2011-05-19

Spatiotemporal Organization of Atrial Fibrillation Using Cross-Bicoherence with Surrogate Data

Rafael Jaimes
Worcester Polytechnic Institute

Spatiotemporal Organization of Atrial Fibrillation with Rotigaptide Treatment Using Cross-Bicoherence Index with Surrogate Data

By

RAFAEL JAIMES III

A Thesis

Submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

In

Biomedical Engineering

May 2011

Thesis Committee:

Ki H. Chon, Ph.D., Advisor & Head of Department, Biomedical Engineering

X _____

Domhnull Granquist-Fraser, Ph.D., Assistant Professor, Biomedical Engineering

X _____

David McManus, M.D., University of Massachusetts Medical School

X _____

Abstract

Atrial fibrillation (AF) is a troublesome disease often overlooked by more serious myocardial infarctions. Up until now, there has been very little or no use of high order spectral techniques in order to evaluate the organization of the atrium during AF. Cross-bicoherence algorithm can be used alongside a surrogate data threshold in order to determine significant phase coupling interactions, giving rise to an organizational metric. This proposed algorithm is used to show rotigaptide, a gap junction coupling drug, significantly increases the organization of the atria during episodes of AF due to improvement of cell-to-cell coupling.

Acknowledgments

I would like to thank to thank my thesis committee members as well as the members of Prof. Chon's lab for providing me with guidance throughout this project:

Chris Scully, M.S.

Bufan Yang, M.S.

Nandakumar Selvaraj, Ph.D.

Jinseok Lee, Ph.D.

Table of Contents

Table of Figures

Table of Tables

Table of Equations

I. Introduction

Motivation

The goal of this project is to quantitatively analyze the spatiotemporal organization of the electrical activity of the right atrium during atrial fibrillation (AF), in the absence and presence of the antiarrhythmic peptide rotigaptide. High-order spectral analysis is used in this effort for ultimately a better understanding of AF, for improved targeting of defibrillation techniques and surgical procedures such as the maze procedure and catheter ablation.

Atrial Fibrillation

AF is a very prevalent disease in the United States, affecting over 2.2 million patients in 2009 [1]. The mortality rate for AF in the U.S. was 11,555 in 2009 [1]. The primary concern with AF is that is a main contributor in causing strokes. Among patients with AF the risk of ischemic stroke increases by 4 to 5 fold, and overall AF is believed to be responsible for at least 15 to 20% of all ischemic strokes [1]. AF is not usually treated directly, but rather its symptoms are treated. Anti-coagulants are usually prescribed in order to prevent the formulation of thrombosis. However on occasion, anti-arrhythmics are prescribed or cardioversion is employed for defibrillation.

AF occurs when areas of the atrial tissue generate electrical pulses randomly. The presence of secondary pacemaker cells overpowers the electrical impulse of the sinoatrial (SA) node, which under normal conditions should regulate the pacing of the heart. With AF there is no ordered stimulus to the downstream myocardium, and seemingly random patterns occur. The rapid and random electrical activity results in inefficient myocardial contraction and decreased ejection fraction. A typical ECG during AF is displayed in Fig. 1.

Fig. 1: ECG of Atrial Fibrillation (Lead II) [2]

AF only involves the atria, typically the atrioventricular (AV) node is able to reset the electrical pulse and the ventricles perform normally. The P-wave in the ECG, corresponding to atrial depolarization, is normally not present during AF. The QRS complex is clearly visible in the ECG during AF and analysis such as R-R interval or heart rate can be calculated. It has been previously shown that R-R interval is irregular during AF, and can be used to aid in differentiating other arrhythmias such as atrial flutter [3].

Current Treatments

The most common effect of atrial fibrillation is the risk of clot formation due to stagnant blood in the right atrial or left atrial appendages, which may potentially lead to pulmonary embolism or stroke, respectively, among other complications. Because of the large risk of thrombosis, the most common forms of treatment are blood thinners and anti-coagulants such as heparin.

Anti-platelets may be effective at removing some of the dangers involved with AF. However, they do not target the disease itself. In addition to other risks, AF leads to decreased ejection fraction and inefficient cardiac output. Amiodarone is one of the most popular antiarrhythmic pharmacologic currently used to maintain sinus rhythm in long-term AF patients. However, it is associated with a recurrence rate of 30% after 1 year of treatment [4]. Iodine is a major component of amiodarone and along with its lipophilic nature, it is associated with pulmonary toxicity, thyroid abnormalities, and ocular injury [4]. Other pharmacotherapy interventions such as sotalol or propafenome have an abysmal 60% recurrence rate [4].

Surgical procedures such as the maze procedure [5] and catheter ablation [6] are commonly used in order to isolate those parts of the heart (or surrounding tissue) that cause ectopic beats. Pulmonary vein isolation (PVI) is the most common, due to the pulmonary vein often causing AF through an ectopic beat [7-9]. The maze procedure also isolates the atrial appendages, in order to help eliminate the formation of blood clots. The most common

An additional treatment used to achieve sinus rhythm from arrhythmias is electrical cardioversion or electrical defibrillation. Normally, a shock is delivered at the crest of the R-wave within the QRS complex, if present. However, Everett et al have shown that atrial organization plays a vital role in successful defibrillation [10] and timing the shock based off organization can lead to even greater success [11]. Everett and his group performed 182 shocks between 10 dogs, 95 of the shocks were successful in re-achieving sinus rhythm and 87 of the shocks were unsuccessful. Electrogram data was collected prior to each shock using a bipolar electrode setup and organization index was calculated using a method discussed later. It was found that the mean organization index for the successful shocks was 0.505 and for unsuccessful shocks was 0.352, with a p-value < 0.0001.

Higher organization of the atrium leads to higher chances of defibrillation success. A proposed drug known as rotigaptide, while not able to suppress AF independently, may increase the organization of the atrium and improve the chances of successful cardioversion.

Rotigaptide & Connexin43

Heart failure and arrhythmias are often associated with decreased gap junction intercellular communication. It has been previously reported that antiarrhythmic peptides (AAP) exert antiarrhythmic actions by increasing the gap junction intercellular communication between cardiac myocytes [12]. A stable AAP analogue known as rotigaptide ("ZP123", Zealand Pharma, Copenhagen, Denmark) has been shown to increase the electrical coupling between ventricular myocytes [13], [14]. During myocardium ischemia, the cardiac myocytes have decreased gap junction coupling and undergo an increased dispersion of action potential duration and slowed conduction velocity which facilitates arrhythmias including but not limited to ventricular and/or atrial tachycardia or fibrillation. Rotigaptide increases the conduction velocity by opening gap junctions, while having little to no effect on the effective refractory period [15], heart rate, contractility or mean coronary flow as shown in isolated rabbit hearts [13]. Rotigaptide augments gap junction conductance improving cell-to-cell coupling at both electrical and metabolic levels [13]. Everett et al have shown that with 10 nM treatment of rotigaptide, there is a 19% increase in conduction velocity in the atrium during an AF episode. With 100 nM there was a 42% increase over no treatment, and with 300 nM there was a 51% increase compared to no treatment [16]. This conduction velocity increase should lead to a greater organization of the atrium.

There are several gap junction proteins present in the heart: connexin43 (Cx43) is one of the primary connexin isoforms along with Cx40 and Cx45, especially in the atrium [17]. The SA node only expresses Cx43 at the border zone with the atrium, while the AV node expresses primarily Cx40 and Cx43 with very little Cx45 [17]. Beardslee et al reported in 2000 that electrical uncoupling induced by acute ischemia is associated with changes in phosphorylation of Cx43 specifically [18]. In 2006, Axelsen et al have shown that rotigaptide greatly lengthens the time to asystole after ischemia, suggesting rotigaptide has a role in Cx43 phosphorylation [14].

Current Atrial Organization Methods

In 1989, Ropella et al were one of the first groups attempting to quantify the atrial organization during fibrillation. They attempted to use the magnitude-squared coherence function based off power spectral densities (PSD). Although the method failed at providing anything useful concerning atrial fibrillation organization, it did prove to be an effective way to differentiate fibrillatory from non-fibrillatory arrhythmias [19]. The method was based off a 2^{nd} order technique using frequency analysis which is useful for analyzing linear and time-invariant (LTI) systems. However, AF is a time-varying, non-linear process.

In 1995, Botteron and Smith were one of the first groups to successfully develop a method of quantifying the atrial organization during fibrillation: using harmonic analysis [20]. They used an indirect way of observing changes in phase over time by comparing the area under the dominant frequency and its harmonics with that of the entire spectral area up until that point. When Botteron and Smith used their method of measuring organization index, they found the average organization was between 0.32 and 0.54 for AF compared with 0.91 and 0.95 for sinus rhythm [21]. Although the method proved to be somewhat reliable for determining organization, when Everett and his group used the method to check for differentiation in organization due to rotigaptide treatment, no discernment could be made [unpublished]. Botteron and Smith's method is still reliant on the PSD, and does not accurately represent phase relations and non-linear relations between signals. Normally, the PSD discards phase information. For example, the PSD of a sine wave is the same as a cosine wave at the same frequency. If the only difference between signals is the phase, that difference will not be detectable by using PSD alone.

Literature is devoid of a more elegant method of measuring the organization of the atrium: using cross-bispectrum technique in order to properly analyze the non-linear interactions. By implementing cross-bispectrum over time the time-varying aspect can be addressed as well.

Cross-Bispectrum Techniques

The magnitude-squared coherence function and cross-PSD are variants of a method in order to determine linear interactions, or frequency coupling, between two signals. They are able to find coherence or similarity in frequency spectral content between two inputs. The cross-bispectrum is a 3rd order technique suited to detecting interaction between two signals involving both linear and non-linear interaction [22]. Non-linear interactions, more specifically: quadratic phase coupling (QPC) between two signals, implies that not only two signals are coherent with respect to a particular frequency, but that the frequencies are in phase. The cross-bispectrum between two signals is defined as the triple product of the third order cumulants of two signals:

$$B_{xy}(f_1, f_2) = P_x(f_1)P_y(f_2)P_x^*(f_1 + f_2)$$

Eq. 1: Cross-bispectrum between two signals

The cross-bispectrum is sensitive to frequency and/or phase coupling between signals; both linear and non-linear interactions contribute to the magnitude of the output. Using the cross-bispectrum alone, discernment cannot be made between phase and frequency coupling.

By observing the phase coupling between signals, taken from different regions of the atria, it is believe that by quantifying these interactions they would lead to a more efficient and accurate metric of determining atrial organization. Using this improved organizational metric, it can then be used to determine if there is an organizational difference between no treatment and varying treatments of rotigaptide doses.

The following simulated data was analyzed with the cross-bispectrum algorithm in order to evaluate its efficacy. Two signals with pre-determined frequency and phase coupling were generated using a method proposed by Siu and Chon [23] described in Eq. 2 (code in Appendix A).

$$x_1(n) = e^{j(2\pi f_x(1)n\Delta+\varphi_x(1))}$$

$$x_2(n) = e^{j(2\pi f_x(2)n\Delta+\varphi_x(2))}$$

$$y_1(n) = e^{j(2\pi f_y(1)n\Delta+\varphi_y(1))}$$

$$y_2(n) = e^{j(2\pi f_y(2)n\Delta+\varphi_y(2))}$$

$$x(n) = x_1(n) + x_2(n) + x_1(n)x_2(n)$$

$$y(n) = y_1(n) + y_2(n) + y_1(n)x_2(n)$$

Eq. 2: Simulated signals with pre-determined coupling

…where $f_x(1) = f_y(1)$=0.03 Hz, $f_x(2) = f_y(2)$=0.12 Hz, n=4096 samples, Δ=1 sec, and φ=rand(0…2π).

The two signals x(n) and y(n) have precisely the same frequency content as shown in the PSD generated by an FFT in Fig. 2.

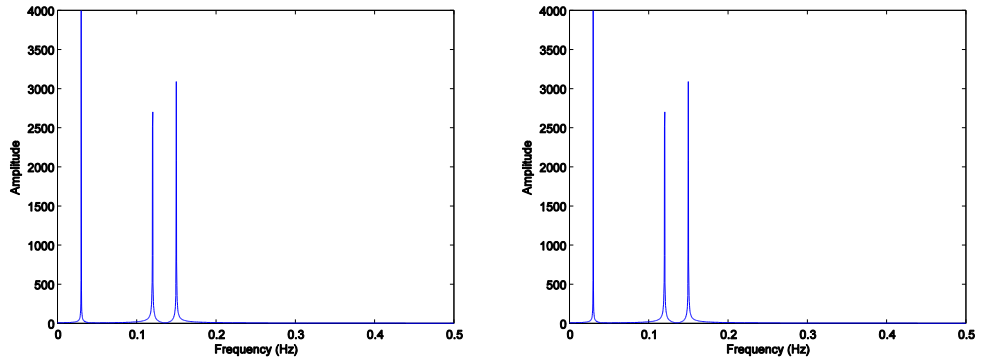


Fig. 2: a) PSD for x(n), left and b) y(n), right

Peaks arise at both 0.03 Hz and 0.12 Hz, in addition to the frequency coupled harmonic at 0.15 Hz (f_1+f_2). Frequency coupling arises from the composite sum from the first two terms in x(n) and y(n). Despite the similarity in frequency content apparent in the PSD, the phase content is quite different between these two signals due to unidirectional phase coupling from y(n) to x(n) arisen from the 3rd term product in y(n). In order to evaluate the phase coupling, cross-bispectrum is used. The cross-bispectrum between signal 1 and signal 2 gives rise to the following image in Fig. 3.

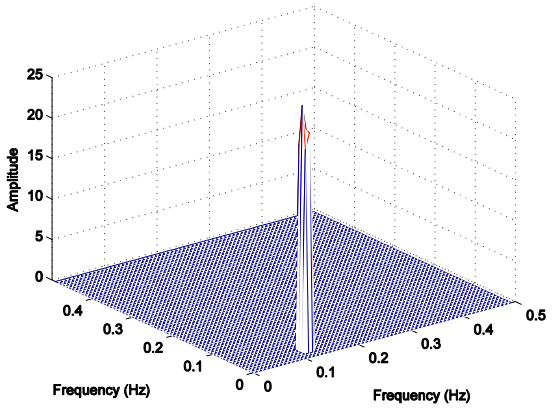


Fig. 3: Cross-bispectrum between two signals with known phase coupling

A significant peak of magnitude nearly 25 arises at the intersection of 0.03 and 0.12 Hz, indicating interaction between the two signals. However, since the cross-bispectrum is non-specific it is believed this peak is due to both non-linear and linear interactions (phase and frequency coupling). If the non-linear interactions are destroyed (using a surrogate method described in the next section), the peak is severely suppressed to a magnitude of just fewer than 4, albeit still present when there is only frequency coupling (Fig. 4). The cross-bispectrum is not normalized, and the amplitude values are of relative magnitude.

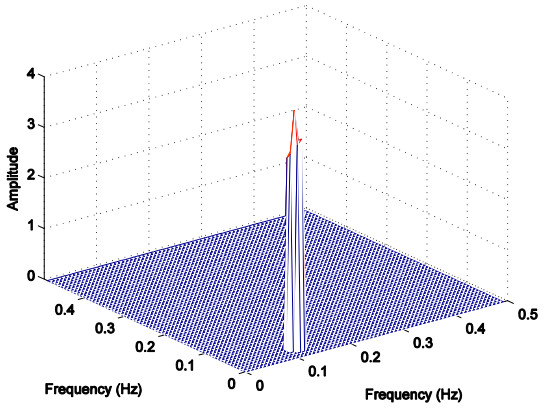


Fig. 4: Cross-bispectrum between two signals with only frequency coupling

A normalized cross-bispectrum can be performed as the cross-bicoherence index, where the result of the cross-bispectrum is divided by the products of the power spectra of the two signals evaluated at each frequency and the composite frequency. The cross-bicoherence is described by Siu and Chon [23] and calculated as in Eq. 3. The cross-bicoherence is more sensitive than the cross-bispectrum: it requires both linear and non-linear interactions to be present. Because of normalization, the amplitude is also presented with a value between 0 and 1.

$$Bic_{xy}(f_1, f_2) = \frac{|B_{xy}(f_1, f_2)|^2}{P_x(f_1)P_y(f_2)P_x(f_1 + f_2)}$$

Eq. 3: Cross-bicoherence index between two signals

The cross-bicoherence was evaluated on the two signals x(n) and y(n) as previously described and result is shown in Fig. 5. The amplitude of the peak is slightly greater than 0.9, indicating significant linear and non-linear interaction between the two signals.

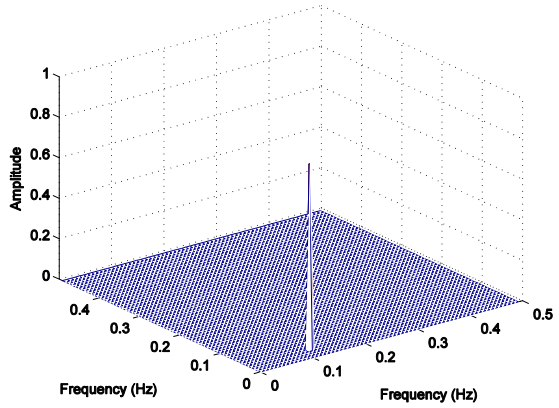


Fig. 5: Cross-bicoherence index between two signals with known frequency and phase coupling.

If the cross-bicoherence is ran again between the two signals when there are only linear interactions present, no significant peaks arise. However, a baseline floor with minimal linear interactions remains (Fig. 6).

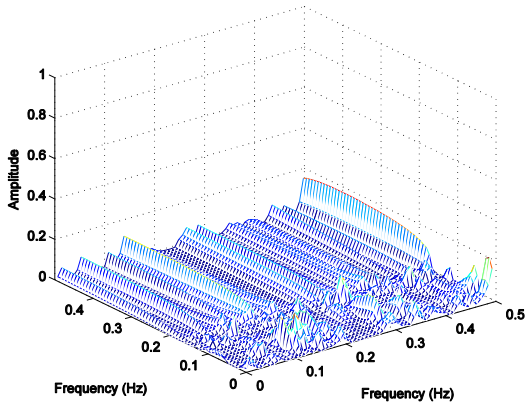


Fig. 6: Cross-bicoherence between two signals with only frequency coupling

Although the cross-bicoherence requires both linear and non-linear interactions to be present for significant peaks, there is still a portion of amplitude caused by linear interactions alone. For the scope of this project, quadratic phase coupling must be solely assessed.

In order to determine statistical significance of the cross-bicoherence result, and to eliminate magnitude due to linear interactions, a threshold must be implemented. Shils et al proposed a √3/√N threshold where N is the number of segments [24], however, this threshold is rather arbitrary and does not provide statistical significance on a frequency to frequency basis. It has been shown to erroneously reject true amplitudes in certain conditions [23]. A surrogate data threshold technique can be implemented to achieve higher specificity and sensitivity used in conjunction with cross-bicoherence.

Surrogate Data

A surrogate time-series can be generated in which the surrogate data has the same linear characteristics of the original signal, but with randomized phase. The iteratively refined surrogate data technique (IRSDT) was used as described by Schreiber and Schmitz [25]. By definition, the PSD of a signal should be identical to the PSD of its own surrogate data. The IRSDT method of producing surrogate data is even more effective because it iteratively corrects for amplitude deviations in the PSD, maintaining the correct distribution of the signal. A time-series signal including two known sinusoids of frequencies 0.03 Hz and 0.12 Hz is shown in Fig. 7 with signal length of 4096 data points.

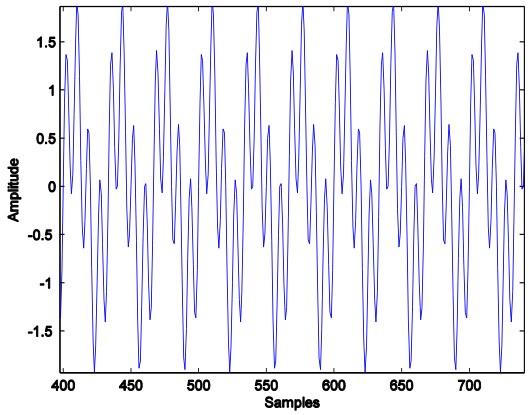


Fig. 7: Epoch of time series of two known sinusoids

The PSD for the signal was computed using an FFT and is shown in Fig. 8. The PSD presents the expected energy peaks at 0.03 Hz and 0.12 Hz.

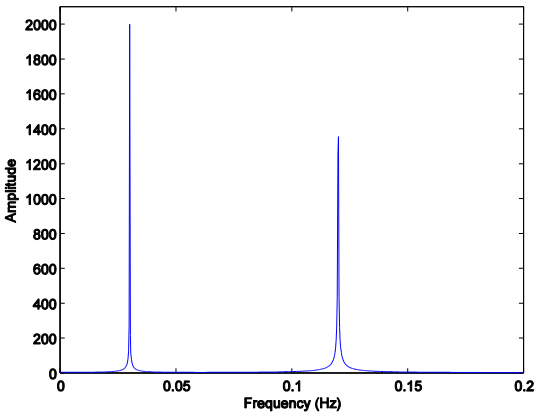


Fig. 8: PSD of random signal with two known frequencies

Surrogate data for the signal was generated using the algorithm by Gautama shown in Appendix F. As expected, when the PSD is computed for the surrogate data, a nearly identical PSD arises, as shown in Fig. 9. The PSD is a 2nd order technique, and discards any phase related data (the PSD for a sine and cosine wave of the same frequency are identical). Not only is the frequency content identical, but the spectral energy (amplitude) are equivalent as well, due to the IRSDT approach.

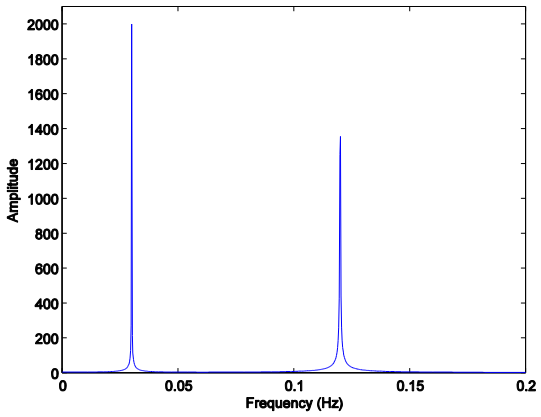


Fig. 9: PSD of signal's surrogate data

Using surrogate data the phase coupling between two signals can be destroyed, making it possible to find the linear interactions between two signals. Subtracting linear interactions from general interactions is an effective way to determine non-linear interactions: e.g. surrogate data in conjunction with cross-bicoherence between two signals can detect quadratic phase coupling.

Cross-bicoherence with Surrogate Data Threshold

Cross-bicoherence is an effective technique to determine quadratic phase coupling (QPC) between two signals. This technique of determining QPC can be performed more specifically by using a statistically determined threshold by surrogate data described by Siu et al [22, 23]. Using this method, only non-linear interaction will contribute to the final result and linear interactions are almost completely attenuated as determined by an iterative statistical approach.

To perform this strategy between two input signals: the cross-bicoherence is first run between arbitrary signals 1 and 2. Next, 100 iterations of surrogate data are generated for signal 1. The cross-bicoherence is run between each and every iteration of signal 1 surrogates vs. the original signal 2. The result of each of these cross-bicoherence values are seemingly random, but after all 100 iterations a pattern begins to emerge representing the cross-bicoherence due to linear interactions of signal 1 and signal 2. The surrogate data threshold is determined by the mean plus two standard deviations of the amplitude at each frequency combination, in order to define statistical significance. Siu et al confirmed the surrogate data realizations come from a normally distributed population using a Kolmogorov-Smirnov goodness of fit test, justifying the use of descriptive statistics mean and standard deviation which require normality [23]. Mean plus two standard deviations provides 95% confidence, for the upper limit of the normal distribution.

If this method is used on the two simulated signals with frequency and phase coupling previously described, the result will emerge with a peak due to only non-linear interaction (Fig. 10). The result looks very similar to that in Fig. 5 when the cross-bicoherence was used alone, although amplitude is slightly less (roughly 0.8 vs. 0.9) due to the subtraction of the linear interaction, given by the surrogate data statistical threshold (approximately 0.1 at the desired frequency location). The

amplitude of a peak from cross-bicoherence alone has both linear and non-linear contribution. The surrogate data threshold will eliminate the linear contribution, providing an efficient way to evaluate non-linear interactions alone.

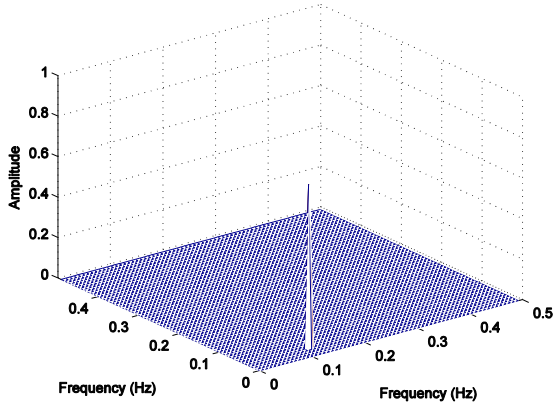


Fig. 10: Cross-bicoherence after surrogate data threshold

Of course if the method of cross-bicoherence with surrogate data threshold (CBicS) is used on two signals which are only frequency coupled and do not possess phase coupling, there will be no significant results as shown in Fig. 11.

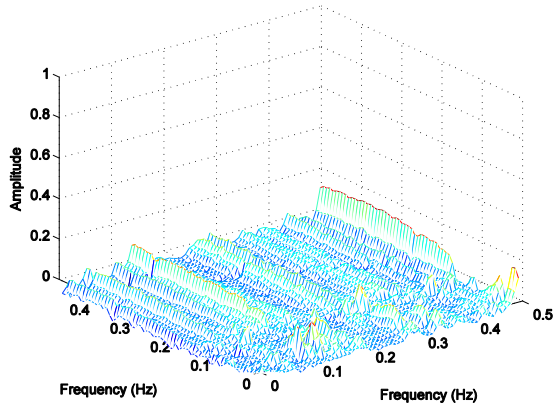


Fig. 11: CBicS between two signals with only frequency coupling

Although the result is not flat, there are no significant peaks. In the absence of phase coupling, the cross-bicoherence with surrogate method will not provide any results, even when there is frequency coupling. Surprisingly, this result between two signals with no phase coupling is not zero. Thus, an additional global cutoff value must be implemented. This global cutoff can be carefully selected in order to discard any floor measurements, but leave clearly prominent peaks.

The cross-bicoherence gives a normalized coupling magnitude between the values of 0 and 1. If the magnitude at a particular combination was already low, say 0.3, and the surrogate threshold for that value was only 0.1, then the coupling magnitude would still be greater than 0. This further exemplifies the need for a global cutoff, to eliminate any low magnitude interactions. A global surrogate threshold should not be used, because a surrogate threshold is determined at every frequency combination unique for each interaction between the two signals.

Robustness of the Cross-bicoherence with Surrogate data

It has previously been shown that given the choice between cross-bicoherence, cross-bispectrum alone, or either option with surrogate data, that it is most effective to use cross-bicoherence with surrogate data (CBicS) [23]. This method is the least susceptible to low SNR and does not superfluously identify linear interactions when there is a lack of non-linear interaction. Cross-bicoherence index is also preferred to the cross-bispectrum because it provides more meaningful amplitude, normalized from 0 to 1. Cross-bicoherence index also is less susceptible to variability in segment length when compared to the cross-bispectrum [23]. Following the notion CBicS is the most efficient cross-bispectral method, it is tested in the following section.

CBicS was performed between two signals generated with frequency and phase coupling as in Appendix A with a segment length of 128. Additive, Gaussian white noise was added to each of the signals to simulate varying levels of SNR. The algorithm was performed at each 1 dB step from -30 to +30 dB SNR and detected frequencies are shown in Fig. 12 and Fig. 13.

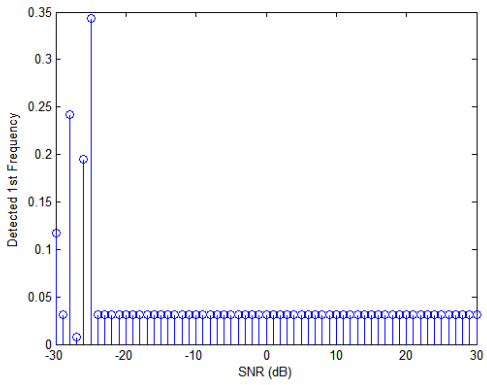


Fig. 12: Effect of SNR on CBicS (f1)

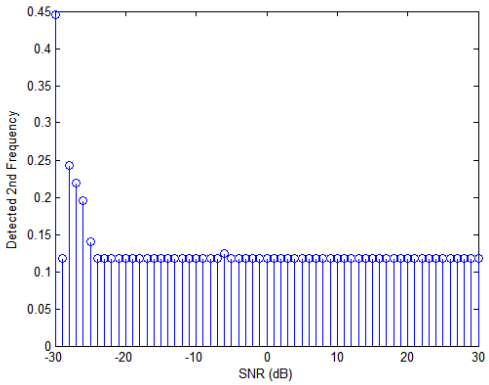


Fig. 13: Effect of SNR on CBicS (f2)

The correct detected frequencies should be 0.03 Hz and 0.12 Hz for f1 and f2 respectively. The frequencies were detected correctly for an SNR as low as -25 dB. Further evaluation of the robustness of the CBicS was determined by Siu and Chon, and in fact they claim the method can be accurate for SNR as low as even -30 dB [23]. They evaluated the effect that signal length can have on the various methods (cross-bicoherence alone, cross-bicoherence with surrogate threshold, and cross-bispectrum with surrogate threshold). They also evaluated the requirement of percentage of coupling between two signals. The cross-bicoherence with surrogate threshold proved to be the most accurate and sensitive of the three methods in all three tests: varying noise levels, signal length, and percentage of coupling [23].

Example: Surrogate-Data Threshold with Cross-Bicoherence

Throughout the next section, the process and plots will be shown for an iteration of CBicS between two different cardiac electrograms and will help solidify the necessity of a surrogate-data threshold for use with cross-bicoherence.

First, the cross-bispectrum is taken between the two channels. For this example, the 991_82_RB (Dog 4, 10 nM treatment, epoch at 20 minutes) dataset was randomly selected, between channels 42 and 68 as signal 1 and 2 respectively. Both channels contain 6.25 Hz content which is phase coupled between the two.

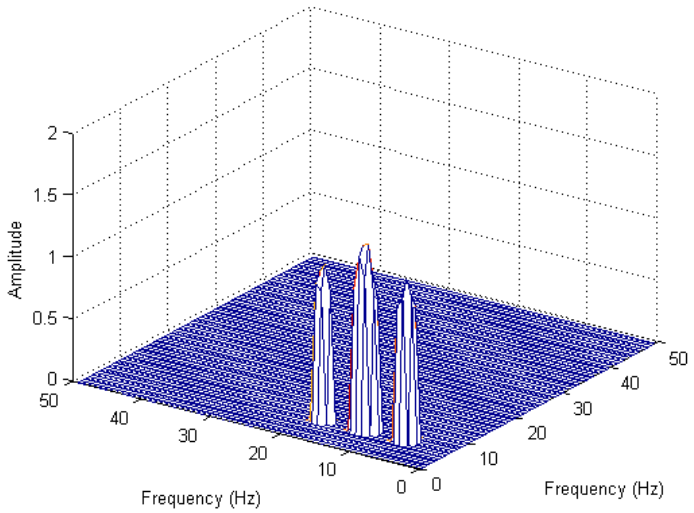


Fig. 14: Cross-bispectrum peaks

The cross-bispectrum found several significant peaks, a few which are illustrated in the first quadrant above. The highest peak was found at 11.7 Hz and 6.25 Hz. The desired peak is also visible at 6.25 Hz and 6.25 Hz, however, the magnitude of the peak is slightly lower, and is normally not easily recognized. Regardless, the locations of all peaks found from the cross-bispectrum are noted, and used later in verification.

Next, the cross-bicoherence is run between the two signals. (Fig. 15) There are several more peaks identified, including both non-linear and linear interactions. The ability to discern between the two is not yet possible without the introduction of surrogate data. If simply the maximum peak is taken, the strongest bicoherency was found between frequencies 6.25 Hz and 12.5 Hz. These frequencies are harmonics: they must have frequency and phase coupling components. However, with cross-bicoherence alone there is no way to determine the contribution from each. Thus there is a requirement for CBicS.

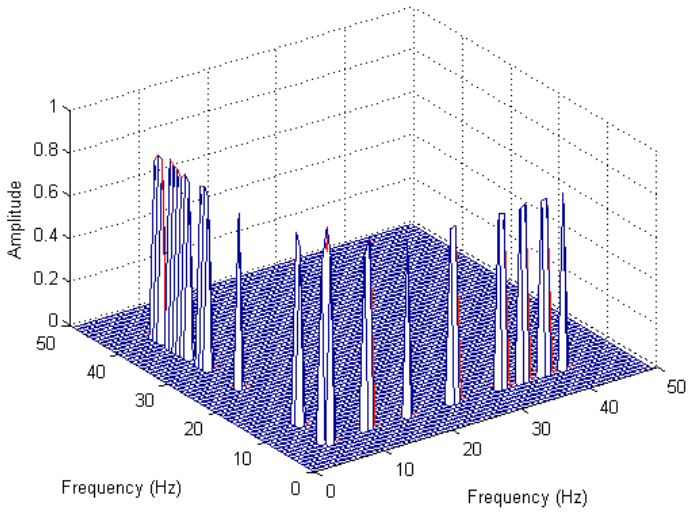


Fig. 15: Cross-bicoherence peaks

100 different iterations of surrogate data is generated for signal 1. The surrogate data contains all of the same linear characteristics of signal 1, but with randomized phase. The cross-bicoherence is run between an iteration of signal 1 surrogate data with signal 2. The result of one run is shown in Fig. 16.

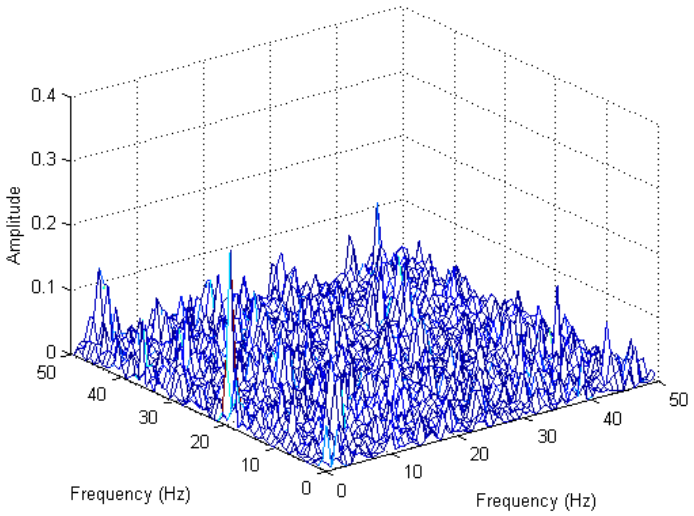


Fig. 16: Cross-bicoherence between Signal 1 Surrogates and Signal 2

Clearly after iteration, the results look random and no visible peaks appear. However, after all 100 iterations of signal 1 surrogate data are cross-bicoherenced with signal 2, patterns begin to emerge and peaks are identified for linear interactions only. The mean of the amplitudes is taken at each frequency, plus two standard deviations. The visible peaks are shown in Fig. 17, representing locations of linear interaction between signal 1 and signal 2.

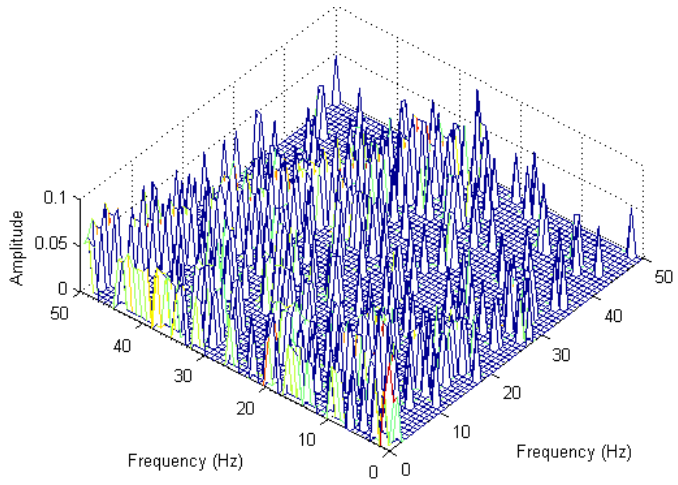


Fig. 17: Cross-bicoherence with Surrogates Statistical Averaging

If the linear interactions are subtracted from the raw cross-bicoherence, a single peak occurs in the first quadrant as shown in Fig. 18. To improve specificity, only the locations at which there were peaks found in the cross-bispectrum are used.

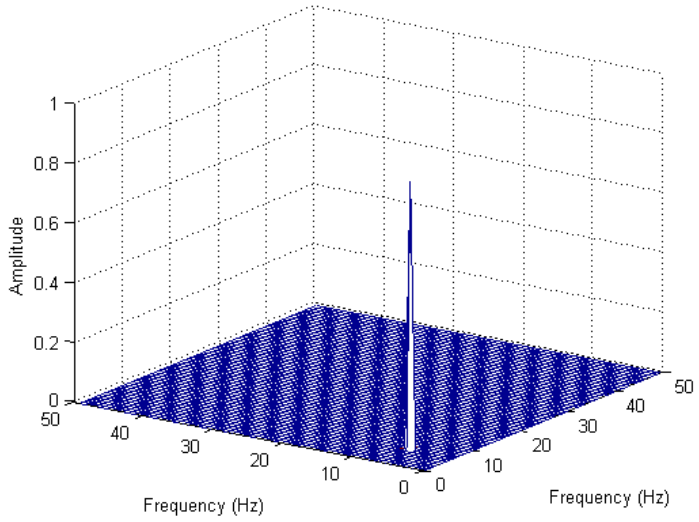


Fig. 18: Cross-Bicoherence after Surrogate Threshold

The peak shown in Fig. 18 represents the primary location at which two frequencies are phase coupled between signal 1 and 2. The peak is at 6.25 Hz and 6.25 Hz with amplitude of 0.8974 after threshold subtraction, with a true bicoherency peak of 0.9918. Cross-bicoherence using a surrogate threshold is the most specific method at identifying interaction between two phase coupled. If used with the absence of a surrogate data threshold, cross-bicoherence will give rise to peaks with amplitude contributed by both linear and non-linear interaction. If only quadratic phase coupling interactions are desired, it is absolutely necessary to employ a statistical surrogate time-series threshold used in conjunction with cross-bicoherence.

II. Methods

Data Collection

The data presented was collected from four dogs under the care of Thomas Everett et al under Jeffery Olgin's group at University of California: Berkeley (Berkeley, CA). First, congestive heart failure (CHF) was induced by rapid ventricular pacing via a lead in the right ventricle at 240 BPM (4 Hz) for four weeks. The CHF model has been previously shown to create atrial conduction abnormalities detected by epicardial mapping [26]. Episodes of AF were initiated using rapid atrial burst pacing from the left atrium using the following parameters: cycle length of 50 ms (20 Hz), pulse width of 9.9 ms, and output of 9.9 mA. The AF episodes were recorded with epicardial surface electrode plaques first with no treatment, and after with 10 nM and later 300 nM doses of rotigaptide treatment, taken at varying intervals described in Appendix G.

The 90 electrode epicardial surface plaque was placed on the right atrium spanning the right atrial Bachman's bundle (RBB) spanning inferiorly to the medial right atrial appendage (MRAA). The electrode configuration is shown in Fig. 19. The data was collected at a sampling rate of 2000 Hz for epochs up to 50 seconds in length.

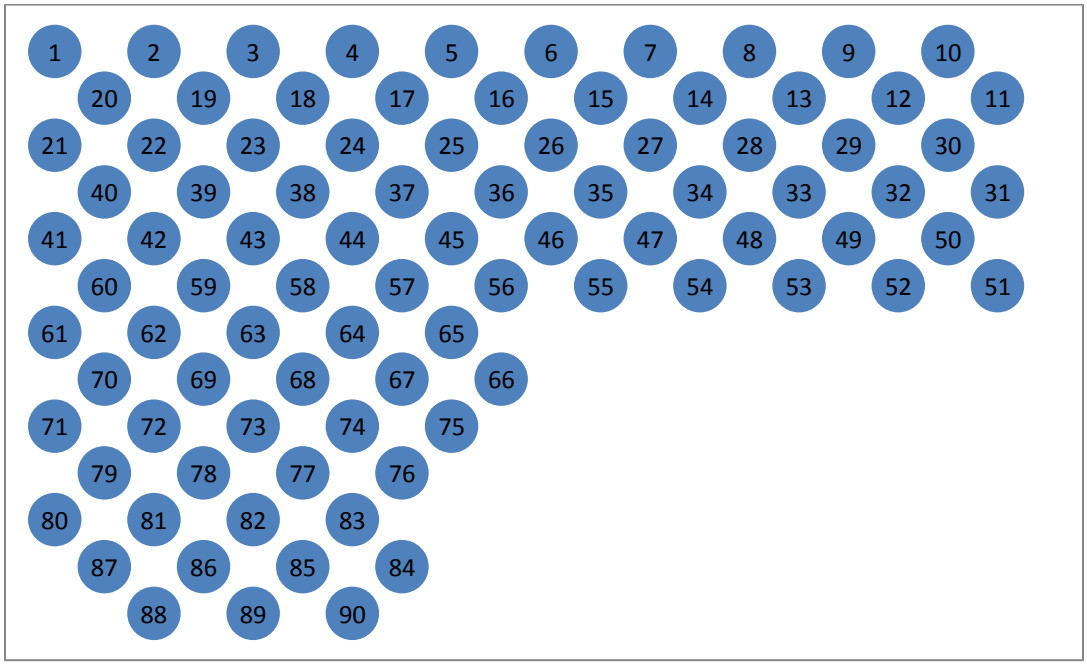


Fig. 19: Electrode plaque configuration for superior RA

Cross-bicoherence with Surrogate Data Algorithm

Materials

A 64-bit Windows environment running 64-bit MATLAB 7.11.0.584 (2010b) was used on a dual Intel Xeon (X5680) CPU platform. The original electrode plaque for the superior right atria (SRA) contained 90 channels. MATLAB supports up to eight threads per algorithm, so for symmetric data distribution the number of channels must be factorable by eight. For multi-threading purposes the last two electrodes were discarded for a total of 88. The data is split into eight even portions and is computed in parallel. The average computation time is between 8 and 12 hours for an 88 channel dataset, 50 seconds long, with a down-sampled rate of 100 Hz. The recommended data length for cross-bicoherence is about 4096 samples; therefore the length of 5000 provided ample information for the calculation. The algorithm was run between every possible permutation of the 88 channels, except the case where an electrode would be analyzed against itself, for a total of 7,656 runs per dataset (88x88-88).

Process

The data is first down-sampled using a built-in MATLAB function: decimate. It is recommended to avoid down-sampling greater than a factor of 13, so the data is down-sampled in two steps: once by a factor of 10, and then by a factor of 2, for a total factor of 20. The decimate function automatically implements a low pass Chebyshev Type I filter with cutoff frequency of 80 Hz for the decimation down to 200 Hz and cutoff frequency of 40 Hz before decimating to 100 Hz. The 8^{th} order filter is implemented in both forward and reverse directions for zero-phase implementation, doubling the order to 16. Initial sampling rate was 2000 Hz, after decimation it is 100 Hz. The down-sampled data is then divided into eight slices.

After the data is evenly divided, a CPU thread is assigned to each slice using the single-process, multiple-data functionality (spmd) in MATLAB. The main program call can be found in Appendix B. The main algorithm (see flow chart in Appendix N part 1) begins with the calculation of a cross-bispectrum using a slightly modified version of bispecdx.m (Appendix C) from the High-Order Spectral Analysis (HOSA) Toolbox by Swami [27]. The cross-bispectrum was modified to incorporate a different sampling rate and also to disable any plotting features included. The following parameters were used for the cross-bispectrum: an FFT length (NFFT) of 128 samples, segment length of 128 samples (optimally it is equal to NFFT), window size of 5, and 50% overlap between segments. Given the sampling frequency of 100 Hz and defined NFFT, the frequency resolution is 0.78125 Hz. NFFT is best chosen as a power of 2 for maximum efficiency, although if the next step of 256 is chosen, the time to compute the algorithm roughly doubles. The choice of 128 samples gives a fair balance between computation time and frequency resolution. The window length of five is the default value and is used as the length of the side of the Rao-Gabr optimal window.

The cross-bispectrum is the most general form of coupling detection, thus maximum numbers of peaks appear. The peaks are checked to ensure they are the local maxima, and the location of the peaks are noted for further reference henceforth referred to as the significant peaks.

Next the cross-bicoherence is run between the two signals. Similarly a modified version of the bicoherx.m ([27]) is used with sampling rate input modification and disabled plotting (Appendix D). The cross-bicoherence was used with the same parameters as mentioned above for the cross-bispectrum, with the only difference being that of the window. The cross-bicoherence algorithm uses a Hanning window, and thus the empty set ([]) variable was used for the default value by the script. The cross-bicoherence result data is extracted only at the significant peaks that have been previously determined. Surrogate data is generated (Appendix E) for signal 1 using a version of generate_surrogate.m (Appendix F) to collect 100 different iterations of surrogate data all with randomized phase. The cross-bicoherence is performed between every iteration of signal 1 surrogate data with signal 2. The result of the cross-bicoherence is the linear interaction between signal 1 and 2, all non-linear characteristics are suppressed. Every permutation of channels is performed for the overall algorithm: every electrode has a chance at being signal 1 since the cross-bispectral techniques are unidirectional. There is no advantage to generating surrogate data for signal 2. The phase is randomized in signal 1; therefore non-linear interactions are already destroyed.

The threshold value is then found at each of the locations of the significant peaks by taking the mean plus two standard deviations. Global maximum peak was not used; rather each frequency location (at significant peak) had its own threshold value because of fundamental amplitude variation at different frequencies from the cross-bicoherence.

Lastly, the values from the original cross-bicoherence output between signal 1 and 2 are subtracted by the corresponding surrogate data statistical threshold values. The number of significant peaks from the cross-bispectrum should always be equal to or greater than the number of significant peaks found after the cross-bicoherence with surrogate data. Because the threshold from linear interactions is subtracted from the output of the original cross-bicoherence, any value greater than 0 suggests there are non-linear interactions. However, due to the sheer large number of significant interactions even after threshold implementation, only amplitudes relatively high represent significant interaction. Despite the statistical surrogate data threshold, an additional threshold must be user-defined for practical analytical purposes. The amplitude can take on any value from 0 to 1, and sometimes even greater than 1 in high correlation scenarios. There is no upper limit to the result as mentioned by Hinich et al [28]. Previously, global surrogate data threshold has been used [23]. However, the cross-bicoherence weighs the amplitude differently for different frequency combinations due to the natural inverse relationship between amplitude and frequency when traversing into the frequency domain. Elgar & Guza have previously reported this phenomena in 1988 when using the bicoherence function [29]. Therefore, it is more accurate to use varying surrogate data threshold values for each individual frequency location. However if there is low initial output, and the surrogate data threshold is also low for that frequency combination, a secondary, greater threshold must be used in order to suppress those insignificant interactions rather than taking any result above 0 even after surrogate

threshold subtraction. Threshold values after surrogate threshold subtraction used for analysis were 0.5 and 0.8, for example.

The final result shows the two frequencies found to be phase matched between the two signals. By definition, the frequencies which are phase coupled will always be either the same value, or a harmonic of each other. The resulting cross-bicoherence amplitude is after threshold subtraction.

Method of Analysis

In order to quantify the organization of the atrium during AF, time-frequency analysis as well as the implementation of an organization index was used.

Time-frequency analysis was first used in order to determine dominant coupled frequencies (DCFs) as determined from the CBicS output using code in

Appendix H. An ideal "low-pass filter" is implemented with a cutoff value of 20 Hz in order to prevent the ventricular pacing frequency from being reported as the DCF. For some of the episodes, the pacing frequency completely masked the other frequencies of interest, as shown in Fig. 20. It is important to note that the DCF is almost always vastly different from the true DF found from the PSD, because the DCFs found from CBicS are phase coupled and often represent only relatively low spectral energy throughout the PSD.

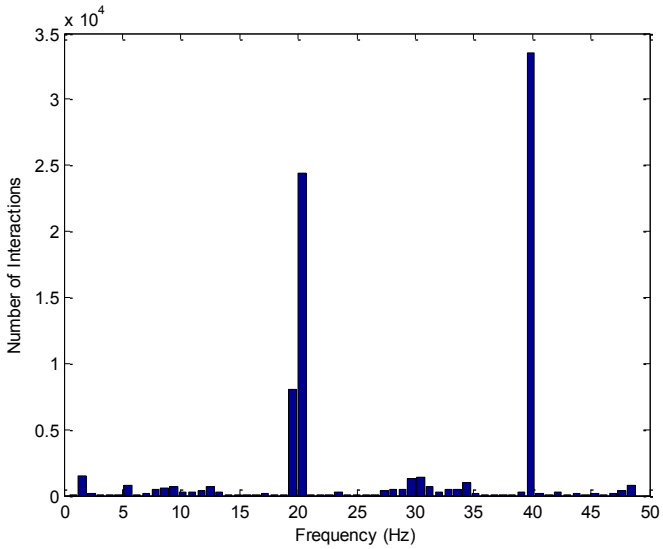


Fig. 20: Dominance of pacing frequency

Organization index was implemented at specific frequencies. Dominant and significant frequencies were determined using pseudo-time-varying cross-bicoherence algorithm (along with surrogate data threshold). Background frequency content was analyzed by summation of coupling incidence. The program found in Appendix I determines which electrograms are in synchronization, as determined by the DCF and its harmonics (also see flowchart in Appendix N, part 2). The cross-bicoherence organization index (CBOI) was found by dividing the number of channels phase coupled at the desired frequency or its harmonics, divided by the total number of channels as shown in Eq. 4.

$$CBOI = \frac{\#coupled\ electrodes\ at\ DCF}{total\ \#\ channels}$$

Eq. 4: Cross-Bicoherence Organization Index

An alternative method was also used, more broad and less specific, but more widely applicable for all mechanisms of AF even when a DCF is not determined.

$$CBOI = \frac{\#electrodes\ with\ amplitude > 0.8}{total\ \#\ channels}$$

Eq. 5: Cross-Bicoherence Organization Index 2

Statistical Analysis

Statistical significance was determined by the distribution free Wilcoxon Rank-sum test (also known as Mann-Whitney Wilcoxon test). The results that were analyzed do not follow normal distribution as confirmed by both Shapiro-Wilks and Kolmogorov-Smirnov tests, so a non-parametric test was found appropriate for comparisons.

III. Results

Time-Varying Cross-Bicoherence Analysis

The first step in analysis is to sort through the amplitude output of the cross-bicoherence with surrogates (CBicS) and determine an appropriate threshold for significance. Pseudo time-varying cross-bicoherence was performed in order to view the coupled frequency content over time by using the output of the CBicS with sliding time windows (with no overlap, because the epochs are separate). First, a global 0.5 amplitude threshold was implemented (after surrogate threshold subtraction) shown in Fig. 21.

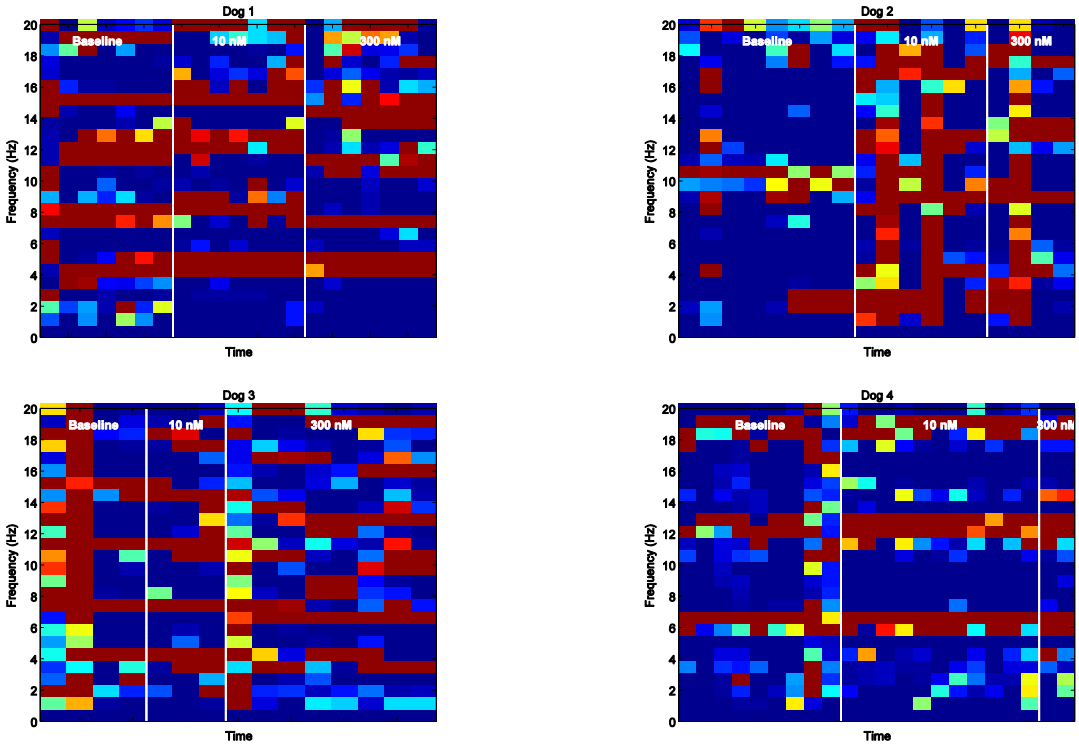


Fig. 21: Time-varying CBicS output for Dogs 1-4 with 0.5 amplitude threshold

Dark blue represents none or low coupled frequency content. Dark red shows the most common frequencies involved with phase coupling (≥ 176 instances). The time axis represents the order of collected electrograms as described in Appendix G (taken every 1-5 minutes on average). From left to right: baseline AF, 10 nM rotigaptide treatment, and 300 nM treatment (separated by white lines). It was not possible to implement a true time-varying cross-bicoherence algorithm because of the irregularity of the acquisition of data. Recordings were not always taken as regular intervals, due to unsustainability of the AF episode. The only Dog which was able to sustain all 3 AF episodes (baseline, 10nM, 300nM) without interruption was Dog 1. Each dataset for Dog 1 was collected at 5 minute intervals from 0-30 minutes.

Even after surrogate threshold subtraction and 0.5, there is still a large amount of coupling present in the signals. To further isolate the more significant interactions, an additional threshold of 0.8 was implemented. The results of the time-varying cross-bicoherence analysis after the 0.8 amplitude threshold are shown in the following figure.

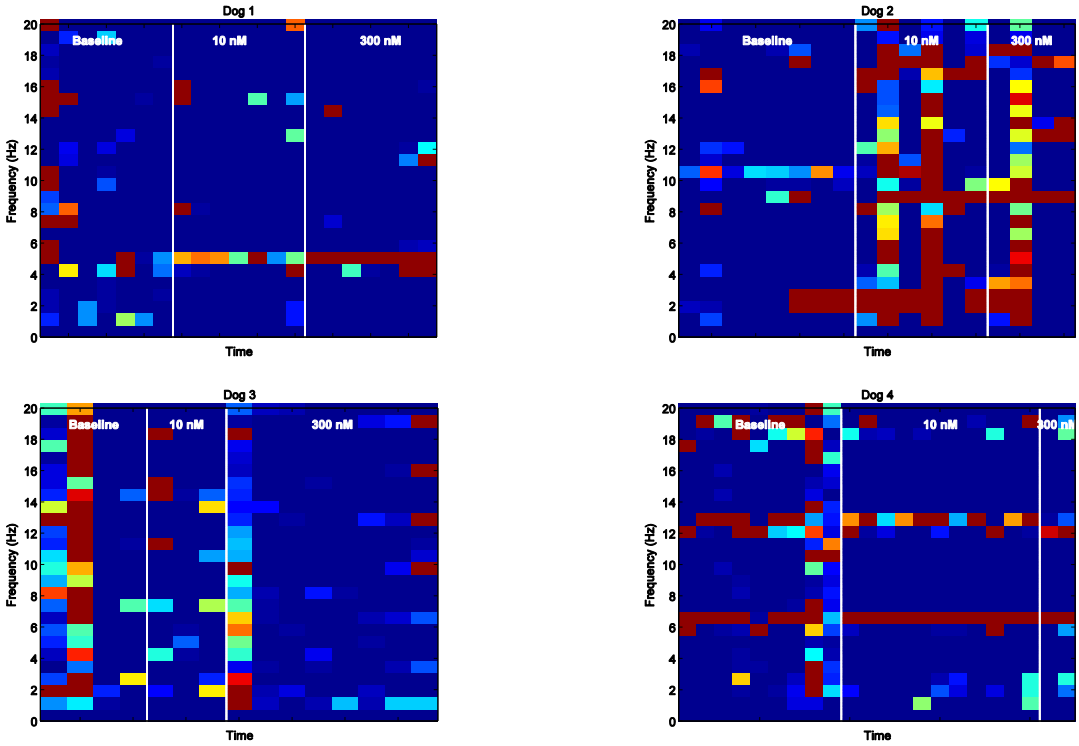


Fig. 22: Time-varying CBicS output for Dogs 1-4 with 0.8 amplitude threshold

By using amplitude of 0.8, the significant frequency content is much more visible. For Dog 1, there was spectral content in the 6-14 Hz band at baseline, most of which is attenuated after the administration of rotigaptide. There is a consistent frequency of 4.6875 Hz over time, becoming more apparent in 10 nM, and even more apparent in 300 nM. This frequency for Dog 1 is evaluated later. Dog 2 has little frequency content at baseline, and with the introduction of the drug begins to show more coupled spectral energy across nearly the entire bandwidth. 1.5625 Hz was the most consistent frequency across time, though not as prominent as the respective DCFs in Dog 1 and 4. Dog 3 had spectral content across the entire band for a few baseline datasets and the first 300 nM dataset, but no consistent frequencies over time. Dog 4 had a consistent frequency of 6.25 Hz over time. Additionally, Dog 4 had spectral power in many bands for some of the baseline datasets.

The dominant coupled frequency for each dog is shown in Table 1 with error up to ± 0.78125 (Frequency resolution). Dog 2 and 3 did not have a single dominant frequency across all epochs.

Table 1: Time-varying CBicS determined Dominant Coupled Frequencies

Dog	DCF (Hz)
1	4.6875
2	N/A
3	N/A
4	6.2500

The dominant frequencies found from time-varying cross-bicoherence analysis were used as a foundation for the implementation of a more stringent form of quantifying organization during the AF episodes: an organization index.

The characterization of background frequencies (all those except the DCF) is also provided in Table 2. An interaction involves channel to channel coupling with an amplitude greater than 0.8. The effect from no treatment to treatment is noted.

Table 2: Total number of interactions involving background frequencies

Dog	Baseline	10 nM	300 nM	Effect
1	39,877	1,703	1,908	Suppression
2	10,843	28,375	28,719	Additive
3	11,484	2,449	3,737	Suppression
4	21,874	5,626	667	Suppression

Organization Index

For Dog 1, the previously reported frequency of 4.6875 Hz was checked for synchronization. With an amplitude threshold of 0.8, the mean ± SD of organization is depicted for Dog 1 in Fig. 23 and given in Table 3.

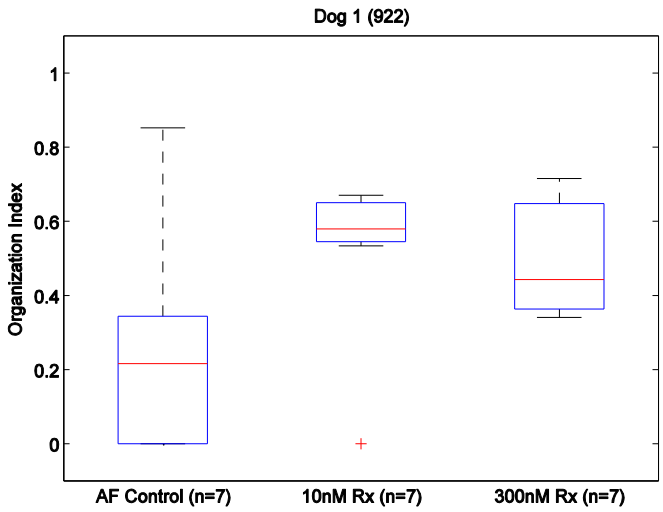


Fig. 23: Organization Index at 4.6875 Hz for Dog 1

A rank-sum test between the AF control and 10nM treatment reveals a p-value of 0.0979. However, if the outlier is removed (as determined by Q1-1.5*IQR) the p-value decreases to 0.0280. If AF control is compared to 300nM, the p-value is 0.0460. 10nM vs. 300nM treatment reveals a 0.2145 p-value.

A similar procedure was followed for Dog 4, but with the frequency of 6.25 Hz as determined from the time-frequency analysis. The mean ± SD of organization for Dog 4 is given in Table 3 and boxplots available in Fig. 24 . Rank-sum between AF control and 10nM, AF control and 300nM, and 10nM vs. 300nM all gave p-values > 0.05.

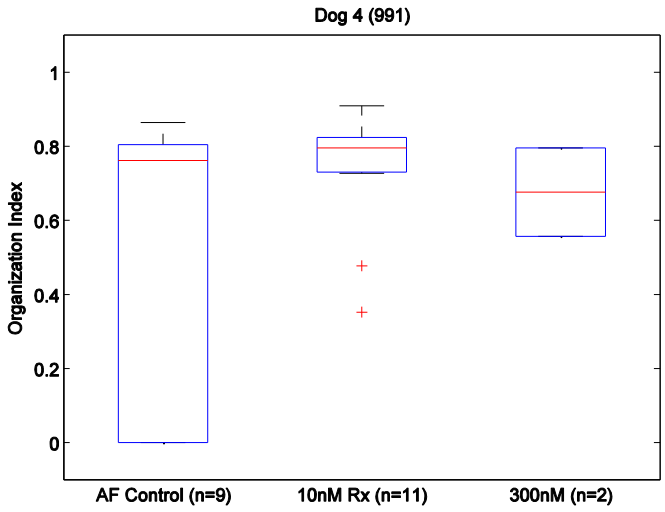


Fig. 24: Organization Index at 6.25 Hz for Dog 4

Although not as dominate as the respective DCFs in Dog 1 and 4, in Dog 2 there was moderate consistency at 1.5625 Hz. Synchronization was checked and reported in Table 3. There was very little 1.5625 Hz content in Dog 2 initially, but after the introduction of the drug the frequency began to emerge, albeit at not very high organization (Fig. 25).

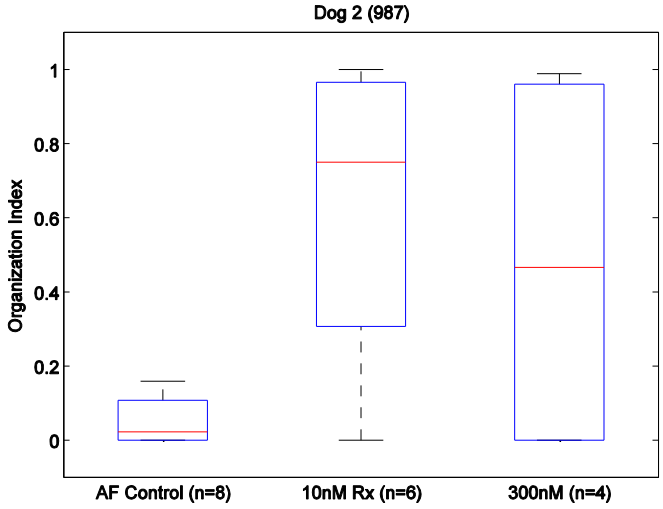


Fig. 25: Organization Index at 1.5625 Hz for Dog 2

The rank-sum test gave a p-value of 0.0173 when comparing AF control to 10nM for Dog 2 at 1.5625 Hz. Although 300nM had two datasets with greater than 0.9 organization, it also had two datasets with 0 organization at 1.5625 Hz, leading to a >0.05 p-value when compared to either the AF control and 10nM sets.

Table 3: Mean Organization Index of Right Atrium (Mean ± SD) with Set Frequency

Dog	AF Control	10 nM	300 nM	Freq. (Hz)
1	0.24 ± 0.3	0.52 ± 0.2	0.49 ± 0.2	4.6875
2	0.05 ± 0.1	0.63 ± 0.4	0.48 ± 0.6	1.5625
4	0.45 ± 0.4	0.73 ± 0.2	0.68 ± 0.2	6.2500

For further evaluation, the global amplitude threshold was dropped back down to 0.5 in order to allow more coupled frequency content for analysis. Almost all of the sets have shown at least some degree of organization when looking at 3.90625 Hz with an amplitude threshold of 0.5, except Dog 2 AF control. If the threshold is increased to 0.8, almost all of the 3.90625 Hz is suppressed. The hearts were previously paced to 240 BPM (4 Hz) for four weeks in order to induce CHF, and it is believed this frequency is still intrinsically present in the cells. There was no significant difference among treatment groups other than Dog 2 AF control vs. 10nM (p-value 0.0300) and 300nM (p-value 0.0364).

Table 4: Mean Organization Index of Right Atrium (Mean ± SD) at 3.90625

Dog	AF Control	10 nM	300 nM
1	0.60 ± 0.2	0.73 ± 0.1	0.67 ± 0.2
2	0.00 ± 0	0.17 ± 0.2	0.23 ± 0.2
3	0.43 ± 0.2	0.41 ± 0.4	0.58 ± 0.3
4	0.15 ± 0.2	0.04 ± 0.1	0.26 ± 0.1

Dog 3, which did not show any consistent frequency content over time with the 0.8 threshold, showed a consistent organization of mean greater than 40% for both no treatment and treatment at 3.90625 Hz with a threshold of 0.5.

Rather than look at organization for specific frequencies, the organization can also be determined by simply counting the number of channels that possess amplitude greater than a certain value, say 0.8. This method of determining the organization index gives an overall picture of how organized the atrium is under certain time and treatment conditions, regardless of the frequency prominence and/or suppression caused by treatment (Table 5). The boxplots showing the organization index for the consolidated epochs by treatment groups are shown in (Fig. 26). The consolidation includes epoch recordings from sustained AF episodes as well as individual AF episodes (where there is only one epoch from a particular AF episode). Refer to Appendix G for the summary of acquired data.

Table 5: Organization Index of RA with no specified frequency with 0.8 amplitude cutoff (Mean ± SD)

Dog	Control	10 nM	300 nM
1	0.81 ± 0.2	0.88 ± 0.1	0.98 ± 0.02
2	0.60 ± 0.3	0.98 ± 0.04	1.00 ± 0.006
3	0.71 ± 0.4	0.85 ± 0.1	0.63 ± 0.2
4	0.93 ± 0.1	0.79 ± 0.1	0.74 ± 0.2

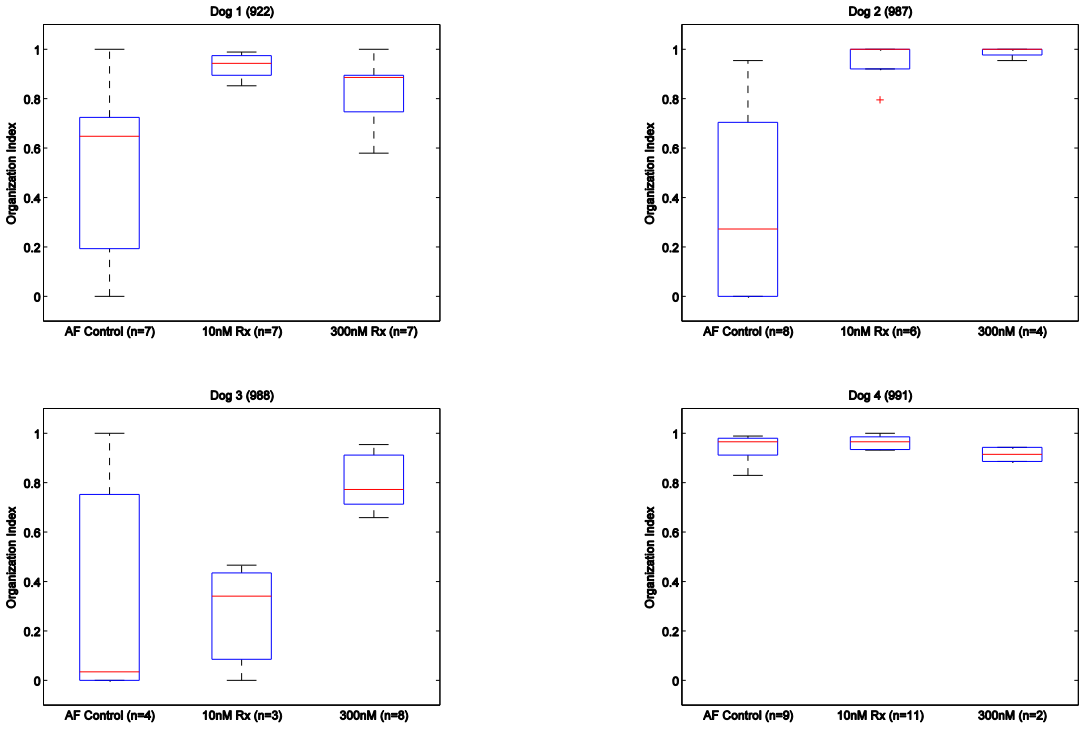


Fig. 26: Boxplot of average organization index for Dogs 1-4 (no specified frequency)

A rank-sum test was run between every possible combination (Table 6):

Table 6: Summary of Rank-sum results for organization indices (P-values)

Dog	Control vs. 10 nM	Control vs. 300 nM	10 nM vs. 300 nM
1	0.0239*	0.0769	0.1265
2	0.0047*	0.0081*	0.6670
3	1.000	0.1364	0.0167*
4	0.3772	0.5091	0.2051

IV. Discussion

Dominant Frequency Prominence & Secondary Frequency Suppression

The frequency content of Dogs 1 and 4 stabilized over time with the introduction of rotigaptide treatment. Not only did 4.6875 Hz and 6.25 Hz have a consistent presence over the time-varying cross-bicoherence analysis, but with the introduction of the drug other secondary frequency content was suppressed as well as shown in Table 2. Dog 2 had nearly no coupled 1.5625 Hz content at all, and with the introduction of the drug the frequency was introduced; although rather than observing secondary frequency suppression, new frequencies were introduced with the treatment as well. With Dog 3, no consistent frequency was observed over time with the 0.8 amplitude threshold. However, secondary frequency suppression was also observed as in Dog 1 and Dog 2.

Organization Index

The average organization is in contrast to the reported organization index determined by Everett et al of 0.3 for AF control and for both treatment levels. They used a vastly different method to determine an organization index, by observing the alteration of frequency over time using harmonic analysis developed by Botteron and Smith[20]. Everett et al could not show any significant difference in organization between control and rotigaptide treatments in the right atrium during AF [unpublished].

Not all of the rank-sums show a significant difference between control and treatment groups: the boxplots in Fig. 26 and increasing mean organization in Table 5 show the organizational improvement. The organization of the right atrium during AF may already have some inherent organization, however with great variance. If the organization is already high (Dog 4), then the treatment will have little improving effect. In these scenarios, the organization during AF is already high, causing the rank-sum test to show no significant difference between no-treatment and treatment groups. The organization of the atrium with rotigaptide treatment is more stable with increasing rotigaptide doses, showing a consistently higher organization factor over all epochs, except in the case of Dog 1, where 300 nM had a slightly negative global organization effect when compared to 10 nM.

When organization is evaluated with the amplitude cutoff of 0.5 and at 3.90625 Hz, both Dog 1 and Dog 3 had organization over 40% for all datasets, with or without treatment. Therefore, AF does still have organization even without treatment, especially due to cell memory. The myocytes were tachypaced at 4 Hz for four weeks. Dog 2 showed significant improvement in organization with treatment at this frequency range, and Dog 4 showed little effect to the treatment at this frequency range.

Frequency Phenomena

Most of the datasets have shown some degree of organization at 3.90625 Hz as shown in Table 4 with amplitude cutoff of 0.5. With a cutoff of 0.8, most of the 4 Hz content is attenuated. Prior to the AF episodes, the hearts were paced at 240 BPM (4 Hz) for four weeks. It has been previously shown that the atrial substrate adjusts to long-term effects [30]. The significance of other dominant coupled frequency values is unknown (4.6875 Hz, 1.5625 Hz, and 6.2500 Hz).

Mechanisms of Atrial Fibrillation

The different frequencies present and variation in response to rotigaptide imply different AF mechanisms. Dogs 1 and 4 had prominent, steady dominant coupled frequencies that would re-enforce the mother rotor theory [31-34]. However the sheer high number of other coupled frequency content before treatment could imply the multiple-wavelet theory [35]. A more rigorous evaluation would have to be performed with higher temporal resolution in order to determine conduction vector paths to support mechanistic theories.

Regardless, the similarity in response by Dogs 1 and 4, and possibly 3, imply similar AF mechanisms. Dog 2's unique response could imply a different mechanism, further supporting the theory that AF mechanisms are not independent nor concrete, and could be additive and/or distinctive in nature [34].

V. Conclusion

Adding rotigaptide significantly increases the organization of the right atrium during episodes of AF due to steadying of a dominant coupled frequency (n=2 of 4) and elimination of secondary frequencies (n=3 of 4). It maintains or improves organization of the atrium using a cross-bicoherence with surrogate threshold based organizational metric (n=4 of 4). This improved organization may lead to greater electrical cardioversion efficacy.

Steadying of the dominant frequency with treatment was observed in two dogs. A higher number of channels are phase matched with the rotigaptide treatment as shown by the Rank-sum test for one dog (Dog 1). However in a dog (Dog 4), the organization at the DF did increase although it was not found to be statistically significant due to baseline organization already being high. Suppression of secondary frequency content was also observed in Dog 1 and Dog 4, in addition to Dog 3.

In one dog (Dog 2), the rotigaptide treatment introduced more background coupled spectral content. However, the frequency non-specific, global organization was improved. Global organization was improved in Dog 1, 2, and 3. It stayed steady in Dog 4, because the organization of the baseline AF was already high.

Although Rotigaptide may provide an overall increased organization to the atrium during fibrillation, clearly it is not able to act as a pure anti-arrhythmic to halt the episode in the CHF model. Its gap junction coupling properties have been shown to suppress AF vulnerability in acute ischemia models but not CHF [13, 15]. The gap junction coupling properties do lead to conduction velocity increase [16]and as a result greater global organization.

Downfalls

The frequency resolution of 0.78125 Hz is very rough, which has a negative quantizing effect. Also, this project only evaluates the right atrium; it omits the left atrium although the data under the same conditions is available.

When Botteron and Smith used their method of measuring organization index, they found the average organization was between 0.32 and 0.54 for AF compared with 0.91 and 0.95 for sinus rhythm [21]. Due to the lack of availability of sinus rhythm data, it is unknown how the CBicS algorithm would perform under sinus rhythm. However it can be hypothesized that at minimum 90% organization would be expected.

Rotigaptide has poor oral bioavailability due to enzymatic degradation and inadequate mucosal penetration[4], however other drug delivery methods may prove to be reliable and effective. It is also currently unknown if rotigaptide would provide any preventative measure of AF.

Appendices

Appendix A

Signal Generator (Siu & Chon)

```
function [x,y]=kin_crossbi_gen(signal_size,seg_size)
fx1=0.03;
fx2=0.12;
fy1=0.03;
fy2=0.12;

%getting number of segments
num_seg=floor(signal_size/seg_size);

%creating random phase
rand_phase=rand(4,num_seg);
for i=1:4
    rand_phase(i,:)=rand_phase(i,:)./max(rand_phase(i,:));
end
rand_phase=abs(rand_phase);
rand_phase=rand_phase./max(max(rand_phase));
rand_phase=rand_phase.*2.*pi;


%looping for the signal
for a=1:num_seg
    for b=(a-1)*seg_size+1:a*seg_size
        x1(b)=exp(-j.*((2.*pi.*fx1.*b)+rand_phase(1,a)));
        x2(b)=exp(-j.*((2.*pi.*fx2.*b)+rand_phase(2,a)));
        y1(b)=exp(-j.*((2.*pi.*fy1.*b)+rand_phase(3,a)));
        y2(b)=exp(-j.*((2.*pi.*fy2.*b)+rand_phase(4,a)));
        x(b)=x1(b)+x2(b)+(x1(b).*x2(b));
        y(b)=y1(b)+y2(b)+(y1(b).*x2(b));
    end
end

x=x';
y=y';
```

Appendix B

Main Call

```
%% Data input & downsampler, pre-processing
cd C:\Users\Administrator\Desktop\Algorithms\Primary\
direc='D:\Rafael\For Ki Chon BME SUNY\512 channel Plaque data\CHF Dog 988\';
stringer='988_169';
s=importdata([direc, stringer, '.txt'],'\t',1);
RBArray=s.data(1:end,422:509);
num_chans=numel(RBArray(1,1:end));
timeL=length(RBArray);
stringer=[stringer,'_RB'];

data=cell(1,num_chans);
for k=1:num_chans
    data{k}=RBArray(1:timeL,k);
end

% 11-29-2010 Pre-processing, DECIMATION
parfor k=1:num_chans
    data{k}=decimate(data{k}, 10);
end

parfor k=1:num_chans
    data{k}=decimate(data{k}, 2);
end


%%
nfft=128; % # of FFT points, make sure at least equal to or twice nsamp.
wind=5; %Window Size for bispec only
nsamp=128; %Number of samples per segment
overlap=50; % 50% overlap
samp=100; %Sampling frequency

num_chans=length(data);
pieceL=ceil(num_chans/8); %Number of chans must be a multiple of 8 in order to have 0 residuals.
residuals=pieceL*8-num_chans;% CHECK RESIDUALS BEFORE RUNNING!!!!!!!

randn('state',sum(100*clock)); %#ok<RAND>
amp=zeros(num_chans,pieceL);
f1=zeros(num_chans,pieceL);
f2=zeros(num_chans,pieceL);
large_Bicthres=cell(size(amp));

spmd
if labindex==1
    E=1;clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Started']);
    for D=1:pieceL;
        for C=1:num_chans
            signal1 = data{C};
            signal2 = data{D};

Bicthres=kin_xbi_general_Bicthres_Optimized(signal1,signal2,nfft,wind,nsamp,overlap,samp);
                if (Bicthres~=0)
                    [~,max_index]=max(Bicthres(:,3));
                    amp(C,E)=Bicthres(max_index(1,1),3);
                    f1(C,E)=Bicthres(max_index(1,1),1);
                    f2(C,E)=Bicthres(max_index(1,1),2);
                    large_Bicthres{C,E}=Bicthres;
                end
        end
        if E==3; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 25% complete']); end
        if E==7; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 50% complete']); end
```

```
        if E==9; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 75% complete']); end
        E=E+1;
    end
    clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Complete'])
end

if labindex==2
    E=1;clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Started']);
    for D=pieceL+1:2*pieceL;
        for C=1:num_chans
            signal1 = data{C};
            signal2 = data{D};

Bicthres=kin_xbi_general_Bicthres_Optimized(signal1,signal2,nfft,wind,nsamp,overlap,samp);
                if (Bicthres~=0)
                    [~,max_index]=max(Bicthres(:,3));
                    amp(C,E)=Bicthres(max_index(1,1),3);
                    f1(C,E)=Bicthres(max_index(1,1),1);
                    f2(C,E)=Bicthres(max_index(1,1),2);
                    large_Bicthres{C,E}=Bicthres;
                end
        end
        if E==3; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 25% complete']); end
        if E==7; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 50% complete']); end
        if E==9; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 75% complete']); end
        E=E+1;
    end
    clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Complete'])
end

if labindex==3
    E=1;clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Started']);
    for D=2*pieceL+1:3*pieceL;
        for C=1:num_chans
            signal1 = data{C};
            signal2 = data{D};

Bicthres=kin_xbi_general_Bicthres_Optimized(signal1,signal2,nfft,wind,nsamp,overlap,samp);
                if (Bicthres~=0)
                    [~,max_index]=max(Bicthres(:,3));
                    amp(C,E)=Bicthres(max_index(1,1),3);
                    f1(C,E)=Bicthres(max_index(1,1),1);
                    f2(C,E)=Bicthres(max_index(1,1),2);
                    large_Bicthres{C,E}=Bicthres;
                end
        end
        if E==3; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 25% complete']); end
        if E==7; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 50% complete']); end
        if E==9; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 75% complete']); end
        E=E+1;
    end
    clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Complete'])
end

if labindex==4
    E=1;clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Started']);
    for D=3*pieceL+1:4*pieceL;
        for C=1:num_chans
```

```
            signal1 = data{C};
            signal2 = data{D};

Bicthres=kin_xbi_general_Bicthres_Optimized(signal1,signal2,nfft,wind,nsamp,overlap,samp);
                if (Bicthres~=0)
                    [~,max_index]=max(Bicthres(:,3));
                    amp(C,E)=Bicthres(max_index(1,1),3);
                    f1(C,E)=Bicthres(max_index(1,1),1);
                    f2(C,E)=Bicthres(max_index(1,1),2);
                    large_Bicthres{C,E}=Bicthres;
                end
        end
        if E==3; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 25% complete']); end
        if E==7; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 50% complete']); end
        if E==9; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 75% complete']); end
        E=E+1;
    end
    clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Complete'])
end

if labindex==5
    E=1;clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Started']);
    for D=4*pieceL+1:5*pieceL;
        for C=1:num_chans
            signal1 = data{C};
            signal2 = data{D};

Bicthres=kin_xbi_general_Bicthres_Optimized(signal1,signal2,nfft,wind,nsamp,overlap,samp);
                if (Bicthres~=0)
                    [~,max_index]=max(Bicthres(:,3));
                    amp(C,E)=Bicthres(max_index(1,1),3);
                    f1(C,E)=Bicthres(max_index(1,1),1);
                    f2(C,E)=Bicthres(max_index(1,1),2);
                    large_Bicthres{C,E}=Bicthres;
                end
        end
        if E==3; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 25% complete']); end
        if E==7; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 50% complete']); end
        if E==9; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 75% complete']); end
        E=E+1;
    end
    clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Complete'])
end


if labindex==6
    E=1;clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Started']);
    for D=5*pieceL+1:6*pieceL;
        for C=1:num_chans
            signal1 = data{C};
            signal2 = data{D};

Bicthres=kin_xbi_general_Bicthres_Optimized(signal1,signal2,nfft,wind,nsamp,overlap,samp);
                if (Bicthres~=0)
                    [~,max_index]=max(Bicthres(:,3));
                    amp(C,E)=Bicthres(max_index(1,1),3);
                    f1(C,E)=Bicthres(max_index(1,1),1);
                    f2(C,E)=Bicthres(max_index(1,1),2);
                    large_Bicthres{C,E}=Bicthres;
                end
        end
```

```
        if E==3; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 25% complete']); end
        if E==7; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 50% complete']); end
        if E==9; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 75% complete']); end
        E=E+1;
    end
    clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Complete'])
end

if labindex==7
    E=1;clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Started']);
    for D=6*pieceL+1:7*pieceL;
        for C=1:num_chans
            signal1 = data{C};
            signal2 = data{D};

Bicthres=kin_xbi_general_Bicthres_Optimized(signal1,signal2,nfft,wind,nsamp,overlap,samp);
                if (Bicthres~=0)
                    [~,max_index]=max(Bicthres(:,3));
                    amp(C,E)=Bicthres(max_index(1,1),3);
                    f1(C,E)=Bicthres(max_index(1,1),1);
                    f2(C,E)=Bicthres(max_index(1,1),2);
                    large_Bicthres{C,E}=Bicthres;
                end
        end
        if E==3; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 25% complete']); end
        if E==7; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 50% complete']); end
        if E==9; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 75% complete']); end
        E=E+1;
    end
    clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Complete'])
end

if labindex==8
    E=1;clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Started']);
    for D=7*pieceL+1:8*pieceL;
        for C=1:num_chans
            signal1 = data{C};
            signal2 = data{D};

Bicthres=kin_xbi_general_Bicthres_Optimized(signal1,signal2,nfft,wind,nsamp,overlap,samp);
                if (Bicthres~=0)
                    [~,max_index]=max(Bicthres(:,3));
                    amp(C,E)=Bicthres(max_index(1,1),3);
                    f1(C,E)=Bicthres(max_index(1,1),1);
                    f2(C,E)=Bicthres(max_index(1,1),2);
                    large_Bicthres{C,E}=Bicthres;
                end
        end
        if E==3; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 25% complete']); end
        if E==7; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 50% complete']); end
        if E==9; clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' 75% complete']); end
        E=E+1;
    end
    clocker=(clock); disp([num2str(clocker(2)),'-',num2str(clocker(3)),'
',num2str(clocker(4)),':',num2str(clocker(5)), ' Complete'])
end

end
```

```
master_Bicthres=[large_Bicthres{1}, large_Bicthres{2}, large_Bicthres{3}, large_Bicthres{4},
large_Bicthres{5}, large_Bicthres{6}, large_Bicthres{7}, large_Bicthres{8}];
save(['D:\Rafael\Results\002b\amps-',stringer,'.mat'], 'master_Bicthres')
master_amp=[amp{1}, amp{2}, amp{3}, amp{4}, amp{5}, amp{6}, amp{7}, amp{8}];
master_f1=[f1{1}, f1{2}, f1{3}, f1{4}, f1{5}, f1{6}, f1{7}, f1{8}];
master_f2=[f2{1}, f2{2}, f2{3}, f2{4}, f2{5}, f2{6}, f2{7}, f2{8}];
```

Appendix C

Cross-bispectrum (Swami, [27])

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Cross bispectrum program
function [Bspec,waxis] = ...
    bispecdx_samp (x, y, z, nfft, wind, nsamp, overlap,samp,plotflag)
%BISPECDX Cross-Bispectrum estimation using the direct (fft-based) approach.
%   [Bspec,waxis] = bispecdx (x,y,z, nfft, wind, segsamp,overlap,plotflag)
%   x    - data vector or time-series
%   y    - data vector or time-series  (same dimensions as x)
%   z    - data vector or time-series  (same dimensions as x)
%   nfft - fft length [default = power of two > segsamp]
%   wind - window specification for frequency-domain smoothing
%          if 'wind' is a scalar, it specifies the length of the side
%             of the square for the Rao-Gabr optimal window  [default=5]
%          if 'wind' is a vector, a 2D window will be calculated via
%             w2(i,j) = wind(i) * wind(j) * wind(i+j)
%          if 'wind' is a matrix, it specifies the 2-D filter directly
%   segsamp - samples per segment [default: such that we have 8 segments]
%             - if x is a matrix, segsamp is set to the number of rows
%   overlap - percentage overlap, allowed range [0,99]. [default = 50];
%             - if x is a matrix, overlap is set to 0.
%   plotflag- if 0, cross-bispectrum will not be displayed [default=1]
%   Bspec   - estimated bispectrum: an nfft x nfft array, with origin
%             at the center, and axes pointing down and to the right.
%   waxis   - vector of frequencies associated with the rows and columns
%             of Bspec;  sampling frequency is assumed to be 1.

%  Copyright (c) 1991-2001 by United Signals & Systems, Inc.
%       $Revision: 1.8 $
%  A. Swami   January 20, 1995

%     RESTRICTED RIGHTS LEGEND
% Use, duplication, or disclosure by the Government is subject to
% restrictions as set parforth in subparagraph (c) (1) (ii) of the
% Rights in Technical Data and Computer Software clause of DFARS
% 252.227-7013.
% Manufacturer: United Signals & Systems, Inc., P.O. Box 2374,
% Culver City, Caliparfornia 90231.
%
%  This material may be reproduced by or parfor the U.S. Government pursuant
%  to the copyright license under the clause at DFARS 252.227-7013.

% ----------------------- Parameter checks  ---------------------------
    [lx, lrecs] = size(x);
    [ly, nrecs] = size(y);
    [lz, krecs] = size(z);
    if (lx ~= ly | lrecs ~= nrecs | ly ~= lz | nrecs ~= krecs)
       error(' x, y and z should have identical dimensions')
    end

    if (ly == 1)
       x = x(:);  y = y(:);  z = z(:); ly = nrecs; nrecs = 1;
    end

    if (exist('plotflag') ~= 1)    plotflag = 1;   end
    if (exist('nfft') ~= 1)            nfft = 128; end
```

```
  if (exist('overlap') ~= 1)      overlap = 50;  end
  overlap = min(99, max(overlap,0));
  if (nrecs > 1)                  overlap =  0;  end
  if (exist('nsamp') ~= 1)          nsamp = 0;   end
  if (nrecs > 1)                    nsamp = ly;  end
  if (nrecs == 1 & nsamp <= 0)
     nsamp = fix(ly/ (8 - 7 * overlap/100));
  end
  if (nfft  < nsamp)   nfft = 2^nextpow2(nsamp); end
  overlap  = fix(overlap/100  * nsamp);
  nadvance = nsamp - overlap;
  nrecs    = fix ( (ly*nrecs - overlap) / nadvance);


% ------------------- create the 2-D window -------------------------
  if (exist('wind') ~= 1) wind = 5; end
  [m,n] = size(wind);
  window = wind;
  if (max(m,n) == 1)     % scalar: wind is size of Rao-Gabr window
     winsize = wind;
     if (winsize < 0) winsize = 5; end        % the window length L
     winsize = winsize - rem(winsize,2) + 1;  % make it odd
     if (winsize > 1)
        mwind   = fix (nfft/winsize);            % the scale parameter M
        lby2    = (winsize - 1)/2;

        theta  = -lby2:lby2;
        opwind = ones(winsize,1) * (theta .^2);       % w(m,n)=m^2
        opwind = opwind + opwind' + theta' * theta;   % m^2 + n^2 + mn
        opwind = 1 - (2*mwind/nfft)^2 * opwind;       %
        hex    = ones(winsize,1) * theta;             % m
        hex    = abs(hex) + abs(hex') + abs(hex+hex');
        hex    = (hex < winsize);
        opwind = opwind .* hex;
        opwind = opwind * (4 * mwind^2) / (7 * pi^2) ;
     else
        opwind = 1;
     end

  elseif (min(m,n) == 1)  % 1-D window passed: convert to 2-D
     window = window(:);
     if (any(imag(window) ~= 0 ))
        disp(['1-D window has imaginary components: window ignored'])
        window = 1;
     end
     if (any(window < 0))
        disp(['1-D window has negative components: window ignored'])
        window = 1;
     end
     lwind  = length(window);
     windf  = [window(lwind:-1:2); window];    % the full symmetric 1-D
     window = [window; zeros(lwind-1,1)];
     opwind = (windf * windf')       ...
              .* hankel(flipud(window), window); % w(m)w(n)w(m+n)
     winsize = length(window);

  else                    % 2-D window passed: use directly
    winsize = m;
    if (m ~= n)
       disp('2-D window is not square: window ignored')
       window = 1;
       winsize = m;
    end
    if (rem(m,2) == 0)
       disp('2-D window does not have odd length: window ignored')
       window = 1;
       winsize = m;
    end
    opwind  = window;
  end
```

```
% ---------------- accumulate triple products ----------------------

    Bspec    = zeros(nfft,nfft);

    mask = hankel([1:nfft],[nfft,1:nfft-1] );   % the hankel mask (faster)
    locseg = [1:nsamp]';
    for krec = 1:nrecs
        xseg   = x(locseg);
    yseg   = y(locseg);
        zseg   = z(locseg);
        Xf     = fft(xseg-mean(xseg), nfft)/nsamp;
        Yf     = fft(yseg-mean(yseg), nfft)/nsamp;
        CZf    = fft(zseg-mean(zseg), nfft)/nsamp;
        CZf    = conj(CZf);
        Bspec  = Bspec + (Xf * Yf.') .* ...
             reshape(CZf(mask), nfft, nfft);
        locseg = locseg + nadvance;
    end

    Bspec = fftshift(Bspec)/(nrecs);

% ----------------- frequency-domain smoothing ------------------------

  if (winsize > 1)
      lby2 = (winsize-1)/2;
      Bspec = conv2(Bspec,opwind);
      Bspec = Bspec(lby2+1:lby2+nfft,lby2+1:lby2+nfft);
  end
% ------------ contout plot of magnitude bispectum --------------------

   if (rem(nfft,2) == 0)
       waxis = [-nfft/2:(nfft/2-1)]'/nfft.*samp;
   else
       waxis = [-(nfft-1)/2:(nfft-1)/2]'/nfft.*samp;
   end

% if (plotflag)
%    hold off, clf
%   contour(abs(Bspec),4,waxis,waxis),grid
%    contour(waxis,waxis,abs(Bspec),4), grid on
%    mesh(waxis, waxis, abs(Bspec))
%    title('Cross-Bispectrum ')
%    xlabel('f1'), ylabel('f2')
%    set(gcf,'Name','Hosa BISPECDX')
% end
return
```

Appendix D

Cross-bicoherence (Swami, [27])

```
%Cross bicoherence program
function [bic,waxis] = bicoherx_noplot_samp (w,x,y,  nfft, wind, nsamp, overlap,samp)
%BICOHERX - Direct (FD) method for estimating cross-bicoherence
%   [bic,waxis] = bicoherx (w,x,y,  nfft, wind, segsamp, overlap)
%   w,x,y - data vector or time-series
%         - should have identical dimensions
%   nfft - fft length [default = power of two > nsamp]
%          actual size used is power of two greater than 'nsamp'
%   wind - specifies the time-domain window to be applied to each
%          data segment; should be of length 'segsamp' (see below);
%       otherwise, the default Hanning window is used.
%   segsamp - samples per segment [default: such that we have 8 segments]
%           - if x is a matrix, segsamp is set to the number of rows
%   overlap - percentage overlap, 0 to 99  [default = 50]
%           - if y is a matrix, overlap is set to 0.
%   bic     - estimated cross-bicoherence: an nfft x nfft array, with
%             origin at center, and axes pointing down and to the right.
%   waxis   - vector of frequencies associated with the rows and columns
%             of bic;  sampling frequency is assumed to be 1.

%  Copyright (c) 1991-2001 by United Signals & Systems, Inc.
%       $Revision: 1.7 $
%  A. Swami   January 20, 1995

%     RESTRICTED RIGHTS LEGEND
% Use, duplication, or disclosure by the Government is subject to
% restrictions as set forth in subparagraph (c) (1) (ii) of the
% Rights in Technical Data and Computer Software clause of DFARS
% 252.227-7013.
% Manufacturer: United Signals & Systems, Inc., P.O. Box 2374,
% Culver City, California 90231.
%
%  This material may be reproduced by or for the U.S. Government pursuant
%  to the copyright license under the clause at DFARS 252.227-7013.


% --------------------- parameter checks -----------------------------

    if (size(w) ~= size(x) | size(x) ~= size(y) )
       error(' w, x, and y should have identical dimensions')
    end
    [ly, nrecs] = size(y);
    if (ly == 1)
        ly = nrecs; nrecs = 1;
         w = w(:);  x = x(:);  y = y(:);
    end

    if (exist('nfft') ~= 1)            nfft = 128; end
    if (exist('overlap') ~= 1)      overlap = 50;  end
    overlap = max(0,min(overlap,99));
    if (nrecs > 1)                  overlap = 0;   end
    if (exist('nsamp') ~= 1)          nsamp = 0;  end
    if (nrecs > 1)                    nsamp = ly;  end
    if (nrecs == 1 & nsamp <= 0)
       nsamp = fix(ly/ (8 - 7 * overlap/100));
    end
    if (nfft  < nsamp)   nfft = 2^nextpow2(nsamp); end

    overlap  = fix(overlap/100  * nsamp);
    nadvance = nsamp - overlap;
    nrecs    = fix ( (ly*nrecs - overlap) / nadvance);

% ------------------------------------------------------------------------
    if (exist('wind') ~= 1) wind = hanning(nsamp); end
    [rw,cw] = size(wind);
```

```
    if (min(rw,cw) ~= 1 | max(rw,cw) ~= nsamp)
%       disp(['Segment size  is ',int2str(nsamp)])
%       disp(['"wind" array  is ',int2str(rw),' by ',int2str(cw)])
%       disp(['Using default window'])
        wind = hanning(nsamp);
    end
    wind = wind(:);
% ---------------- accumulate triple products ----------------------

    bic  = zeros(nfft,nfft);
    Pyy  = zeros(nfft,1);          Pww = Pyy; Pxx = Pyy;

    mask = hankel([1:nfft],[nfft,1:nfft-1] );   % the hankel mask (faster)
    Yf12 = zeros(nfft,nfft);
    ind  = [1:nsamp]';

    for k = 1:nrecs
        ws  = w(ind); ws = (ws-mean(ws)).* wind;
        Wf  = fft(ws,nfft)  / nsamp;  CWf = conj(Wf);
        Pww = Pww + Wf .* CWf;

        xs  = x(ind); xs = (xs-mean(xs)).* wind;
        Xf  = fft(xs,nfft)  / nsamp;  CXf = conj(Xf);
        Pxx = Pxx + Xf .* CXf;

        ys  = y(ind);   ys = (ys(:) - mean(ys)) .* wind;
    Yf  = fft(ys,nfft)  / nsamp;  CYf = conj(Yf);
    Pyy = Pyy + Yf .* CYf;

        Yf12(:)  = CYf(mask);
        bic = bic + (Wf * Xf.') .* Yf12;

        ind = ind + nadvance;
    end

    bic     = bic / nrecs;
    Pyy     = Pyy  / nrecs;   Pww = Pww / nrecs;  Pxx = Pxx / nrecs;
    mask(:) = Pyy(mask);
    bic = abs(bic).^2 ./ (Pww * Pxx.' .* mask);
    bic = fftshift(bic) ;

% ------------ contour plot of magnitude bispectum --------------------

   if (rem(nfft,2) == 0)
       waxis = [-nfft/2:(nfft/2-1)]'/nfft.*samp;
   else
       waxis = [-(nfft-1)/2:(nfft-1)/2]'/nfft.*samp;
   end
%
%    hold off, clf
% %   contour(bic,4,waxis,waxis),grid
%    contour(waxis,waxis,bic,4), grid on
%    title('Cross-Bicoherence')
%    xlabel('f1'), ylabel('f2')
%    set(gcf,'Name','Hosa BICOHERX')
%
%    [colmax,row] = max(bic)  ;
%    [maxval,col] = max(colmax);
%    row = row(col);
%    disp(['Max: bic(',num2str(waxis(row)),',',num2str(waxis(col)),') = ', ...
%          num2str(maxval) ])
%
% return
```

Appendix E

Surrogate Loop (Siu & Chon)

```
function [surr] = surr_gen(signal)
surr=zeros(length(signal),100);
parfor i=1:100
    [surr(:,i)]=generate_surrogate(signal);
end
```

Appendix F

Generate Surrogates (Gautama, [36])

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Surrogate data generation program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate surrogate data with matching amplitude spectrum and
% amplitude distribution (Schreiber and Schmitz, 1996).
% Multivariate extension as described in Schreiber and Schmitz (2000)
%
% Usage: [Xs E] = generate_surrogate (X, specflag);
%   specflag    exact amplitude spectrum (1, default), otherise amp distr
%   X [pp x dim]

function [Xs,E] = generate_surrogate(X)

max_it = 500;
[pp dim] = size(X);

if (dim==1)
    X = X(:);
    pp = length(X);

    % Initial Conditions
    rn = X(randperm(pp));
    Yamp = abs(fft(X)); % Desired amplitude spectrum
    Xsorted = sort(X);  % Desired signal distribution

    E = zeros(1,max_it);
    c = 1;
    prev_err = 1000000;
    err = prev_err - 1;
    while (prev_err>err) && (c<max_it)

        % Match Amplitude Spec
        Yrn = fft(rn);
        Yang = angle(Yrn);
        sn = real(ifft(Yamp.*(cos(Yang)+sqrt(-1).*sin(Yang))));

        % Match Signal Distribution
        [~, INDs] = sort(sn);
        rn(INDs) = Xsorted;

        % Eval Convergence
        prev_err = err;
        A2 = abs(Yrn);
        %err = mean(mean(abs(A2-Yamp)));
        err = mean(abs(A2-Yamp));
        E(c) = err;

        c = c+1;
    end
    E = E(1:c-1);
```

```
%     if (specflag==1)
        Xs = sn;     % Exact Amp Spectrum
%     else
%         Xs = rn;  % Exact Amp Distribution
%     end

end

if (dim>1)
    Y = fft(X);      % fft every column
    Yamp = abs(Y);
    Porig = angle(Y);

    % Initial Conditions
    rn = zeros(size(X));
    for k=1:dim
        rn(:,k) = X(randperm(pp),k);
    end
    Xsorted = sort(X);

    prev_err = 1000000;
    err = prev_err - 1;
    c = 1;
    while (prev_err>err) && (c<max_it)
        % Match Amplitude Spec
        Prn = angle(fft(rn));
        goal = Prn - Porig;
        AUX1 = sum(cos(goal),2);
        alpha = repmat((AUX1~=0).*atan(sum(sin(goal),2)./sum(AUX1+(AUX1==0),2)),1,dim);
        alpha = alpha + repmat(pi.*(sum(cos(alpha-goal),2)<0),1,dim);
        Pcurr = Porig + alpha;
        sn = real(ifft(Yamp.*(cos(Pcurr)+sqrt(-1).*sin(Pcurr))));

        % Match Signal Distribution
        [~, INDs] = sort(sn);
        for k=1:dim
            rn(INDs(:,k),k) = Xsorted(:,k);
        end

        % Eval Convergence
        prev_err = err;
        A2 = abs(fft(rn));
        err = mean(mean(abs(A2-Yamp)));
        E(c) = err;
        c = c+1;
    end

    if (flag==1)
        Xs = sn;     % Exact Amp Spectrum
    else
        Xs = rn;     % Exact Amp Distribution
    end

end
```

Appendix G

Summary of Acquired Electrograms

Each dog had a minimum of one AF episode for each treatment group (no treatment, 10 nM, and 300 nM). Some dogs had several AF episodes within the same treatment group. The start of every new episode is indicated by a "1", recorded one minute after the start of that new episode. If the episode is sustained across more than one recording, the value indicates the time point after the start of the episode in minutes. Sustained episodes are shaded in gray. Additional sustained episodes for the same dog and treatment group are shaded in light blue.

Table 7: Summary of Acquired Electrograms (time of acquisition in minutes)

Dog 1 922			Dog 2 987			Dog 3 988			Dog 4 991		
no Rx	**10 nM**	**300 nM**	**no Rx**	**10 nM**	**300 nM**	**no Rx**	**10 nM**	**300 nM**	**no Rx**	**10 nM**	**300 nM**
1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	1	1	1	1	5	1	3	5	2
10	10	10	5	1	1	1	8	1	5	10	
15	15	15	10	1	2	2		5	9	15	
20	20	20	15	1				10	1	20	
25	25	25	1	2				15	5	25	
30	30	30	1					20	8	1	
			1					25	1	1	
									1	1	
										5	
										10	

Appendix H

Frequency Finder

```
function [uniqFreqs, DFreq, DChan]=sig_bars(stringer, plotflag)
% This function finds the signifiant amplitudes in the output array, and
% plots channels and frequencies with bars.
% Open matlab variable file, variable name = master_Bicthres
% Note: This function requires the non-standard function count.m
direc='D:\Rafael\Results\002b\';
% stringer='922_21_RB';  % Dataset name
% plotflag=1;
data=open([direc, 'amps-', stringer, '.mat']);
threshold=0.5;

%% Significance Finder, Greater than amp threshold as defined above
events=0;
chan=[];
freq=[];
amps=[];
for b=1:length(data.master_Bicthres)
    for k=1:length(data.master_Bicthres)
        if b==k;
            break
        else
            temp=data.master_Bicthres{b,k};
            if temp~=0
                [m, ~]=size(temp);
                for p=1:m
                    if  temp(p,3) > threshold && 0 < temp(p,1) &&...
                          temp(p,1) < 20 && 0 < temp (p,2) && temp(p,2)...
                           < 20
                        events=events+1;
                        chan=[chan; [b k]];                 %#ok<AGROW>
                        freq=[freq; temp(p,1) temp(p,2)];   %#ok<AGROW>
                        amps=[amps; temp(p,3)];             %#ok<AGROW>

                    end
                end
            end
            clearvars temp
        end
    end
end

%% Channel Counting & Bar Plot
uniqChan=unique(chan);
temp=zeros(length(uniqChan),2);
for k= 1:length(uniqChan)
    stringChan=num2str(uniqChan(k),15); % 15 decimal precision
    temp(k,1)= count(chan(:,1),['==',stringChan]);
    temp(k,2)= count(chan(:,2),['==',stringChan]);
end

uniqChan(:,2)=temp(:,1)+temp(:,2);

if plotflag==1;
figure(1)
bar(uniqChan(:,1),uniqChan(:,2))
xlabel('Channel'); ylabel('Number of Interactions')
xlim([0 88])
title([stringer(1), stringer(2), stringer(3), ' ', stringer(5)...
    stringer(6), ' ', stringer(8), stringer(9)])
else

end


[~, locDChan]=max(uniqChan(:,2));
```

```
DChan=uniqChan(locDChan);


%% Frequency Counting & Bar Plot
uniqFreqs=unique(freq);
temp=zeros(length(uniqFreqs),2);
for k= 1:length(uniqFreqs)
    stringFreq=num2str(uniqFreqs(k),15);
    temp(k,1)= count(freq(:,1),['==',stringFreq]);
    temp(k,2)= count(freq(:,2),['==',stringFreq]);
end

uniqFreqs(:,2)=temp(:,1)+temp(:,2);


if plotflag==1;
figure(2)
bar(uniqFreqs(:,1),uniqFreqs(:,2),0.5);
xlabel('Frequency (Hz)'); ylabel('Number of Interactions')
xlim([0 25])
title([stringer(1), stringer(2), stringer(3), ' ', stringer(5)...
    stringer(6), ' ', stringer(8), stringer(9)])
else

end

[~, locDFreq]=max(uniqFreqs(:,2));
DFreq=uniqFreqs(locDFreq);
```

Appendix I

Organization Index Loop For All Files

```
function orgzer=init_looper(dataset,DF)
% input "DF" after dataset

global orgzer k
orgzer=cell(1,3);

% Dataset: 922
if strcmp(dataset, '922_RB')
    for k=1:21;

        if k==1;
            stringer='922_19_RB';
            comp_str(stringer, DF)
        end

        if k==2;
            stringer='922_20_RB';
            comp_str(stringer, DF)
        end

        if k==3;
            stringer='922_21_RB';
            comp_str(stringer, DF)
        end

        if k==4;
            stringer='922_22_RB';
            comp_str(stringer, DF)
        end

        if k==5;
            stringer='922_23_RB';
            comp_str(stringer, DF)
        end

        if k==6;
            stringer='922_24_RB';
            comp_str(stringer, DF)
        end

        if k==7;
            stringer='922_25_RB';
            comp_str(stringer, DF)
        end

        if k==8;
            stringer='922_79_RB';
            comp_str(stringer, DF)
        end

        if k==9;
            stringer='922_80_RB';
            comp_str(stringer, DF)
        end

        if k==10;
            stringer='922_81_RB';
            comp_str(stringer, DF)
        end

        if k==11;
            stringer='922_82_RB';
            comp_str(stringer, DF)
        end
```

```
        if k==12;
            stringer='922_83_RB';
            comp_str(stringer, DF)
        end

        if k==13;
            stringer='922_84_RB';
            comp_str(stringer, DF)
        end

        if k==14;
            stringer='922_85_RB';
            comp_str(stringer, DF)
        end

        if k==15;
            stringer='922_144_RB';
            comp_str(stringer, DF)
        end

        if k==16;
            stringer='922_145_RB';
            comp_str(stringer, DF)
        end

        if k==17;
            stringer='922_146_RB';
            comp_str(stringer, DF)
        end

        if k==18;
            stringer='922_147_RB';
            comp_str(stringer, DF)
        end

        if k==19;
            stringer='922_148_RB';
            comp_str(stringer, DF)
        end

        if k==20;
            stringer='922_149_RB';
            comp_str(stringer, DF)
        end

        if k==21;
            stringer='922_150_RB';
            comp_str(stringer, DF)
        end
    end
end

% Dataset: 987
if strcmp(dataset, '987_RB')
    for k=1:18;

        if k==1;
            stringer='987_43_RB';
            comp_str(stringer, DF)
        end

        if k==2;
            stringer='987_44_RB';
            comp_str(stringer, DF)
        end

        if k==3;
            stringer='987_45_RB';
            comp_str(stringer, DF)
        end
```

```
if k==4;
    stringer='987_46_RB';
    comp_str(stringer, DF)
end

if k==5;
    stringer='987_47_RB';
    comp_str(stringer, DF)
end

if k==6;
    stringer='987_48_RB';
    comp_str(stringer, DF)
end

if k==7;
    stringer='987_49_RB';
    comp_str(stringer, DF)
end

if k==8;
    stringer='987_50_RB';
    comp_str(stringer, DF)
end

if k==9;
    stringer='987_87_RB';
    comp_str(stringer, DF)
end

if k==10;
    stringer='987_88_RB';
    comp_str(stringer, DF)
end

if k==11;
    stringer='987_89_RB';
    comp_str(stringer, DF)
end

if k==12;
    stringer='987_90_RB';
    comp_str(stringer, DF)
end

if k==13;
    stringer='987_91_RB';
    comp_str(stringer, DF)
end

if k==14;
    stringer='987_92_RB';
    comp_str(stringer, DF)
end

if k==15;
    stringer='987_173_RB';
    comp_str(stringer, DF)
end

if k==16;
    stringer='987_174_RB';
    comp_str(stringer, DF)
end

if k==17;
    stringer='987_175_RB';
    comp_str(stringer, DF)
end

if k==18;
```

```
            stringer='987_176_RB';
            comp_str(stringer, DF)
        end
    end
end

% Dataset: 988
if strcmp(dataset, '988_RB')
    for k=1:15;
        if k==1;
            stringer='988_38_RB';
            comp_str(stringer, DF)
        end

        if k==2;
            stringer='988_39_RB';
            comp_str(stringer, DF)
        end

        if k==3;
            stringer='988_40_RB';
            comp_str(stringer, DF)
        end

        if k==4;
            stringer='988_41_RB';
            comp_str(stringer, DF)
        end

        if k==5;
            stringer='988_78_RB';
            comp_str(stringer, DF)
        end

        if k==6;
            stringer='988_79_RB';
            comp_str(stringer, DF)
        end

        if k==7;
            stringer='988_80_RB';
            comp_str(stringer, DF)
        end

        if k==8;
            stringer='988_162_RB';
            comp_str(stringer, DF)
        end

        if k==9;
            stringer='988_163_RB';
            comp_str(stringer, DF)
        end

        if k==10;
            stringer='988_164_RB';
            comp_str(stringer, DF)
        end

        if k==11;
            stringer='988_165_RB';
            comp_str(stringer, DF)
        end

        if k==12;
            stringer='988_166_RB';
            comp_str(stringer, DF)
        end

        if k==13;
            stringer='988_167_RB';
```

```
            comp_str(stringer, DF)
        end

        if k==14;
            stringer='988_168_RB';
            comp_str(stringer, DF)
        end

        if k==15;
            stringer='988_169_RB';
            comp_str(stringer, DF)
        end
    end
end

% Dataset: 991
if strcmp(dataset, '991_RB')
    for k=1:22;

        if k==1;
            stringer='991_37_RB';
            comp_str(stringer, DF)
        end

        if k==2;
            stringer='991_38_RB';
            comp_str(stringer, DF)
        end

        if k==3;
            stringer='991_39_RB';
            comp_str(stringer, DF)
        end

        if k==4;
            stringer='991_40_RB';
            comp_str(stringer, DF)
        end

        if k==5;
            stringer='991_41_RB';
            comp_str(stringer, DF)
        end

        if k==6;
            stringer='991_42_RB';
            comp_str(stringer, DF)
        end

        if k==7;
            stringer='991_43_RB';
            comp_str(stringer, DF)
        end

        if k==8;
            stringer='991_44_RB';
            comp_str(stringer, DF)
        end

        if k==9;
            stringer='991_45_RB';
            comp_str(stringer, DF)
        end

        if k==10;
            stringer='991_82_RB';
            comp_str(stringer, DF)
        end

        if k==11;
            stringer='991_83_RB';
```

```
            comp_str(stringer, DF)
        end

        if k==12;
            stringer='991_84_RB';
            comp_str(stringer, DF)
        end

        if k==13;
            stringer='991_85_RB';
            comp_str(stringer, DF)
        end

        if k==14;
            stringer='991_86_RB';
            comp_str(stringer, DF)
        end

        if k==15;
            stringer='991_87_RB';
            comp_str(stringer, DF)
        end

        if k==16;
            stringer='991_88_RB';
            comp_str(stringer, DF)
        end

        if k==17;
            stringer='991_89_RB';
            comp_str(stringer, DF)
        end

        if k==18;
            stringer='991_90_RB';
            comp_str(stringer, DF)
        end

        if k==19;
            stringer='991_91_RB';
            comp_str(stringer, DF)
        end

        if k==20;
            stringer='991_92_RB';
            comp_str(stringer, DF)
        end

        if k==21;
            stringer='991_170_RB';
            comp_str(stringer, DF)
        end

        if k==22;
            stringer='991_171_RB';
            comp_str(stringer, DF)
        end
    end
end

function comp_str(stringer, DF)
global orgzer k
 [stringer,org,freq]=sig_Freq_Chan(stringer,0,DF);

    orgzer{k,1}=stringer;
    orgzer{k,2}=org;
    orgzer{k,3}=freq;

return
```

Appendix J

Master looping file for Organization Index with Manual Frequency Input

```
matlabpool 4

spmd
    if labindex==1
        orgzer=init_looper('922_RB', 3.125); %specify frequency here
    end

    if labindex==2
        orgzer=init_looper('987_RB', 3.125); %specify frequency here
    end

    if labindex==3
        orgzer=init_looper('988_RB', 3.125); %specify frequency here
    end

    if labindex==4
        orgzer=init_looper('991_RB', 3.125); %specify frequency here
    end
end

amplitudes=vertcat(orgzer{1},orgzer{2},orgzer{3},orgzer{4});
output=vertcat(amplitudes{1:end,2});

matlabpool close
```

Appendix K

Organization Index

```
function [stringer,Freq_Chan_Org,DF]=sig_Freq_Chan(stringer,plotflag, DF)
% Freq_Chan_Org (Output): Organization Index
% Plotflag: 1 or 0
% DF: Specify frequency for channel organization
% DEPENDENCIES: vec2row.m count.m round2.m

%% x and y space for RB (taken from electrode placement array)
x=[0
2
4
6
8
10
12
14
16
18
19
17
15
13
11
9
7
5
3
1
0
2
4
6
8
10
12
14
16
18
19
17
15
13
11
9
7
5
3
1
0
2
4
6
8
10
12
14
16
18
19
17
15
13
11
9
7
5
```

```
3
1
0
2
4
6
8
9
7
5
3
1
0
2
4
6
8
7
5
3
1
0
2
4
6
7
5
3
1
2];

y=[12
12
12
12
12
12
12
12
12
12
11
11
11
11
11
11
11
11
11
11
11
10
10
10
10
10
10
10
10
10
10
9
9
9
9
9
9
9
9
9
9
```

```
8
8
8
8
8
8
8
8
8
8
8
7
7
7
7
7
7
7
7
7
7
6
6
6
6
6
5
5
5
5
5
4
4
4
4
4
3
3
3
3
2
2
2
2
1
1
1
1
0];

%%
direc='D:\Rafael\Results\002b\';
% stringer='922_24_RB'; % Dataset name
% plotflag=0;
data=open([direc, 'amps-', stringer, '.mat']);
threshold=0.5;
NumChans=length(data.master_Bicthres);
%% Significance Finder, Greater than amp threshold, maintains structure
events=0;
chan=cell(NumChans,NumChans);
freq=cell(NumChans,NumChans);
for b=1:NumChans
    for k=1:NumChans
        if b==k;
            freq{b,k}=[];
            chan{b,k}=[];
        else
            temp=data.master_Bicthres{b,k};
            if temp~=0
                [m, ~]=size(temp);
                for p=1:m
                    if temp(p,3) > threshold
```

```
                    events=events+1;
                    freq{b,k}=[freq{b,k}; temp(p,1) temp(p,2)];
                    chan{b,k}=[chan{b,k}; [b k]];
                end
            end
        end
        clearvars temp
    end
end


% Coupled Frequency Finder
% freqRes=0.78125;
freqRes=0;
DFHarm=round2([DF DF*2 DF*3 DF-freqRes DF+freqRes DF*2-freqRes DF*2+freqRes DF*3-freqRes
DF*3+freqRes DF*4 DF*5 DF*6],0.0001);

if DF~=0

coupledFreqs=zeros(NumChans,NumChans);
for b=1:NumChans
    for k=1:NumChans
        temp=single(freq{b,k});
        if temp~=0
        [m,~]=size(temp);
        for p=1:m
            if (round2(temp(p,1),0.0001)==DF ||round2(temp(p,1),0.0001)==DFHarm(4)
||round2(temp(p,1),0.0001)==DFHarm(5))...
                && (round2(temp(p,2),0.0001)==DFHarm(1) ||round2(temp(p,2),0.0001)==DFHarm(2)
||round2(temp(p,2),0.0001)==DFHarm(3)||round2(temp(p,2),0.0001)==DFHarm(4)||round2(temp(p,2),0.00
01)==DFHarm(5)||round2(temp(p,2),0.0001)==DFHarm(6)||round2(temp(p,2),0.0001)==DFHarm(7)||round2(
temp(p,2),0.0001)==DFHarm(8)||round2(temp(p,2),0.0001)==DFHarm(9)||round2(temp(p,2),0.0001)==DFHa
rm(10)||round2(temp(p,2),0.0001)==DFHarm(11)||round2(temp(p,2),0.0001)==DFHarm(12))
                coupledFreqs(b,k)=1;
            end
            if (round2(temp(p,2),0.0001)==DF || round2(temp(p,2),0.0001)==DFHarm(4)  ||
round2(temp(p,2),0.0001)==DFHarm(5))...
                && (round2(temp(p,1),0.0001)==DFHarm(1) ||round2(temp(p,1),0.0001)==DFHarm(2)
||round2(temp(p,1),0.0001)==DFHarm(3)||round2(temp(p,1),0.0001)==DFHarm(4)||round2(temp(p,1),0.00
01)==DFHarm(5)||round2(temp(p,1),0.0001)==DFHarm(6)||round2(temp(p,1),0.0001)==DFHarm(7)||round2(
temp(p,1),0.0001)==DFHarm(8)||round2(temp(p,1),0.0001)==DFHarm(9)||round2(temp(p,1),0.0001)==DFHa
rm(10)||round2(temp(p,1),0.0001)==DFHarm(11)||round2(temp(p,1),0.0001)==DFHarm(12))
                coupledFreqs(b,k)=1;
            end
        end
        end
    end
end


% Coupled Channel Finder based off Dominant Frequency
VectorChan=[];
for b=1:NumChans
    for k=1:NumChans
        if coupledFreqs(b,k)
            VectorChan=[VectorChan, vec2row(unique(chan{b,k}))];                    %#ok<AGROW>
        end
    end
end

uniqChans=unique(VectorChan);
coupledChans=zeros(NumChans,1);
for p=1:NumChans
    if count(uniqChans,['==',num2str(p)]);
        coupledChans(p)=1;
    end
end

Freq_Chan_Org=length(uniqChans)/NumChans;
```

```
%% Scatter Plot Results using X and Y coordinates of electrode plaque

if plotflag==1;
figure
colormap(flipud(flag));
scatter(x,y,200,coupledChans,'filled');
axis([-1 20 -1 13]);axis off;
caxis([0 1])
% title([stringer(1), stringer(2), stringer(3), ' ', stringer(5)...
%     stringer(6), ' ', stringer(8), stringer(9)])
% saveas(gcf, ['D:\Rafael\Plots\FreqChanPlot\',stringer,'.tiff'], 'tiff')
% saveas(gcf, ['D:\Rafael\Plots\EPS\',stringer,'.eps'], 'eps')
end
else
    Freq_Chan_Org=0;
    DF=0;
end
```

Appendix K

Time-Frequency Analysis

```
matlabpool 4
freqband=0:0.78125:20;

spmd
    if labindex==1;
        colormatrix=freqLOOPERband('922_RB');
    end
    if labindex==2;
        colormatrix=freqLOOPERband('987_RB');
    end
    if labindex==3;
        colormatrix=freqLOOPERband('988_RB');
    end
    if labindex==4;
        colormatrix=freqLOOPERband('991_RB');
    end
end


[~,N]=size(colormatrix{1});
figure(1)
imagesc(1:N,freqband,colormatrix{1});
ylabel('Frequency (Hz)')
xlabel('Time')
set(gca,'YDir','Normal')
set(gca,'XTickLabel', '')
hold on
plot(7.5,0:0.01:20,'color','white')
plot(14.5,0:0.01:20,'color','white')
ylim([0 20])
title('Dog 1')
text(3.5,19,'Baseline','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
text(10.5,19,'10 nM','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
text(18.5,19,'300 nM','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
caxis([0 176]) % for 0.8 amp

[~,N]=size(colormatrix{2});
figure(2)
imagesc(1:N,freqband,colormatrix{2});
ylabel('Frequency (Hz)')
xlabel('Time')
set(gca,'YDir','Normal')
set(gca,'XTickLabel', '')
hold on
plot(8.5,0:0.01:20,'color','white')
plot(14.5,0:0.01:20,'color','white')
ylim([0 20])
title('Dog 2')
text(4.5,19,'Baseline','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
text(11.5,19,'10 nM','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
text(16.5,19,'300 nM','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
caxis([0 176]) % for 0.8 amp

[~,N]=size(colormatrix{3});
figure(3)
imagesc(1:N,freqband,colormatrix{3});
ylabel('Frequency (Hz)')
xlabel('Time')
set(gca,'YDir','Normal')
```

```
set(gca,'XTickLabel', '')
hold on
plot(4.5,0:0.01:20,'color','white')
plot(7.5,0:0.01:20,'color','white')
ylim([0 20])
title('Dog 3')
text(2.5,19,'Baseline','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
text(6,19,'10 nM','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
text(11.5,19,'300 nM','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
caxis([0 176]) % for 0.8 amp

[~,N]=size(colormatrix{4});
figure(4)
imagesc(1:N,freqband,colormatrix{4});
ylabel('Frequency (Hz)')
xlabel('Time')
set(gca,'YDir','Normal')
set(gca,'XTickLabel', '')
hold on
plot(9.5,0:0.01:20,'color','white')
plot(20.5,0:0.01:20,'color','white')
ylim([0 20])
title('Dog 4')
text(5,19,'Baseline','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
text(15,19,'10 nM','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
text(21.5,19,'300 nM','HorizontalAlignment','center',...
    'FontWeight','bold','Color',[1 1 1])
caxis([0 176]) % for 0.8 amp

matlabpool close
```

Appendix L

Looper for Time-Frequency Analysis

```
function colormatrix=freqLOOPERband(dataset)
global colormatrix k
colormatrix=zeros(26,1);

% Dataset: 922
if strcmp(dataset, '922_RB')
    for k=1:21;

        if k==1;
            stringer='922_19_RB';
            comp_str(stringer)
        end

        if k==2;
            stringer='922_20_RB';
            comp_str(stringer)
        end

        if k==3;
            stringer='922_21_RB';
            comp_str(stringer)
        end

        if k==4;
            stringer='922_22_RB';
            comp_str(stringer)
        end

        if k==5;
            stringer='922_23_RB';
            comp_str(stringer)
        end

        if k==6;
            stringer='922_24_RB';
            comp_str(stringer)
        end

        if k==7;
            stringer='922_25_RB';
            comp_str(stringer)
        end

        if k==8;
            stringer='922_79_RB';
            comp_str(stringer)
        end

        if k==9;
            stringer='922_80_RB';
            comp_str(stringer)
        end

        if k==10;
            stringer='922_81_RB';
            comp_str(stringer)
        end

        if k==11;
            stringer='922_82_RB';
            comp_str(stringer)
        end

        if k==12;
            stringer='922_83_RB';
```

```
            comp_str(stringer)
        end

        if k==13;
            stringer='922_84_RB';
            comp_str(stringer)
        end

        if k==14;
            stringer='922_85_RB';
            comp_str(stringer)
        end

        if k==15;
            stringer='922_144_RB';
            comp_str(stringer)
        end

        if k==16;
            stringer='922_145_RB';
            comp_str(stringer)
        end

        if k==17;
            stringer='922_146_RB';
            comp_str(stringer)
        end

        if k==18;
            stringer='922_147_RB';
            comp_str(stringer)
        end

        if k==19;
            stringer='922_148_RB';
            comp_str(stringer)
        end

        if k==20;
            stringer='922_149_RB';
            comp_str(stringer)
        end

        if k==21;
            stringer='922_150_RB';
            comp_str(stringer)
        end
    end
end

% Dataset: 987
if strcmp(dataset, '987_RB')
    for k=1:18;

        if k==1;
            stringer='987_43_RB';
            comp_str(stringer)
        end

        if k==2;
            stringer='987_44_RB';
            comp_str(stringer)
        end

        if k==3;
            stringer='987_45_RB';
            comp_str(stringer)
        end

        if k==4;
            stringer='987_46_RB';
```

```
    comp_str(stringer)
end

if k==5;
    stringer='987_47_RB';
    comp_str(stringer)
end

if k==6;
    stringer='987_48_RB';
    comp_str(stringer)
end

if k==7;
    stringer='987_49_RB';
    comp_str(stringer)
end

if k==8;
    stringer='987_50_RB';
    comp_str(stringer)
end

if k==9;
    stringer='987_87_RB';
    comp_str(stringer)
end

if k==10;
    stringer='987_88_RB';
    comp_str(stringer)
end

if k==11;
    stringer='987_89_RB';
    comp_str(stringer)
end

if k==12;
    stringer='987_90_RB';
    comp_str(stringer)
end

if k==13;
    stringer='987_91_RB';
    comp_str(stringer)
end

if k==14;
    stringer='987_92_RB';
    comp_str(stringer)
end

if k==15;
    stringer='987_173_RB';
    comp_str(stringer)
end

if k==16;
    stringer='987_174_RB';
    comp_str(stringer)
end

if k==17;
    stringer='987_175_RB';
    comp_str(stringer)
end

if k==18;
    stringer='987_176_RB';
    comp_str(stringer)
```

```
        end
    end
end

% Dataset: 988
if strcmp(dataset, '988_RB')
    for k=1:15;
        if k==1;
            stringer='988_38_RB';
            comp_str(stringer)
        end

        if k==2;
            stringer='988_39_RB';
            comp_str(stringer)
        end

        if k==3;
            stringer='988_40_RB';
            comp_str(stringer)
        end

        if k==4;
            stringer='988_41_RB';
            comp_str(stringer)
        end

        if k==5;
            stringer='988_78_RB';
            comp_str(stringer)
        end

        if k==6;
            stringer='988_79_RB';
            comp_str(stringer)
        end

        if k==7;
            stringer='988_80_RB';
            comp_str(stringer)
        end

        if k==8;
            stringer='988_162_RB';
            comp_str(stringer)
        end

        if k==9;
            stringer='988_163_RB';
            comp_str(stringer)
        end

        if k==10;
            stringer='988_164_RB';
            comp_str(stringer)
        end

        if k==11;
            stringer='988_165_RB';
            comp_str(stringer)
        end

        if k==12;
            stringer='988_166_RB';
            comp_str(stringer)
        end

        if k==13;
            stringer='988_167_RB';
            comp_str(stringer)
        end
```

```
        if k==14;
            stringer='988_168_RB';
            comp_str(stringer)
        end

        if k==15;
            stringer='988_169_RB';
            comp_str(stringer)
        end
    end
end

% Dataset: 991
if strcmp(dataset, '991_RB')
    for k=1:22;

        if k==1;
            stringer='991_37_RB';
            comp_str(stringer)
        end

        if k==2;
            stringer='991_38_RB';
            comp_str(stringer)
        end

        if k==3;
            stringer='991_39_RB';
            comp_str(stringer)
        end

        if k==4;
            stringer='991_40_RB';
            comp_str(stringer)
        end

        if k==5;
            stringer='991_41_RB';
            comp_str(stringer)
        end

        if k==6;
            stringer='991_42_RB';
            comp_str(stringer)
        end

        if k==7;
            stringer='991_43_RB';
            comp_str(stringer)
        end

        if k==8;
            stringer='991_44_RB';
            comp_str(stringer)
        end

        if k==9;
            stringer='991_45_RB';
            comp_str(stringer)
        end

        if k==10;
            stringer='991_82_RB';
            comp_str(stringer)
        end

        if k==11;
            stringer='991_83_RB';
            comp_str(stringer)
        end
```

```
        if k==12;
            stringer='991_84_RB';
            comp_str(stringer)
        end

        if k==13;
            stringer='991_85_RB';
            comp_str(stringer)
        end

        if k==14;
            stringer='991_86_RB';
            comp_str(stringer)
        end

        if k==15;
            stringer='991_87_RB';
            comp_str(stringer)
        end

        if k==16;
            stringer='991_88_RB';
            comp_str(stringer)
        end

        if k==17;
            stringer='991_89_RB';
            comp_str(stringer)
        end

        if k==18;
            stringer='991_90_RB';
            comp_str(stringer)
        end

        if k==19;
            stringer='991_91_RB';
            comp_str(stringer)
        end

        if k==20;
            stringer='991_92_RB';
            comp_str(stringer)
        end

        if k==21;
            stringer='991_170_RB';
            comp_str(stringer)
        end

        if k==22;
            stringer='991_171_RB';
            comp_str(stringer)
        end
    end
end



function comp_str(stringer)
global colormatrix k coloric
coloric=freqColor(stringer);
colormatrix(1:26,k)=coloric;
return
```

Appendix L

Collection of Sig_bars (Frequency finder) results

```
function coloric=freqColor(stringer)
freqband=(0:0.78125:20)';
coloric=zeros(26,1);

uniqFreqs=sig_bars(stringer, 0);
[m,~]=size(uniqFreqs);
for k=1:m
    for p=1:length(freqband)
        if uniqFreqs(k,1)==freqband(p)
            coloric(p)=uniqFreqs(k,2);
        end
    end
end
```

Appendix M

Publically available functions used as dependencies

```
function Result = count(data,condition)
% COUNT (A,B)
% Counts the number of elements in A that match the criteria specified in B.
% Example:
% Data = [1 2 3 4 3 2 7 6 9 1 1 2 5 9 9];%
% Count(Data,'==9')
% ans =  3
% Richard Medlock, 2001.
nElements = length(data);
IndexIDs = 1:nElements;
Result = eval(['data' condition]);
Result = IndexIDs(Result);
Result = length(Result);
```

```
function v_row = vec2row(v)
% Force a vector to be a row vector
if(size(v,1) > max(1,size(v,2)))
    v_row = v';
else
    v_row = v;
end
```

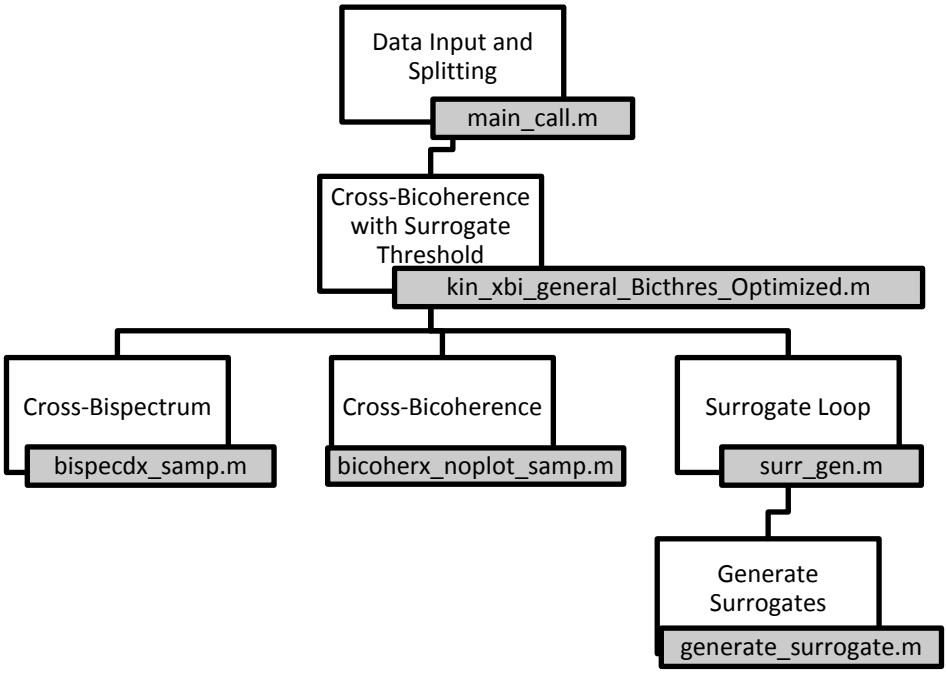
```
function z = round2(x,y)
%ROUND2 rounds number to nearest multiple of arbitrary precision.
%   Z = ROUND2(X,Y) rounds X to nearest multiple of Y.
%
%% defensive programming
error(nargchk(2,2,nargin))
error(nargoutchk(0,1,nargout))
if numel(y)>1
  error('Y must be scalar')
end

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

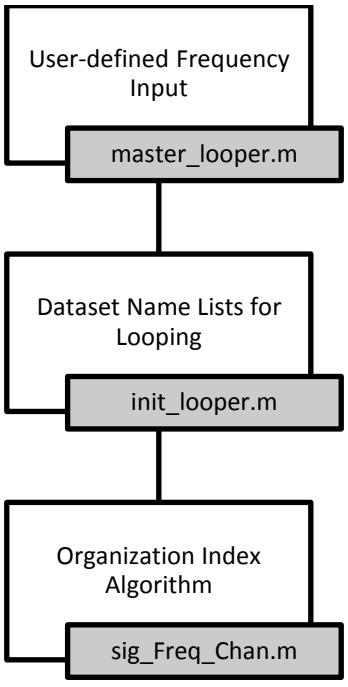
Appendix N

Flow Chart for Programs

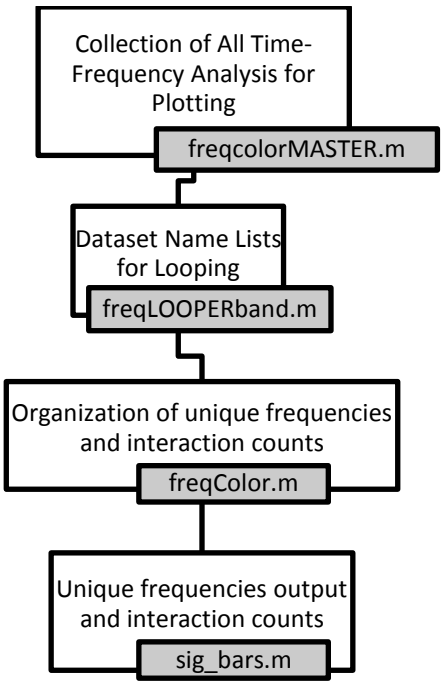
1. ***Cross-Bicoherence with Surrogate Data Threshold***



2. ***Organization Index Algorithm for Datasets In a Group,***



3. Time-Frequency Analysis



References

1. Lloyd-Jones, D., et al., *Heart Disease and Stroke Statistics--2009 Update: A Report From the American Heart Association Statistics Committee and Stroke Statistics Subcommittee.* Circulation, 2009. **119**(3): p. e21-181.
2. *slide0062_image009.jpg (JPEG Image, 1774x451 pixels)*.
3. Dash, S., et al., *Automatic Real Time Detection of Atrial Fibrillation.* Annals of Biomedical Engineering, 2009. **37**(9): p. 1701-1709.
4. Mazzini, M.J. and K.M. Monahan, *Pharmacotherapy for atrial arrhythmias: Present and future.* Heart Rhythm, 2008. **5**(6, Supplement 1): p. S26-S31.
5. Cox, J.L., *The first Maze procedure.* The Journal of Thoracic and Cardiovascular Surgery, 2011. **141**(5): p. 1093-1097.
6. Brenyo, A.J. and M.K. Aktas, *Non-Pharmacologic Management of Atrial Fibrillation.* The American Journal of Cardiology. **In Press, Corrected Proof**.
7. Haïssaguerre, M., et al., *Spontaneous Initiation of Atrial Fibrillation by Ectopic Beats Originating in the Pulmonary Veins.* New England Journal of Medicine, 1998. **339**(10): p. 659-666.
8. Yamabe, H., et al., *Analysis of the mechanisms initiating random wave propagation at the onset of atrial fibrillation using noncontact mapping: Role of complex fractionated electrogram region.* Heart Rhythm. **In Press, Uncorrected Proof**.
9. Zemlin, C.W., B.G. Mitrea, and A.M. Pertsov, *Spontaneous onset of atrial fibrillation.* Physica D, 2009. **238**(11-12): p. 969–975.
10. Everett, T.H., et al., *Frequency domain algorithm for quantifying atrial fibrillation organization to increase defibrillation efficacy.* Biomedical Engineering, IEEE Transactions on, 2001. **48**(9): p. 969-978.
11. Everett, T.H., IV, et al., *Assessment of Global Atrial Fibrillation Organization to Optimize Timing of Atrial Defibrillation.* Circulation, 2001. **103**(23): p. 2857-2861.
12. Dhein, S., et al., *Improving cardiac gap junction communication as a new antiarrhythmic mechanism: the action of antiarrhythmic peptides.* Naunyn-Schmiedeberg's Archives of Pharmacology, 2010. **381**(3): p. 221-234.
13. Guerra, J.M., et al., *Effects of the Gap Junction Modifier Rotigaptide (ZP123) on Atrial Conduction and Vulnerability to Atrial Fibrillation.* Circulation, 2006. **114**(2): p. 110-118.
14. Axelsen, L.N., et al., *Identification of ischemia-regulated phosphorylation sites in connexin43: A possible target for the antiarrhythmic peptide analogue rotigaptide (ZP123).* Journal of Molecular and Cellular Cardiology, 2006. **40**(6): p. 790-798.
15. Shiroshita-Takeshita, A., et al., *Model-Dependent Effects of the Gap Junction Conduction-Enhancing Antiarrhythmic Peptide Rotigaptide (ZP123) on Experimental Atrial Fibrillation in Dogs.* Circulation, 2007. **115**(3): p. 310-318.
16. Everett, T.H., *Enhancement of cell coupling reduces AF Vulnerability in a canine model of chronic mitral regurgitation.* Heart Rhythm, 2005. **2**(5): p. 596-597.
17. Dhein, S., *Gap junction channels in the cardiovascular system: pharmacological and physiological modulation.* Trends in Pharmacological Sciences, 1998. **19**(6): p. 229-241.
18. Beardslee, M.A., et al., *Dephosphorylation and Intracellular Redistribution of Ventricular Connexin43 During Electrical Uncoupling Induced by Ischemia.* Circ Res, 2000. **87**(8): p. 656-662.
19. Ropella, K., et al., *The coherence spectrum. A quantitative discriminator of fibrillatory and nonfibrillatory cardiac rhythms.* Circulation, 1989. **80**(1): p. 112-119.
20. Botteron, G.W. and J.M. Smith, *A technique for measurement of the extent of spatial organization of atrial activation during atrial fibrillation in the intact human heart.* Biomedical Engineering, IEEE Transactions on, 1995. **42**(6): p. 579-586.

21. Botteron, G.W. and J.M. Smith, *Quantitative Assessment of the Spatial Organization of Atrial Fibrillation in the Intact Human Heart.* Circulation, 1996. **93**(3): p. 513-518.
22. Siu, K.L., et al., *Statistical Approach to Quantify the Presence of Phase Coupling Using the Bispectrum.* Biomedical Engineering, IEEE Transactions on, 2008. **55**(5): p. 1512-1520.
23. Siu, K. and K. Chon, *On the Efficacy of the Combined Use of the Cross-Bicoherence with Surrogate Data Technique to Statistically Quantify the Presence of Nonlinear Interactions.* Annals of Biomedical Engineering, 2009. **37**(9): p. 1839-1848.
24. Shils, J.L., et al., *Bispectral analysis of visual interactions in humans.* Electroencephalography and Clinical Neurophysiology, 1996. **98**(2): p. 113-125.
25. Schreiber, T. and A. Schmitz, *Surrogate time series.* Physica D: Nonlinear Phenomena, 2000. **142**(3-4): p. 346-382.
26. Li, D., et al., *Promotion of Atrial Fibrillation by Heart Failure in Dogs : Atrial Remodeling of a Different Sort.* Circulation, 1999. **100**(1): p. 87-95.
27. Swami, A. *High-Order Spectral Analysis (HOSA) Toolbox ver. 2.0.3*. 2000 Dec 27th, 2000; Available from: http://www.mathworks.com/matlabcentral/fileexchange/3013-hosa-higher-order-spectral-analysis-toolbox.
28. Hinich, M.J. and M. Wolinsky, *Normalizing bispectra.* Journal of Statistical Planning and Inference, 2005. **130**(1-2): p. 405-411.
29. Elgar, S. and R.T. Guza, *Statistics of bicoherence.* Acoustics, Speech and Signal Processing, IEEE Transactions on, 1988. **36**(10): p. 1667-1668.
30. Everett, T.H., et al., *Structural atrial remodeling alters the substrate and spatiotemporal organization of atrial fibrillation: a comparison in canine models of structural and electrical atrial remodeling.* American Journal of Physiology - Heart and Circulatory Physiology, 2006. **291**(6): p. H2911-H2923.
31. Everett, T. and J. Olgin, *Basic mechanisms of atrial fibrillation.* Cardiol Clin, 2004. **22**(1): p. 9-20.
32. Jalife, J., O. Berenfeld, and M. Mansour, *Mother rotors and fibrillatory conduction: a mechanism of atrial fibrillation.* Cardiovascular Research, 2002. **54**(2): p. 204-216.
33. Berenfeld, O., et al., *Frequency-Dependent Breakdown of Wave Propagation Into Fibrillatory Conduction Across the Pectinate Muscle Network in the Isolated Sheep Right Atrium.* Circ Res, 2002. **90**(11): p. 1173-1180.
34. Jalife, J., *Déjà vu in the theories of atrial fibrillation dynamics.* Cardiovascular Research, 2011. **89**(4): p. 766-775.
35. Mandapati, R., et al., *Stable Microreentrant Sources as a Mechanism of Atrial Fibrillation in the Isolated Sheep Heart.* Circulation, 2000. **101**(2): p. 194-199.
36. Gautama, T. *Surrogate Data*. 2005 March 2nd, 2005; Available from: http://www.mathworks.com/matlabcentral/fileexchange/4612.