'); axis([t(1) t(end) 0 8]);
end

%%
```

GUI: Main_Fisher_data_proc m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%This program  provides a graphical user interface (GUI) for performing
%Fisher information(FI)calculations. It is designed to simplify the
%procedure by allowing the user to select formatted data files, set
%parameter values and process the data which calculates FI over numerous
%tightening levels (from strict to relaxed tightening) providing an average
%FI result. The computed FI is plotted in the GUI, along with a plot of the
%FI results that have been "smoothed" using an "m" point average (<FI>).
%The user is able to save the FI computations and plot for further use.  It %also provides access for the advanced study of regime shifts, if they exist.

%Procedure for use
%
%Select the formatted data files (*time.xls, *data.xls and *sost.xls; see
%Appendix 6-B.2 for file formatting details) and press the
%'START' button to process the data and generate a plot of the
%time-averaged fisher.  After the data is processed, the plot and
%calculations may be saved by using the 'Save Plot' and 'Save Calculations'
%buttons. The 'Advanced' button calls the AdvancedFisher GUI for further
% Fisher information analysis.

%Main_Fisher_data_proc was developed by Tarsha Eason (2008-2010)as a
%graphical extension to work done from 2004 to 2008 by Cabezas, Pawloski,
%Karunithini and Eason to calculate FI in the Gfisher.m and
%Nfisherpdf.m code.  Please feel free to refer to the source code of those
%files for additional references.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
function varargout = Main_Fisher_data_proc(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Main_Fisher_data_proc_OpeningFcn, ...
                   'gui_OutputFcn',  @Main_Fisher_data_proc_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end

handles.processDataCompleted = 0; %
%%
function Main_Fisher_data_proc_OpeningFcn(hObject, eventdata, handles, varargin)
 handles.output = hObject;
%set(hObject,'toolbar','figure'); %enables toolbar
%this variable used to prevent users from breaking the GUI
%the variable is set to 1 once the data has been processed
```

```
%this command asks the user to confirm closing of GUI
set(handles.figure1,'CloseRequestFcn','closeGUI');

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Main_Fisher_data_proc wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%%
% --- Outputs from this function are returned to the command line.
function varargout = Main_Fisher_data_proc_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


%%
function inputFiles_listbox_Callback(hObject, eventdata, handles)
%no code needed for this callback, the listbox is only used as a visual

%%
function addFiles_pushbutton_Callback(hObject, eventdata, handles)

%gets input file(s) from user
[input_file,pathname] = uigetfile( ...
      {'*.xls', 'Data Files (*.xls)'; ...
       '*.*', 'All Files (*.*)'}, ...
       'Select files', ...
       'MultiSelect', 'on');

%if file selection is cancelled, pathname should be zero
%and nothing should happen
if pathname == 0
   return
end

%gets the current data file names inside the listbox
inputFileNames = get(handles.inputFiles_listbox,'String');
 %if they only select one file, then the data will not be a cell
if iscell(input_file) == 0

   %add the most recent data file selected to the cell containing
   %all the data file names
   inputFileNames{length(inputFileNames)+1} = fullfile(pathname,input_file);
```

```
%else, data will be in cell format
else
   %stores full file path into inputFileNames
   for n = 1:length(input_file)
      inputFileNames{length(inputFileNames)+1} = fullfile(pathname,input_file{n});
   end
end

%updates the gui to display all filenames in the listbox
set(handles.inputFiles_listbox,'String',inputFileNames);

%make sure first file is always selected so it doesn't go out of range
%the GUI will break if this value is out of range
set(handles.inputFiles_listbox,'Value',1);

% Update handles structure
guidata(hObject, handles);

%%
function deleteFiles_pushbutton_Callback(hObject, eventdata, handles)
%this function allows the user to delete files they accidentally
%added to the list box

%get the current list of file names from the listbox
inputFileNames = get(handles.inputFiles_listbox,'String');

%get the values for the selected file names
option = get(handles.inputFiles_listbox,'Value');

%is there is nothing to delete, nothing happens
if (isempty(option) == 1 || option(1) == 0 || isempty(inputFileNames))
   return
end
%erases the contents of highlighted item in data array
inputFileNames(option) = [];

%updates the gui, erasing the selected item from the listbox
set(handles.inputFiles_listbox,'String',inputFileNames);

%moves the highlighted item to an appropriate value or else will get error
if option(end) > length(inputFileNames)
   set(handles.inputFiles_listbox,'Value',length(inputFileNames));
end

% Update handles structure
guidata(hObject, handles);
%%
function edit_hwin_Callback(hObject, eventdata, handles)
% hObject        handle to edit_hwin (see GCBO)
% eventdata      reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputText_hwin as text
%        str2double(get(hObject,'String')) returns contents of edit_hwin as a double
```

```
HW=str2num(get(hObject,'String'));

%checks to see if input is empty.  If so, default input1_editText to zero
if(isempty(HW))
   set(hObject,'String','0')
end
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function edit_hwin_CreateFcn(hObject, eventdata, handles)
% hObject        handle to edit_hwin (see GCBO)
% eventdata      reserved - to be defined in a future version of MATLAB
% handles        empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
%%
function edit_winspace_Callback(hObject, eventdata, handles)
% hObject        handle to edit_winspace (see GCBO)
% eventdata      reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputText_winspace as text
%        str2double(get(hObject,'String')) returns contents of inputText_winspace as a double
WS=str2num(get(hObject,'String'));

%checks to see if input is empty.  If so, default input1_editText to zero
if(isempty(WS))
   set(hObject,'String','0')
end

guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function edit_winspace_CreateFcn(hObject, eventdata, handles)
% hObject        handle to edit_winspace (see GCBO)
% eventdata      reserved - to be defined in a future version of MATLAB
% handles        empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
%%
function edit_Smoothpts_Callback(hObject, eventdata, handles)
% hObject        handle to edit_winspace (see GCBO)
% eventdata      reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of inputText_winspace as text
%        str2double(get(hObject,'String')) returns contents of inputText_winspace as a double
nS=str2num(get(hObject,'String'));

%checks to see if input is empty.  If so, default input1_editText to zero
if(isempty(nS))
   set(hObject,'String','0')
end
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function edit_Smoothpts_CreateFcn(hObject, eventdata, handles)
% hObject          handle to edit_winspace (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
%%
function reset_pushbutton_Callback(hObject, eventdata, handles)
%resets the GUI by clearing all relevant fields
handles.processDataCompleted = 0;
cla(handles.axes1,'reset');

set(handles.edit_hwin,'String','8');
set(handles.edit_winspace,'String','1');
set(handles.edit_Smoothpts,'String','3');
%set(handles.inputFiles_listbox,'String','');
%set(handles.inputFiles_listbox,'Value',0);
guidata(hObject, handles);
%%
function start_pushbutton_Callback(hObject, eventdata, handles)

inputFileNames = get(handles.inputFiles_listbox,'String');
%checks to see if the user selected any input files
%if not, nothing happens
if isempty(inputFileNames)
   return
end

%disables the button while data is processing
disableButtons(handles);
refresh(Main_Fisher_data_proc);

%initialize the cell structures
handles.data = {};
handles.time = {};
sost=[];
%Read input data
```

```
for x=1:length(inputFileNames)
   A=inputFileNames{x};
   a=size(A,2);
   b=a-7;
   if(A(b:a)=='data.xls')
      %display('Loading Variable Data');
      data=xlsread(inputFileNames{x});
      %size(data)

   elseif(A(b:a)=='time.xls')
      %display('Loading Time Data');
      time=xlsread(inputFileNames{x});
      %size(time)
   elseif(A(b:a)=='sost.xls')
      %display('Loading Size of states Data');
      sost=xlsread(inputFileNames{x});
      %size(sost)
   else
      display('Not correct input filename.  See Appendix 6-B.2');
   end
end

%Calculate Fisher information
t=time;
handles.data=data;
handles.t=time;

Flag=isempty(sost); %test to see if *sost.xls file was uploaded

if Flag==1
%display('Since no size of states file was added.  The size of states will be calculated from the first 5 data points')

errordlg('Since no size of states file was added.  The size of states will be calculated from the first 5 data points','File
Error');
sost=2*std(data(1:5,:));
end

handles.sost=sost;
%get hwin and winspace from input form
H=get(handles.edit_hwin,'String');
W=get(handles.edit_winspace,'String');
nS=get(handles.edit_Smoothpts,'String');

hwin=str2num(H);
winspace=str2num(W);
nSpts=str2num(nS);

handles hwin=hwin;
handles.winspace=winspace;

TL=[]; FisherMat=[];

%Using search algorithm for finding the size of states
%[s_sost,region_year,diff_test]=statesize(4,hwin,winspace,data,time);
%sost=s_sost;
```

```
%Calculating lowest tightening level before all FI results are 8
%(max): PO_TL
for i=1:100
   Tol=100-(i-1);
 [FI, midt_win, t_win]=GFisher(t,data,sost,hwin,winspace,Tol,0);
   %YMat(:,i)=t_win;
   %YMat(:,i)=t_win(i);
   FMat(:,i)=FI';

n=size(FI,2);  %# of FI Calculations

%Calculates FI for each time window by seeking the tightening
%level at which all resulting fisher calculation for each widow is at its
%max = 8, perfect order.  The average fisher is calculated
%from the FisherMat which stores all of the FI results from each time
%window less the windows where all of the FI results are 8.

  if sum(FMat(:,i))<8*n %Tightening level with all FI values = 8, perfect order
   % display('Yes');
    FisherMat(:,i)=FMat(:,i);

    TL=Tol; %Final value is lowest tightening level before all FI results are 8
   else
   %display('End TL Calcs');

  end
end %End Tol loop

if isempty(TL)==1
display('The system is in perfect order for this size of states at all tightening levels.  Please feel free to make an
adjustment accordingly!!');
f = warndlg('The system is in perfect order for this size of states at all tightening levels.  Please feel free to make an
adjustment accordingly!!', 'Warning Dialog');
 PO_TL=100;

else
PO_TL=TL-1;
end

handles.PO_TL=PO_TL;

%Provides an ouput when the system is competely orderly at all tighten
%levels.  This results in no collection of FI values before the system is a
%maximum level.  Accordingly, FisherMat is empty indicates either true
%perfect order or a need to adjust the size of states (smaller)

if isempty(FisherMat)==1;
   AvgFish=8*ones(size(FI));
else

%Calculates Avg FI for each time window
AvgFish=mean(FisherMat');

end
```

```
T=zeros(size(t_win));
P=T|t_win;

t_win;

   handles.AvgFish=AvgFish;
   %AvgFish

for i=1:size(AvgFish,2)
   AvgFI(:,i)=P(:,i)*AvgFish(i);
end

handles.AvgFI_win=AvgFI;

handles.PO_TL=PO_TL;
%PO_TL
%plot mean FI over time window
axes(handles.axes1)

YMat=t(1:size(handles.AvgFI_win,1)); %time period
handles.YMat=YMat;

%%%%Plot Avg FI over time window%%%%%%%%%%%%%%%%
%plot(YMat,handles.AvgFI_win)
%title('Average FI vs. Time');

%%%%%Plot average FI in center of window%%%%%%%%%%%%%%%%
for j=1:size(t_win,2)
   R(:,j)=nonzeros(t_win(:,j));
   Z(:,j)=AvgFish(j)*ones(size(R,1),1);
Y(j)=ceil(mean(R(:,j))); %mean year for each time window
end

plot(Y,AvgFish,'*b-')
handles.Year=Y;
title('Average FI over time');
hold
n=nSpts; %n=3;  %make sure to create an edit text to change this value
[FIm]=SmoothFI(n,Y',AvgFish',0);
handles.MnYr=FIm(:,1);
handles.MnFI=FIm(:,2);

plot(FIm(:,1),FIm(:,2),'*r-')

legend('Fisher information(FI)','Smoothed FI (<FI>)')
legend(handles.axes1,'boxoff')
axis([Y(1,1) Y(end) 0 8]); xlabel('Year'); ylabel('Fisher information');

enableButtons(handles);
handles.processDataCompleted = 1;
guidata(hObject, handles);
%%
function savePlot_pushbutton_Callback(hObject, eventdata, handles)
```

```
%if the data hasn't been processed yet,
%nothing happens when this button is pressed

if (handles.processDataCompleted == 0)
   return
end

disableButtons(handles);
refresh(Main_Fisher_data_proc);

savePlot(handles.axes1); %figure out how to save the legend?

enableButtons(handles);
guidata(hObject, handles);
%%
function export_pushbutton_Callback(hObject, eventdata, handles)
%This function saves the mean FI values, date and time of
%calculation, as well as the smoothed mean FI results into an Excel file in
%the format:
%Row1 Col1: Date
%Row2 Col1: Time (military)
%Row3 Col1: Perfect order tightening level: PO_TL
%Row4 to Row4+size(FI) Col2: [Year FI]
%The Smoothed mean FI results follow

%if the data hasn't been processed yet,
%nothing happens when this button is pressed
if (handles.processDataCompleted == 0)
  return
end

calctime=clock;

%stores savepath for results
[filename, pathname] = uiputfile ({'*.xls', 'Data Files (*.xls)'; ...
      '*.*', 'All Files (*.*)'}, ...
      'Save file as','default');

%if user cancels save command, nothing happens
if isequal(filename,0) || isequal(pathname,0)
   return
end

%saves the data
dlmwrite(fullfile(pathname, filename),[calctime(2),calctime(3),calctime(1)],'delimiter','/','-append');
dlmwrite(fullfile(pathname, filename),[calctime(4) calctime(5)],'delimiter',':','precision','%4.2d','-append');
dlmwrite(fullfile(pathname, filename),handles.PO_TL,'-append');
%dlmwrite(fullfile(pathname, filename),'Mean FI','-append')
dlmwrite(fullfile(pathname, filename),[handles.Year',handles.AvgFish'],'-append','delimiter','\t','roffset',1)
dlmwrite(fullfile(pathname, filename),[handles.MnYr,handles.MnFI],'-append','delimiter','\t','roffset',2)
%%
```

```
function disableButtons(handles)
set(handles.figure1,'Pointer','watch');
set(handles.start_pushbutton,'Enable','off');
set(handles reset_pushbutton,'Enable','off');
set(handles.addFiles_pushbutton,'Enable','off');
set(handles.savePlot_pushbutton,'Enable','off');
set(handles.deleteFiles_pushbutton,'Enable','off');
set(handles.inputFiles_listbox,'Enable','off');
set(handles.export_pushbutton,'Enable','off');
set(handles.Advanceduser_pushbutton,'Enable','off');
set(handles.Help_pushbutton,'Enable','off');
%%
function enableButtons(handles)
set(handles.figure1,'Pointer','arrow');
set(handles.start_pushbutton,'Enable','on');
set(handles reset_pushbutton,'Enable','on');
set(handles.addFiles_pushbutton,'Enable','on');
set(handles.savePlot_pushbutton,'Enable','on');
set(handles.deleteFiles_pushbutton,'Enable','on');
set(handles.inputFiles_listbox,'Enable','on');
set(handles.export_pushbutton,'Enable','on');
set(handles.Advanceduser_pushbutton,'Enable','on');
set(handles.Help_pushbutton,'Enable','on');
%%
function inputFiles_listbox_CreateFcn(hObject, eventdata, handles)

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Advanceduser_pushbutton.
function Advanceduser_pushbutton_Callback(hObject, eventdata, handles)
% hObject          handle to Advanceduser_pushbutton (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          structure with handles and user data (see GUIDATA)

AdvancedFisherGUI()  %Call subGUI for further analysis

guidata(hObject, handles);

function Help_pushbutton_Callback(hObject, eventdata, handles)
% hObject          handle to edit_hwin (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          structure with handles and user data (see GUIDATA)

%testflag=250000

HelpPath = which('helpMain.html');
web(HelpPath);  %opens GUI help file

guidata(hObject,handles);
```

AdvancedFisherGUI m

```
function varargout = AdvancedFisherGUI(varargin)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%This program is called from the Fisher_data_proc GUI and provides a
%graphical user interface for studying the pervasiveness and intensity of
%regime shifts by plotting the Fisher information (FI) results for tightening
%levels (TL) from 100%-10% overlaid by the average FI. Pervasiveness indicates
%the number of variables affected by the shift and is determined by finding
%the TL value at which the system is perfectly ordered (i.e. FI is at its
%max (8) for the entire period.  Pervasiveness is computed by subtracting
%the perfect order tightening level (POTL) from 100) and denotes the percentage
%of variables that were impacted. The intensity relates to the amplitude of
%the shift and is determined by how far the FI drops between two stable regimes.

%NOTE: Data files must be selected through the Fisher_data_proc GUI and
%processed prior to doing advanced analysis through this GUI.
%
%AdvancedFisher was developed by Tarsha Eason(2008-2010) as a graphical
%extension to work done from 2004 to 2008 by Cabezas, Pawlowski, Karuninithi
%and Eason to calculate FI in the Revfisher m, fisherpdf m code.
%Please feel free to refer to the source code of those files for additional
%references.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%

%Matlab GUIDE information

% ADVANCEDFISHERGUI M-file for AdvancedFisherGUI.fig
% ADVANCEDFISHERGUI, by itself, creates a new ADVANCEDFISHERGUI or raises
% the existing singleton*.
%
% H = ADVANCEDFISHERGUI returns the handle to a new ADVANCEDFISHERGUI or
% the handle to the existing singleton*.
%
%ADVANCEDFISHERGUI('CALLBACK',hObject,eventData,handles,...) calls the
%local function named CALLBACK in ADVANCEDFISHERGUI.M with the given input
%arguments.
%
%ADVANCEDFISHERGUI('Property','Value',...) creates a new ADVANCEDFISHERGUI
%or raises the existing singleton*.  Starting from the left, property value
%pairs are applied to the GUI before AdvancedFisherGUI_OpeningFcn gets
%called.  An unrecognized property name or invalid value makes property
%application stop.  All inputs are passed to AdvancedFisherGUI_OpeningFcn
%via varargin.

 % Last Modified by GUIDE v2.5 17-Jun-2010 12:44:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                'gui_OpeningFcn', @AdvancedFisherGUI_OpeningFcn, ...
                'gui_OutputFcn',  @AdvancedFisherGUI_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

 % --- Executes just before AdvancedFisherGUI is made visible.
function AdvancedFisherGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to AdvancedFisherGUI (see VARARGIN)

% Choose default command line output for AdvancedFisherGUI
handles.output = hObject;

%this command asks the user to confirm closing of GUI
set(handles.figure1,'CloseRequestFcn','closeGUI');

%set(hObject,'toolbar','figure');  % activates toolbar

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes AdvancedFisherGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = AdvancedFisherGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function inputText_hwin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to inputText_hwin (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function inputText_winspace_Callback(hObject, eventdata, handles)
% hObject          handle to inputText_winspace (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputText_winspace as text
%        str2double(get(hObject,'String')) returns contents of inputText_winspace as a double
%WS=str2num(get(hObject,'String'));

%checks to see if input is empty.  If so, default input1_editText to zero
%if(isempty(WS))
%    set(hObject,'String','0')
%end
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function inputText_winspace_CreateFcn(hObject, eventdata, handles)
% hObject          handle to inputText_winspace (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function inputText_TL_Callback(hObject, eventdata, handles)
% hObject          handle to inputText_TL (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of inputText_TL as text
%        str2double(get(hObject,'String')) returns contents of inputText_TL as a double

%Tol=str2num(get(hObject,'String'));

%checks to see if input is empty.  If so, default input1_editText to zero
if(isempty(Tol))
    set(hObject,'String','0')
end
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function inputText_TL_CreateFcn(hObject, eventdata, handles)
% hObject          handle to inputText_TL (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%Calculates and plots FI different tightening levels

% --- Executes on button press in FishTL_pushbutton.
function FishTL_pushbutton_Callback(hObject, eventdata, handles)
% hObject          handle to FishTL_pushbutton (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          structure with handles and user data (see GUIDATA)

YMat=[];FMat=[];
mainFigureHandle=Main_Fisher_data_proc;  %stores the figure handle of main GUI here
mainGUIdata = guidata(mainFigureHandle);

%Pulling data, time, AvgFish, and midyear from Main_Fisher_data_proc m

t=mainGUIdata.t;
data=mainGUIdata.data;
AvgFish=mainGUIdata.AvgFish;
midyear=mainGUIdata.Year;
PO_TL=mainGUIdata.PO_TL;
SOST=mainGUIdata.sost;
hwin=mainGUIdata.hwin;
winspace=mainGUIdata.winspace;

%Studying Pervasiveness and Intensity from Main_Fisher data_proc parameter
%settings

for i=1:10
    Tol=100-(i-1)*10;

 [FI, midt_win, t_win]=GFisher(t,data,SOST,hwin,winspace,Tol,0);


 FMat(:,i)=FI';
 YMat(:,i)=midt_win;

end

axes(handles.axes3); plot(YMat(:,1),FMat(:,1),YMat(:,2),FMat(:,2),YMat(:,3),FMat(:,3),YMat(:,4),FMat(:,4),YMat(:,
5),FMat(:,5));


% Create legend
legend1 = legend(handles.axes3,'TL=100%','TL=90%','TL=80%','TL=70%','TL=60%');
set(legend1,'YColor',[1 1 1],'XColor',[1 1 1],...
    'Location','NorthEast',...
    'FontSize',8,...
    'FontName','Arial');
```

```
hold

plot(midyear,AvgFish,'-gd','LineWidth',2);

title('Pervasiveness and Intensity (100%-60%)');
axis([YMat(1,1) YMat(size(YMat,1)) 0 10]); xlabel('Year'); ylabel('Fisher information');

hold off

axes(handles.axes4); plot(YMat(:,6),FMat(:,6),YMat(:,7),FMat(:,7),YMat(:,8),FMat(:,8),YMat(:,9),FMat(:,9),YMat(:,
10),FMat(:,10))

% Create legend
legend2 = legend(handles.axes4,'TL=50%','TL=40%','TL=30%','TL=20%','TL=10%');
set(legend2,'YColor',[1 1 1],'XColor',[1 1 1],...
   'Location','NorthEast',...
    'FontSize',8,...
    'FontName','Arial');
 hold
plot(midyear,AvgFish,'-gd','LineWidth',2);

title('Pervasiveness and Intensity (50%-0%)');
axis([YMat(1,1) YMat(size(YMat,1)) 0 10]); xlabel('Year'); ylabel('Fisher information');

hold off

PVL=100-PO_TL;
set(handles.edit_POTL,'String',num2str(PO_TL));
set(handles.edit_Pervasiveness,'String',num2str(PVL))

%Save result of User Analysis

%dlmwrite('Results.xls',[hwin,winspace],'-append');
%dlmwrite('Results.xls',[YMat(:,1) AFish'],'-append','delimiter','\t','precision','%4.4f');
%dlmwrite('Results.xls',[midyear' AvgFish'],'-append','delimiter','\t','precision','%4.4f');
warning off
%%%%%

guidata(hObject, handles); %updates the handles

function edit_Pervasiveness_Callback(hObject, eventdata, handles)
% hObject        handle to edit_Pervasiveness (see GCBO)
% eventdata      reserved - to be defined in a future version of MATLAB
% handles        structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_Pervasiveness as text
%        str2double(get(hObject,'String')) returns contents of edit_Pervasiveness as a double

 % --- Executes during object creation, after setting all properties.
function edit_Pervasiveness_CreateFcn(hObject, eventdata, handles)
% hObject        handle to edit_Pervasiveness (see GCBO)
% eventdata      reserved - to be defined in a future version of MATLAB
% handles        empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in RESET_Pushbutton.
function RESET_Pushbutton_Callback(hObject, eventdata, handles)
% hObject          handle to RESET_Pushbutton (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          structure with handles and user data (see GUIDATA)
cla(handles.axes3,'reset');
cla(handles.axes4,'reset');
set(handles.inputText_TL,'String',100)
set(handles.inputText_hwin,'String',10)
set(handles.inputText_winspace,'String',5)
set(handles.edit_Pervasiveness,'String',0);
set(handles.edit_POTL,'String',0);
%handles.buttonCounter = 0; %reset buttoncounter

guidata(hObject, handles); %updates the handles

function edit_POTL_Callback(hObject, eventdata, handles)
% hObject          handle to edit_POTL (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_POTL as text
%        str2double(get(hObject,'String')) returns contents of edit_POTL as a double

% --- Executes during object creation, after setting all properties.
function edit_POTL_CreateFcn(hObject, eventdata, handles)
% hObject          handle to edit_POTL (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in Help_pushbutton.
function Help_pushbutton_Callback(hObject, eventdata, handles)
% hObject          handle to Help_pushbutton (see GCBO)
% eventdata        reserved - to be defined in a future version of MATLAB
% handles          structure with handles and user data (see GUIDATA)

HelpPath = which('helpAdvGUI2.html');
web(HelpPath);

guidata(hObject,handles);
```

round2 m : Floating Point Correction

```
function z = round2(x,y)
%ROUND2 rounds number to nearest multiple of arbitrary precision.
%Z = ROUND2(X,Y) rounds X to nearest multiple of Y.
%Floating point work around designed by Robert Bemis 15 Dec 2003 and updated
%(29 Jun 2006): http://www.mathworks.com/matlabcentral/fileexchange/4261

%% defensive programming
error(nargchk(2,2,nargin))
ror(nargoutchk(0,1,nargout))
if prod(size(y))>1
  error('n must be scalar')
end

z = round(x/y)*y;
```

CloseGUI.m

```
function closeGUI

selection = questdlg('Do you want to close the GUI?',...
                  'Close Request Function',...
                  'Yes','No','Yes');
switch selection,
   case 'Yes',
    delete(gcf)
   case 'No'
     return
end
```

EPA
United States
Environmental Protection
Agency

PRESORTED STANDARD
POSTAGE & FEES PAID
EPA
PERMIT NO. G-35

Office of Research and Development (8101R)
Washington, DC 20460