Structural Health Monitoring Tools (SHMTools)

Function Library

LANL/UCSD Engineering Institute

LA-CC-14-046
LA-UR-14-21142

April 1, 2019

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 addResp2Geom_shm.m File Reference

Optimal Sensor Placement: Add response shape to geometry layout.

Call Methods

[respLayout, respScale] = addResp2Geom_shm (geomLayout, respXYZ, scale)

[Response Layout, Response Scale]= Add Response to Geometry (Geometry Layout, Response XYZ, Scale)

Function Description

Adds response shape in three coordinates to geometry layout.

Parameters

→ ***respLayout*** (POINTINFO, POINTS, INSTANCES) : Layout of geometric points with response shape added, POINTINFO is the ordered set [ID XCoord, YCoord, ZCoord]

→ ***respScale*** (1, POINTS, INSTANCES) : Norm of the response vector at each geometric point

← ***geomLayout*** (POINTINFO, POINTS) : Layout array defining the IDs and coordinates of geometric points, POINTINFO is the ordered set [coordID, X, Y, Z]

← ***respXYZ*** (DIRECTIONS, COORDINATES, INSTANCES) : Array of response in the X, Y, and Z directions for the coordinates defined in "geomLayout", DIRECTIONS is the ordered set [X, Y, Z]

← ***scale*** (scalar) : scale to multiply response by before it is added to the geometry (default is 10 of geometry size / maximum response)

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.2 analyticSignal_shm.m File Reference

Feature Extraction: Converts signals to their analytic form.

Call Methods

[analyticMatrix] = analyticSignal_shm (X)

[Analytic Signal Matrix] = Analytic Signal (Signal Matrix)

Function Description

Returns a matrix of analytic signals converted by Hilbert transform using the FFT. The analytic signal consists of a real part of a signal as well as it's imaginary parts. To compute analytic signal, the FFT of the real signal is calculated and its positive frequency components are doubled; its negative frequency components are set to zero. By taking the inverse Fourier transform of the manipulated frequency domain, the analytic signal is formed.

Parameters

→ ***analyticMatrix*** (SAMPLES, CHANNELS, INSTANCES) : analytic signal matrix

← ***X*** (SAMPLES, CHANNELS, INSTANCES) : signal matrix

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.3 arModel_shm.m File Reference

Feature Extraction: AutoRegressive model (AR).

Call Methods

[arParametersFV, RMSresidualsFV, arParameters, arResiduals, arPrediction] = arModel_shm (X, arOrder)

[AR Parameters Feature Vectors, RMS Residuals Feature Vectors, AR Parameters, AR Residuals, AR Prediction] = AR Model (Time Series Data, AR Model Order)

Function Description

Returns feature vectors of AR parameters and root mean square of the AR residual errors in concatenated format. Additionally, for each time series it returns the AR parameters, residual errors, and predicted signals.

Parameters

- → ***arParametersFV*** (INSTANCES, FEATURES) : Feature vectors of AR parameters in concatenated format, FEATURES = CHANNELS∗arOrder
- → ***RMSresidualsFV*** (INSTANCES, FEATURES) : Feature vectors of root mean squared AR residual errors in concatenated format, FEATURES = CHANNELS
- → ***arParameters*** (ARORDER, CHANNELS, INSTANCES) : AR parameters
- → ***arResiduals*** (TIME, CHANNELS, INSTANCES) : AR residuals
- → ***arPrediction*** (TIME, CHANNELS, INSTANCES) : AR prediction
- ← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data
- ← ***arOrder*** (integer) : AR model order

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

Figueiredo, E., Park, G., Figueiras, J., Farrar, C., and Worden, K. (2009). Structural Health Monitoring Algorithm Comparisons Using Standard Datasets, Los Alamos National Laboratory Report: LA-14393.

Sample Data Set

www.lanl.gov/projects/ei

2.4 arModelOrder_shm.m File Reference

Feature Extraction: Determine appropriate AutoRegressive model order (AR).

Call Methods

[meanARorder, arOrders, model] = arModelOrder_shm (X, method, arOrderMax, tol)

[Mean AR Order, AR Orders, Model] = AR Model Order (Time Series Data, Method, Maximum AR Order, Tolerance)

Function Description

This function computes the suitable AR model order based on one out of four available methods. The methods are based on Akaike's information criterion (AIC), partial autocorrelation function (PAF), root mean squared (RMS) error, singular value decomposition (SVD), and Bayesian Information Criterion (BIC).

Parameters

→ ***meanARorder*** (1, CHANNELS) : mean AR model order

→ ***arOrders*** (1, CHANNELS, INSTANCES) : suitable AR model order for each time series

→ ***model*** (struct) : output parameters of the model, fields are as follows: .outData (1, OUTPUTS) : vector containing the outputs of method

→ ***.arOrderList*** (1, ORDERS) : vector containing the sequential AR model order up to arOrderMax

→ ***.controlLimit*** (scalar) : threshold to pick the AR model order (for PAF method the output is a vector with upper and lower limits)

→ ***.method*** (string) : method used to find the suitable order

→ ***.arOrderMax*** (integer) : maximum AR model order computed model

→ ***.tol*** (scalar) : tolerance

← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***method*** (string) : optimization method; 'AIC', 'PAF', 'SVD', 'RMS', or 'BIC'

← ***arOrderMax*** (integer) : maximum AR model order to be computed

← ***tol*** (scalar) : tolerance

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

Figueiredo, E., Park, G., Figueiras, J., Farrar, C., and Worden, K. (2009). Structural Health Monitoring Algorithm Comparisons Using Standard Data Sets, Los Alamos National Laboratory Report: LA-14393.

Sample Data Set

http://institute.lanl.gov/ei/software-and-data/

2.5 arsAccel_shm.m File Reference

Feature Extraction: Resamples signals to angular domain using accel.

Call Methods

[xARSMatrix, samplesPerRev] = arsAccel_shm (X, Fs, nFilter, samplesPerRev, Fc, Fsb, meshOrder, nGearTeeth)

[Angular Resampled Signal Matrix, Samples per Revolution] = Angular Resampling for Acceleration (Signal Matrix, Sampling Frequency, Number of Filter Coefficients, Desired Samples per Revolution, Band Pass Central Frequency, Filter Side Band Width, Order of Frequency, Number of Gear Teeth)

Function Description

Angular resampling of signals for acceleration. If there exist a periodic component in an vibration signal that can be attributed to a gear mesh order, it is possible to angular resample a signal to rotations of that gear. The algorithm uses a band pass filter to separate the gear mesh order from the rest of the signal. This separated mesh order signal is converted to and analytic signal to determine the phase associated with a particular gear mesh order that was filtered out along with knowledge of the number of gear teeth for the particular gear and the order that was separated is used to estimate the phase of the shaft. That is then used to angular resample the signal to a specified samples per revolution. The function resamples to a specified samples per revolution of the shaft but if no sample per revolution is specified it resamples to the minimum sample per revolution resolution of the signal. If the desired samples per revolution is specified as higher than the minimum resolution, the signal will be upsampled to a resolution that allows for generating the splines required to resample to the desires SPR. This form of resampling is not appropriate for vibration signals where the shaft speed is subject to large variations or quick fluctuations but can be valuable in further resampling to a particular order after a signal has been order tracked using a tachometer signal. Fc is to be the Central Frequency of the gear harmonic order to be band pass filtered and Fsb the frequency width from the central frequency to the nearest first order sideband. The meshorder and number of teeth in the gear that is being tracked are to be specified so that the shaft phase can be interpolated from the band passed signal. nFilter is the desired filter length of the filtering operations used in the algorithm for band pass filtering the harmonic as well as antialiasing filtering used after interpolations.

Parameters

→ ***xARSMatrix*** (SAMPLES, CHANNELS, INSTANCES) : angular resampled signal matrix, SAMPLES = samplesPerRev∗REVOLUTIONS

→ ***samplesPerRev*** (integer): samples per revolution in xARSMatrix

← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***Fs*** (scalar) : sampling frequency of signal matrix

← ***nFilter*** (integer) : number of filter coefficients in band pass filter

← ***samplesPerRev*** (integer): samples per revolution in xARSMatrix

← ***Fc*** (scalar) : band pass filter central frequency (mesh frequency)

← ***Fsb*** (scalar) : frequency interval between meshing frequency and first sideband

← ***meshOrder*** (integer) : filtered harmonic order

← ***nGearTeeth*** (integer) : number of gear teeth

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Sets

1. `signal matrix, SAMPLES = samplesPerRev*REVOLUTIONS`
2. `samplesPerRev (integer): samples per revolution in xARSMatrix`

2.6 arsTach_shm.m File Reference

Feature Extraction: Resamples signals to angular domain using tachyometer.

Call Methods

[xARSMatrix, samplesPerRev] = arsTach_shm (X, nFilter, samplesPerRev, gearRatio)

[Angular Resampled Signal Matrix, Samples per Revolution] = Angular Resampling from Tachyometer (Signal Matrix, Number of Filter Coefficients, Samples Per Revolution, Gear Ratio)

Function Description

Resamples a matrix of signals individually in the time domain to a specified equally spaced angular domain using a tachometer matrix with tachometer signals corresponding to each instance. If X consists of multiple channels then there is one tach signal for each channel. Primarily used for order tracking of signals by resampling a signal to an equally sampled angular domain, frequency components that may be smeared due to varying shaft speed in the time domain have increased resolution by resampling to the angular domain. This function determines a shaft phase based on a tachometer signal. The tach signals are generated using an optical sensor that gives one pulse per revolution of a shaft which has a reflector that passes once per revolution. By determining the zero crossings of the tach signal the shaft phase can be interpolated to the time domain. This phase signal is used to resample to angular resample the time domain signal to a set samples per revolution of the shaft but if no sample per revolution is specified it resamples to the minimum samples per revolution resolution of the signal. If the desired samples per revolution is specified as higher than the minimum angular resolution the signal will be upsampled to a angular resolution that allows for generating the splines required to interpolate the signal to the desires samples per revolution. If it is desired to resample the signal to a shaft speed that is separate from the tach pulse signal shaft a gear ratio can be specified that modifies the phase interpolated from the main shaft where the tachometer is located to the alternate shaft separated by a gear or belt drive. NOTE: FIRST CHANNEL OF SIGNAL MATRIX X MUST BE THE TACHOMETER SIGNAL.

Parameters

- → ***xARSMatrix*** (SAMPLES, CHANNELS, INSTANCES) : angular resampled signal matrix, SAMPLES = samplesPerRev∗REVOLUTIONS
- → ***samplesPerRev*** (integer): samples per revolution in xARSMatrix
- ← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data, first channel must be tachometer signal
- ← ***nFilter*** (integer) : number of filter coefficients.
- ← ***samplesPerRev*** (integer) : number of samples per revolution
- ← ***gearRatio*** (scalar) : gear ratio

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Sets

1. `signal matrix, SAMPLES = samplesPerRev*REVOLUTIONS`
2. `samplesPerRev (integer): samples per revolution in xARSMatrix`

2.7 arsvmModel_shm.m File Reference

Feature Extraction: AutoRegressive Support Vector Machine model (ARSVM).

Call Methods

[arsvmModels, arsvmModelOrder] = arsvmModel_shm (X, arsvmModelOrder, s, t, e)

[ARSVM Models, ARSVM Model Order] = ARSVM Model (Time Series Data, ARSVM Model Order, SVM Type, Kernel Function Type, Tolerance)

Function Description

Build an AutoRegressive Support Vector Machine (ARSVM) model to be used with evalARmodel_shm.

Parameters

→ ***arsvmModels*** {CHANNELS,1} (struct) : AutoRegressive Support Vector Machine models for each channel, input to evalARmodel_shm

→ ***arsvmModelOrder*** (integer) : ARSVM model order

← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***arsvmModelOrder*** (integer) : ARSVM model order

← ***s*** (integer) : type of SVM, options are 0 = C-SVC, 1 = nu-SVC, 2 = one-class SVM, 3 = epsilon-SVR, or 4 = nu-SVR (default is 0)

← ***t*** (integer) : type of kernel function, options are 0 = linear, 1 = polynomial, 2 = radial basis function, 3 = sigmoid, or 4 = precomputed kernel (default is 2)

← ***e*** (scalar) : tolerance of termination criterion

Author

Luke Bornn
Dustin Harvey
LA-CC-14-046

Modifications

1. none

Reference

```
Bornn, L., Farrar, C.R., Park, G., Farinholt,K., (2009).
Structural Health Monitoring With Autoregressive Support
Vector Machines.  Journal of Vibration And Acoustics,
131, 021004.
```

Sample Data Set

```
none
```


2.8 arxModel_shm.m File Reference

Feature Extraction: AutoRegressive model with eXogenous inputs (ARX).

Call Methods

[arxParametersFV, RMSresidualsFV, arxParameters, arxResiduals, arxPrediction, arxOrders] = arxModel_shm (X, arxOrders)

[ARX Parameters Feature Vectors, RMS Residuals Feature Vectors, ARX Parameters, ARX Residuals, ARX Prediction, ARX Model Orders] = ARX Model (Time Series Data, ARX Model Orders)

Function Description

This function returns the ARX parameters and the root mean square (RMS) of the residual errors in concatenated format by using the least squares approach. This function assumes multi-output and single-input data. The first column is assumed to be the input time series.

Parameters

→ ***arxParametersFV*** (INSTANCES, FEATURES) : n feature vectors of AXR parameters in concatenated format, FEATURES = CHANNELS∗(a+b)

→ ***RMSresidualsFV*** (INSTANCES, FEATURES) : n feature vectors of root mean squared ARX residual errors in concatenated format, FEATURES = OUTPUTCHANNELS

→ ***arxParameters*** (ORDER, OUTPUTCHANNELS, INSTANCES) : ARX parameters, ORDER = a+b

→ ***arxResiduals*** (TIME, OUTPUTCHANNELS, INSTANCES) : ARX residuals

→ ***arxPrediction*** (TIME, OUTPUTCHANNELS, INSTANCES) : ARX prediction

→ ***arxOrders*** (1, ORDERS) : vector of integers specifying the orders of the ARX model; ORDERS is the ordered set [a b tau] where a and b are the orders of the outputs and input respectively, and tau is the delay of the input

← ***X*** (TIME, CHANNELS, INSTANCES) : matrix composed of input and output time series (the first column in CHANNELS is the input time series, CHANNELS = OUTPUTCHANNELS + 1)

← ***arxOrders*** (1, ORDERS) : vector of integers specifying the orders of the ARX model; ORDERS is the ordered set [a b tau] where a and b are the orders of the outputs and input respectively, and tau is the delay of the input (0 by default)

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

www.lanl.gov/projects/ei

2.9 assembleOutlierDetector_shm.m File Reference

Outlier Detection: Assemble outlier detector.

Call Methods

assembledDetector = assembleOutlierDetector_shm (suffix, DirBase)

Assembled Detector Structure = Assemble Outlier Detector(Suffix, Base Directory)

Function Description

Use this function to automatically assemble an outlier detector. After a series of questions are answered, a training routine is produced to be used along with detectOutlier_shm for detection (the training module will return all relevant information to the detection routine).

Parameters

- → ***assembledDetector*** (struct): contains the parameters selected during assembly. The actual assembled training routine is saved in AssembledDetectors directory.
- ← ***suffix*** (string): the training function created will be named trainOutlierDetector_{suffix}. By default the suffix is just a time stamp.
- ← ***DirBase*** (string): should be the full path to the "Detection" directory. If not provided, then the directory where this file is installed will be used.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.10 bandLimWhiteNoise_shm.m File Reference

Data Acquisition: Generate band-limited white noise.

Call Methods

waveform = bandLimWhiteNoise_shm (arraySize, cutoffs, rms)

Waveform = Band Limited White Noise (Array Size, Cutoffs, RMS)

Function Description

Generates band limited white noise for the normalized frequency range specifed by "cutoffs" with specified rms. This is accomplished by forward and reverse filtering the signal with a fourth order butterworth.

Parameters

→ ***waveform*** (POINTS, COLUMNS, ...) : band limited white noise

← ***arraySize*** (1, DIMENSIONS) : size of the waveform matrix

← ***cutoffs*** (1, FREQUENCIES) : normalized frequency cutoffs, FREQUENCIES is the ordered set [low high], high is optional

← ***rms*** (scalar) : RMS of the signal

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.11 buildContainedGrid_shm.m File Reference

Active Sensing: Lists uniformly spaced 2D coordinates within border.

Call Methods

[pointList, pointMask, xMatrix, yMatrix] = buildContainedGrid_shm (borderStruct, xSpacing, ySpacing)

[Grid Points, Points of Interest Mask, X Matrix, Y Matrix] = Build Contained Grid (Border Structure, X Spacing, Y Spacing

Function Description

Constructs a list of uniformly spaced coordinates which are contained within the line segments listed in "border.outside" and outside of the line segments listed in "border.inside". Can include any number of closed outside and inside border polygons. Distince closed polygons can be stored as cells or NaN delimited arrays.Also returns a logical mask matrix and X and Y coordinate matrices so that pointList=[xMatrix(pointMask); yMatrix(pointMask)]. "xMatrix" and "yMatrix" span the rectangle defined by the extremities of "outsideBorder". "pointMask" is intended for filling two dimensional maps of data which correspond the points of interest.

Parameters

- → ***pointList*** (COORDINATES, POINTS) : List of uniformly spaced contained points; COORDINATES is the ordered set [x, y]
- → ***pointMask*** (XINDEX, YINDEX) : Logical mask of points from "xMatrix" and "yMatrix" which are contained within "outsideBorder"
- → ***xMatrix*** (XINDEX, YINDEX) : Matrix of uniformly spaced X coordinates
- → ***yMatrix*** (XINDEX, YINDEX) : Matrix of uniformly spaced Y coordinates
- ← ***borderStruct*** (struct) : contains following fields: .outside {BORDERS,1} (ENDPOINTS, SEGMENTS) : Cell array of matrices of line segments which define a continous plate border. ENDPOINTS is ordered set [startX startY endX endY] describing each segment
- ← ***.inside*** {BORDERS,1} (ENDPOINTS, SEGMENTS) : Cell array of matrices of line segments which define a continous plate border. ENDPOINTS is ordered set [startX startY endX endY] describing each segment
- ← ***xSpacing*** (scalar) : Desired spacing between X coordinates
- ← ***ySpacing*** (scalar) : Desired spacing between Y coordinates

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.12 buildCoverTree_shm.m File Reference

Outlier Detection: Build cover tree data structure.

Call Methods

coverTree = buildCoverTree_shm (X, l, fastDist)

Cover Tree = Build Cover Tree (Training Data, Maximum Level, Fast Metric Function)

Function Description

Build a data structure (cover tree) for fast estimation of kernel weights. The main data structure consists of r-nets, r = diameter of X down to r = diam(X)/2^l. The tree is a parent child relationship over these nets. Fields are used by rangeQuery to quickly find training points within a range of a given test point for kernel evaluation.

Parameters

- → ***coverTree*** (struct) : includes the following fields: .tree (struct) : main data structure (r-nets)
- → ***.ic*** (1, INSTANCES) : ordering of the points
- ← ***X*** (INSTANCES, FEATURES) : training features
- ← ***l*** (scalar) : the maximum level of the tree, i.e. the covers, r, are from diam(X) to diam(X)/2^l. By default, if l is not provided, then l is set to infinity, i.e. the covers are built down to r is min distances between the points in X.
- ← ***fastDist*** (handle) : metric to use, one of the functions in FastMetricKernelEstimation/DistanceMetrics, default is a Euclidean implementation

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.13 buildPairList_shm.m File Reference

Data Acquisition: Build list of sensor pair combinations.

Call Methods

pairList = buildPairList_shm (channels, isBiDirectional, isPitchCatch, isPulseEcho)

[Pair List] = Build Pair List (Channels, Is Bidirectional, Is Pitch Catch, Is Pulse Echo)

Function Description

Build a list of actuator-sensor pair combinations for active sensing from a list of available channels.

Parameters

→ ***pairList*** (SENSORIDS, PAIRS) : Matrix of actuator-sensor PAIRS, SENSORIDS is the ordered set of integer IDs [actuatorID, sensorID]

← ***channels*** (1, CHANNELS) : List of integer channels to multiplex through

← ***isBiDirectional*** (logical) : whether actuators and sensors should be paired up in both directions, i.e. include both pairs [i,j] and [j,i]

← ***isPitchCatch*** (logical) : whether to include pairs [i,j] where i∼=j

← ***isPulseEcho*** (logical) : whether to include pairs [i,j] where i=j

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.14 cepstrum_shm.m File Reference

Feature Extraction: Computes the cepstrum of a real signal.

Call Methods

[cepstrumMatrix] = cepstrum_shm (X, nFFT, cepstrumType)

[Cepstrum Matrix] = Cepstrum (Signal Matrix, Number of FFT Bins, Cepstrum Type)

Function Description

Computes real cepstrums or complex cepstrums of a signal matrix. The real cepstrum is calculated by taking the natural log of the magnitude of a signals Fourier transform and then taking its Fourier transform. The complex cepstrum is determined by taking the natural log of the magnitude of the signals Fourier transform then corrects the phase component by removing a linear trend such that there is no phase discontinuity at +/- pi then taking the inverse Fourier transform of the modified fft to get the complex cepstrum. By default the length of the Fourier transform used is set to the length of the input signals but if desired it can be specified such that the signal is zero padded in the fft calculations and returns an nFFT length cepstrum.

Parameters

→ ***cepstrumMatrix*** (NFFT, CHANNELS, INSTANCES) : cepstrum matrix

← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***nFFT*** (integer) : Number of Bins in FFT

← ***cepstrumType*** (character) : Cepstrum Type 'C' - Complex Cepstrum 'R' - Real Cepstrum

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.15 cmif_shm.m File Reference

Feature Extraction: Computes the complex mode indicator function (CMIF).

Call Methods

h = cmif_shm (G)

Complex Mode Indicator Function = Complex Mode Indicator Function (Frequency Response Function)

Function Description

Computes the complex mode indicator function to help identify mode shapes

Parameters

→ ***h*** (INSTANCES, FREQUENCY) : complex mode indicator function

← ***G*** (FREQUENCY, DOFS, INSTANCES) : matrix of complex frequency response functions

Author

Scott W. Doebling
Phillip J. Cornwell
Erik G. Straser
Charles R. Farrar
LA-CC-14-046

Modifications

1. March 4, 2009, by Stuart Taylor, for inclusion with the SHM software modal analysis group

Reference

`none`

Sample Data Set

`www.lanl.gov/projects/ei`

2.16 coherentMatchedFilter_shm.m File Reference

Active Sensing: Coherent matched filter.

Call Methods

filterResult = coherentMatchedFilter_shm (waveform, matchedWaveform)

Filter Result = Coherent Matched Filter (Waveform, Matched Waveform)

Function Description

Applies a coherent matched filter to the waveform. This is equivalent to taking the dot product of the orignal waveform with the matched waveform. If the original waveform is not the same length as the matched waveform, then the filter is applied at each point in waveform through convolution. The start and end of the convolution result are truncated so that the filtered waveform is the same size as the original. Coherent matched filters are useful in the optimal detection of completely known signals.

Parameters

→ ***filterResult*** (TIME, ...) : multi-dimensional matrix of filter results when length(waveform)==length(matchedWaveform), multi-dimensional matrix of filtered waveforms when length(waveform)>length(matchedWaveform)

← ***waveform*** (TIME1, ...) : multi-dimensional matrix of waveforms, time is along first dimension, all other dimensions are arbitrary

← ***matchedWaveform*** (TIME2, 1) : vector containing matched waveform

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

```
Kay, SM. Fundamentals of Statistical Signal Processing,
Volume 2:  Detection Theory.  Prentice Hall PTR, 1998.
```

Sample Data Set

```
none
```


2.17 comac_shm.m File Reference

Feature Extraction: Computes coordinate modal assurance criteria (COMAC).

Call Methods

CM = comac_shm (modes1, modes2)

COMAC matrix = Coordinate Modal Assurance Criteria (Mode Shape Array 1, Mode Shape Array 2)

Function Description

Computes the coordinate modal assurance criteria

Parameters

→ ***CM*** (MODES, MODES) : Modal Assurance Criterion array

← ***modes1*** (DOFS, MODES) : Array containing first set of mode shapes

← ***modes2*** (DOFS, MODES) : Array containing second set of mode shapes

Author

Scott W. Doebling
Phillip J. Cornwell
Erik G. Straser
Charles R. Farrar
LA-CC-14-046

Modifications

1. March 4, 2009, by Stuart Taylor, for inclusion with the SHM software modal analysis group

Reference

`none`

Sample Data Set

`www.lanl.gov/projects/ei`

2.18 CombinedASPlot_shm.m File Reference

Feature Extraction: Plots color map, borders, and sensors together.

Call Methods

[axesHandle] = combinedASPlot_shm (xMatrix, yMatrix, dataMap2D, border, sensorLayout, axesHandle)

[Axes Handle] = Active Sensing Combined Plot (X Matrix, Y Matrix, Data Map 2D, Border, Sensor Layout, Axes Handle)

Function Description

Plots 2D color map, border line segments, and sensors on a single axis. The 2D colormap is plotted with zbuffer renderer, no lines, face interpolation, equal axis, and a colorbar.

Parameters

- → ***axesHandle*** (handle) : Matlab axes handle
- ← ***xMatrix*** (XINDEX, YINDEX) : Matrix of uniformly spaced X coordinates formed by meshgrid
- ← ***yMatrix*** (XINDEX, YINDEX) : Matrix of uniformly spaced Y coordinates formed by meshgrid
- ← ***dataMap2D*** (XINDEX, YINDEX) : 2D matrix of filled values with NaNs for empty space
- ← ***border*** (ENDPOINTS, SEGMENTS): line segment list, ENDPOINTS is the ordered set [x1, y1, x2, y2]
- ← ***sensorLayout*** (SENSORINFO, SENSOR) : sensor layout IDs, coordinates, and sensitivity direction. Length of SENSORINFO dimension may vary. SENSORINFO is one of the following ordered sets: [sensorID, xCoord] [sensorID, xCoord, yCoord] [sensorID, xCoord, yCoord, zCoord] [sensorID, xCoord, xDir, yDir, zDir] [sensorID, xCoord, yCoord, xDir, yDir, zDir] [sensorID, xCoord, yCoord, zCoord, xDir, yDir, zDir]
- ← ***axesHandle*** (handle) : Matlab axes handle

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.19 cosineKernel_shm.m File Reference

Outlier Detection: Kernel weights for the cosine kernel.

Call Methods

W = cosineKernel_shm (Delta)

Weights = Cosine Kernel Weights (Evaluation Points)

Function Description

Estimate the triangular kernel at points in Delta. In one dimension, this is given as K(u) = (pi/4)cosine((pi/2).u)_+ where u = (x -x_i)/h. In higher dimensions we just take the product of the one dimensional kernel at each coordinate.

Parameters

→ ***W*** (WEIGHTS, 1) : evaluated kernel weights

← ***Delta*** (INSTANCES, FEATURES) : each row is (x-x_i)/bandwidth where the kernel is centered at x, and x_i are the training points.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`http://en.wikipedia.org/wiki/Kernel_(statistics)`

Sample Data Set

`none`

2.20 crestFactor_shm.m File Reference

Feature Extraction: Calculate crest factor feature.

Call Methods

[cf] = crestFactor_shm (X)

[Crest Factor Feature Matrix] = Crest Factor Feature (Conditioned Raw Signal Matrix)

Function Description

Computes the crest factor feature matrix of a raw signal matrix. The crest factor operates on the conditioned raw signal. It is a measure of the peak amplitude of the signal divided by its root mean square. The crest factor has been found to be more sensitive to damage incured in early stages of gear and bearing failure.

Parameters

→ ***cf*** (INSTANCES, FEATURES) : feature vectors of crest factor feature in concatenated format, FEATURES = CHANNELS

← ***X*** (SAMPLES, CHANNELS, INSTANCES) : signal matrix

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.21 cwtScalogram_shm.m File Reference

Feature Extraction: Compute continuous wavelet scalograms.

Call Methods

[scaloMatrix, f, t] = cwtScalogram_shm (X, Fs, fMin, fMax, nScale, waveOrder, waveType, useAnalytic)

[Scalogram Matrix, Frequency Vector, Time Vector] = Continuous Wavelet Scalogram (Signal Matrix, Sampling Frequency, Minimum Frequency, Maximum Frequency, Number of Wavelet Scales, Wavelet Order Parameter, Wavelet Type, Use Complex Wavelet)

Function Description

Computes the continuous wavelet scalograms of a matrix of digital signals using mirrored Morlet wavelets. The continuous wavelet transform function loops through a filter bank of wavelets based of the parameters and are convolved with a signal whose ends are mirrored depending on the wavelet length. The cwt wavelet scalogram has a dyadic time frequency output where the time resolution and frequency resolutions vary over the time frequency domain. High frequency scales have better time resolution and poor frequency resolution where as the low frequency scales have poor time resolution and better frequency resolution. Fs specifies the sampling frequency that the signal was sampled at. fMin and fMax specify the minimum and maximum central frequency of the wavelets used corresponding to its scale. nScale specifies the number of scales to be sampled between fMin and fMax on a logarithmic sampling domain.

Parameters

→ ***scaloMatrix*** (NSCALE, TIME, CHANNELS, INSTANCES) : continuous wavelet transform scalogram matrix

→ ***f*** (NSCALE, 1) : frequency vector corresponding to CWT scalogram

→ ***t*** (TIME, 1) : time vector corresponding to CWT scalogram

← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***Fs*** (scalar) : sampling frequency in Hz

← ***fMin*** (scalar) : minimum scalogram frequency, f(1)

← ***fMax*** (scalar) : maximum scalogram frequency, f(nScale)

← ***nScale*** (integer) : number of scalogram frequency scales

← ***waveOrder*** (integer) : wavelet order, imaginary implies analytic wavelets

← ***waveType*** (string) : wavelet type to be used in computing CWT scalogram 'morlet' : Morlet wavelet 'shannon' : Shannon wavelet 'bspline' : 3rd-order b-spline wavelet

← ***useAnalytic*** (logical) : use complex wavelets

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.22 defopshape_shm.m File Reference

Feature Extraction: Define operating shape.

Call Methods

[residue, freq, damp] = defopshape_shm (G, df, freq)

[Residues, Natural Frequency, Damping Ratio] = Define Operating Shape (FRF array, Delta Frequency, Frequency)

Function Description

Defines operating shape based on user frequency input and estimates damping values using the half power method.

Parameters

→ ***residue*** (DOFS, 1) : vector of complex modal residues

→ ***freq*** (scalar) : modal frequency

→ ***damp*** (scalar) : damping ratio

← ***G*** (FREQUENCY, DOFS, INSTANCES) : matrix of complex frequency response functions

← ***df*** (scalar) : frequency resolution

← ***freq*** (scalar) : selected frequency in Hz

Author

Scott W. Doebling
Phillip J. Cornwell
Erik G. Straser
Charles R. Farrar
LA-CC-14-046

Modifications

1. March 17, 2009, by Stuart Taylor, for inclusion with the SHM software modal analysis group

Reference

`none`

Sample Data Set

`www.lanl.gov/projects/ei`

2.23 demean_shm.m File Reference

Feature Extraction: Removes signal mean.

Call Methods

[Y] = demean_shm (X)

[Demeaned Signal Matrix] = Demean Signal (Signal Matrix)

Function Description

Removes the signal means of a matrix of signals. Also called DC offset removal or constant detrending.

Parameters

$\rightarrow$ ***Y*** ((SAMPLES, CHANNELS, INSTANCES)) : demeaned signals

$\leftarrow$ ***X*** (SAMPLES, CHANNELS, INSTANCES) : signal matrix

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.24 detectorMultiSiteWrapper_shm.m File Reference

Damage detection: Wrappers for use cases.

Call Methods

[results, confidences] = detectorMultiSiteWrapper_shm (X, siteIds, retrainModel, trainFun, modelFilePrefix, oldDataFileName)

[Results, Confidences] = Wrapper for Multi-Site(Data, Site IDs, Retrain Model Flag, Training Function Handle, Model File Prefix, Old Data File Name)

Function Description

This is a wrapper for the multi-site use case where a separate model is trained for each site. The wrapper automatically keeps track of the models and uses the right model to test a new point. The models are kept track of in the SAVEDIR/ directory where the model files are tagged per site id.

Parameters

→ ***results*** (EXAMPLES, SITES) : matrix of complex modal residues

→ ***confidences*** (EXAMPLES, SITES) : matrix of confidences between [0 1] (in the result) per site

← ***X*** (EXAMPLES, FEATURES, SITES): feature matrix of the data to be tested or trained on

← ***siteIds*** (SITES, 1): the site ids corresponding to the third column of X

← ***retrainModel*** (scalar): a flag specifying whether to retrain the model or not. Default is 1 in which case the past data is added to the new data and a new model is relearned for each site

← ***trainFun*** (handle): the training function to be used. Must have the same signature as trainOutlierDetector_shm which is the default

← ***modelFilePrefix*** (string): specifies a prefix to the filename to save the model under. The final filename is simply (modelFilePrefix + 'Site_' + siteId + '.mat'). The default prefix is 'SAVEDIR/UndamagedModel'

← ***oldDataFileName*** (string): the filename to save past data in. The past data is then added to the new data for retraining

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.25 detectOutlier_shm.m File Reference

Outlier Detection: Detect outliers in test features.

Call Methods

[results, confidences, scores, threshold] = detectOutlier_shm (Xtest, modelFile, models, threshold, sensorCodes)

[Results, Confidences, Scores, Threshold] = Detect Outlier (Test Features, Model File Name, Models, Threshold, Sensor Codes)

Function Description

Given a statistical model of the training data, flag the test points as outliers if the score values under the model are under a certain threshold. For instance, if the model is a statistical distribution of the training data, then the scores could be log-likelihood values of the test points under the distribution. In the context of SHM, the outliers would correspond to a damaged structure.

Parameters

→ ***results*** (INSTANCES, 1) : vector of 0, and 1. A '1' in the ith position indicates that the ith observation is flagged as damaged as its score is below a certain threshold

→ ***confidences*** (INSTANCES, 1) : confidence values between 0-1, the confidence value c_i in the ith position indicates that the ith observation scores less than c_i fraction of the data used to train the model

→ ***scores*** (INSTANCES, 1): vector of scores for each instance

→ ***threshold*** (scalar) : threshold used for flagging outliers

← ***Xtest*** (INSTANCES, FEATURES) : test features

← ***modelFile*** (string) : the name of the file containing the density model to use, default value is 'UndamagedModel.mat' in the working directory, modelFile is only used if models is empty

← ***models*** (struct) : density models as generated by trainOutlierDetector functions; if model is given, modelFile is ignored

← ***threshold*** (scalar) : threshold to use for flagging outliers. Overwrite default which is provided within the model.

← ***sensorCodes*** (codes, 1) : vector of sensor codes (integers) associated with each example in Xtest, that is, for each example, a corresponding code for the sensor the example was drawn from. By default we just use -1 for each example.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.26 discRandSeparation_shm.m File Reference

Feature Extraction: Separates signal into periodic and random components.

Call Methods

[Xperiodic, Xrandom] = discRandSeparation_shm (X, nWin, nOvlap, nFFT, nWinDelay)

[Periodic Signal Matrix, Random Signal Matrix] = Discrete/Random Separation (Signal Matrix, Window Sample Length, Window Overlap Length, Number of FFT Bins, Secondary Window Sample Delay Length]

Function Description

Returns a matrix of periodic signals, and a corresponding matrix of remaining random signals. discRandSeparation_shm uses windowed segments of an order tracked signal and its corresponding delayed versions to estimate a transfer function in the same way as a frequency response function would be computed. The theory is that periodic components will remain in both windows and random noise varies. By computing an averaged transfer function it is possible to develop a filter that will separate the periodic signal from the random signal. Periodic components should have a value near one in the transfer function of the filter and random components a value close to zero. This method is an alternative to self adaptive noise cancellation or time synchronous averaging. Gear mesh signals are predominantly periodic while bearing signals tend to be more random. The periodic part is then determined by filtering the signal using the impulse response and the random part by subtracting the periodic signal from the order tracked signal.

Parameters

→ ***Xperiodic*** (SAMPLES, CHANNELS, INSTANCES) : periodic signal matrix

→ ***Xrandom*** (SAMPLES, CHANNELS, INSTANCES) : Random Signal Matrix

← ***X*** (TIME, CHANNELS, INSTANCES) : order tracked signal matrix

← ***nWin*** (integer) : samples per window

← ***nOvlap*** (integer) : number of overlapping samples between windows

← ***nFFT*** (integer) : number of FFT frequency bins

← ***nWinDelay*** (integer): number of samples between delayed windows

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

Xrandom (SAMPLES, CHANNELS, INSTANCES) : Random Signal Matrix

2.27 distance2Index_shm.m File Reference

Active Sensing: Convert distance to index of acquired data.

Call Methods

indices = distance2Index_shm (propDistance, sampleRate, velocity, offset)

[Indices] = Distance To Index (Propagation Distance, Sample Rate, Velocity, Offset)

Function Description

Convert Distance to Index of Aquired Data

Parameters

→ ***indices*** (...) : arbitrarily matrix of indices corresponding to propDistance

← ***propDistance*** (...) : matrix of propagation distances

← ***sampleRate*** (scalar) : sample rate data was aquired at (Hz)

← ***velocity*** (scalar) : propogation velocity

← ***offset*** (scalar) : offset of impulse from start of waveform (in seconds or points)

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.28 dwvd_shm.m File Reference

Feature Extraction: Computes discrete Wigner-Ville distributions.

Call Methods

[dwvdMatrix, f, t] = dwvd_shm (X, nWin, nOvlap, nFFT, Fs)

[Wigner Ville Distribution Matrix, Frequency Vector, Time Vector] = Discrete Wigner-Ville Distribution (Signal Matrix, Samples per Window, Number of Overlapping Window Samples, Number of FFT Bins, Sampling Frequency)

Function Description

Computes the discrete Wigner-Ville matrix from signals. The function computes a Wigner-Viller distribution for each signal in the signal matrix by segmenting the signal into nWin length signals of with nOvlap overlapping samples. From the segments an analytic signal is computed and weighted kernel functions are formed to compute the Wigner-Ville Distribution via DFT. The Wigner Ville Distribution has better time and frequency resolution than the STFT but can suffer from interference negative values which are not actually present in the signal being analyzed.

Parameters

- → ***dwvdMatrix*** (NFFT, TIME, CHANNELS, INSTANCES) : dscrete Wigner-Ville distribution
- → ***f*** (NFFT, 1) : frequency vector corresponding to dwvdMatrix
- → ***t*** (TIME, 1) : time vector corresponding to dwvdMatrix
- ← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data
- ← ***nWin*** (integer) : samples per window length
- ← ***nOvlap*** (integer) : number of overlapping samples between windows
- ← ***nFFT*** (integer) : number of frequency bins in FFT
- ← ***Fs*** (double) : sampling frequency in Hz

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.29 envelope_shm.m File Reference

Feature Extraction: Envelope signals.

Call Methods

[envelopeMatrix] = envelope_shm (X)

[Enveloped Signal Matrix] = Envelope Signal (Signal Matrix)

Function Description

Returns a matrix of enveloped signals. The envelope signal is the magnitude of the analytic signal. It is used in feature NB4M which takes the 4th order statistical moment of the enveloped signal and is normalized by and averaged variance of a baseline enveloped signal squared. Envelope analysis is also a useful tool in bearing analysis as not much can be gained from the frequency domain alone for bearing failures. Oftentimes changes in the frequency domain due to bearing damage are buried by deterministic gear mesh components. As such, frequency bands of vibration signals are bandpass filtered and the envelope of the band pass signal is then used for bearing diagnostics. Frequency bands of interest can be determined using the fast kurtogram method which looks at the spectral kurtosis of different frequency bands.

Parameters

→ ***envelopeMatrix*** (SAMPLES, CHANNELS, INSTANCES) : enveloped signal matrix

← ***X*** (SAMPLES, CHANNELS, INSTANCES) : signal matrix

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.30 epanechnikovKernel_shm.m File Reference

Outlier Detection: Kernel weights for the epanechnikov kernel.

Call Methods

W = epanechnikovKernel_shm (Delta)

Weights = Epanechnikov Kernel Weights (Evaluation Points)

Function Description

Estimate the epanechnikov kernel at points in Delta. In one dimension, this is given as K(u) = 3/4.(1-u^2)_+ where u = (x -x_i)/h. In higher dimensions we just take the product of the one dimensional kernel at each coordinate.

Parameters

→ ***W*** (WEIGHTS, 1) : evaluated kernel weights

← ***Delta*** (INSTANCES, FEATURES) : each row is (x-x_i)/bandwidth where the kernel is centered at x, and x_i are the training points.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`http://en.wikipedia.org/wiki/Kernel_(statistics)`

Sample Data Set

`none`

2.31 estimateGroupVelocity_shm.m File Reference

Active Sensing: Estimate wave propogation speed.

Call Methods

[estimatedVelocity, velocityList] = estimateGroupVelocity_shm (waveform, pairList, sensorLayout, sampleRate, actuationWidth, lineOfSight)

[Estimated Speed, Speed List] = Estimate Wavespeed (Waveform, Pair List, Sensor Layout, Sampling Rate, Actuation Width, Line of Sight)

Function Description

The wavespeed is estimated by calculating the time of flight to the first significant peak in each waveform. The function therefore only calculates the wavespeed of the first arriving mode. If any part of the waveform is negative, its envelope is used instead. The estimated speed is calculated by removing speeds that are more than a standard deviation from the median, and using the mean of the remaining speeds. Also returned are the invidual speeds calculated from each waveform.

Parameters

→ ***estimatedVelocity*** (scalar) : estimated speed (distance/time)

→ ***velocityList*** (1, WAVEFORMS) : calculated speed for each waveform

← ***waveform*** (TIME, ...) : multi-dimensional matrix of waveforms

← ***pairList*** (SENSORIDS, PAIRS) : Matrix of actuator-sensor PAIRS, SENSORIDS is the ordered set of integer IDs [actuatorID, sensorID]

← ***sensorLayout*** (SENSORINFO, SENSORS) : sensor layout IDs and coordinates, SENSORINFO is the ordered set [sensorID, xCoord, yCoord]

← ***sampleRate*** (scalar) : sampling rate (points/time)

← ***actuationWidth*** (integer) : length (in points) of acutation signal

← ***lineOfSight*** (1, PAIRS) : vector of logicals, whether line of sight exists between each pair

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

none

2.32 evalARmodel_shm.m File Reference

Feature Extraction: Evaluate AutoRegressive model (AR).

Call Methods

[RMSresidualsFV, arResiduals, arPrediction] = evalARmodel_shm (Y, arParameters)

[RMS Residuals Feature Vectors, AR residuals, AR Prediction] = Evaluate AR Model Performance (Time Series Data, AR Parameters)

Function Description

This approach consists of using the AR model, with parameters estimated from the baseline condition, to predict the response of data obtained from future time histories. This approach is based on the assumption that damage will introduce either linear deviation from the baseline condition or nonlinear effects in the signal, and therefore, the linear model developed with the baseline data will no longer accurately predict the response of the damaged system.

Parameters

→ ***RMSresidualsFV*** (INSTANCES, FEATURES) : Feature vectors of root mean squared AR residual errors in concatenated format, FEATURES = CHANNELS

→ ***arResiduals*** (TIME, CHANNELS, INSTANCES) : AR residuals

→ ***arPrediction*** (TIME, CHANNELS, INSTANCES) : AR prediction

← ***Y*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***arParameters*** (ARORDER, CHANNELS, INSTANCES) : AR parameters from baseline condition output from arModel_shm

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`www.lanl.gov/projects/ei`

2.33 evalARSVMmodel_shm.m File Reference

Feature Extraction: Evaluate AR Support Vector Machine model (ARSVM).

Call Methods

[RMSresidualsFV, arsvmResiduals, arsvmPrediction] = evalARSVMmodel_shm (Y, arsvmModels, arsvmModelOrder)

[RMS Residuals Feature Vectors, ARSVM residuals, ARSVM Prediction] = Evaluate ARSVM Model Performance (Time Series Data, ARSVM Models, ARSVM Model Order)

Function Description

This approach consists of using the ARSVM model created from the baseline condition, to predict the response of data obtained from future time histories. This approach is based on the assumption that damage will introduce deviations in the response of the damaged system that the model developed with the baseline data will not predict.

Parameters

→ ***RMSresidualsFV*** (INSTANCES, FEATURES) : Feature vectors of root mean squared AR residual errors in concatenated format, FEATURES = CHANNELS

→ ***arsvmResiduals*** (TIME, CHANNELS, INSTANCES) : AR residuals

→ ***arsvmPrediction*** (TIME, CHANNELS, INSTANCES) : AR prediction

← ***Y*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***arsvmModels*** {CHANNELS,1} (struct) : AutoRegressive Support Vector Machine models for each channel, output from arsvmModel_shm

← ***arsvmModelOrder*** (integer) : ARSVM model order

Author

Luke Bornn
Dustin Harvey
LA-CC-14-046

Modifications

1. none

Reference

Bornn, L., Farrar, C.R., Park, G., Farinholt,K., (2009). Structural Health Monitoring With Autoregressive Support Vector Machines. Journal of Vibration And Acoustics, 131, 021004.

Sample Data Set

none

2.34 evalARXmodel_shm.m File Reference

Feature Extraction: Evaluate AR model with eXogenous inputs (ARX).

Call Methods

[RMSresidualsFV, arxResiduals, arxPrediction] = evalARXmodel_shm (Y, arxParameters, arxOrders)

[RMS Residuals Feature Vectors, ARX residuals, ARX Prediction] = Evaluate ARX Model Performance (Time Series Data, ARX Parameters, ARX Model Orders)

Function Description

This approach consists of using the ARX model, with parameters estimated from the baseline condition, to predict the response of data obtained from future time histories. This approach is based on the assumption that damage will introduce either linear deviation from the baseline condition or nonlinear effects in the signal, and therefore, the linear model developed with the baseline data will no longer accurately predict the response of the damaged system.

Parameters

→ ***RMSresidualsFV*** (INSTANCES, FEATURES) : n feature vectors of root mean squared ARX residual errors in concatenated format, FEATURES = OUTPUTCHANNELS

→ ***arxResiduals*** (TIME, CHANNELS, INSTANCES) : ARX residuals

→ ***arxPrediction*** (TIME, CHANNELS, INSTANCES) : ARX prediction

← ***Y*** (TIME, CHANNELS, INSTANCES) : matrix composed of input and output time series (the first column in CHANNELS is the input time series, CHANNELS = OUTPUTCHANNELS + 1)

← ***arxParameters*** (ORDER, OUTPUTCHANNELS, INSTANCES) : ARX parameters output from arxModel_shm, ORDER = a+b

← ***arxOrders*** (1, ORDERS) : vector of integers specifying the orders of the ARX model; ORDERS is the ordered set [a b tau] where a and b are the orders of the outputs and input respectively, and tau is the delay of the input (0 by default)

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Sets

1. www.lanl.gov/projects/ei
2. Set Parameteres

2.35 exciteAndAquire_shm.m File Reference

Data Acquisition: Basic one-shot traditional analog input-output session.

Call Methods

respWaveforms = exciteAndAquire_shm (adaptor, aiID, aiChans, aoID, sampleRate, exciteWaveforms, aiNumSamples, runCount)

Response Waveforms = Excite and Aquire(Adaptor, Analog Input ID, Analog Input Channels, Analog Output ID, Sample Rate, Excitation Waveforms, Number of Samples, Number of Runs)

Function Description

Initializes analog input and analog output devices, sends the excitation waveform to the analog output devices, syncronizes their starts, and returns the acquired data. Supports only one analog output and one analog input device at a time. If the number of columns of exciteWaveforms is less than runCount, then the excitation waveforms are used in order and then repeated until all runs are completed. Analog output is assumed to be on channel 0.

Parameters

→ ***respWaveforms*** (TIME, CHANNELS) : Matrix of acquired time data

← ***adaptor*** (string) : DAQ Toolbox device adaptor

← ***aiID*** (string) : DAQ Toolbox analog input device ID

← ***aiChans*** (1, CHANNELS) : Array of integer analog input channels, example: aiChans=[0 1 23]

← ***aoID*** (string) : DAQ Toolbox analog input device ID

← ***sampleRate*** (scalar) : sampleRate, in Hz, of input and output devices

← ***exciteWaveforms*** (TIME, 1) : excitation waveforms to send to analog output device

← ***aiNumSamples*** (integer) : number of samples to acquire on analog input channels

← ***runCount*** (integer) : number of runs to complete

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

www.ni.com

Sample Data Set

none

2.36 extractSubsets_shm.m File Reference

Active Sensing: Extract data subsets using start indices and window.

Call Methods

dataSubsets = extractSubsets_shm (data, startIndices, subsetWindow)

Data Subsets = Extract Subsets (Data, Start Indices, Subset Window)

Function Description

Takes a 2D matrix "data" of size MxN and windows S subsets along the first dimension to produce a 3D matrix "dataSubsets" of size WxNxS, where W is the length of the window "subsetWindow". The start of the subsets are defined by the matrix "startIndices", which is of size NxS. Subsets whose indices are larger than M are filled with zeros.

Parameters

→ ***dataSubsets*** (TIMESUB, INSTANCES, SUBSETS) : extracted and windowed subsets of data

← ***data*** (TIME, INSTANCES) : full data matrix

← ***startIndices*** (INSTANCES, SUBSETS) : matrix of start indices

← ***subsetWindow*** (TIMESUB, 1) : window to apply to subsets

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.37 fastKurtogram_shm.m File Reference

Feature Extraction: Computes fast spectral kurtograms.

Call Methods

[fastKurtMatrix,levels,f] = fastKurtogram_shm (X,Fs,nKlevels,flagPave3)

[Fast Kurtogram Matrix, Level Vector, Frequency Vector] = Fast Spectral Kurtogram (Signal Matrix, Sampling Frequency, Number of Levels, 1/3 Scale Paving Flag)

Function Description

Returns a matrix of spectral kurtogram images for a signal matrix. It is a measure of the spectral kurtosis relative to frequency band of varying widths and ranges of a signal. Frequency bins with high spectral kurtosis tend to be cause due to transients within the signal that are often caused by damage in machinery. The fast kurtogram is a quick low resolution method of finding frequency bands for demodulation in envelope analysis. By default it uses 1/2 scale paving of the kurtogram but an option exists to also allow for 1.3 scale paving by setting the input variable flagPave3 to true for additional frequency band spectral kurtosis values. From finer frequency bands a larger nKlevels value is required. The frequency vector relative to the fastKurtMatrix are the bin center locations of the image tiles. Frequency Band widths can be calculated using the cursor tool on the imagesc plot of the images.

Parameters

→ ***fastKurtMatrix*** (NKLEVELS,FREQ,CHANNELS,INSTANCES): fast kurtogram data matrix

→ ***levels*** (NKLEVELS,1): levels vector corresponding to fastKurtMatrix

→ ***f*** (FREQ, 1) : frequency vector corresponding to fastKurtMatrix

← ***X*** (SAMPLES, CHANNELS, INSTANCES) : signal matrix

← ***Fs*** (double): sampling frequency in Hz

← ***nKlevels*** (integer) : number of kurtogram levels

← ***flagPave3*** (logical) : 1/3 scale paving flag

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Sets

1. Fs (double): sampling frequency in Hz
2. nKlevels (integer) : number of kurtogram levels
3. flagPave3 (logical) : 1/3 scale paving flag

2.38 fastMetricKernelDensity_shm.m File Reference

Outlier Detection: Fast metric estimation of kernel density.

Call Methods

eval_x = fastMetricKernelDensity_shm (x, Xtrain, h, coverTree, kernelType, fastDist)

Density = Fast Metric Estimation of Kernel Density (Query Point, Training Data, Bandwidth, Tree Data Structure, Kernel Type, Fast Metric Function)

Function Description

Use the input training data and cover tree data structure to perform fast evaluation of the density at x, using a general metric. For a given h, the cover tree is used to quickly identify an r-net in the tree, r= h/4, which is subsequently used to estimate the density at x. The cover tree allows us to avoid computing the distance to each training point in a loop and instead will just compute distances to some of the relevant points.

Parameters

→ ***eval_x*** (scalar) : The evaluated density at x.

← ***x*** (1, FEATURES) : test point at which to evaluate the density

← ***Xtrain*** (INSTANCES, FEATURES) : training features

← ***h*** (scalar) : bandwidth parameter

← ***coverTree*** (struct) : data structure for fast evaluation generated by buildCoverTree_shm

← ***kernelType*** (integer): 1-triangle, 2-epanechnikov, 3-quartic, 4-triweights

← ***fastDist*** (handle) : metric to use, one of the functions in FastMetricKernelEstimation/DistanceMetrics, default is a Euclidean implementation

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`http://en.wikipedia.org/wiki/Kernel_(statistics) Fast, smooth and adaptive regression in metric spaces. Samory Kpotufe. NIPS 2009.`

Sample Data Set

`none`

2.39 fill2DMap_shm.m File Reference

Active Sensing: Fill 2D map from 1D vector according to mask.

Call Methods

dataMap2D = fill2DMap_shm (data1D, mask)

[Data Map 2D] = Fill 2D Map (Data 1D, Mask)

Function Description

Fill 2D Map From 1D Vector According to Mask

Parameters

→ ***dataMap2D*** (XINDEX, YINDEX) : 2D matrix of filled values with NaNs for empty space

← ***data1D*** (DATA, 1) : data vector of same length as number of "true" values in mask matrix

← ***mask*** (XINDEX, YINDEX) : logical mask matrix

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.40 filter_shm.m File Reference

Feature Extraction: Filters signals with FIR filter.

Call Methods

[Y] = filter_shm (X,filterCoef)

[Filtered Signal Matrix] = Filter Signal (Signal Matrix, Filter Coefficients)

Function Description

Filters a signal matrix using fast fft convolution with specified filter impulse response coefficients. The fft of the signal and the filter impulse response are zero padded to be of adequate length and multiplied in the frequency domain. The inverse Fourier transform is then taken. After the signal has been convolved by the filter impulse response, the end is cropped where the impulse response begins to run off the length of the signal such that the signal length of the input signal is equal to the signal length of the output signal.

Parameters

→ ***Y*** (SAMPLES, CHANNELS, INSTANCES) : filtered signal matrix

← ***X*** (SAMPLES, CHANNELS, INSTANCES) : signal matrix

← ***filterCoef*** (NFILTER, 1): filter tap coefficients

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.41 fir1_shm.m File Reference

Feature Extraction: Generate finite impulse response (FIR) filter taps.

Call Methods

[G] = fir1_shm (nFilter, Wn, filterType, win)

[FIR Filter Taps] = Finite Impulse Response Filter Design (Number of Filter Taps, Frequency Range, Filter Type, Window)

Function Description

Returns the filter taps of a specified windowed FIR filter design. Output is not scaled such that the impulse response at the central frequency is 0dB. This only has an appreciable effect for short length filters. nFilter is the length of the filter. Wn is a vector of frequency information for the filter design to be specified in the range (0,0.5) which is the normalized frequency range from 0 to the Nyquist frequency. It is to be a scalar for low pass and high pass filters representing the cutoff frequency. For bandpass and stop band filter design it is to be a vector of two frequency component specifying the range of the pass band or stop band. winParam is a cell containing the window design information. For most window types it only contains the window type. But for Chebyshev or Kaiser filters its second element can be specified to be the dB dropoff or Beta value respectively The FIR filter is designed using an ideal windowed filter response.

Parameters

→ ***G*** (NFILTER, 1) : windowed finite impulse response filter taps

← ***nFilter*** (integer) : number of filter taps, nFilter must be greater than 3 and must be odd for high-pass and band-stop filters

← ***Wn*** (scalar) : normalized frequency filter specifications. In the case of high- and low-pass filter design, this value is a scalar representative of the cutoff frequency. In the case of band-pass or band-stop, Wn is a two element vector [LOWF,HIGHF] consisting of the frequency range to pass or stop.

← ***filterType*** (string) : filterType specifies the type of filter design desired from the following options: 'low' = low-pass filter design (default) 'high' = high pass filter design 'bandpass' = band-pass filter design 'bandstop' = band-stop filter design

← ***win*** (NFILTER, 1) : window column vector, must be same length as number of filter taps

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.42 flexLogicFilter_shm.m File Reference

Active Sensing: Flexible logic filter.

Call Methods

[filteredData] = flexLogicFilter_shm (data, logicFilter, dims)

Filtered Data = Flexible Logic Filter (Data, Logic Filter, Dimensions)

Function Description

Logically filters along specified dimensions, "dims", of "data" according to "logicFilter". Elements in "data" corresponding to the zero elements of "logicFilter" will be rendered equal to zero. The size of the dimensional subset of data specified by "dims" must equal the size of "logicFilter", or, sizeOfData(dims)=size(logicFilter). If no dimensions are given, then the first dimensions of "data" matching the dimension lengths of "logicFilter" are used.

Parameters

→ ***filteredData*** (...) : Filtered data matrix with the same size as "data"

← ***data*** (...) : Original data with arbitrary dimensions

← ***logicFilter*** (...) : Logical matrix with dimensions that are a subset of the dimensions of "data"

← ***dims*** (1, DIMENSIONS) : dimensions of "data" to filter according to "logicFilter". Default is first matching dimensions of data

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.43 fm0_shm.m File Reference

Feature Extraction: Calculate FM0 feature.

Call Methods

[fm0] = fm0_shm (X,fundMeshFreq,trackOrders,nFFT,nBinSearch)

[FM0 Feature Matrix] = FM0 Feature (Conditioned Raw Signal Ensemble, Fundamental Mesh Frequency, Harmonic Orders, FFT Bins, FFT Bin Search Width)

Function Description

Computes the FM0 feature space of a multidimensional time synchronous averaged/angular resampled signal space. The FM0 feature is the magnitude of a signal peak to peak amplitude divided by the sum of the frequency amplitudes corresponding to its fundamental gearmesh frequencies. trackOrders is a vector of the orders associated with the fundamental gear mesh frequency to be summed. As well a bin width parameter is to be assigned to find the maximum amplitude of the nearest neighbor to the specified order in the power spectral density. The FM0 feature has been reported to not be a good detector for minor tooth damage but adequate for larger tooth damage. FM0 can also be used in conjunction with the Hoelder exponent time series which is more sensitive to non-linearities cause by gear tooth impacts which dissipate with tooth wear. When using the Hoelder time series the FM0 feature tends to be more responsive in detecting minor tooth damage.

Parameters

- → ***fm0*** (INSTANCES, FEATURES) : feature vectors of FM0 feature in concatenated format, FEATURES = CHANNELS
- ← ***X*** (SAMPLES, CHANNELS, INSTANCES) : signal matrix
- ← ***fundMeshFreq*** (scalar) : fundamental gear mesh frequency (normalized)
- ← ***trackOrders*** (NORDERS,1) : orders of fundamental mesh freq. to track
- ← ***nFFT*** (integer) : number of bins in FFT power spectral density
- ← ***nBinSearch*** (integer) : FFT bin search parameter

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Sets

1. fundMeshFreq (scalar) : fundamental gear mesh frequency (normalized)
2. trackOrders (NORDERS,1) : orders of fundamental mesh freq. to track
3. nFFT (integer) : number of bins in FFT power spectral density
4. nBinSearch (integer) : FFT bin search parameter

2.44 fm4_shm.m File Reference

Feature Extraction: Calculate FM4 feature.

Call Methods

[fm4] = fm4_shm (D)

[FM4 Feature Matrix] = FM4 Feature (Difference Signal Matrix)

Function Description

Computes the FM4 feature matrix of a difference signal matrix. The FM4 feature is defined as the fourth statistical moment about the mean of a difference signal normalized by its variance squared. FM4 was developed to detect surface damage on a limited number of gear teeth.

Parameters

→ ***fm4*** (INSTANCES, FEATURES) : feature vectors of FM4 feature in concatenated format, FEATURES = CHANNELS

← ***D*** (SAMPLES, CHANNELS, INSTANCES) : difference signal matrix

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.45 frf_shm.m File Reference

Feature Extraction: Computes frequency response function (FRF).

Call Methods

frfdata = frf_shm (data, blockSize, overlap, winhandle, singleSided)

FRF Data = Frequency Response Function (Time Data, Block Size in Points, Percent Overlap, Window Function Handle, Single Sided)

Function Description

Computes frequency response function based on user inputs

Parameters

→ ***frfdata*** (FREQUENCY, OUTPUTCHANNELS, INSTANCES) : multi-dimensional array of complex valued FRF data where OUTPUTCHANNELS = CHANNELS-1

← ***data*** (TIME, CHANNELS, INSTANCES) : raw time histories, input should be first channel

← ***blockSize*** (integer) : number of points on which to perform FFT for each average

← ***overlap*** (scalar) : percent overlap expressed as a decimal

← ***winhandle*** (handle) : Matlab function handle for windowing function (e.g. @hann)

← ***singleSided*** (logical) : whether a single sided FRF should be used; if true, the second half of frfdata will be removed

Author

Stuart Taylor
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`http://institute.lanl.gov/ei/software-and-data/`

2.46 gaussianKernel_shm.m File Reference

Outlier Detection: Kernel weights for the gaussian kernel.

Call Methods

W = gaussianKernel_shm (Delta)

Weights = Gaussian Kernel Weights (Evaluation Points)

Function Description

Estimate the gaussian kernel at points in Delta. In one dimension, this is given as K(u) = (1/sqrt(2 pie))exp(u^2) where u = (x -x_i)/h. In higher dimensions we just take the product of the one dimensional kernel at each coordinate.

Parameters

→ ***W*** (WEIGHTS, 1) : evaluated kernel weights

← ***Delta*** (INSTANCES, FEATURES) : each row is (x-x_i)/bandwidth where the kernel is centered at x, and x_i are the training points.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`http://en.wikipedia.org/wiki/Kernel_(statistics)`

Sample Data Set

`none`

2.47 getElementCentroids_shm.m File Reference

Optimal Sensor Placement: Calculates centroid of each element.

Call Methods

centCoords = getElementCentroids_shm (elements, nodes)

[Centroid Coordinates] = Get Element Centroids (Elements, Nodes)

Function Description

Calculates centroid of each element

Parameters

→ ***centCoords*** (ELEMENTS, COORDINATES) : coordinates of element CGs, COORDINATES is the ordered set [X Y Z]

← ***elements*** (ELEMENTS, NODES) : Indices to nodes constructing each element

← ***nodes*** (NODES, COORDINATES) : Coordinates of element nodes, COORDINATES is the ordered set [X Y Z]

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.48 getGausModSin_shm.m File Reference

Data Acquisition: Generate gaussian modulated sine.

Call Methods

[gausModSinSig, numPoints] = getGausModSin_shm (centerFreq, sampleRate, normBandwidth, numPoints)

[Gaussian Modulated Sine, # of Points] = Gaussian Modulated Sine (Center Frequency, Sample Rate, Normalized Bandwidth, # of Points)

Function Description

Builds a gaussian modulated sin wave with specified center frequency and normalized bandwidth. The standard 3dB bandwidth of the frequency response of the returned wave will be equal to the normalized bandwidth times the center frequency.

Parameters

→ ***gausModSinSig*** (TIME, 1) : gaussian modulated sine waveform data

→ ***numPoints*** (integer) : resulting length of signal if numPoints input was not specified

← ***centerFreq*** (scalar) : frequency of unwindowed sinusoid in Hz

← ***sampleRate*** (scalar) : sampling rate of waveform in Hz

← ***normBandwidth*** (scalar) : 3dB bandwidth normalized to center frequency

← ***numPoints*** (integer) : desired length of signal

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.49 getLineOfSight_shm.m File Reference

Active Sensing: Detects line of sight presence between two points.

Call Methods

isLineOfSight = getLineOfSight_shm (pointA, pointB, border)

[Is Line Of Sight] = Get Line Of Sight(Point A, Point B, Border)

Function Description

Returns a logical array corresponding to the presence of direct line of sight from points A to points B. If any line segment in "border" crosses the direct path between the points, the corresponding logic value will be false.

Parameters

→ ***isLineOfSight*** (1, PAIRS) : logical array of lines of sight presence from points A to points B

← ***pointA*** (COORDINATES, PAIRS) : list of starting points, COORDINATES is the ordered set [XCoord, YCoord]

← ***pointB*** (COORDINATES, PAIRS) : list of ending points, COORDINATES is the ordered set [XCoord, YCoord]

← ***border*** (ENDPOINTS, SEGMENTS): Matrix of line segments that define external or internal plate boundaries, ENDPOINTS is the ordered set [x1, y1, x2, y2]

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.50 getPropDist2Boundary_shm.m File Reference

Active Sensing: Propagation distance from actuator to boundary to sensor.

Call Methods

[propDist, minPropDist] = getPropDist2Boundary_shm (pairList, sensorLayout, border)

[Propagation Distance, Min Propagation Distance] = Get Propagation Dist to Boundary (Pair List, Sensor Layout, Border)

Function Description

For each sensor pair, calculates the total propagation distance from sensor 1, to the nearest point on each boundary, to sensor 2. Also returns each pair's distance to the nearest boundary. Boundary segments do not have to be continuous or joined. If size(pairList,1)==1, then pairList is replicated so that each sensor is treated as its own pair.

Parameters

- → ***propDist*** (SEGMENTS, PAIRS) : total propapagtion distance from each pair to each boundary
- → ***minPropDist*** (1, PAIRS) : total propapagtion distance from each pair to nearest boundary
- ← ***pairList*** (SENSORIDS, PAIRS) : Matrix of actuator-sensor PAIRS, SENSORIDS is the ordered set of integer IDs [actuatorID, sensorID]
- ← ***sensorLayout*** (SENSORINFO, SENSORS) : sensor layout IDs and coordinates, SENSORINFO is the ordered set [sensorID, xCoord, yCoord]
- ← ***border*** (ENDPOINTS, SEGMENTS): Matrix of line segments that define external or internal plate boundaries, ENDPOINTS is the ordered set [x1, y1, x2, y2]

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.51 getSensorLayout_shm.m File Reference

Optimal Sensor Placement: Get sensor layout from DOF indices.

Call Methods

[sensorLayout] = getSensorLayout_shm (sensorIndices, respDOF, geomLayout)

[Sensor Layout] = Get Sensor Layout (Sensor Indices, Response DOF, Geometry Layout)

Function Description

Get sensor layout from DOF indices

Parameters

→ ***sensorLayout*** (SENSORINFO, SENSORS): ID, coordinates, and directional sensitivity of sensors, SENSORNINFO is the ordered set [ID, XCoord, YCoord, ZCoord, XDir, YDir, ZDir]

← ***sensorIndices*** (SENSORS, 1) : Vector of DOF indices of sensors corresponding to DOF definitions in "respDOF"

← ***respDOF*** (DOFS, DOFINFO) : Definitions of the degrees of freedom for the response vector "respVec". DOFINFO is the ordered set [coordID, responseDirection]. The first column contains the IDs of the geometric points corresponding to those listed in "geomLayout". The second column contains the direction of the DOF (1-X, 2-Y, 3-Z, 4-YZ, 5-XZ, 6-XY).

← ***geomLayout*** (POINTINFO, POINTS) : Layout array defining the IDs and coordinates of geometric points, POINTINFO is the ordered set [coordID, X, Y, Z]

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.52 getThresholdChi2_shm.m File Reference

Feature Classification: Calculate threshold for Chi-squared distribution.

Call Methods

threshold = getThresholdChi2_shm (numDOF, confidence)

Threshold = Get Threshold Chi2 (# of DOF, Confidence)

Function Description

Calculate threshold for Chi-squared distribution.

Parameters

→ ***threshold*** (scalar) : One-sided threshold

← ***numDOF*** (scalar) : Number of degrees of freedom

← ***confidence*** (scalar) : Confidence level between 0 and 1

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.53 hoelderExp_shm.m File Reference

Feature Extraction: Calculate Hoelder exponent series.

Call Methods

[hoelderMatrix] = hoelderExp_shm (scaloMatrix, f)

[Hoelder Exponent Matrix] = Hoelder Exponent (Scalogram, Scalogram Frequency Vector)

Function Description

Computes the matrix of Hoelder exponent series from a matrix of time frequency scalograms. If desired it can also operate on any time frequency matrix with a lesser ability to detect non-linearities in the signals. The Hoelder exponent is essentially the change in the slope along the frequency domain of a time frequency matrix as a function of time. When used with scalograms of a signal, if the signal contains non-linearities in it the high frequency content in the time-frequency domain increases and increases the slope. This allows for non-linearities to be more easily detected in vibration signals.

Parameters

→ ***hoelderMatrix*** (TIME, CHANNELS, INSTANCES) : Hoelder exponent series matrix

← ***scaloMatrix*** (NFREQ, TIME, CHANNELS, INSTANCES) : continuous wavelet transform scalogram matrix (or other time-frequency matrix)

← ***f*** (NFREQ, 1) : CWT scalogram frequency vector

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.54 incoherentMatchedFilter_shm.m File Reference

Active Sensing: Incoherent matched filter.

Call Methods

filterResult = incoherentMatchedFilter_shm (waveform, matchedWaveform)

Filter Result = Incoherent Matched Filter (Waveform, Matched Waveform)

Function Description

Applies an incoherent matched filter to the waveform. This is equivalent to taking the magnitude of the dot product of the orignal waveform with the analytic signal of the matched waveform. If the original waveform is not the same length as the matched waveform, then the filter is applied at each point in waveform through convolution. The start and end of the convolution result are truncated so that the filtered waveform is the same size as the original. The analytic signal of a waveform is equal to the waveform plus its hilbert transform times i. Incoherent matched filters are useful in the optimal detection of signals with unknown phase.

Parameters

→ ***filterResult*** (TIME, ...) : multi-dimensional matrix of filter results when length(waveform)==length(matchedWaveform), multi-dimensional matrix of filtered waveforms when length(waveform)> length(matchedWaveform)

← ***waveform*** (TIME1, ...) : multi-dimensional matrix of waveforms, time is along first dimension, all other dimensions are arbitrary

← ***matchedWaveform*** (TIME2, 1) : vector containing matched waveform

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

```
Kay, SM. Fundamentals of Statistical Signal Processing,
Volume 2:  Detection Theory.  Prentice Hall PTR, 1998.
```

Sample Data Set

```
none
```


2.55 kdTree_shm.m File Reference

Data Analysis: Partition the data using kd tree.

Call Methods

[idx, tree] = kdTree_shm (X, k, method)

[partitioning of X, tree structure] = KD Partition Tree (Data Set, Number of Partitions, method)

Function Description

Builds a hierarchical partition of X down to depth = floor(log(k)/log(2)). The final partition is retained and described in idx. The algorithm works by recursively bisecting the data space along coordinates.

Parameters

→ ***idx*** (CLUSTERINDICES, 1) : vector of cluster membership

→ ***tree*** (class): contains the binary rptree as a cell array over the nodes

← ***X*** (INSTANCES, FEATURES) : training features

← ***k*** (scalar) : number of partition cells

← ***method*** (scalar): 1 or 2, method 1 cycles through the coordinates at each level of the tree, method 2 picks the coordinate with highest variance

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.56 kMeans_shm.m File Reference

Data Analysis: Partition the data using kmeans.

Call Methods

idx = kMeans_shm (X, k, niter)

Partitioning of X = Wrapper for Kmeans (Data Set, Number of Partitions, Maximum Iterations)

Function Description

K-means clustering algorithm

Parameters

→ ***idx*** (CLUSTERINDICES, 1) : vector of cluster membership

← ***X*** (INSTANCES, FEATURES) : training features

← ***k*** (scalar) : number of partition cells

← ***niter*** (scalar) : max number of iterations of the algorithm

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.57 kMedians_shm.m File Reference

Data Analysis: Partition the data using kmedians.

Call Methods

[idx, i_c] = kMedians_shm (X, k)

[Partition of X, Cluster Centers]= K Median Clustering (Data Set, Number of Partitions, Maximum Iterations)

Function Description

K-median clustering algorithm

Parameters

→ ***idx*** (CLUSTERINDICES, 1) : vector of cluster membership

→ ***i_c*** (CENTERS, 1) : vector of cluster centers

← ***X*** (INSTANCES, FEATURES) : training features

← ***k*** (scalar) : number of partition cells

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.58 l2Dist_shm.m File Reference

Outlier Detection: Compute L2 (euclidean) distances.

Call Methods

A = l2Dist_shm (M1,M2)

Distances = Compute L2 Distances (First Set of Points, Second set of Points)

Function Description

Compute Euclidean distances from M1 to M2

Parameters

→ ***A*** (M1POINTS, M2POINTS) : L2 distances where M1POINTS and M2POINTS are the number of points in M1 and M2, respectively. A(i, j) = distance (M1(i, :), M2(j, :))

← ***M1*** (INSTANCES, FEATURES): feature vectors

← ***M2*** (INSTANCES, FEATURES): feature vectors

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.59 labelPlot_shm.m File Reference

Plotting: Add a title, axis labels, and legend to plot.

Call Methods

[axesHandle] = labelPlot_shm (axesHandle, titleStr, xlabelStr, ylabelStr, zlabelStr, legendStrs)

[Axes Handle] = Label Plot (Axes Handle, Title, X Axis Label, Y Axis Label, Z Axis Label, Legend)

Function Description

Add a title, axis labels, and legend to plot.

Parameters

→ ***axesHandle*** (handle) : MATLAB axes handle

← ***axesHandle*** (handle) : MATLAB axes handle

← ***titleStr*** (string) : plot title, empty value will be ignored

← ***xlabelStr*** (string) : label for x-axis, empty value will be ignored

← ***ylabelStr*** (string) : label for y-axis, empty value will be ignored

← ***zlabelStr*** (string) : label for z-axis, empty value will be ignored

← ***legendStrs*** ({DATASETS,1} (string) : cell array of strings for legend, empty value will be ignored

Author

Dustin Harvey
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.60 learnFactorAnalysis_shm.m File Reference

Outlier Detection: Learn factor analysis (FA).

Call Methods

[model] = learnFactorAnalysis_shm (X, numFactors, estMethod)

[Model] = Learn Factor Analysis (Training Features, # Comum Factors, Factor Scores, Estimation Method)

Function Description

Compute lambda (loadings) and psi (specific variances) matrices from the factor analysis model using the training data matrix X. The matrices lambda and psi are computed based on the maximum likelihood estimates (MLEs). Matrix X is standardized to zero mean and unit variance before estimating lambda and psi. Note that this does not affect the model fit because MLEs in this common factor analysis model are invariant to scale.

Parameters

→ ***model*** (struct) : parameters of the model, the fields are as follows: .lambda (FEATURES, NUMFACTORS) : matrix of factor loadings

→ ***.psi(FEATURES,FEATURES)*** : specific variances matrix

→ ***.dataMean*** (1, FEATURES) : mean vector of the variables

→ ***.dataStd*** (1, FEATURES) : standard deviation of the variables

→ ***.numFactors*** (integer) : number of common factors

→ ***.estMethod*** (string) : factor scores estimation method

← ***X*** (INSTANCES, FEATURES) : training features where each row (INSTANCES) is a feature vector

← ***numFactors*** (integer) : number of common factors

← ***estMethod*** (string) : factor scores estimation method; either 'thomson','regression', or 'bartlett'

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

Kullaa, J. (2003). Is Temperature Measurement Essential in Structural Health Monitoring? Proceedings of the 4th International Workshop on Structural Health Monitoring 2003: From Diagnostic & Prognostics to Structural Health Monitoring (pp. 717–724). DEStech Publications, Inc.

Sample Data Set

http://institute.lanl.gov/ei/software-and-data/

2.61 learnFastMetricKernelDensity_shm.m File Reference

Outlier Detection: Learn fast non parametric kernel density estimation.

Call Methods

model = learnFastMetricKernelDensity_shm (X, h, kernelType, fastDist, bs_method)

Model = Learn Fast Metric Kernel Density Estimation (Training Features, Bandwidth Matrix, Kernel Type, Fast Metric Function, Bandwidth Selection Method)

Function Description

Put together a model object for kernel density estimation using a cover tree for fast estimation over a general metric space.

Parameters

→ ***model*** (struct) : includes training data, bandwidth, kernel, and metric

← ***X*** (INSTANCES, FEATURES) : training features where each row (INSTANCES) is a feature vector

← ***h*** (scalar) : bandwidth parameter

← ***kernelType*** (integer) : 1-triangle, 2-epanechnikov, 3-quartic, 4-triweights.

← ***fastDist*** (handle) : metric to use, one of the functions in FastMetricKernelEstimation/DistanceMetrics, default is a Euclidean implementation

← ***bs_method*** (scalar): 1 or 2, bandwith selection method to use. Method 1 performs cross validation and sets the bandwidth as (b ∗ average diameter) in each coordinate, where the fraction b is selected through cross validation. Method 2 just sets b to (1/n)^(1/(D+2)) to match the minimax l2 rate in each coordinate. Method 1 is preferred although it takes more time to compute. A bandwidth selection method is used if and only if bandwidth matrix is empty ([]).

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.62 learnGMM_shm.m File Reference

Outlier Detection: Learn gaussian mixture model.

Call Methods

model = learnGMM_shm (X, idx)

[Model] = Learn Gaussian Mixture Model (Training Features, Clustering Indices)

Function Description

Learn a mixture of gaussians over the given partition idx of the points in X. If idx is not passed in, then a single Gaussian is learned.

Parameters

→ ***model*** (struct): parameters of the model, the fields are as follows: .p (1, RATES) : mixture rates

→ ***.mu*** (COORDINATES, MEANS) : means of the gaussians

→ ***.s2*** (COORDINATES, COORDINATES, COORDINATES) : covariance matrices of the gaussians

← ***X*** (INSTANCES, FEATURES) : training features where each row (INSTANCES) is a feature vector

← ***idx(CLUSTERINDICES,1)*** : vector of cluster membership

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.63 learnGMMSemiParametricModel_shm.m File Reference

Outlier Detection: Learn GMM semi-parametric density model.

Call Methods

dModel = learnGMMSemiParametricModel_shm(X, partitionFun, k)

Density Model = Learn GMM semi-Parametric Density Model (Training Features, Partitioning Function, Partition Size)

Function Description

Serves to assemble a semi-parametric model learning function, using the partitioning function provided. The partitioning function is used to partition the data, and a Gaussian Mixture Model (GMM) is learned over the partition.

Parameters

- → ***dModel*** (struct) : parameters of the model
- ← ***X*** (INSTANCES, FEATURES) : training features where each row (INSTANCES) is a feature vector
- ← ***partitionFun*** (struct) : function handle taking parameters (X, k), returning idx, see e.g. kMeans_shm
- ← ***k*** (integer): number of partition cells

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.64 learnKernelDensity_shm.m File Reference

Outlier Detection: Learn non parametric kernel density estimation.

Call Methods

model = learnKernelDensity_shm (X, H, kernelFun, bs_method)

Model = Learn Kernel Density Estimation (Training Features, Bandwidth Matrix, Kernel Function, Bandwidth Selection Method)

Function Description

Construct a model object for kernel density estimation.

Parameters

- → ***model*** (struct): includes training data, bandwidth and kernel function
- ← ***X*** (INSTANCES, FEATURES) : training features where each row (INSTANCES) is a feature vector
- ← ***H*** (FEATURES, FEATURES): matrix of bandwidth, restricted to diagonal bandwidth m matrices.
- ← ***kernelFun*** (handle): kernel to use (function handle), options can be found in NonParametricDetectors/Kernels.
- ← ***bs_method*** (scalar): 1 or 2, bandwith selection method to use. Method 1 performs cross validation and set the bandwidth as b∗std in each coordinate, where the fraction b is selected through cross validation. Method 2 just sets b to (1/n)^(1/3) to match the minimax l2 rate in each coordinate. Method 1 is preferred although it takes more time to compute. A bandwidth selection method is used if and only if bandwidth matrix is empty ([]).

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.65 learnMahalanobis_shm.m File Reference

Outlier Detection: Learn mahalanobis squared distance.

Call Methods

[model] = learnMahalanobis_shm (X)

[Model] = Learn Mahalanobis (Training Features)

Function Description

Compute the mean vector and the convariance matrix of the feature vectors in the matrix X.

Parameters

→ ***model*** (struct) : parameters of the model, the fields are as follows: .dataMean (1, FEATURES) : mean vector of the features

→ ***.dataCov*** (FEATURES, FEATURES) : covariance matrix of the features

← ***X*** (INSTANCES, FEATURES) : training features where each row (INSTANCES) is a feature vector

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

Worden, K., & Manson, G. (2000). Damage Detection using Outlier Analysis. Journal of Sound and Vibration , 229 (3), 647-667.

Sample Data Set

http://institute.lanl.gov/ei/software-and-data/

2.66 learnNLPCA_shm.m File Reference

Outlier Detection: Learn nonlinear principal component analysis (NLPCA).

Call Methods

[model] = learnNLPCA_shm (X, b, M1, M2, paramIte, paramPer)

[Model] = Learn NLPCA (Training Features, # Nodes Bottleneck Layer, # Nodes Mapping Layer, # Nodes de-Mapping Layer, # Iterations, Performance)

Function Description

Learn the correlation among the features to reveal the unobserved variables, which drives any changes in the data. This correlation is represented at the bottleneck layer, where the number of nodes depends on the number of unobserved variables expected. (Currently, M1 is taken equal to M2.)

Parameters

→ ***model*** (struct) : parameters of the model, the fields are as follows: .net (object) : neural network object defining the basic features of the trained network

→ ***.b*** (integer) : number of nodes in the bottleneck layer

→ ***.M1*** (integer) : number of nodes in the mapping layer

→ ***.M2*** (integer) : number of nodes in the de-mapping layer (M1=M2 by default)

→ ***.E*** (scalar) : mean square error of the training error

← ***X*** (INSTANCES, FEATURES) : training features where each row (INSTANCES) is a feature vector

← ***b*** (integer) : number of nodes in the bottleneck layer

← ***M1*** (integer) : number of nodes in the mapping layer

← ***M2*** (integer) : number of nodes in the de-mapping layer (M1=M2 by default)

← ***paramIte*** (integer) : number of iterations

← ***paramPer*** (scalar) : performace

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

References

1. Sohn, H., Worden, K., & Farrar, C. R. (2002). Statistical Damage Classification under Changing Environmental and Operational Conditions. Journal of Intelligent Material Systems and Structures , 13 (9), 561-574.

2. Kramer, M. A. (1991). Nonlinear Principal Component Analysis using Autoassociative Neural Networks. AIChE Journal , 37 (2), 233-243.

Sample Data Set

http://institute.lanl.gov/ei/software-and-data/

2.67 learnPCA_shm.m File Reference

Outlier Detection: Learn principal component analysis (PCA).

Call Methods

[model] = learnPCA_shm (X, perVar, stand)

[Model] = Learn Principal Component Analysis (Training Features, Percentage of Variance, Standardization)

Function Description

This function returns the principal components of the data set X required by the percentage of variance.

Parameters

→ ***model*** (struct) : parameters of the model, the fields are as follows: .loadings (FEATURES, PRINCIPALCOMP) : loadings matrix where columns are the number of principal components

→ ***.dataParam*** (2, FEATURES) : matrix composed of (1) means and (2) standard deviations

← ***X*** (INSTANCES, FEATURES) : training features where each row (ISNTANCES) is a m-dimensional feature vector

← ***perVar*** (scalar) : minimal percentage of the variance to explain the variability in the matrix X

← ***stand(scalar)*** : 0 - standardization or 1 - non standardization

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

```
Figueiredo, E., Park, G., Figueiras, J., Farrar, C.,
and Worden, K. (2009).  Structural Health Monitoring
Algorithm Comparisons Using Standard Data Sets, Los
Alamos National Laboratory Report:  LA-14393.
```

Sample Data Set

```
www.lanl.gov/projects/ei
```


2.68 learnSVD_shm.m File Reference

Outlier Detection: Learn singular value decomposition (SVD).

Call Methods

model = learnSVD_shm (X, paramStand)

[Model] = Learn Singular Value Decomposition (Training Features, Standardization)

Function Description

Compute the singular values of the state matrix X. In the context of damage detection, if the rank of M is r and if a vector from Y comes from a damaged condition, the rank of Mc will be equal to r+1. If the rank of M and Mc are the same, this algorithm assumes that when Mc is composed by a damaged condition, the damage detection is based on changes in magnitude of the singular values.

Parameters

→ ***model*** (struct) : parameters of the model, the fields are as follows: .X (INSTANCES, FEATURES) : training data matrix where each row (INSTANCES) is a feature vector

→ ***.S*** (FEATURES, 1) : singular values of X

→ ***.dataMean*** (1, FEATURES) : Column vector composed of the mean of the variables

→ ***.dataStd*** (1, FEATURES) : Column vector composed of the standard deviations of the variables

← ***X*** (INSTANCES, FEATURES) : training features where each row (INSTANCES) is a feature vector

← ***paramStand(logical)*** : whether standardization should be used

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

References

1. Ruotolo, R., & Surage, C. (1999). Using SVD to Detect Damage in Structures with Different Operational Conditions. Journal of Sound and Vibration , 226 (3), 425-439.

2. Figueiredo, E., Park, G., Figueiras, J., Farrar, C., and Worden, K. (2009). Structural Health Monitoring Algorithm Comparisons Using Standard Data Sets, Los Alamos National Laboratory Report: LA-14393.

Sample Data Set

www.lanl.gov/projects/ei

2.69 lkDist_shm.m File Reference

Outlier Detection: Compute Lk distances.

Call Methods

A = lkDist_shm (M1, M2, k)

Distances = Compute Lk Distances (First Set of Points, Second set of Points, Norm)

Function Description

Compute Lk norm distances from M1 to M2

Parameters

- → ***A*** (M1POINTS, M2POINTS) : Lk norm distances where M1POINTS and M2POINTS are the number of points in M1 and M2, respectively; A(i, j) = distance (M1(i, :), M2(j, :)).
- ← ***M1*** (INSTANCES, FEATURES): feature vectors
- ← ***M2*** (INSTANCES, FEATURES): feature vectors
- ← ***k*** (scalar) : choice of Lk norm to use

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.70 lpcSpectrogram_shm.m File Reference

Feature Extraction: Computes spectrogram using LPC coefficients.

Call Methods

[lpcSpecMatrix, f, t] = lpcSpectrogram_shm (X, modelOrder, nWin, nOvlap, nFFT, Fs)

[LPC Spectrogram Matrix, Frequency Vector, Time Vector] = Linear Predictive Spectrogram (Signal Matrix, Linear Predictive Coefficient Model Order, Samples per Window, Number of Overlapping Window Samples, Number of FFT Bins, Sampling Frequency)

Function Description

Computes a matrix stack of the time frequency domain for a matrix of digital signals using a linear predictive coefficient spectrogram algorithm.

Parameters

→ ***lpcSpecMatrix*** (NFREQ, TIME, CHANNELS, INSTANCES) : linear predictive coefficient spectrogram matrix

→ ***f*** (NFREQ, 1) : frequency vector corresponding to lpcSpecMatrix

→ ***t*** (TIME, 1) : time vector borresponding to lpcSpecMatrix

← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***modelOrder*** (integer) : linear predictive model order

← ***nWin*** (integer) : number of samples per window

← ***nOvlap*** (integer) : number of overlapping window samples

← ***nFFT*** (integer) : number of FFT frequency bins

← ***Fs*** (double) : sampling frequency

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.71 m6a_shm.m File Reference

Feature Extraction: Calculate M6A feature.

Call Methods

[m6a] = m6a_shm (D)

[M6A Feature Matrix] = M6A Feature (Difference Signal Matrix)

Function Description

Computes the M6A feature space of a multidimensional difference signal space. The M6A feature is defined as the sixth statistical moment about the mean of a difference signal normalized by its variance to the 3rd power. M6A is similar to the FM4 feature but is reported to be more sensitive to damage.

Parameters

→ ***m6a*** (INSTANCES, FEATURES) : feature vectors of M6A feature in concatenated format, FEATURES = CHANNELS

← ***D*** (SAMPLES, CHANNELS, INSTANCES) : difference signal matrix

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.72 m8a_shm.m File Reference

Feature Extraction: Calculate M8A feature.

Call Methods

[m8a] = m8a_shm (D)

[M8A Feature MAtrix] = M8A Feature (Difference Signal Space)

Function Description

Computes the M8A feature space of a multidimensional difference signal space. The M8A feature is defined as the eighth statistical moment about the mean of a difference signal normalized by its variance to the 4th power. M8A is similar to the FM4 feature but is reported to be more sensitive to damage.

Parameters

→ ***m8a*** (INSTANCES, FEATURES) : feature vectors of M8A feature in concatenated format, FEATURES = CHANNELS

← ***D*** (SAMPLES, CHANNELS, INSTANCES) : difference signal matrix

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.73 mac_shm.m File Reference

Feature Extraction: Computes modal assurance criteria (MAC).

Call Methods

mc = mac_shm (t1, t2, Q)

MAC matrix = Modal Assurance Criteria (Mode Shape Array 1, Mode Shape Array 2, Normalization Matrix)

Function Description

Computes modal assurance criteria

Parameters

→ ***mc*** (MODES, MODES) : Modal Assurance Criterion array

← ***t1*** (DOFS, MODES) : Array containing first set of mode shapes

← ***t2*** (DOFS, MODES) : Array containing second set of mode shapes

← ***Q*** (DOFS, DOFS) : Matrix (often a mass matrix) to normalize the MAC

Author

Scott W. Doebling
Phillip J. Cornwell
Erik G. Straser
Charles R. Farrar
LA-CC-14-046

Modifications

1. March 4, 2009, by Stuart Taylor, for inclusion with the SHM software modal analysis group

Reference

`none`

Sample Data Set

`www.lanl.gov/projects/ei`

2.74 metricKernel_shm.m File Reference

Outlier Detection: Metric kernel function.

Call Methods

weights = metricKernel_shm (distances, kerneltype)

Weights = Metric Kernel Function (Distances, Kernel Type)

Function Description

Defines many kernel functions over general metrics (see kerneltype). Distances from training points to a query point x are passed in.

Parameters

→ ***weights*** (WEIGHTS, 1) : kernel weights computed.

← ***distances*** (DISTANCES, 1) : disntances from training points to a query point x. These distances are defined using any metric, e.g. l_1, l_2.

← ***kerneltype*** (scalar) : 1, 2, 3, 4 for types of kernel. Let d_i be ith distance: 1 - triangle kernel - w_i = (1-d_i)_+ 2 - epanechnikov kernel - w_i = (1-d_i^2)_+ 3 - quartic kernel - w_i = ((1-d_i^2)_+)^2 4 - triweight kernel - w_i = ((1-d_i^2)_+)^3

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.75 na4m_shm.m File Reference

Feature Extraction: Calculate NA4M feature.

Call Methods

[na4m, m2] = na4m_shm (R, m2)

[NA4M Feature Matrix, Average Baseline Signal Variance] = NA4M Feature (Residual Signal Matrix, Average Baseline Signal Variance)

Function Description

Computes the NA4M damage feature for an matrix of residual signals. The NA4M damage feature is the fourth order statistical moment normalized by an average baseline signal variance to the second power. The residual signal is the raw signal with the gear mesh and shaft frequencies removed from the signal.

Parameters

→ ***na4m*** (INSTANCES, FEATURES) : feature vectors of NA4M feature in concatenated format, FEATURES = CHANNELS

→ ***m2*** (1, CHANNELS) : average baseline signal variance

← ***R*** (SAMPLES, CHANNELS, INSTANCES) : residual signal matrix

← ***m2*** (1, CHANNELS) : average baseline signal variance

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

```
m2 (1, CHANNELS) : average baseline signal variance
```


2.76 nb4m_shm.m File Reference

Feature Extraction: Calculate NB4M feature.

Call Methods

[nb4m, m2] = nb4m_shm (X, m2)

[NB4M Feature Vectors] = NB4M Feature (Band Passed Mesh Signal, Average Envelope Variance)

Function Description

Computes the NB4M feature matrix for a 3D matrix of band pass mesh signals. Traditionally NB4 is calculated continuously over the coarse of a machines life and M2 is a runtime average of the envelope signal variance. For the purposes of this module the numerator of the feature (m2) is to be an averaged envelope signal variance from a baseline state, similar to the NA4$*$ damage feature, as the data supplied is a binary data set with only a good condition and bad condition and not a run to failure data set. NB4 was developed to detect damage on gear teeth. The NB4M feature provided in this function is the 4th order statistical moment of the band passed mesh signal normalized by an average variance of a baseline gearbox signal squared.

Parameters

→ ***nb4m*** (INSTANCES, FEATURES) : feature vectors of NB4M feature in concatenated format, FEATURES = CHANNELS

→ ***m2*** (1, CHANNELS) : average envelope signal variance

← ***X*** (SAMPLES, CHANNELS, INSTANCES) : band pass mesh signal matrix

← ***m2*** (1, CHANNELS) : average envelope signal variance from a baseline gearbox state

← ***See*** also: analyticSignal_shm, envelope_shm

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Sets

1. `m2 (1, CHANNELS) : average envelope signal variance from a baseline gearbox state`
2. `See also: analyticSignal_shm, envelope_shm`

2.77 NI_FGEN_InitConfig_shm.m File Reference

Data Acquisition: Initialize and configure NI-FGEN session.

Call Methods

niFgenHandle = NI_FGEN_InitConfig_shm (niFgenOptions)

[NI_FGEN Handle] = NI-FGEN Init & Configure (NI_FGEN Options)

Function Description

Initializes high speed function generation session to be used with NI high speed boards with Fgen drivers.

Parameters

→ ***niFgenHandle*** (handle) : Handle of NI_FGEN session

← ***niFgenOptions*** (struct) : NI_FGEN options structure constructed using NI-FGEN_SetOptions_shm.

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`www.ni.com`

Sample Data Set

`none`

2.78 NI_FGEN_PrepWave_shm.m File Reference

Data Acquisition: Prepare waveform for NI-FGEN session.

Call Methods

NI_FGEN_PrepWave_shm (niFgenHandle, waveform, niFgenOptions)

[] = NI-FGEN Prepare Waveform (NI_FGEN Handle, Waveform, NI-FGEN Options)

Function Description

Sets up the waveform waveformDataArray for output by analog output session niFgen-Handle. Waveform must be between -1 and 1.

Parameters

→ ***none***

← ***niFgenHandle*** (handle) : Handle of NI_FGEN session

← ***waveform*** (TIME, 1) : waveform to sent to output device

← ***niFgenOptions*** (struct) : NI_FGEN options structure constructed using NI_-FGEN_SetOptions_shm.

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`www.ni.com`

Sample Data Set

`none`

2.79 NI_FGEN_SetOptions_shm.m File Reference

Data Acquisition: Set options for NI-FGEN session.

Call Methods

niFgenOptions = NI_FGEN_SetOptions_shm (deviceID, channelNum, sampleRate, gain)

NI_FGEN Options = NI-FGEN Set Options (Device ID, Channel #, Sample Rate, Gain)

Function Description

Set the basic options for analog output using a National Instruments high speed function generator and the FGEN driver set.

Parameters

→ ***niFgenOptions*** (struct) : NI-FGEN options structure

← ***deviceID*** (string) : Name assigned to device in NI MAX

← ***channelNum*** (integer) : Output channel number

← ***sampleRate*** (integer) : Sampling rate in Hz

← ***gain*** (scalar) : Gain on output waveform (gain < 6)

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`www.ni.com`

Sample Data Set

`none`

2.80 NI_multiplexSession_shm.m File Reference

Data Acquisition: Run NI active sensing multiplex session.

Call Methods

[waveforms] = NI_multiplexSession_shm (niSyncHandle, niSwitchHandle, niScopeHandle, niScopeOptions, pairList, numberOfAverages, delay, outputChSep)

[Waveforms] = Run NI Multiplex Session (Sync Handle, NISwitch Handle, NIScope Handle, NIScope Options, Sensor Pairs, # of Averages, Delay, Output Channel Separation)

Function Description

Aquires waveforms for all combinations of channel pairs. For each pair (ch_a, ch_b), ch_a will be routed to 'com0' and ch_b+outputChSep will be routed to 'com1'. This corresponds to a 2-bank multiplexor configuration where the first bank corresponds to actuation outputs and the second bank corresponds to sensing inputs. Assumes the ai session "ai" is included in the sync session defined by "niSyncHandle". Aquisitions for each channel pair are averaged.

Parameters

→ ***waveforms*** (TIME, PAIRS) : Matrix of acquired time data

← ***niSyncHandle*** (handle) : Handle of session intialized by syncPrep

← ***niSwitchHandle*** (handle) : Handle of initialized switch

← ***niScopeHandle*** (handle) : Handle of analog input object

← ***niScopeOptions*** (struct) : analog input options structure

← ***pairList*** (SENSORIDS, PAIRS) : Matrix of actuator-sensor PAIRS, SENSORIDS is the ordered set of integer IDs [actuatorID, sensorID]

← ***numberOfAverages*** (integer) : number of averages to perform for each channel pair

← ***delay*** (scalar) : delay in milliseconds between acquisitions

← ***outputChSep*** (scalar) : channel sepparation between input and output lines (example: 16 or 32)

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

www.ni.com

Sample Data Set

none

2.81 NI_SCOPE_FetchWaves_shm.m File Reference

Data Acquisition: National Instruments NI-SCOPE fetch waves.

Call Methods

waveforms = NI_SCOPE_FetchWaves_shm (niScopeHandle, niScopeOptions)

Waveforms = NI-SCOPE Fetch Waves (NI_SCOPE Handle, NI_SCOPE Options)

Function Description

Returns data acquired by session ai. This function assumes that the entire waveform has already been collected and can only be called after the DAQ session has been started (using syncStart).

Parameters

→ ***waveforms*** (TIME, CHANNELS) : matrix of waveforms

← ***niScopeHandle*** (handle) : Handle of analog input object

← ***niScopeOptions*** (struct) : analog input options structure constructed using NI_SCOPE_SetOptions

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`www.ni.com`

Sample Data Set

`none`

2.82 NI_SCOPE_InitConfig_shm.m File Reference

Data Aquisition: Initialize and configure NI-SCOPE session.

Call Methods

[niScopeHandle, niScopeOptions] = NI_SCOPE_InitConfig_shm (niScopeOptions)

[NI_SCOPE Handle, NI_SCOPE Options] = NI-SCOPE Init & Configure (NI_SCOPE Options)

Function Description

Initializes high speed data acquisition session to be used with NI high speed boards with niScope drivers.

Parameters

→ ***niScopeHandle*** (handle) : Handle of analog input object

→ ***niScopeOptions*** (struct) : updated analog input options structure

← ***niScopeOptions*** (struct) : analog input options structure constructed using NI_SCOPE_SetOptions.

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`www.ni.com`

Sample Data Set

`none`

2.83 NI_SCOPE_SetOptions_shm.m File Reference

Data Acquisition: Set options for NI-SCOPE session.

Call Methods

niScopeOptions = NI_SCOPE_SetOptions_shm (deviceID, channelList, minSampleRate, minNumPts)

NI_SCOPE Options = NI-SCOPE Set Options (Device ID, Channel List, Min Sample Rate, Min Number of Points)

Function Description

Set the basic options for using NI_SCOPE device.

Parameters

→ ***niScopeOptions*** (struct) : Structure containing all the option parameters

← ***deviceID*** (string) : Name assigned to device in NI MAX

← ***channelList*** (1, CHANNELS) : Acquisition Channel(s), integers

← ***minSampleRate*** (scalar) : Sampling Rate

← ***minNumPts*** (integer) : Number of Points to Aquire

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`www.ni.com`

Sample Data Set

`none`

2.84 NI_SWITCH_Connect_shm.m File Reference

Data Acquisition: Connect NI-Switch channels.

Call Methods

NI_SWITCH_Connect_shm (niSwitchHandle, chan1, chan2, resetConnections, wait4Switch)

[]= NI-SWITCH Connect(NI_SWITCH Handle, Channel 1, Channel 2, Reset Connections, Wait For Switch)

Function Description

Routes channels chan1 to chan2 within the NI switch referenced by switchHandle. If chan1 and chan2 are cell arrays, then the first entry of chan1 will be connected to the first entry of chan2 and so on. This function is intended to be used with a NI switch programed for 2-bank switching. An example set of channels would be chan1={'com0','com1'}, chan2={'ch3,'ch25'}

Parameters

→ ***none***

← ***niSwitchHandle*** (handle) : Handle of initialized switch

← ***chan1*** {1, CHANNELS} (string) : Cell array of strings identifying channels

← ***chan2*** {1, CHANNELS} (string): Cell array of strings identifying channels

← ***resetConnections*** (logical) : whether to reset all connections first

← ***wait4Switch*** (logical) : whether to wait for the switch to debounce before returning

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`www.ni.com`

Sample Data Set

`none`

2.85 NI_SWITCH_Init_shm.m File Reference

Data Acquisition: NI-Switch initialization.

Call Methods

niSwitchHandle = NI_SWITCH_Init_shm (deviceID)

NISwitch Handle = NI-SWITCH Initialize (Device ID)

Function Description

Initializes NI Switch for channel multiplexing

Parameters

→ ***niSwitchHandle*** (handle) : Handle of switch object

← ***deviceID*** (string) : Name assigned to switch device in Automation Explorer

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.86 NI_TCLK_SyncPrep_shm.m File Reference

Data Acquisition: Prepare NI-TCLK syncronization.

Call Methods

niTclkSyncHandle = NI_TCLK_SyncPrep_shm (varargin)

Sync Handle = NI-TCLK Sync Preparation (session1)

Function Description

Builds syncronization session consisting of compatible data acquisition sessions using NI_TCLK library. Triggering of the session results in simultaneous triggering of the session members. Triggering is performed by the sepparate function **NI_TCLK_Trigger_shm.m** (p. 116).

Parameters

→ ***niTclkSyncHandle*** (handle) : syncronized session handle

← ***session1*** (handle) : data acquisition handle

← ***session2*** (handle) : data acquisition handle

← ***session3*** (handle) : data acquisition handle

← ***session4*** (handle) : data acquisition handle

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.87 NI_TCLK_Trigger_shm.m File Reference

Data Acquisition: Trigger NI-TCLK session.

Call Methods

NI_TCLK_Trigger_shm (syncHandle)

[] = NI-TCLK Trigger (Sync Handle)

Function Description

Trigger NI_TCLK Session

Parameters

→ ***none***

← ***syncHandle*** (handle) : syncronization session handle

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.88 nodeElementPlot_shm.m File Reference

Optimal Sensor Placement: Plot structure nodes and elements.

Call Methods

axesHandle = nodeElementPlot_shm (nodeLayout, elements, nodeScale, axesHandle)

Axes Handle = Node-Element Plot (Nodes, Elements, Response, Response Direction, Response Scale, Axes Handle)

Function Description

Plots the nodes and elements of a structure. Supports 2-, 3-, and 4-node 2D elements and 8-node 3D elements. Faces of the elements are colored according to interpolation of nodeScale using the default colormap. Faces are semi-transparent with alpha of 0.3.

Parameters

→ ***axesHandle*** (handle) : handle of axes that structure was plotted on

← ***nodeLayout*** (NODEINFO, NODES, INSTANCES): ID and coordinates of each node, NODEINFO is the ordered set [ID XCoord YCoord ZCoord]

← ***elements*** (NODES, ELEMENTS) : Indices of the nodes that construct each element

← ***nodeScale*** (NODES, INSTANCES) : scales for determining element colors through interpolation(optional)

← ***axesHandle*** (handle) : handle of axes that structure will be plotted on

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.89 normmodes_shm.m File Reference

Feature Extraction: Normalizes mode shapes using specified method.

Call Methods

modes_norm = normmodes_shm (modes, mpf, dppair, method)

Normalized Mode Shapes = Normalize Mode Shapes (Mode Shape Array, Modal Participation Factors, Drive Point Pair, Normalization Method)

Function Description

Normalizes the modes shapes according to user-input method

Parameters

→ ***modes_norm*** (DOFS, MODES) : Normalized mode shapes

← ***modes*** (DOFS, MODES) : Array of mode shapes

← ***mpf*** (1, MODES) : complex modal participation factors

← ***dppair*** (1, SENSORPAIR) : drive point pair, SENSORPAIR is the ordered set of integers [sensor1 sensor2]

← ***method*** (string) : user input for normalization method, either 'Drive Point' or 'Unit Mass'

Author

Scott W. Doebling
Phillip J. Cornwell
Erik G. Straser
Charles R. Farrar
LA-CC-14-046

Modifications

1. March 4, 2009, by Stuart Taylor, for inclusion with the SHM software modal analysis group

Reference

`none`

Sample Data Set

`www.lanl.gov/projects/ei`

2.90 OSP_FisherInfoEIV_shm.m File Reference

Optimal Sensor Placement: Fisher information matrix EIV method for OSP.

Call Methods

[opList, detQ] = OSP_FisherInfoEIV_shm (numSensors, modeShapes, covMatrix)

[Optimal List, Fisher Determinant] = OSP Fisher Information EIV (# of Sensors, Mode Shapes, Covariance Matrix)

Function Description

Determines optimal sensor placement using the Fisher Information Matrix method. The optimal sensor locations are chosen to maximize the determinant of the Fisher Information Matrix of the mode shape matrix masked to those locations. The optimal locations are chosen by removing, one at a time, the sensor locations with the lowest Effective Independance Value. This usually results in a near maximum for the determinant.

Parameters

- → ***opList*** (SENSORS) : Indices of 'modeShapes' corresponding to optimal locations
- → ***detQ*** (scalar) : Determinant of the Fisher information matrix reduced to "numSensors" sensors
- ← ***numSensors*** (integer) : The number of sensors to place
- ← ***modeShapes*** (DOFS, MODES) : Mode shapes of interest
- ← ***covMatrix*** (DOFS, DOFS) : DOF measurement covariance matrix. A vector may be inputed to represent the elements of a diagonal covariance matrix

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`DC Kammer, and RD Brillhart, Optimal sensor placement for modal identification using system-realization methods, Journal of Guidance, Control and Dynamics 19(1996), pp. 729-731.`

Sample Data Set

`none`

2.91 OSP_MaxNorm_shm.m File Reference

Optimal Sensor Placement: Maximum norm method for OSP.

Call Methods

[opList] = OSP_MaxNorm_shm (numSensors, modeShapes, weights, duelingDistance, respDOF, geomLayout)

[Optimal List] = OSP Using Maximum Norm (Number of Sensors, Mode Shapes, Weights, Dueling Distance, Response DOF, Geometry Layout)

Function Description

Determines optimal sensor placement using maximum norm method. The optimal sensor locations are chosen as the maxima of the squared sum of the weighted mode shapes. Since this often produces closely spaced sensor locations, dueling may be enabled. Dueling involves taking neighboring sensor locations, determining which has a higher norm value, and permantely removing (killing) the loser location. The location with the next highest norm then takes its place and the process repeats.

Parameters

→ ***opList*** (SENSORS, 1) : Indices of 'modeShape' corresponding to optimal locations

← ***numSensors*** (integer) : The number of sensors to place

← ***modeShapes*** (DOFS, MODES) : Mode shapes of interest

← ***weights*** (MODES, 1) : Mode shape influence weights

← ***duelingDistance*** (scalar) : Maximum distance for dueling.

← ***respDOF*** (DOFS, DOFINFO) : Definitions of the degrees of freedom for the response vector "respVec". DOFINFO is the ordered set [coordID, responseDirection]. The first column contains the IDs of the geometric points corresponding to those listed in "geomLayout". The second column contains the direction of the DOF (1-X, 2-Y, 3-Z, 4-YZ, 5-XZ, 6-XY).

← ***geomLayout*** (POINTINFO, POINTS) : Layout array defining the IDs and coordinates of geometric points, POINTINFO is the ordered set [coordID, X, Y, Z]

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.92 PCA_shm.m File Reference

Feature Extraction: Principal component analysis (PCA).

Call Methods

[loadings, scores, latents, Xprime, residuals] = PCA_shm (X, perVar, stand)

[Loadings, Scores, Latents, Xprime, Residuals] = Principal Component Analysis (Data, Percentage of Variance, Standardization)

Function Description

This function performs the PCA on the covariance (or correlation) matrix of X.

Parameters

→ ***loadings*** (VARIABLES, PRINCIPALCOMP) : loadings matrix where columns are the principal components

→ ***scores*** (OBSERVATIONS, PRINCIPALCOMP) : scores matrix where rows correspond to observations and columns to principal components

→ ***latents*** (VARIABLES, 1) : vector containing the eigenvalues of the covariance (or correlation) matrix

→ ***Xprime*** (OBSERVATIONS, VARIABLES) : projection back on the original space using only the principal components that account to the required variance

→ ***residuals*** (OBSERVATIONS, VARIABLES) : corresponds to the information lost on the projection back to the original space

← ***X*** (OBSERVATIONS, VARIABLES) : data matrix where rows are observations and columns are variables

← ***perVar*** (scalar) : minimal percentage of the variance to explain the variability in the matrix X (only to compute Xprime and residuals)

← ***stand*** (logical) : whether data should be standardized or not

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

```
Figueiredo, E., Park, G., Figueiras, J., Farrar, C.,
and Worden, K. (2009).  Structural Health Monitoring
```


Algorithm Comparisons Using Standard Data Sets, Los Alamos National Laboratory Report: LA-14393.

Sample Data Set

www.lanl.gov/projects/ei

2.93 pdTree_shm.m File Reference

Data Analysis: Builds a principal direction partition tree.

Call Methods

[idx, tree] = pdTree_shm (X, k)

[Partitioning of X, Tree Structure] = Principal Direction Partition Tree (Data Set, Number of Partitions)

Function Description

Builds a hierarchical partition of X down to depth = floor(log(k)/log(2)). The final partition is retained and described in idx. The algorithm works by recursively bisecting the data space along principal directions of the data.

Parameters

→ ***idx*** (CLUSTERINDICES, 1) : vector of cluster membership

→ ***tree*** (class): contains the binary rptree as a cell array over the nodes

← ***X*** (INSTANCES, FEATURES) : training features

← ***k*** (scalar) : number of partition cells

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.94 plot2DMap_shm.m File Reference

Feature Extraction: Plot basic 2D color map.

Call Methods

axesHandle = plot2DMap_shm (xMatrix, yMatrix, dataMap2D, axesHandle)

Axes Handle = Plot 2D Map (X Matrix, Y Matrix, Data Map 2D, Axes Handle)

Function Description

Plots 2D colormap with zbuffer renderer, no lines, face interpolation, equal axis, and a colorbar.

Parameters

→ ***axesHandle*** (handle) : Matlab axes handle

← ***xMatrix*** (XINDEX, YINDEX) : Matrix of uniformly spaced X coordinates

← ***yMatrix*** (XINDEX, YINDEX) : Matrix of uniformly spaced Y coordinates

← ***dataMap2D*** (XINDEX, YINDEX) : 2D matrix of filled values with NaNs for empty space

← ***axesHandle*** (handle) : Matlab axes handle (default: new figure and axes)

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.95 plotBorder_shm.m File Reference

Feature Extraction: Plots set of line segments listed in border.

Call Methods

axesHandle = plotBorder_shm (border, axesHandle)

[Axis Handle] = Plot Border (Border, Axis Handle)

Function Description

Plots set of line segments listed in border

Parameters

→ ***axesHandle*** (handle) : Matlab axes handle

← ***border*** (ENDPOINTS, SEGMENTS): line segment list, ENDPOINTS is the ordered set [x1, y1, x2, y2]

← ***axesHandle*** (handle) : Matlab axes handle

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.96 plotFeatures_shm.m File Reference

Feature Extraction: Plot feature vectors as a subplot for each feature.

Call Methods

[axesHandle] = plotFeatures_shm (featureVectors, instanceIndices, featureIndices, subplotTitles, subplotYLabels, axesHandle)

[Axes Handle] = Plot Features (Feature Vectors, Intances to Plot, Features to Plot, Titles for Subplots, Y-Axis Labels for Subplots, Axes Handle)

Function Description

Plot feature vectors or a set of samples from feature vectors. Each feature uses its own subplot.

Parameters

→ ***axesHandle*** (handle) : MATLAB axes handle

← ***featureVectors*** (INSTANCES, FEATURES) : features where each row (INSTANCES) is a feature vector

← ***instanceIndices*** (1, INSTANCESCHOSEN) : list of indices of instances to be plotted, all instances are plotted by default

← ***featureIndices*** (1, FEATURESCHOSEN) : list of indices of features to be plotted, all features are plotted by default, maximum of 20

← ***subplotTitles*** ({1, FEATURESCHOSEN} (string) : titles for each subplot, enter in same order as featureIndices

← ***subplotYLabels*** ({1, FEATURESCHOSEN} (string) : y-axis labels for each subplot, enter in same order as featureIndices

← ***axesHandle*** (handle) : MATLAB axes handle

Author

Dustin Harvey
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.97 plotKurtogram_shm.m File Reference

Feature extraction: Create a kurtogram plot.

Call Methods

[axesHandle] = plotKurtogram_shm (fastKurtMatrix, channelIndex, instance, freqVector, levelVector, axesHandle)

[Axes Handle] = Plot Kurtogram (Kurtogram Data, Channel Index, Instance Index, Frequency Vector, Level Vector, Axes Handle)

Function Description

Create a kurtogram plot from output of fastKurtogram_shm.

Parameters

→ ***axesHandle*** (handle) : axes handle for plot

← ***fastKurtMatrix*** (LEVELS,FREQ,CHANNELS,INSTANCES) : kurtogram data matrix

← ***channelIndex*** (integer) : channel to plot

← ***instance*** (integer) : instance to plot

← ***freqVector*** (FREQ,1) : frequency values corresponding to fastKurtMatrix

← ***levelVector*** (NKLEVELS,1): levels vector corresponding to fastKurtMatrix

← ***axesHandle*** (handle) : handle for plotting axes

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.98 plotPSD_shm.m File Reference

Feature extraction: Create a plot of power spectral densities.

Call Methods

[axesHandle] = plotPSD_shm (psdMatrix, channelIndex, is1sided, freqVector, useDBscale, plotAverage, axesHandle)

[Axes Handle] = Plot Power Spectral Density (PSD Matrix, Channel Index, PSD Range, Frequency Vector, Use dB Magnitude, Plot Average PSD, Axes Handle)

Function Description

Create a plot of power spectral densities across all instances in a power spectral density matrix.

Parameters

→ ***axesHandle*** (handle) : axes handle for PSD image, vector of 2 axes handles if plotAverage is true

← ***psdMatrix*** (FREQ, CHANNELS, INSTANCES) : power spectral density matrix

← ***channelIndex*** (integer) : channel to plot

← ***is1sided*** (string) : specifies type of psd, one-sided or two-sided

← ***freqVector*** (FREQ,1) : frequency values corresponding to psdMatrix

← ***useDBscale*** (string) : use dB for magnitude scale instead of linear

← ***plotAverage*** (logical) : flags function to include a subplot of the average the power spectral densities across all instances.

← ***axesHandle*** (handle) : axes handle for PSD image

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.99 plotROC_shm.m File Reference

Feature Classification: Plot receiver operating characteristic curve.

Call Methods

[axesHandle] = plotROC_shm (TPR, FPR, scaling, axesHandle)

[Axes Handle] = Plot Receiver Operating Characteristic Curve (True Positive Rate, False Positive Rate, Scaling, Axes Handle)

Function Description

Plot receiver operating characteristic curve(s). A curve is plotted for each column of TPR and FPR

Parameters

→ ***axesHandle*** (handle) : Matlab axes handle

← ***TPR*** (POINTS, CURVES) : matrix composed of true positive rates between 0 and 1

← ***FPR*** (POINTS, CURVES) : matrix composed of false positive rates between 0 and 1

← ***scaling*** (string) : axis scaling 'linear' - Linear scale 'logx' - X-axis log scale 'logy' - Y-axis complimentary log scale 'logxy' - X-axis log scale and Y-axis complimentary log scale 'normal' - Normal probability paper scale

← ***axesHandle*** (handle) : Matlab axes handle

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.100 plotScalogram_shm.m File Reference

Feature extraction: Create a scalogram plot.

Call Methods

[axesHandle] = plotScalogram_shm (scaloMatrix, channelIndex, instance, timeVector, freqVector, axesHandle)

[Axes Handle] = Plot Scalogram (Scalogram Data, Channel Index, Instance Index, Time Vector, Frequency Vector, Axes Handle)

Function Description

Create a time-frequency plot from output of cwtScalogram_shm.

Parameters

→ ***axesHandle*** (handle) : axes handle for plot

← ***scaloMatrix*** (NSCALE,TIME,CHANNELS,INSTANCES) : scalogram data matrix

← ***channelIndex*** (integer) : channel to plot

← ***instance*** (integer) : instance to plot

← ***timeVector*** (TIME,1) : sampling times corresponding to scaloMatrix

← ***freqVector*** (NSCALE,1) : frequency vector corresponding to scales in scaloMatrix

← ***axesHandle*** (handle) : handle for plotting axes

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.101 plotScoreDistributions_shm.m File Reference

Feature Classification: Plot distribution of scores using KDE.

Call Methods

axesHandle = plotScoreDistributions_shm (scores, states, stateNames, thresholds, flipSigns, useLogScores, smoothing, axesHandle)

Axes Handle = Plot Score Distributions (Scores, Damage States, State Names, Thresholds, Flip Signs, Use Log Scores, Kernel Smoothing Parameter, Axes Handle)

Function Description

Plots distributions of the scores resulting from a scoring process by using a gaussian kernel density function. Each unique state is plotted with a different color. If provided, the score thresholds will be plotted as vertical dashed lines.

Parameters

→ ***axesHandle*** (handle) : Matlab axes handle

← ***scores*** (INSTANCES, 1) : Vector of scores from scoring process corresponding to each of the measured instances. Scores are essentialy the reduced single dimension decision variable that is the thresholded to determine the detected state.

← ***states*** (INSTANCES, 1) : Integers ranging from zero to N specifying the known state damage level correponding to each of the measured instances

← ***stateNames*** {STATES, 1} (string) : cell array of strings corresponding, in order, to each of the possible states (optional)

← ***thresholds*** (STATES, 1) : vector of thresholds corresponding to each of the possible states (optional)

← ***flipSigns*** (logical) : whether signs of scores and thresholds should be flipped

← ***useLogScores*** (logical) : whether log scores should be used

← ***smoothing*** (scalar) : kernel density function smoothing parameter

← ***axesHandle*** (handle) : Matlab axes handle

Author

Eric Flynn
Dustin Harvey
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.102 plotScores_shm.m File Reference

Feature Classification: Plot bar graph showing detection results.

Call Methods

axesHandle = plotScores_shm (scores, states, stateNames, thresholds, flipSigns, useLogScores, axesHandle)

Axes Handle = Plot Scores (Scores, States, State Names, Thresholds, Flip Signs, Use Log Scores, Axes Handle)

Function Description

Plots a bar graph of the scores resulting from a scoring process. Each unique state is plotted with a different color. If provided, the score thresholds will be plotted as dashed lines on top of the bar graph.

Parameters

→ ***axesHandle*** (handle) : Matlab axes handle

← ***scores*** (INSTANCES, 1) : Vector of scores from scoring process corresponding to each of the measured instances. Scores are essentialy the reduced single dimension decision variable that is the thresholded to determine the detected state.

← ***states*** (INSTANCES, 1) : Integers ranging from zero to N specifying the state (either known or detected) correponding to each of the measured instances

← ***stateNames*** {STATES, 1} (string) : cell array of strings corresponding, in order, to each of the possible states (optional)

← ***thresholds*** (STATES, 1) : vector of thresholds corresponding to each of the possible states (optional)

← ***flipSigns*** (logical) : whether signs of scores and thresholds should be flipped

← ***useLogScores*** (logical) : whether log scores should be used

← ***axesHandle*** (handle) : Matlab axes handle

Author

Eric Flynn
Dustin Harvey
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.103 plotSensors_shm.m File Reference

Optimal Sensor Placement: Plot sensors.

Call Methods

[axesHandle, sensorHandle] = plotSensors_shm (sensorLayout, axesHandle)

[Axes Handle, Sensor Handle] = Plot Sensors (Sensor Layout, Axes Handle)

Function Description

Plots sensors as black on white circles with sensor IDs and line segments indicating sensor directional sensitivity.

Parameters

→ ***axesHandle*** (handle) : MATLAB axes handle

→ ***sensorHandle*** (handle) : MATLAB handle to sensor plot data

← ***sensorLayout*** (SENSORINFO, SENSOR) : sensor layout IDs, coordinates, and sensitivity direction. Length of SENSORINFO dimension may vary. SENSORINFO is one of the following ordered sets: [sensorID; xCoord] [sensorID; xCoord; yCoord] [sensorID; xCoord; yCoord; zCoord] [sensorID; xCoord; xDir; yDir; zDir] [sensorID; xCoord; yCoord; xDir; yDir; zDir] [sensorID; xCoord; yCoord; zCoord; xDir; yDir; zDir]

← ***axesHandle*** (handle) : MATLAB axes handle

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.104 plotTestBase_shm.m File Reference

Feature Extraction: Plot test and baseline waveforms on a single axes.

Call Methods

[axesHandle] = plotTestBase_shm (testWave, baseWave, samplingRate, axesHandle)

[Axes Handle] = Plot Test and Baseline Waveforms (Test Waveform, Base Waveform, Sampling Rate, Axes Handle)

Function Description

Plot test and baseline waveforms on a single axes

Parameters

→ ***axesHandle*** (scalar) : Axes handle

← ***testWave*** (SAMPLES, WAVEFORMS) : Array of waveforms

← ***baseWave*** (SAMPLES, WAVEFORMS) : Array of baselines

← ***samplingRate*** (scalar) : Sampling rate (samples per second)

← ***axesHandle*** (scalar) : Axes handle for plotting

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Sets

1. `baseWave (SAMPLES, WAVEFORMS) : Array of baselines`
2. `samplingRate (scalar) : Sampling rate (samples per second)`
3. `axesHandle (scalar) : Axes handle for plotting`

2.105 plotTimeFreq_shm.m File Reference

Feature extraction: Create a time-frequency plot.

Call Methods

[axesHandle] = plotTimeFreq_shm (timeFreqMatrix, channelIndex, instance, timeVector, freqVector, useDBscale, axesHandle)

[Axes Handle] = Plot Time-Frequency (Time-Frequency Data, Channel Index, Instance Index, Time Vector, Frequency Vector, Use dB Magnitude,Axes Handle)

Function Description

Create a time-frequency plot from output of lpcSpectrogram_shm, stft_shm or dvwd_shm.

Parameters

→ ***axesHandle*** (handle) : axes handle for plot

← ***timeFreqMatrix*** (FREQ,TIME,CHANNELS,INSTANCES) : time-frequency data matrix

← ***channelIndex*** (integer) : channel to plot

← ***instance*** (integer) : instance to plot

← ***timeVector*** (TIME,1) : sampling times corresponding to timeFreqMatrix

← ***freqVector*** (FREQ,1) : frequency values corresponding to timeFreqMatrix

← ***useDBscale*** (string) : use dB for magnitude scale instead of linear

← ***axesHandle*** (handle) : handle for plotting axes

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.106 propagationDist2Points_shm.m File Reference

Active Sensing: Propagation distance from actuator to point to sensor.

Call Methods

[propDistance] = propagationDist2Points_shm (pairList, sensorLayout, pointList)

[Propogation Distance] = Propagation Distance to POIs (Pair List, Sensor Layout, Points of Interest)

Function Description

For each sensor pair, calculates the total propagation distance from sensor 1 to each spatial point to sensor 2.

Parameters

→ ***propDistance*** (PAIRS, POIS) : total propapagtion distances

← ***pairList*** (SENSORIDS, PAIRS) : Matrix of actuator-sensor PAIRS, SENSORIDS is the ordered set of integer IDs [actuatorID, sensorID]

← ***sensorLayout*** (SENSORINFO, SENSORS) : sensor layout IDs and coordinates, SENSORINFO is the ordered set [sensorID, xCoord, yCoord]

← ***pointList*** (COORDINATES, POIS) : List of points of interest (POIS) coordinates; COORDINATES is the ordered set [x, y]

Author

Eric Flynn
LA-CC-14-046

Modifications

1. January 20 , 2009, by Eric Flynn, for inclusion in the SHM software

Reference

`none`

Sample Data Set

`none`

2.107 psdWelch_shm.m File Reference

Feature Extraction: Estimate power spectral density via Welch's method.

Call Methods

[psdMatrix, f, is1sided] = psdWelch_shm (X, nWin, nOvlap, nFFT, Fs, useOneSided)

[PSD Matrix, Frequency Vector, PSD Range] = Power Spectral Density via Welch's Method (Signal Matrix, Window Length, Overlap Length, FFT Bins, Sampling Frequency, Use One-Sided PSD)

Function Description

Computes the power spectral density of a digital signal using Welch's method. Welch's method segments a signal into a specified number of windowed signals with a specified number of overlapping samples and computes their Fourier transforms independently then averages these Fourier transforms to get an estimate of the power spectral density of a signal. Typically the number of overlapping segments should be between 50 to 75 percent of the window length. The sampling frequency can be specified but if left empty [] it defaults to 1 making the frequency vector centered and normalized. For one-sided power spectral densities, the range is between [0,0.5]∗Fs and [-0.5,0.5]∗Fs for two sided power spectral densities the

Parameters

→ ***psdMatrix*** (NFFT, CHANNELS, INSTANCES) : power spectral density matrix

→ ***f*** (NFFT, 1) : frequency vector corresponding to psdMatrix

→ ***is1sided*** (string) : specifies type of psd, one-sided or two-sided

← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data

← ***nWin*** (integer) : samples per window

← ***nOvlap*** (integer) : number of overlapping samples between windows

← ***nFFT*** (integer) : number of FFT frequency bins

← ***Fs*** (double) : sampling frequency in Hz

← ***useOneSided*** (logical) : create one-sided PSD instead of two-sided

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.108 quarticKernel_shm.m File Reference

Outlier Detection: Kernel weights for the quartic kernel.

Call Methods

W = quarticKernel_shm (Delta)

Weights = Quartic Kernel Weights (Evaluation Points)

Function Description

Estimate the quartic kernel at points in Delta. In one dimension, this is given as K(u) = (15/16).(1-|u|^2)^2_+ where u = (x -x_i)/h. In higher dimensions we just take the product of the one dimensional kernel at each coordinate.

Parameters

→ ***W*** (WEIGHTS, 1) : evaluated kernel weights

← ***Delta*** (INSTANCES, FEATURES) : each row is (x-x_i)/bandwidth where the kernel is centered at x, and x_i are the training points.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`http://en.wikipedia.org/wiki/Kernel_(statistics)`

Sample Data Set

`none`

2.109 reduce2PairSubset_shm.m File Reference

Active Sensing: Extract parameter and data subsets based on sensor subset.

Call Methods

[pairSubset, layoutSubset ,waveformBaseSub, waveformTestSub] = reduce2PairSubset_shm (sensorSubset, sensorLayout, pairList, waveformBase, waveformTest)

[Pair Subset, Layout Subset, Baseline Waveform, Test Waveform] = Reduce to Pair Subset (Sensor Subset, Sensor Layout, Pair List, Baseline Waveforms, Test Waveforms)

Function Description

Extract parameter and data subsets according to sensor subset

Parameters

→ ***pairSubset*** (SENSORS,PAIRS) : subset of sensor pairs, SENSORIDS is the ordered set of integer IDs [actuatorID, sensorID]

→ ***layoutSubset*** (SENSORINFO, SENSORS) : subset of sensorLayout, SENSORINFO is the ordered set [sensorID, xCoord, yCoord]

→ ***waveformBaseSub*** (TIME, PAIRS, ...) : subset of pair data

→ ***waveformTestSub*** (TIME, PAIRS, ...) : subset of pair data

← ***sensorSubset*** (SENSORS,1) : list of sensor subset IDs

← ***sensorLayout*** (SENSORINFO, SENSORS) : sensor layout IDs and coordinates, SENSORINFO is the ordered set [sensorID, xCoord, yCoord]

← ***pairList*** (SENSORIDS, PAIRS) : Matrix of actuator-sensor PAIRS, SENSORIDS is the ordered set of integer IDs [actuatorID, sensorID]

← ***waveformBase*** (TIME, PAIRS, ...) : pair data

← ***waveformTest*** (TIME, PAIRS, ...) : pair data

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.110 res2modes_shm.m File Reference

Feature Extraction: Extract mode shapes from complex residues.

Call Methods

[modalvector, mpf] = res2modes_shm (ic2rmeth, res)

[Modal Vectors, Modal Participation Factors] = Residues to Modes (Method, Resdiues)

Function Description

Computes modal vectors and modal participation factors from complex residues.

Parameters

→ ***modalvector*** (DOFS, MODES) : vector(s) of modeshapes

→ ***mpf*** (1, MODES) : complex modal participation factors

← ***ic2rmeth*** (string) : method used to convert residues to modes; 'Real Part', 'Imag Part', or 'Phase Rounding'

← ***res*** (DOFS, MODES) : matrix of complex modal residues

Author

Scott W. Doebling
Phillip J. Cornwell
Erik G. Straser
Charles R. Farrar
LA-CC-14-046

Modifications

1. February 8, 2009, by Stuart Taylor, for inclusion with the SHM software modal analysis group

Reference

```
Richardson, M.H. & Formenti, D.L., "Parameter
Estimation from Frequency Response Measurements using
Rational Fraction Polynomials", Proceedings of the
1st International Modal Analysis Conference, Orlando,
Florida, November 8-10, 1982.
```

Sample Data Set

```
www.lanl.gov/projects/ei
```


2.111 responseInterp_shm.m File Reference

Optimal Sensor Placement: Interpolate structure response.

Call Methods

respXYZ = responseInterp_shm (geomLayout, respVec, respDOF, use3DInterp)

[Displacement XYZ]= Response Interpolation (Geometry Layout, Response Vector, Response DOF, Use 3D Interpolation)

Function Description

Fills a 2D array "respXYZ" containing the X, Y, Z responses for the coordinates defined in "geomLayout" using the response vector "respVec" and DOF definitions "respDOF". The response at coordinates not given in "respVec" are determined through interpolation.

Parameters

→ ***respXYZ*** (DISPLACEMENTS, COORDINATES, INSTANCES) : Array of displacements in the X, Y, and Z directions for the coordinates defined in "geomLayout", DISPLACEMENTS is the ordered set [X Y Z]

← ***geomLayout*** (POINTINFO, POINTS) : Layout array defining the IDs and coordinates of geometric points, POINTINFO is the ordered set [coordID, X, Y, Z]

← ***respVec*** (DOFS, INSTANCES) : Vector of responses at the degrees of freedom listed in respDOF

← ***respDOF*** (DOFS, DOFINFO, INSTANCES) : Definitions of the degrees of freedom for the response vector "respVec". DOFINFO is the ordered set [coordID, responseDirection]. The first column contains the IDs of the geometric points corresponding to those listed in "geomLayout". The second column contains the direction of the DOF (1-X, 2-Y, 3-Z, 4-YZ, 5-XZ, 6-XY).

← ***use3DInterp*** (logical) : whether 3D interpolation should be used as a first step, 3D interpolation does not perform well when the points are sparse or extrapolation is required

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.112 rms_shm.m File Reference

Feature Extraction: Calculate root mean square feature.

Call Methods

[rms] = rms_shm (X)

[RMS Feature Vectors] = RMS Feature (Signal Matrix)

Function Description

Computes the root mean square feature.

Parameters

→ ***rms*** (INSTANCES, FEATURES) : feature vectors of root mean square feature in concatenated format, FEATURES = CHANNELS

← ***X*** (SAMPLES, CHANNELS, INSTANCES) : signal matrix

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.113 ROC_shm.m File Reference

Feature Classification: Receiver operating characteristic (ROC) curve.

Call Methods

[TPR, FPR] = ROC_shm (scores, damageStates, numPts, thresholdType)

[True Positive Rate, False Positive Rate] = Receiver Operating Characteristic (Scores, Damaged States, # of Points, Threshold Type)

Function Description

Tool to compare and evaluate the performance of classification algorithms. Note that the scores should decrease for the damaged instances.

Parameters

→ ***TPR*** (POINTS, 1) : vector composed of true positive rates

→ ***FPR*** (POINTS, 1) : vector composed of false positive rates

← ***scores*** (INSTANCES, 1) : vector composed of scores for each instance

← ***damageStates*** (INSTANCES, 1) : binary classification vector of known damage states (0-undamaged and 1-damaged) corresponding to vector of scores

← ***numPts*** (scalar) : number of points to evaluate ROC curve at. If set to default (recommended), each score value from a damaged state (scores(damageStates==1)) is used as a threshold to calculate an roc curve point making numPts equal to sum(damageStates==1)

← ***thresholdType*** (string) : 'above' or 'below' to define if scores above or below a given threshold should be flagged as damaged

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. December 9, 2009 by Eric Flynn. Added optional input to specify number of points to evaluate ROC curve at. Default thresholds to evaulate ROC curve at (if numPts not specified) is now equal to the set of input scores corresponding to the true labels.
2. March 25, 2010 by Eric Flynn Added thresholdType option. Modified code for substantial efficiency improvement.

Reference

`none`

Sample Data Set

`http://institute.lanl.gov/ei/software-and-data/`

2.114 rpfit_shm.m File Reference

Feature Extraction: Rational polynomial curve-fitting.

Call Methods

[res, Freq, Damp] = rpfit_shm (G, df, frq, nmodes, exterms)

[Residues, Frequencies, Damping] = Rational Poly Fit (FRF, Frequency Resolution, Frequency Range, # of Modes, # of Extra Terms)

Function Description

Performs rational polynomial curve-fit using Forsythe Orthogonal Polynomials. Returns complex residues for both reference and response DOF using a singular value decomposition to convert from modal participation factors. Returns residues based on first reference (residues are not computed using multi-reference data).

Parameters

→ ***res*** (DOFS, MODES, INSTANCES) : matrix of complex modal residues

→ ***Freq*** (MODES, INSTANCES) : modal frequencies

→ ***Damp*** (MODES, INSTANCES) : damping ratios

← ***G*** (FREQUENCY, DOFS, INSTANCES) : matrix of complex frequency response functions

← ***df*** (scalar) : frequency resolution

← ***frq*** (RANGES, ENDPOINTS) : frequency ranges in Hz; ENDPOINTS is the ordered set [startRange, endRange]

← ***nmodes*** (RANGES, 1) : number of modes to fit in each frequency range; must be in same order as frq

← ***exterms*** (integer) : number of extra terms to include

Author

Scott W. Doebling
Phillip J. Cornwell
Erik G. Straser
Charles R. Farrar
LA-CC-14-046

Modifications

1. February 8, 2009, by Stuart Taylor, for inclusion with the SHM software modal analysis group

Reference

Richardson, M.H. & Formenti, D.L., "Parameter Estimation from Frequency Response Measurements using Rational Fraction Polynomials", Proceedings of the 1st International Modal Analysis Conference, Orlando, Florida, November 8–10, 1982.

Sample Data Set

www.lanl.gov/projects/ei

2.115 rpTree_shm.m File Reference

Data Analysis: Builds a random projection partition tree.

Call Methods

[idx, tree] = rpTree_shm (X, k)

[partitioning of X, tree structure] = Random Projection Partition Tree (Data Set, Number of Partitions)

Function Description

Builds a hierarchical partition of X down to depth = floor(log(k)/log(2)). The final partition is retained and described in idx. The algorithm works by recursively bisecting the data space with random hyperplanes.

Parameters

→ ***idx*** (CLUSTERINDICES, 1) : vector of cluster membership

→ ***tree*** (class): contains the binary rptree as a cell array over the nodes

← ***X*** (INSTANCES, FEATURES) : training features

← ***k*** (scalar) : number of partition cells

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.116 scaleMinMax_shm.m File Reference

Feature Extraction: Scale data to a minimum and maximum value.

Call Methods

[scaledX] = scaleMinMax_shm (X, scalingDimension, scaleRange)

[Scaled Time Series Data] = Scale Minimum Maximum (Data Matrix, Dimension to Scale, Scale Range)

Function Description

This function scales the data between the values in the scaling range along the specified dimension.

Parameters

→ ***scaledX*** (...) : scaled data matrix

← ***X*** (...) : data matrix

← ***scalingDimension*** (integer) : dimension of X to apply scaling along

← ***scaleRange*** (1, ENDPOINTS) : Minimum and maximum value for scaledX, if no scaleRange is specified data is normalized to z-score, ENDPOINTS is the ordered set [minValue maxValue]

Author

Dustin Harvey
Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`www.lanl.gov/projects/ei`

2.117 scoreFactorAnalysis_shm.m File Reference

Outlier Detection: Score factor analysis (FA).

Call Methods

[scores, uniqueFactors, factorScores] = scoreFactorAnalysis_shm (Y, model)

[Scores, Unique Factors, Factor Scores] = Score Factor Analysis (Test Features, Model)

Function Description

This function returns a score for each instance of the data matrix Y based on the Euclidean norm of the unique factors. The features in Y are standardized to zero mean and unit variance using the mean and standard deviation vectors estimated in the training phase. In the context of damage detection, the objective of factor analysis is to identify the underlying common factors that can explain the correlation among the features. The factors are the latent variables affecting the features, e.g. temperature or humidity. The common factor model is used to estimate the underlying factors by means of the training data and latter to eliminate their effects in the test data in such a way that the unique factors are independent variables and insensitive to the operational and environmental variations.

Parameters

→ ***scores*** (INSTANCES, 1) : vector of scores

→ ***uniqueFactors*** (INSTANCES, FEATURES) : matrix of unique factors or residuals

→ ***factorScores*** (INSTANCES, NUMFACTORS) : matrix of factor scores

← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector

← ***model*** (struct) : parameters of the model, the fields are as follows: .lambda (FEATURES, NUMFACTORS) : matrix of factor loadings

← ***.psi(FEATURES,FEATURES)*** : specific variances matrix

← ***.dataMean*** (1, FEATURES) : mean vector of the variables

← ***.dataStd*** (1, FEATURES) : standard deviation of the variables

← ***.numFactors*** (integer) : number of common factors

← ***.estMethod*** (string) : factor scores estimation method

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

Kullaa, J. (2003). Is Temperature Measurement Essential in Structural Health Monitoring? Proceedings of the 4th International Workshop on Structural Health Monitoring 2003: From Diagnostic & Prognostics to Structural Health Monitoring (pp. 717-724). DEStech Publications, Inc.

Sample Data Set

http://institute.lanl.gov/ei/software-and-data/

2.118 scoreFastMetricKernelDensity_shm.m File Reference

Outlier Detection: Score fast non parametric kernel density estimation.

Call Methods

densities = scoreFastMetricKernelDensity_shm (Y, model, returnLogDensities)

Density Scores = Score Fast Metric Kernel Density Estimation (Test Features, Model, Use Log Densities)

Function Description

Estimate the density at points in Y given a (distribution) model that contains a cover tree used for fast estimation on a metric space.

Parameters

→ ***densities*** (INSTANCES, 1) : evaluated log densities

← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector

← ***model*** (struct): distribution parameters including a cover tree, training sample, kernel function, and bandwidth

← ***returnLogDensities*** (logical): return density values if 1, otherwise return log(densities) which is the default

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.119 scoreGMM_shm.m File Reference

Outlier Detection: Score gaussian mixture model.

Call Methods

densities = scoreGMM_shm (Y, model, returnLogDensities)

Density Scores = Score Gaussian Mixture Model (Test Features, Distribution Parameters, Use Log Densities)

Function Description

Estimate the density at points in Y given the distribution (model).

Parameters

→ ***densities*** (INSTANCES, 1) : evaluated log densities

← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector

← ***model*** (struct): distribution parameters

← ***returnLogDensities*** (logical): return density values if 1, otherwise return log(densities) which is the default

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.120 scoreGMMSemiParametricModel_shm.m File Reference

Outlier Detection: Score GMM semi-parametric density model.

Call Methods

densities = scoreGMMSemiParametricModel_shm (Y, model, returnLogDensities)

Density Scores = Score GMM Semi-Parametric Density Model (Test Features, Model, Use Log Densities)

Function Description

Estimate the density at points in Y given the distribution (model). This function just calls scoreGMM_shm and exists just to maintain the structure of learn/score pairs expected by assembly routines in the package.

Parameters

→ ***densities*** (INSTANCES, 1) : evaluated log densities

← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector

← ***model*** (struct): distribution parameters

← ***returnLogDensities*** (logical): return density values if 1, otherwise return log(densities) which is the default

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.121 scoreKernelDensity_shm.m File Reference

Outlier Detection: Score non parametric kernel density estimation.

Call Methods

densities = scoreKernelDensity_shm (Y, model, returnLogDensities)

Density Scores = Score Kernel Density Estimation (Test Features, Model, Use Log Densities)

Function Description

Estimate the density at points in Y given the (distribution) model.

Parameters

- → ***densities*** (INSTANCES, 1) : evaluated log densities
- ← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector
- ← ***model*** (struct): distribution parameters including a training sample, kernel function, and bandwidth matrix (assumed to be diagonal)
- ← ***returnLogDensities*** (logical): return density values if 1, otherwise return log(densities) which is the default

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.122 scoreMahalanobis_shm.m File Reference

Outlier Detection: Score mahalanobis squared distance.

Call Methods

[scores] = scoreMahalanobis_shm (Y, model)

[Scores] = Score Mahalanobis (Test Features, Model)

Function Description

For each instance of Y, this function returns a score based on Mahalanobis distance data points. The parameters (mean vector and covariance matrix) are set up based on the training data.

Parameters

→ ***scores*** (INSTANCES, 1) : vector of scores

← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector

← ***model*** (struct) : parameters of the model, the fields are as follows: .dataMean (1, FEATURES) : mean vector of the features

← ***.dataCov*** (FEATURES, FEATURES) : covariance matrix of the features

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

Worden, K., & Manson, G. (2000). Damage Detection using Outlier Analysis. Journal of Sound and Vibration , 229 (3), 647-667.

Sample Data Set

http://institute.lanl.gov/ei/software-and-data/

2.123 scoreNLPCA_shm.m File Reference

Outlier Detection: Score nonlinear principal component analysis (NLPCA).

Call Methods

[scores, residuals] = scoreNLPCA_shm (Y, model)

[Scores, Residuals] = Score NLPCA (Test Features, Model)

Function Description

For each instance of Y, this function returns a score based on the Euclidean norm of the residual errors between the target features and the output of the final layer.

Parameters

→ ***scores*** (INSTANCES, 1) : column vector composed by n scores

→ ***residuals(INSTANCES,FEATURES)*** : residual errors

← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector

← ***model*** (struct) : parameters of the model, the fields are as follows: .net (object) : neural network object defining the basic features of the trained network

← ***.b*** (integer) : number of nodes in the bottleneck layer

← ***.M1*** (integer) : number of nodes in the mapping layer

← ***.M2*** (integer) : number of nodes in the de-mapping layer (M1=M2 by default)

← ***.E*** (scalar) : mean square error of the training error

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

References

1. Sohn, H., Worden, K., & Farrar, C. R. (2002). Statistical Damage Classification under Changing Environmental and Operational Conditions. Journal of Intelligent Material Systems and Structures , 13 (9), 561-574.

2. Kramer, M. A. (1991). Nonlinear Principal Component Analysis using Autoassociative Neural Networks. AIChE Journal , 37 (2), 233-243.

Sample Data Set

`http://institute.lanl.gov/ei/software-and-data/`

2.124 scorePCA_shm.m File Reference

Outlier Detection: Score principal component analysis (PCA).

Call Methods

[scores, residuals] = scorePCA_shm (Y, model)

[Scores, Residuals] = Score Principal Component Analysis (Test Features, Model)

Function Description

This function returns one score for each instance (row) of the data test Y, based on the Euclidean norm.

Parameters

→ ***scores*** (INSTANCES, 1) : vector of scores

→ ***residuals*** (INSTANCES, FEATURES) : residual errors

← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector

← ***model(struct)*** : parameters of the model, the fields are as follows: .loadings(FEATURES, PRINCIPALCOMP) : loadings matrix where columns are the number of principal components

← ***.dataParam(2,FEATURES)*** : (1) mean and (2) standard deviation vectors

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

Figueiredo, E., Park, G., Figueiras, J., Farrar, C., and Worden, K. (2009). Structural Health Monitoring Algorithm Comparisons Using Standard Data Sets, Los Alamos National Laboratory Report: LA-14393.

Sample Data Set

www.lanl.gov/projects/ei

2.125 scoreSVD_shm.m File Reference

Outlier Detection: Score singular value decomposition (SVD).

Call Methods

[scores, residuals] = scoreSVD_shm (Y, model)

[Scores, Residuals] = Score Singular Value Decomposition (Test Features, Model)

Function Description

This function returns scores based on the Euclidean norm of the residuals between the singular vectors S and Sc. S and Sc are estimated from M and Mc, respectively. Each state matrix (Mc) is composed by X and one feature vector of Y at a time. In the context of damage detection, if the rank of M is r and if a vector from Y comes from a damaged condition, the rank of Mc will be equal to r+1. If the rank of M and Mc are the same, this algorithm assumes that when Mc is composed by a damaged condition, the damage detection is based on changes in magnitude of the singular values.

Parameters

→ ***scores*** (INSTANCES, 1) : vector of scores

→ ***residuals*** (INSTANCES, FEATURES) : residuals

← ***Y*** (INSTANCES, FEATURES) : test features where each row (INSTANCES) is a feature vector

← ***model*** (struct) : parameters of the model, the fields are as follows: .X (INSTANCES, FEATURES) : training data matrix where each row (INSTANCES) is a feature vector

← ***.S*** (FEATURES, 1) : singular values of X

← ***.dataMean*** (1, FEATURES) : Column vector composed of the mean of the variables

← ***.dataStd*** (1, FEATURES) : Column vector composed of the standard deviations of the variables

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

References

1. Ruotolo, R., & Surage, C. (1999). Using SVD to Detect Damage in Structures with Different Operational Conditions. Journal of Sound and Vibration , 226 (3), 425-439.

2. Figueiredo, E., Park, G., Figueiras, J., Farrar, C., and Worden, K. (2009). Structural Health Monitoring Algorithm Comparisons Using Standard Data Sets, Los Alamos National Laboratory Report: LA-14393.

Sample Data Set

www.lanl.gov/projects/ei

2.126 sdAutoclassify_shm.m File Reference

Feature Extraction: Piezoelectric sensor diagnostics auto-classification.

Call Methods

[sensor_status, data_for_plotting] = sdAutoclassify_shm (slope, thre)

[Sensor Status, Data for Plotting] = Sensor Diagnostics Auto-Classify (slope, Threshold Limit)

Function Description

Automatically classify the operation status of an array of piezoelectric active-sensors in SHM applications. With an array of senosrs, the algorithm instaneously establishes a baseline for sensor status comparison. The assumptions in this approach are 1)one use the same size/material of piezoelectric active-sensors 2)The maximum number of unhealthy patches is less than half of the total number of sensors

Parameters

→ ***sensor_status*** (SENSORS, SENSORINFO) : three columns of information for each sensor: FIRST COLUMN - sensor id number, which correspondes to the input matrix of the sd_feature_shm function; SECOND COLUMN - sensor status (0 is healthy, 1 is de-bonded, 2 is broken/fractured); THIRD COLUMN - capacitance value of piezoelectric active-sensors in nF

→ ***data_for_plotting*** (struct) : data for plotting using the sd_plot_shm function

← ***slope*** (SENSORS, 1) : vector of the slope of the imaginary part of the measured admittance data, this is the output of the sd_feature_shm function

← ***thre*** (scalar) : threshole limit for sensor failure. 0.02 means that the algorothm allows upto 2 variations among healthy sensors

Author

Tim Overly
LA-CC-14-046

Modifications

1. August 20, 2009, by Gyuhae Park, for inclusion with the SHM software modal analysis group

References

1. Overly, T.G., Park, G., Farinholt, K.M., Farrar, C.R. "Piezoelectric Active-Sensor Diagnostics and Validation Using Instantaneous Baseline Data," IEEE Sensors Journal, in press.

2. Park, G., Farrar, C.R., Rutherford, C.A., Robertson, A.N., 2006, "Piezoelectric Active Sensor Self-diagnostics using Electrical Admittance Measurements," ASME Journal of Vibration and Acoustics, 128(4), 469-476.

3. Park, G., Farrar, C.R., Lanza di Scalea, F., Coccia, S., 2006, "Performance Assessment and Validation of Piezoelectric Active Sensors in Structural Health Monitoring," Smart Materials and Structures, 15(6), 1673-1683.

4. Park, S., Park, G., Yun, C.B., Farrar, C.R., 2009, "Sensor Self-Diagnosis Using a Modified Impedance Model for Active-Sensing Structural Health Monitoring," International Journal of Structural Health Monitoring, 8(1),71-82.

Sample Data Set

www.lanl.gov/projects/ei

2.127 sdFeature_shm.m File Reference

Feature Extraction: Piezoelectric sensor diagnostics feature extraction.

Call Methods

[slope] = sdFeature_shm (X)

[Sensor Capacitance] = Sensor Diagnostics Feature (Admittance Data)

Function Description

Extract the capacitance of an array of piezoelectric active-sensors for checking the operational status in SHM applications. Return the slope of piezoelectric active-sensors from the electrical admittance (inverse of impedance) data, which are measured at a relatively low-frequency range (usually <20 kHz). The first column of the slope value is equivalent to the capacitance value of the piezoelectric tranducers.

Parameters

→ ***slope*** (SENSORS, 1) : vector of the slope of the imaginary part of the measured admittance data

← ***X*** (FREQUENCY, SENSORS) : matrix of imaginary electrical admittance. The first column of SENSORS is the frequency range in Hz. After the first column, each column is the admittance data from a sensor. For subsequent analysis, the sensor number is assigned to (column # -1) because the first column is used for the frequency range.

Author

Tim Overly
LA-CC-14-046

Modifications

1. August 20, 2009, by Gyuhae Park, for inclusion with the SHM software modal analysis group

References

1. Overly, T.G., Park, G., Farinholt, K.M., Farrar, C.R. "Piezoelectric Active-Sensor Diagnostics and Validation Using Instantaneous Baseline Data," IEEE Sensors Journal, in press.

2. Park, G., Farrar, C.R., Rutherford, C.A., Robertson, A.N., 2006, "Piezoelectric Active Sensor Self-diagnostics using Electrical Admittance Measurements," ASME Journal of Vibration and Acoustics, 128(4), 469-476.

3. Park, G., Farrar, C.R., Lanza di Scalea, F., Coccia, S., 2006, "Performance Assessment and Validation of Piezoelectric Active Sensors in Structural Health Monitoring," Smart Materials and Structures, 15(6), 1673-1683.

4. Park, S., Park, G., Yun, C.B., Farrar, C.R., 2009, "Sensor Self-Diagnosis Using a Modified Impedance Model for Active-Sensing Structural Health Monitoring," International Journal of Structural Health Monitoring, 8(1),71-82.

Sample Data Set

www.lanl.gov/projects/ei

2.128 sdPlot_shm.m File Reference

Feature Extraction: Piezoelectric sensor diagnostics plot.

Call Methods

sdPlot_shm (A)

Sensor Diagnostics Result Plot (Data for Plotting)

Function Description

Plot the sensor stauts using the sdAutoclassify function. Opens two figures. The first figure describes the outcome of the algorithm of Overly et al. The second figure quantitatively describes the sensor status with the percent deviation from the mean value of the capacitance value of healthy sensors.

Parameters

→ ***none***

← ***A*** (struct) : the second output parameter of the sdAutoclassify_shm function.

Author

Tim Overly
LA-CC-14-046

Modifications

1. August 20, 2009, by Gyuhae Park, for inclusion with the SHM software modal analysis group

References

1. Overly, T.G., Park, G., Farinholt, K.M., Farrar, C.R. "Piezoelectric Active-Sensor Diagnostics and Validation Using Instantaneous Baseline Data," IEEE Sensors Journal, in press.

2. Park, G., Farrar, C.R., Rutherford, C.A., Robertson, A.N., 2006, "Piezoelectric Active Sensor Self-diagnostics using Electrical Admittance Measurements," ASME Journal of Vibration and Acoustics, 128(4), 469-476.

3. Park, G., Farrar, C.R., Lanza di Scalea, F., Coccia, S., 2006, "Performance Assessment and Validation of Piezoelectric Active Sensors in Structural Health Monitoring," Smart Materials and Structures, 15(6), 1673-1683.

4. Park, S., Park, G., Yun, C.B., Farrar, C.R., 2009, "Sensor Self-Diagnosis Using a Modified Impedance Model for Active-Sensing Structural Health Monitoring," International Journal of Structural Health Monitoring, 8(1),71-82.

Sample Data Set

www.lanl.gov/projects/ei

2.129 sensorPairLineOfSight_shm.m File Reference

Active Sensing: Detects line of sight presence from sensors to points.

Call Methods

[pairLineOfSight, sensorLineOfSight] = sensorPairLineOfSight_shm (pairList, sensorLayout, pointList, border)

[Pair Line of Sight, Sensor Line of Sight] = Sensor Pair Line Of Sight (Pair List, Sensor Layout, Points of Interest, Border)

Function Description

Returns a logical matrix corresponding to the presence of direct line of sight from each pair of sensors to each spatial point. If any line segment in "border" crosses the direct path from actuator to spatial point to sensor, the corresponding logic value will be false. Also returns a logical matrix corresponding to the individual line of sight of each actuator/sensor to each point of interest.

Parameters

→ ***pairLineOfSight*** (PAIRS, POINTS) : logical matrix of lines of sight presence from sensor 1 to spatial points to sensor 2

→ ***sensorLineOfSight*** (SENSORS, POINTS) : logical matrix of lines of sight presence from sensor to spatial points

← ***pairList*** (SENSORIDS, PAIRS) : Matrix of actuator-sensor PAIRS, SENSORIDS is the ordered set of integer IDs [actuatorID, sensorID]

← ***sensorLayout*** (SENSORINFO, SENSORS) : sensor layout IDs and coordinates, SENSORINFO is the ordered set [sensorID, xCoord, yCoord]

← ***pointList*** (COORDINATES, POIS) : List of points of interest (POIS) coordinates; COORDINATES is the ordered set [x, y]

← ***border*** (ENDPOINTS, SEGMENTS): Matrix of line segments that define external or internal plate boundaries, ENDPOINTS is the ordered set [x1, y1, x2, y2]

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.130 splitData_shm.m File Reference

Data Management: Split data matrix into two sets along dimension three.

Call Methods

[baselineData, testData, allData] = splitData_shm (data, baselineIndices, testIndices, channelsToUse)

[Baseline Data, Test Data, All Data] = Split Data Into Baseline and Testing (Data, Baseline Indices, Test Indices, Channel Indices to Use)

Function Description

Split data matrix into two sets along third dimension to create baseline and test datasets. Optionally select only a subset of channels.

Parameters

- → ***baselineData*** (POINTS, CHANNELSCHOSEN, BASEINST) : baseline instances of data
- → ***testData*** (POINTS, CHANNELSCHOSEN, TESTINST) : test instances of data
- → ***allData*** (POINTS, CHANNELSCHOSEN, INSTANCES) : all instances of data
- ← ***data*** (POINTS, CHANNELS, INSTANCES) : data matrix to be split
- ← ***baselineIndices*** (1, BASEINST) : list of indices of instances to be used for baseline or logical indexing vector (1, INSTANCES)
- ← ***testIndices*** (1, TESTINST) : list of indices of instances to be used for testing or logical indexing vector (1, INSTANCES)
- ← ***channelsToUse*** (1, CHANNELSCHOSEN) : list of indices of a subset of channels to use

Author

Dustin Harvey
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.131 splitFeatures_shm.m File Reference

Data Management: Split feature vectors into two sets along dimension one.

Call Methods

[trainingFeatures, scoringFeatures, allFeatures] = splitFeatures_shm (features, trainingIndices, scoringIndices, featuresToUse)

[Training Features, Scoring Features, All Features] = Split Features Into Training and Scoring (Features, Training Indices, Scoring Indices, Feature Indices to Use)

Function Description

Split feature vectors into two sets along first dimension to create learning and scoring feature vectors. Optionally select only a subset of features.

Parameters

→ ***trainingFeatures*** (SCORINST, FEATURESCHOSEN) : training instances of features

→ ***scoringFeatures*** (TRAININST, FEATURESCHOSEN) : scoring instances of features

→ ***allFeatures*** (INSTANCES, FEATURESCHOSEN) : all instances of features

← ***features*** (INSTANCES, FEATURES) : feature vectors to be split

← ***trainingIndices*** (1, TRAININST) : list of indices of instances to be used for training or logical indexing vector (1, INSTANCES)

← ***scoringIndices*** (1, SCORINST) : list of indices of instances to be used for scoring or logical indexing vector (1, INSTANCES)

← ***featuresToUse*** (1, FEATURESCHOSEN) : list of indices of a subset of features to use from feature vectors

Author

Dustin Harvey
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.132 statMoments_shm.m File Reference

Feature Extraction: Calculate first four statistical moments.

Call Methods

[statisticsFV] = statMoments_shm (X)

[Statistics Feature Vectors] = Statistical Moments (Time Series Data)

Function Description

Returns the first four statistical moments as damaged sensitive features.

Parameters

→ ***statisticsFV*** (INSTANCES, FEATURES) : First four statistical moments of each channel grouped by moments: (1) mean, (2) standard deviation, (3) skewness, (4) kurtosis; FEATURES = 4∗CHANNELS

← ***X*** (TIME, CHANNELS, INSTANCES) : time series matrix

Author

Eloi Figueiredo
LA-CC-14-046

Modifications

1. none

Reference

Figueiredo, E., Park, G., Figueiras, J., Farrar, C., and Worden, K. (2009). Structural Health Monitoring Algorithm Comparisons Using Standard Data Sets, Los Alamos National Laboratory Report: LA-14393.

Sample Data Set

www.lanl.gov/projects/ei

2.133 stft_shm.m File Reference

Feature Extraction: Computes short-time Fourier transforms.

Call Methods

[stftMatrix, f, t] = stft_shm (X, nWin, nOvlap, nFFT, Fs)

[Short Time Frequency Transform Matrix, Frequency Vector, Time Vector] = Short Time Fourier Transform (Signal Matrix, Number of Samples per Window, Numbers Overlapping Window Samples, Number of FFT Bins, Sampling Frequency)

Function Description

Computes a matrix short-time Fourier transforms for a matrix of signals. The short-time Fourier transform is computed by segmenting a signal into a series of nWin length windowed segments of nOvlap overlapping samples and taking their Fourier transforms to get a time frequency domain representation of the signal. The STFT has a tradeoff between time and frequency resolution. The longer the window lengths the better frequency resolution that can be obtained but the trade off is smearing in the time domain. As well the number of overlapping samples can increase the time resolution but increases computational time and memory resources. Fs is the sampling frequency of the signal but is left empty the frequency domain is normalized.

Parameters

- → ***stftMatrix*** (NFREQ, TIME, CHANNELS, INSTANCES) : short-time Fourier transform matrix
- → ***f*** (NFREQ, 1) : frequency vector corresponding to stftMatrix
- → ***t*** (TIME, 1) : yime vector corresponding to stftMatrix
- ← ***X*** (TIME, CHANNELS, INSTANCES) : matrix of time series data
- ← ***nWin*** (integer) : short-time window length
- ← ***nOvlap*** (integer) : number of overlapping window samples
- ← ***nFFT*** (integer) : number of FFT frequency bins
- ← ***Fs*** (scalar) : sampling frequency in Hz

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.134 structCell2Mat_shm.m File Reference

Active Sensing: Structure-cell mixture to matrix.

Call Methods

combinedMatrix = structCell2Mat_shm (structCell)

[Combined Matrix] = Combine Structure-Cell (Structure-Cell)

Function Description

Takes a structure-cell mixture of matrices, each with M rows, and column-wise combines them together into an M x N x ... matrix, where N is the combined total number of matrix columns in the mixture and so forth.

Parameters

→ ***combinedMatrix*** (DATAROWS, DATACOLUMNS, ...) : combined matrix

← ***structCell*** (...) : a mixture of cells and structures containing matrices with equal row lengths

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.135 sumMultDims_shm.m File Reference

Active Sensing: Sum along multple dimensions.

Call Methods

dataSum = sumMultDims_shm (data, dimensions)

[Data Sum] = Sum Multple Dimensions (Data, Dimensions)

Function Description

Matrix of data is simultaneously summed along multiple dimensions. The summation results in ndims(dataSum)=ndims(data)-length(dimensions).

Parameters

→ ***dataSum*** (...) : data summed along specified dimensions and squeezed, ndims = ndims(data)-length(dimensions)

← ***data*** (...) : data matrix with arbitrary dimensions

← ***dimensions*** (DIMENSIONS,1) : dimensions to sum across

Author

Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.136 thresholdScores_shm.m File Reference

Feature Classification: Threshold a set of scores.

Call Methods

[results, scores, thresholds] = thresholdScores_shm (scores, thresholds, thresholdType, flipThresholdSign, flipOutputSigns)

[Results, Scores, Thresholds] = Threshold Scores (Scores, Thresholds, Threshold Type, Flip Threshold Sign, Flip Output Signs)

Function Description

Threshold a set of scores using 1 or 2 sided threshold.

Parameters

→ ***results*** (INSTANCES, 1) : vector of 0, and 1. A '1' in the ith position indicates that the ith observation is flagged as damaged as its score lies outside the thresholds

→ ***scores*** (INSTANCES, 1): vector of scores for each instance with appropriate signs

→ ***thresholds*** (scalar) : thresholds used for flagging outliers with appropriate sign

← ***scores*** (INSTANCES, 1) : Vector of scores from scoring process corresponding to each of the measured instances. Scores are essentialy the reduced single dimension decision variable that is the thresholded to determine the detected state.

← ***thresholds*** (THRESHOLDS, 1) : vector of thresholds corresponding to each of the possible states where THRESHOLDS is the ordered set [low, high], for a single-sided threshold just enter a scalar; if a scalar is entered for a double-sided threshold, the thresholds will be set at plus or minus the scalar value

← ***thresholdType*** (integer) : type of thresholding procedure to use, options are 0 for single-sided below, 1 for single-sided above, or 2 for double-sided

← ***flipThresholdSign*** (logical) : whether or not to flip the sign of the thresholds before thresholding, set to true if the sign of scores and thresholds do not match

← ***flipOutputSigns*** (logical) : whether or not to flip the sign of the scores and thresholds outputs

Author

Dustin Harvey
Eric Flynn
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.137 timeSyncAvg_shm.m File Reference

Feature Extraction: Time-synchronous average of angularly sampled signals.

Call Methods

[xTSAMatrix] = timeSyncAvg_shm (xARSMatrix, samplesPerRev)

[Time Synchronous Average Signal Matrix] = Time Synchronous Average (Angular Resampled Signal Matrix, Samples per Revolution)

Function Description

This function takes a matrix angular resampled signals and averages each cycle of angular rotation to a single synchronous averaged signal of one revolution for each signal in the input signal matrix. This method is used in an attempt to remove noise from the signals and extract periodic elements. Alternatively it can be repeated and removed from the Angular resampled signal it was generated from signal in an attempt to remove the periodic components from a signal.

Parameters

→ ***xTSAMatrix*** (SAMPLESPERREV, CHANNELS, INSTANCES) : matrix of time synchronous averaged signals

← ***xARSMatrix*** (SAMPLES, CHANNELS, INSTANCES) : matrix of angular resampled signals, SAMPLES = samplesPerRev∗REVOLUTIONS

← ***samplesPerRev*** (integer) : integer valued samples per revolution

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`synchronous averaged signals`

2.138 trainOutlierDetector_shm.m File Reference

Outlier Detection: Train outlier detector.

Call Methods

models = trainOutlierDetector_shm (Xgood, k, confidence, modelFileName, distForScores)

Models = Train Outlier Detector (Training Features, Number of Clusters, Confidence, Model File Name, Distribution Type)

Function Description

Train a semi-parametric model by learning a gaussian over the cells of a partition, then use this model to learn a threshold for outlier detection. Also train a confidence model that will serve to give confidence values between 0-1 to future predictions.

Parameters

→ ***models*** (struct): density models for use by detectOutlier_shm

← ***Xgood*** (INSTANCES, FEATURES) : training features

← ***k*** (scalar) : number of gaussian components to learn, default is 1.

← ***confidence*** (scalar): between 0-1, threshold is picked at this percentile over the training data

← ***modelFileName*** (string): file name to save the model under

← ***distForScores*** (string): specify a distribution-type. This is used to model the actual distribution of the log-likelihoods over the training data. The learned distribution of likelihoods is then used to pick a threshold at the specified confidence value for future detections.This should be one of the distributions supported by the mle function. By default, if no distribution is specified, then the threshold is picked directly at the confidence percentile over the training data with no distributional assumption. Also, if no distribution is specified, a confidence model (i.e. distribution-over-scores used to assess the confidence of future predictions) is learned as a mixture of k Gaussians, k specified above. This confidence model is used by the detectOutlier_shm routine to assess its confidence over detections.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

2.139 triangleKernel_shm.m File Reference

Outlier Detection: Kernel weights for the triangular kernel.

Call Methods

W = triangleKernel_shm (Delta)

Weights = Triangular Kernel Weights (Evaluation Points)

Function Description

Estimate the triangular kernel at points in Delta. In one dimension, this is given as K(u) = (1-|u|)_+ where u = (x -x_i)/h. In higher dimensions we just take the product of the one dimensional kernel at each coordinate.

Parameters

→ ***W*** (WEIGHTS, 1) : evaluated kernel weights

← ***Delta*** (INSTANCES, FEATURES) : each row is (x-x_i)/bandwidth where the kernel is centered at x, and x_i are the training points.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`http://en.wikipedia.org/wiki/Kernel_(statistics)`

Sample Data Set

`none`

2.140 triweightKernel_shm.m File Reference

Outlier Detection: Kernel weights for the triweight kernel.

Call Methods

W = triweightKernel_shm (Delta)

Weights = Triweight Kernel Weights (Evaluation Points)

Function Description

Estimate the triweight kernel at points in Delta.In one dimension, this is given as K(u) = (35/32).(1-u^2)^3_+ where u = (x -x_i)/h. In higher dimensions we just take the product of the one dimensional kernel at each coordinate.

Parameters

→ ***W*** (WEIGHTS, 1) : evaluated kernel weights

← ***Delta*** (INSTANCES, FEATURES) : each row is (x-x_i)/bandwidth where the kernel is centered at x, and x_i are the training points.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`http://en.wikipedia.org/wiki/Kernel_(statistics)`

Sample Data Set

`none`

2.141 uniformKernel_shm.m File Reference

Outlier Detection: Kernel weights for the uniform kernel.

Call Methods

W = uniformKernel_shm (Delta)

Weights = Uniform Kernel Weights (Evaluation Points)

Function Description

Estimate the uniform kernel at points in Delta.In one dimension, this is given as K(u) = sign(1-u) where u = (x -x_i)/h. In higher dimensions we just take the product of the one dimensional kernel at each coordinate.

Parameters

→ ***W*** (WEIGHTS, 1) : evaluated kernel weights

← ***Delta*** (INSTANCES, FEATURES) : each row is (x-x_i)/bandwidth where the kernel is centered at x, and x_i are the training points.

Author

Samory Kpotufe
LA-CC-14-046

Modifications

1. none

Reference

`http://en.wikipedia.org/wiki/Kernel_(statistics)`

Sample Data Set

`none`

2.142 wavelet_shm.m File Reference

Feature Extraction: Generate wavelet of specified type.

Call Methods

[wavelet] = wavelet_shm (waveType, waveParam)

[Wavelet] = Wavelet Generator (Wavelet Type, Wavelet Parameters)

Function Description

Generates a specified wavelet of type 'Morlet', 'Shannon' or 'Bspline'. Wavelet parameters allow user to specify the size of the wavelet desired as well as its central frequency. Lastly the third value of the wavelet parameter is a Boolean flag to specify real or complex wavelets.

Parameters

→ ***wavelet*** (2∗nWavelet+1, 1): symmetric wavelet

← ***waveType*** (string): wavelet type specification, either 'Morlet', 'Shannon', or 'Bspline'

← ***waveParam*** (3, 1) : wavelet parameter vector [Fc, Nw, useComplex] FC (scalar) = wavelet central frequency Nw (integer) = half the wavelet length omitting the central element useComplex (logical) = complex(1) or real(0) valued wavelets

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

`none`

Sample Data Set

`none`

2.143 window_shm.m File Reference

Feature Extraction: Generate window vector of specified type.

Call Methods

[win] = window_shm (winType, winParam, winForm)

[Window Column Vector] = Window Generator (Window Type, Number of Samples in Window, Window Form)

Function Description

Generates window functions used for various DSP functionality including spectral analysis and windowed filtering. The user must specify a window type (listed below) and a window length. It is optional for a user to specify a window format. If winForm is left empty [], then by default a symmetric window form is selected. Alternatively, the user may specify a periodic window form where a N+1 length symmetric window is generated and the samples [1:N] are returned. 'SYMMETRIC' Should be used for filter design and 'PERIODIC' is useful for DFT/FFT and spectral analysis.

Parameters

→ ***win*** (nWindow, 1) : window vector

← ***winType*** (string) : window type specification 'bartlett' - Bartlett window. 'barthannwin' - Modified Bartlett-Hanning window. 'blackman' - Blackman window. 'blackmanharris' - Minimum 4-term Blackman-Harris window. 'blackmannuttall' - Blackman-Nuttall window. 'bohmanwin' - Bohman window. 'chebwin' - Chebychev-Dolph window. 'cosine' - Cosine Window. 'flattopwin' - Flat Top window. 'gausswin' - Gaussian window. 'hamming' - Hamming window. 'hann' - Hanning window. 'kaiser' - Kaiser window. 'lanczos' - Lanczos window. 'nuttallwin' - Nuttall defined minimum 4-term Blackman-Harris window. 'parzenwin' - Parzen (de la Valle-Poussin) window. 'rectwin' - Rectangular window. 'tukeywin' - Tukey window. 'triang' - Triangular window.

← ***winParam*** (scalar) : winParam specifies the window parameters. For most window types, winParam is an integer window length. In the case of a Kaiser or Chebyshev-Dolph window, it is possible to assign an additional parameter as [NWINDOW,ADDPARAM]. In the case of the Kaiser window, ADDPARAM is the shape factor beta (default beta = 0.5). In the case of the Chebyshev-Dolph window, ADDPARAM is the desired relative sidelobe attenuation (dB) (default value 100(dB)).

← ***winForm*** (string): window function form specification, either 'symmetric' - symmetric window function or 'periodic' - periodic window function

Author

Luke Robinson
LA-CC-14-046

Modifications

1. none

Reference

none

Sample Data Set

none

Index

