**Adapting Integral Pulse Frequency Modulation Model of Heart Rate Variability to Individuals**

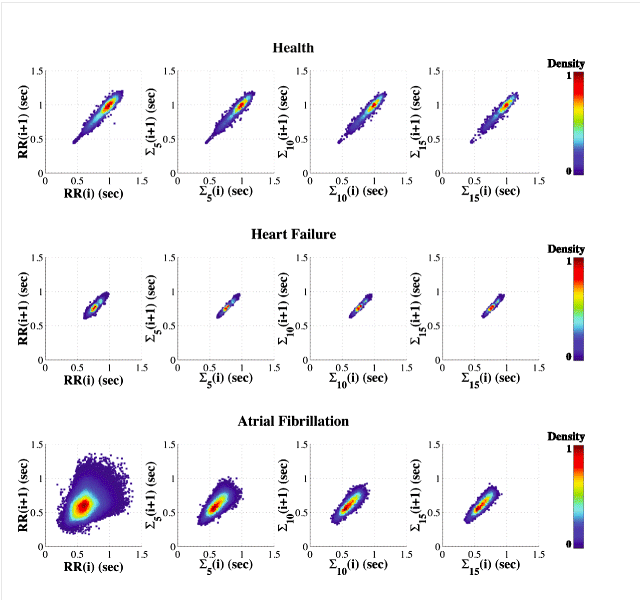
**Cassie Hamilton & Emily Kaplan**

**Biomedical Modeling and Simulation at California Polytechnic SLO**

**8 December 2016**

**Introduction:**

Heart disease is the leading cause of natural death in the U.S., causing over 600,000 deaths each year. [1]. Early detection of these diseases is critical for improved patient outcomes. Heart Rate Variability is a metric which is emerging as an effective way to predict cardiac mortality. HRV measures the variance in duration of consecutive heart beat intervals and assessing abnormalities in HRV can provide insight to a patient’s overall health and examine the effects of treatment [2]. This can be seen in Figure 0. Cardiac models that accurately reproduce HRV could lead to a better understanding of how this metric can be used to predict disease. HRV has been incorporated into existing cardiac models to quantify parasympathetic and sympathetic nervous system activity. However, autonomic activity is highly dependent on the individual. Fitting the cardiac model to these unique physiological changes in HRV will provide a more accurate tool for personal diagnostics and treatment.



**Figure 0:** Poincaré Plots can provide valuable information about cardiac abnormalities[4].

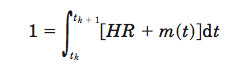
The aim of our project is to personalize a known cardiac model by examining HRV in 2 individuals and adapting the model to fit the personal changes in HRV. We will adapt the Brennan’s integral pulse frequency modulation (IPFM) model, which uses oscillators to model parasympathetic respiratory control and sympathetic control [3]. We will modify the frequency, amplitude, and time delay of these oscillating waves to match waveforms recorded from the individual at rest and under physical stress.

Data will be collected from two participants and individualized IPFM models will be created. The Poincare plots produced by the model will be compared with the plots of the raw data. The model’s accuracy will be assessed using the Ellipse fitting technique [2]. Data generated by the model will be compared to raw data by comparing dispersion of data points parallel and perpendicular to the line of identity.

We will then asses the degree of accuracy with which this model can be tailored to an individual using the ellipse fit method on the data generated by the model’s prediction of the Poincare plot and the Poincare plots generated by the raw data.

**Description of Model:**

Our model adapts the well known IPFM model. The IPFM model portrays cardiac activity by integrating an input signal over time and simulating a heartbeat when the integral reaches a threshold value. This process is described mathematically by Eq. 1.

 (1)

HR is the variable parameter for mean heart rate, 1 is the threshold value, and m(t) is the input signal describing the effects of autonomic activity.

The effects of autonomic activity on cardiac events are modeled by the addition of two oscillator waves to the input signal, described by Eq. 2. Respiration effects on cardiac activity are described by a parasympathetic respiratory oscillator (p), governed by Eq. 3, and sympathetic effects on cardiac activity are described by a sympathetic oscillator (s), governed by Eq. 4.

Screen Shot 2016-12-04 at 1.56.09 PM.png (2)

Screen Shot 2016-12-04 at 1.56.04 PM.png (3)

Screen Shot 2016-12-04 at 1.55.52 PM.png (4)

ωs is a small value, producing slow waves of more than 10 s duration and ωp is a value larger than ωs. Coupling constants Cs and Cp control levels of sympathetic and parasympathetic modulation of cardiac activity.

**Methods and Verification of Accuracy:**

Values of HR, ωs, ωp, Cs, and Cp were initially set to Brennan’s idealized values which corresponded to what they believe represented limited parasympathetic and sympathetic balance as seen in Figure 1. These parameters were then changed to fit raw data recorded from individuals.

Heart Rate Variability was recorded for one hour from each partner using a Polar Heart Strap and HRV Logger app shown in Appendix Figure I. One subject was placed in a high stress environment by being forced to debug code, as well as monitored during rest in order to see resulting differences in R-R interval data. The other subject was monitored at rest. Data from the app was extracted by converting the SQLite database to a csv file which was then read and modeled in MATLAB.

The similarities between RR-intervals of the model and of the recorded were quantified using the Ellipse fitting technique on the resulting Poincare plots. Poincare plots, also known as return maps, are a time-delayed plot of the *current* RR-interval with respect to the *next* RR-interval. An offset of 1 is normally used, but delay plots of any offset can provide different valuable information [2]. Standard deviations were taken along the line of identity and perpendicular to the line of identity. The arrays of RR-intervals and offset RR-intervals were then fit to an ellipse using a MATLAB model provided by Ohad Gal.

**Results and Conclusion:**

We were able to successfully model heart rate variability by recreating the IPFM model in MATLAB. Changing input parameters directly changed variability as seen in Figure 1 and Figure 2. Variability was visualized using the Poincare plot and then quantified using the ellipse fitting technique. When we compared ellipse fitted raw data to our model output using the parameters provided in the Brennan paper, the model was off by up to 150%, averaging 37% error, as seen in Table 1. When parameters were set to fit raw data using a guess and check method, error was vastly decreased. With a maximum error of 11.79% and average error of 5.6%, the individualized models approximated real R-R interval variation with greater accuracy than the Brennan model.



**Figure 1:** Model output when sympathetic input variable = parasympathetic input variable = 0.01. The output displays a graph of wave amplitudes over time (A), R-R interval durations over time (B), and an ellipse fitted Poincare plot (C).The resulting Poincare plot shows low variability.



**Figure 2:** Model output when sympathetic input is larger than parasympathetic input (0.3 and 0.05 respectively). The output displays a graph of wave amplitudes over time (A), R-R interval durations over time (B), and an ellipse fitted Poincare plot (C).The resulting Poincare plot and R-R interval data show greater variability.



**Figure 4:** Raw data trace for Subject 1 at rest (A) and resulting ellipse fitted Poincare plot (B), and model output Poincare plot when input variables have been set to match raw data ellipse trace from Subject 1 (C).



**Figure 5:** Raw data trace for Subject 2 at rest and under stress (blue and green respectively)(A) and resulting ellipse fitted Poincare plot (B), model output Poincare plot when input variables have been set to match raw data ellipse trace from Subject 2 at rest (C), and Model output when input variables have been set to match raw data ellipse trace from Subject 1 under stress.

**Table 1:** IPFM model compared to raw data as measured by Ellipse fitting technique. Values shown as percent differences from raw data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | Radius of X axis | Radius of Y axis | Center at X axis | Center at Y axis |
| **Subject 1**  **Current model** | **11.97%** | **5.24%** | **10.77%** | **10.59%** |
| **Subject 1**  **Brennan model** | **7.81%** | **62.61%** | **4.31%** | **4.47%** |
| **Subject 2 (rest) current model** | **11.79%** | **2.17%** | **0.18%** | **0.47%** |
| **Subject 2 (rest) Brennan model** | **152%** | **44.0%** | **30.6%** | **31.5%** |
| **Subject 2 (stress) current model** | **3.04%** | **7.88%** | **2.46%** | **1.79%** |
| **Subject 2 (stress) Brennan model** | **2.81%** | **64.2%** | **31.7%** | **27.6%** |

By personalizing the model, we are also able to better understand differences in individuals under stress and at rest. As seen in Figure 5, there are stark differences between stressed and unstressed R-R interval data. Physiologically this is due to differences in autonomic nervous system innervation, and our mathematical model was able to account for these differences

Sources of error in the model could be attributed to movement and excitation during heart rate monitoring. For example, during the recording, Subject 2 ran around the building with excitement due to a breakthrough in the model development. This resulted in approximately 10 minutes of dropped data. Raw data might not be consistent with the individualized model in a different time or setting, as only one hour of data was taken. To improve accuracy, data could be taken from a much longer recording. In clinical research, recordings of 24 hours are often used as this provides a more accurate image of the variation throughout the day. Using an optimization function to best approximate values for inputs that match raw data would also increase accuracy.

Moving forward, it would be interesting to test the effect of different kinds of stressors on heart rate data in order to better approximate sympathetic and parasympathetic contributions. Another application of individualizing the model could be to find commonalities in age groups, disease states, or experimental conditions in order to quantify the behavior of the nervous system and heart.

**References:**

1. World Health Organization. "The top 10 causes of death." (2014).

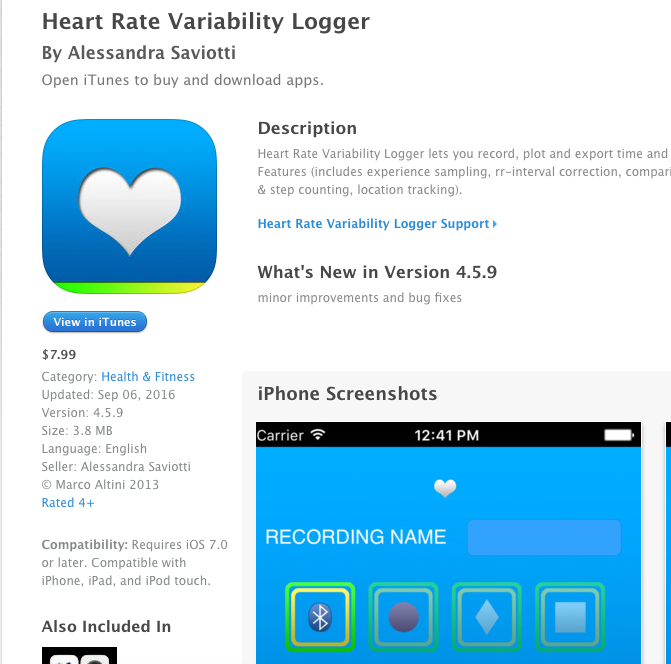
2. Yang, A. C. "Poincare Plots: A MiniReview." PhysioNet Heart Rate Variability-2006 Techniques. Applications and Futute Directions (2006).

3. Brennan, Michael, Marimuthu Palaniswami, and Peter Kamen. "Poincare plot interpretation using a physiological model of HRV based on a network of oscillators." American Journal of Physiology-Heart and Circulatory Physiology 283.5 (2002): H1873-H1886.

4. Henriques, T. S., Mariani, S., Burykin, A., Rodrigues, F., Silva, T. F., & Goldberger, A. L. (2015). Multiscale Poincaré plots for visualizing the structure of heartbeat time series. *BMC Medical Informatics and Decision Making*, *16*, 17. http://doi.org/10.1186/s12911-016-0252-0

**Appendix:**

**Figure I: Methods of Data Recording** Polar Heart Rate Strap Advertisement (left) and Heart Rate Variability Logger (right) were used to record data. SQLite database was downloaded and converted to a csv which was then read and modeled using MATLAB.



**File 1: IPFM.m**

%% Integral Pulse Frequency Modulation Model of heart beats

% integrates an input signal until it reaches a "threshold of unity"; at this point a pulse is produces and the integrator is reset to zero

% MATLABs built in modulate function will perform this for us using a

% rectangular integral approximation

close all;

clear all;

%% Constants

OFFSET = 1;

dt = 0.001;

t = 1:1000000;

%% Inputs based on the Paper

s = sin(2\*pi\*0.025\*t\*dt); % "period of the sympathetic oscillator is set to ?40 s, "

p = sin(2\*pi\*0.344\*t\*dt); % "parasympathetic oscillator is set to a period of ?3 s"..."This oscillator has a value of ?p larger than ?s, typically at the modeled respiration frequency." Normal Respiration rate is 12-20 breaths per minute

Cs = 0.01

Cp = 0.01 %balanced parasymathetic: sympathetic activity

HR0 = 1/.85; %HR corresponds to an R-R interval of 850 ms; HR is a variable parameter that represents mean heart rate

heart\_rate\_inputs = (Cs\*s)+(Cp\*p) + HR0;

% Frequency Modulate Sum of Inputs to the model

thresh = 1;

sum = 0;

beat\_times = zeros(1);

rr\_t = zeros(1);

rr\_y = zeros(1);

for i = 2:1000000

time = i\*dt;

sum = sum + dt\*heart\_rate\_inputs(i);

if sum > thresh

rr\_y(end+1) = time - beat\_times(end);

rr\_t(end+1) = time;

beat\_times(end+1) = time;

thresh = thresh + 1;

end

end

rr\_interval = diff(beat\_times);

length\_rr = length(rr\_interval);

%% Calculate Descriptive Statistics

rr = rr\_interval(1:(end-OFFSET)) .\* 1000;

rr\_delay = rr\_interval((OFFSET+1):end) .\* 1000;

x\_y = rr ./ rr\_delay;

y\_x = rr\_delay ./ rr;

data\_sd1 = std(y\_x); %perpendicular to line of identity

data\_sd2 = std(x\_y); %parallel to line of identity

%% Graph RR intervals

subplot(2,2,1);

plot(t\*dt, heart\_rate\_inputs, 'b.');

xlabel('Time (s)')

ylabel('Amplitude')

legend('Input to IPFM');

subplot(2,2,2);

plot(rr\_t, rr\_y, 'c.');

xlabel('Time (s)')

ylabel('Duration in Seconds')

legend('RR Interval');

subplot(2,1,2);

OFFSET = 1;

plot(rr, rr\_delay, 'k.'); %covert from seconds to ms

models\_ellipse = fit\_ellipse(rr, rr\_delay, gca) %print to see

xlabel('RR(n) in ms');

str = sprintf('RR(n + %d) in ms',OFFSET);

ylabel(str);

legend('Poincare Plot');

**File 2: RRDataViz.m**

%Visualize HRV from Polar Strap CSV

%% CONSTANTS

filename = 'EmilysRR\_Data.csv';

%filename = 'CassiesRR\_Data.csv';

title = filename;

OFFSET = 1;

%%M = csvread(filename,R1,C1,[R1 C1 R2 C2])

%reads only the range bounded by row offsets R1 and R2

%and column offsets C1 and C2.

%Another way to define the range is to use spreadsheet notation,

%such as 'A1..B7' instead of [0 0 6 1].

M = csvread(filename, 2, 0);

time\_stamps = M(2:end, end-1);

rr\_interval = M(2:end, end);

%% Convert to elapsed time

time = zeros(length(time\_stamps));

for i = 1:length(time\_stamps)

time(i) = time\_stamps(i) - time\_stamps(1);

end

%% Calculate Descriptive Statistics

rr = rr\_interval(1:(end-OFFSET));

rr\_delay = rr\_interval((OFFSET+1):end);

x\_y = rr ./ rr\_delay;

y\_x = rr\_delay ./ rr;

data\_sd1 = std(y\_x) %perpendicular to line of identity

data\_sd2 = std(x\_y) %parallel to line of identity

%% Graph RR intervals

figure;

annotation('textbox', [0 0.9 1 0.1], ...

'String', title, ...

'EdgeColor', 'none', ...

'HorizontalAlignment', 'center')

subplot(2,1,1);

plot(time, rr\_interval);

xlabel('Time Elapsed (s)');

ylabel('Duration in milliseconds');

legend('RR Interval');

subplot(2,1,2);

plot(rr, rr\_delay, 'k.');

refline(1,0); %line of identity

xlabel('RR(n) in ms');

str = sprintf('RR(n + %d) is ms',OFFSET);

ylabel(str);

legend('Poincare Plot');

recording\_ellipse = fit\_ellipse(rr, rr\_delay, gca) %

%% Other cool viz

% figure;

% %title('Poincare Plot with Distributions')

% scatterhist(rr, rr\_delay);

% xlabel('RR(n)');

% str = sprintf('RR(n + %d)',OFFSET);

% ylabel(str);

% legend('Poincare Plot');

**File 3: fit\_ellipse.m**

function ellipse\_t = fit\_ellipse( x,y,axis\_handle )

%

% fit\_ellipse - finds the best fit to an ellipse for the given set of points.

%

% Format: ellipse\_t = fit\_ellipse( x,y,axis\_handle )

%

% Input: x,y - a set of points in 2 column vectors. AT LEAST 5 points are needed !

% axis\_handle - optional. a handle to an axis, at which the estimated ellipse

% will be drawn along with it's axes

%

% Output: ellipse\_t - structure that defines the best fit to an ellipse

% a - sub axis (radius) of the X axis of the non-tilt ellipse

% b - sub axis (radius) of the Y axis of the non-tilt ellipse

% phi - orientation in radians of the ellipse (tilt)

% X0 - center at the X axis of the non-tilt ellipse

% Y0 - center at the Y axis of the non-tilt ellipse

% X0\_in - center at the X axis of the tilted ellipse

% Y0\_in - center at the Y axis of the tilted ellipse

% long\_axis - size of the long axis of the ellipse

% short\_axis - size of the short axis of the ellipse

% status - status of detection of an ellipse

%

% Note: if an ellipse was not detected (but a parabola or hyperbola), then

% an empty structure is returned

% =====================================================================================

% Ellipse Fit using Least Squares criterion

% =====================================================================================

% We will try to fit the best ellipse to the given measurements. the mathematical

% representation of use will be the CONIC Equation of the Ellipse which is:

%

% Ellipse = a\*x^2 + b\*x\*y + c\*y^2 + d\*x + e\*y + f = 0

%

% The fit-estimation method of use is the Least Squares method (without any weights)

% The estimator is extracted from the following equations:

%

% g(x,y;A) := a\*x^2 + b\*x\*y + c\*y^2 + d\*x + e\*y = f

%

% where:

% A - is the vector of parameters to be estimated (a,b,c,d,e)

% x,y - is a single measurement

%

% We will define the cost function to be:

%

% Cost(A) := (g\_c(x\_c,y\_c;A)-f\_c)'\*(g\_c(x\_c,y\_c;A)-f\_c)

% = (X\*A+f\_c)'\*(X\*A+f\_c)

% = A'\*X'\*X\*A + 2\*f\_c'\*X\*A + N\*f^2

%

% where:

% g\_c(x\_c,y\_c;A) - vector function of ALL the measurements

% each element of g\_c() is g(x,y;A)

% X - a matrix of the form: [x\_c.^2, x\_c.\*y\_c, y\_c.^2, x\_c, y\_c ]

% f\_c - is actually defined as ones(length(f),1)\*f

%

% Derivation of the Cost function with respect to the vector of parameters "A" yields:

%

% A'\*X'\*X = -f\_c'\*X = -f\*ones(1,length(f\_c))\*X = -f\*sum(X)

%

% Which yields the estimator:

%

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

% | A\_least\_squares = -f\*sum(X)/(X'\*X) ->(normalize by -f) = sum(X)/(X'\*X) |

% ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%

% (We will normalize the variables by (-f) since "f" is unknown and can be accounted for later on)

%

% NOW, all that is left to do is to extract the parameters from the Conic Equation.

% We will deal the vector A into the variables: (A,B,C,D,E) and assume F = -1;

%

% Recall the conic representation of an ellipse:

%

% A\*x^2 + B\*x\*y + C\*y^2 + D\*x + E\*y + F = 0

%

% We will check if the ellipse has a tilt (=orientation). The orientation is present

% if the coefficient of the term "x\*y" is not zero. If so, we first need to remove the

% tilt of the ellipse.

%

% If the parameter "B" is not equal to zero, then we have an orientation (tilt) to the ellipse.

% we will remove the tilt of the ellipse so as to remain with a conic representation of an

% ellipse without a tilt, for which the math is more simple:

%

% Non tilt conic rep.: A`\*x^2 + C`\*y^2 + D`\*x + E`\*y + F` = 0

%

% We will remove the orientation using the following substitution:

%

% Replace x with cx+sy and y with -sx+cy such that the conic representation is:

%

% A(cx+sy)^2 + B(cx+sy)(-sx+cy) + C(-sx+cy)^2 + D(cx+sy) + E(-sx+cy) + F = 0

%

% where: c = cos(phi) , s = sin(phi)

%

% and simplify...

%

% x^2(A\*c^2 - Bcs + Cs^2) + xy(2A\*cs +(c^2-s^2)B -2Ccs) + ...

% y^2(As^2 + Bcs + Cc^2) + x(Dc-Es) + y(Ds+Ec) + F = 0

%

% The orientation is easily found by the condition of (B\_new=0) which results in:

%

% 2A\*cs +(c^2-s^2)B -2Ccs = 0 ==> phi = 1/2 \* atan( b/(c-a) )

%

% Now the constants c=cos(phi) and s=sin(phi) can be found, and from them

% all the other constants A`,C`,D`,E` can be found.

%

% A` = A\*c^2 - B\*c\*s + C\*s^2 D` = D\*c-E\*s

% B` = 2\*A\*c\*s +(c^2-s^2)\*B -2\*C\*c\*s = 0 E` = D\*s+E\*c

% C` = A\*s^2 + B\*c\*s + C\*c^2

%

% Next, we want the representation of the non-tilted ellipse to be as:

%

% Ellipse = ( (X-X0)/a )^2 + ( (Y-Y0)/b )^2 = 1

%

% where: (X0,Y0) is the center of the ellipse

% a,b are the ellipse "radiuses" (or sub-axis)

%

% Using a square completion method we will define:

%

% F`` = -F` + (D`^2)/(4\*A`) + (E`^2)/(4\*C`)

%

% Such that: a`\*(X-X0)^2 = A`(X^2 + X\*D`/A` + (D`/(2\*A`))^2 )

% c`\*(Y-Y0)^2 = C`(Y^2 + Y\*E`/C` + (E`/(2\*C`))^2 )

%

% which yields the transformations:

%

% X0 = -D`/(2\*A`)

% Y0 = -E`/(2\*C`)

% a = sqrt( abs( F``/A` ) )

% b = sqrt( abs( F``/C` ) )

%

% And finally we can define the remaining parameters:

%

% long\_axis = 2 \* max( a,b )

% short\_axis = 2 \* min( a,b )

% Orientation = phi

%

%

% initialize

orientation\_tolerance = 1e-3;

% empty warning stack

warning( '' );

% prepare vectors, must be column vectors

x = x(:);

y = y(:);

% remove bias of the ellipse - to make matrix inversion more accurate. (will be added later on).

mean\_x = mean(x);

mean\_y = mean(y);

x = x-mean\_x;

y = y-mean\_y;

% the estimation for the conic equation of the ellipse

X = [x.^2, x.\*y, y.^2, x, y ];

a = sum(X)/(X'\*X);

% check for warnings

if ~isempty( lastwarn )

disp( 'stopped because of a warning regarding matrix inversion' );

ellipse\_t = [];

return

end

% extract parameters from the conic equation

[a,b,c,d,e] = deal( a(1),a(2),a(3),a(4),a(5) );

% remove the orientation from the ellipse

if ( min(abs(b/a),abs(b/c)) > orientation\_tolerance )

orientation\_rad = 1/2 \* atan( b/(c-a) );

cos\_phi = cos( orientation\_rad );

sin\_phi = sin( orientation\_rad );

[a,b,c,d,e] = deal(...

a\*cos\_phi^2 - b\*cos\_phi\*sin\_phi + c\*sin\_phi^2,...

0,...

a\*sin\_phi^2 + b\*cos\_phi\*sin\_phi + c\*cos\_phi^2,...

d\*cos\_phi - e\*sin\_phi,...

d\*sin\_phi + e\*cos\_phi );

[mean\_x,mean\_y] = deal( ...

cos\_phi\*mean\_x - sin\_phi\*mean\_y,...

sin\_phi\*mean\_x + cos\_phi\*mean\_y );

else

orientation\_rad = 0;

cos\_phi = cos( orientation\_rad );

sin\_phi = sin( orientation\_rad );

end

% check if conic equation represents an ellipse

test = a\*c;

switch (1)

case (test>0), status = '';

case (test==0), status = 'Parabola found'; warning( 'fit\_ellipse: Did not locate an ellipse' );

case (test<0), status = 'Hyperbola found'; warning( 'fit\_ellipse: Did not locate an ellipse' );

end

% if we found an ellipse return it's data

if (test>0)

% make sure coefficients are positive as required

if (a<0), [a,c,d,e] = deal( -a,-c,-d,-e ); end

% final ellipse parameters

X0 = mean\_x - d/2/a;

Y0 = mean\_y - e/2/c;

F = 1 + (d^2)/(4\*a) + (e^2)/(4\*c);

[a,b] = deal( sqrt( F/a ),sqrt( F/c ) );

long\_axis = 2\*max(a,b);

short\_axis = 2\*min(a,b);

% rotate the axes backwards to find the center point of the original TILTED ellipse

R = [ cos\_phi sin\_phi; -sin\_phi cos\_phi ];

P\_in = R \* [X0;Y0];

X0\_in = P\_in(1);

Y0\_in = P\_in(2);

% pack ellipse into a structure

ellipse\_t = struct( ...

'a',a,...

'b',b,...

'phi',orientation\_rad,...

'X0',X0,...

'Y0',Y0,...

'X0\_in',X0\_in,...

'Y0\_in',Y0\_in,...

'long\_axis',long\_axis,...

'short\_axis',short\_axis,...

'status','' );

else

% report an empty structure

ellipse\_t = struct( ...

'a',[],...

'b',[],...

'phi',[],...

'X0',[],...

'Y0',[],...

'X0\_in',[],...

'Y0\_in',[],...

'long\_axis',[],...

'short\_axis',[],...

'status',status );

end

% check if we need to plot an ellipse with it's axes.

if (nargin>2) & ~isempty( axis\_handle ) & (test>0)

% rotation matrix to rotate the axes with respect to an angle phi

R = [ cos\_phi sin\_phi; -sin\_phi cos\_phi ];

% the axes

ver\_line = [ [X0 X0]; Y0+b\*[-1 1] ];

horz\_line = [ X0+a\*[-1 1]; [Y0 Y0] ];

new\_ver\_line = R\*ver\_line;

new\_horz\_line = R\*horz\_line;

% the ellipse

theta\_r = linspace(0,2\*pi);

ellipse\_x\_r = X0 + a\*cos( theta\_r );

ellipse\_y\_r = Y0 + b\*sin( theta\_r );

rotated\_ellipse = R \* [ellipse\_x\_r;ellipse\_y\_r];

% draw

hold\_state = get( axis\_handle,'NextPlot' );

set( axis\_handle,'NextPlot','add' );

plot( new\_ver\_line(1,:),new\_ver\_line(2,:),'r' );

plot( new\_horz\_line(1,:),new\_horz\_line(2,:),'r' );

plot( rotated\_ellipse(1,:),rotated\_ellipse(2,:),'r' );

set( axis\_handle,'NextPlot',hold\_state );

end