Data-driven Characterization of Spruce Tonewood Plates
Documentation of the Repository

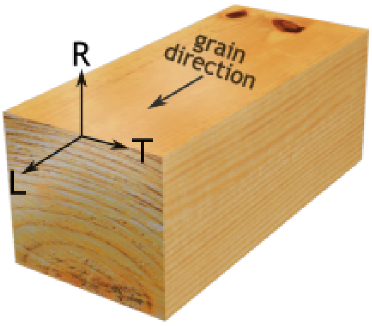
David Giuseppe Badiane

Contents

1 Introduction

This documentation describes the code related to *FRF2Params*, a novel neural-network based method to estimate the mechanical parameters, i.e. the elastic properties, of thin spruce tonewood plates. The method is called *FRF2Params* as it is necessary to measure a frequency response function (FRF) evaluated for prescribed points of the plate to estimate its mechanical parameters. Additionally, the method requires the knowledge of the plate geometry (length, width and thickness) and density.

Following a orthotropic linear elastic model, the elastic behaviour of spruce can be modeled with nine mechanical parameters, 3 Young's moduli $\{E_L, E_R, E_T\}$[Pa], 3 shear moduli $\{G_{LR}, G_{RT}, G_{LT}\}$[Pa] and 3 Poisson's ratios $\{\nu_{LR}, \nu_{RT}, \nu_{LT}\}[\sim]$. Such parameters are defined with reference to the three characteristic directions highlighted in the figure beside, namely the direction parallel to the wood fibers direction (longitudinal, L, corresponding to the growth in height of the tree), the direction radial with respect to the wood growth rings (radial, R, corresponding to the growth in width of the tree) and the direction tangential to the wood growth rings (tangential, T). Our aim is to estimate those parameters.



The flow diagram of *FRF2Params*, depicted in Fig. 2, consists of the following steps:
○ **1)** define a finite element (FE) model of the plate FRF and define where that FRF must be acquired;
○ **2)** use the FE model to generate a dataset containing the eigenfrequencies of the plate and their amplitude in the FRF as the elastic properties, the density, the geometry and the damping of the plate vary;
○ **3)** the dataset is used to train two feedforward neural networks: one for frequency and one for amplitude;
○ **4)** accelerometric measurement to acquire the FRF. The FRF is computed with the H1 estimator and its peaks are identified via peak analysis;
○ **5)** the neural networks are employed in a optimization procedure to minimize the frequency distance between the peaks of the measured FRF and their predictions;
○ **6)** The results are validated by computing a FE simulation of the FRF with the material parameters set to the values obtained with *FRF2Params*.

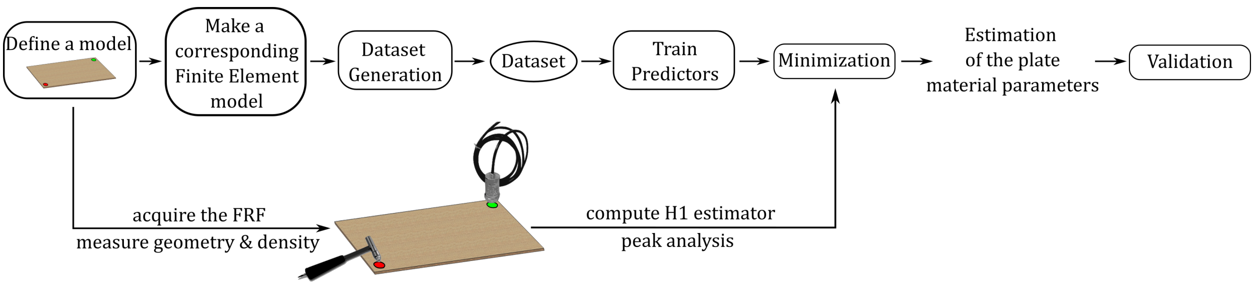


Figure 1: Flow chart of *FRF2Params*.

Here the link of the repository described by this documentation.

2 Directories and Elements

In the repository you can see three directories: "FRF2Params", "functions" and "Figures". "FRF2Params" contains the complete implementation of the method and its application to 10 book matched spruce tonewood plates. "function" contains all the functions used to implement and validate the method. While "Figures" contains the figures fhown in the readMe.md of the repository. We will describe in detail only the "FRF2Params" and "function" directories.

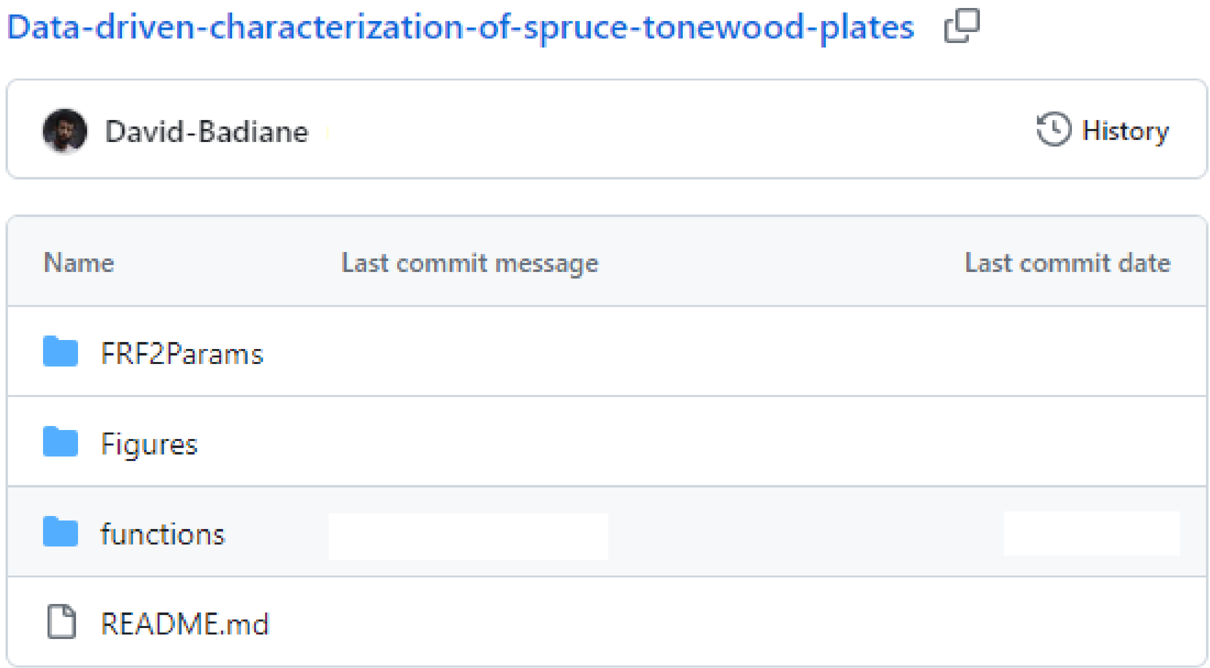


Figure 2: Snapshot of the repository described in this documentation.

3 FRF2Params Directory → FRF2Params application

FRF2Params

- main_FRF2Params.m .... This module applies *FRF2Params* to estimate the material properties of 10 spruce tonewood plates. In this program we have: dataset generation with *Comsol Multiphysics livelink for Matlab*™; FRF2Params application; validation with Comsol
- main_hyperparams.m .... This module finds the optimal number of neurons and number of layers of the feedforward neural networks used in *FRf2Params*, trains them and saves them
- main_compute_exp_FRF.m . This module computes an experimental FRF with the H_1 estimator starting from accelerometric measurements data, performs peak analysis on the estimated FRFs and saves them
- main_modesAnalysis.m .... This module performs modes analysis on the generated dataset and orders the dataset by modal shapes rather than the ascending order of the eigenfrequencies
- main_sensitivity_analysis.m This module analyzes input/output relation of the dataset with or without modes identification, saying us how sensible are the eigenfrequencies of the plate and their amplitude in the FRF to variations of the inputs of the dataset
- measFRFs.mat, FRF_data, geom_mass_measurements Measured FRFs, geometry and density of 10 book-matched thin rectangular plates of spruce tonewood
- gPlate.mph .... The Comsol Multiphysics™ FE model of the plate
- csv_gPlates .... directory with dataset, neural networks and results
  - inputs.csv, outputsAmp.csv, outputsEig.csv .. The dataset without postprocessing
  - datasetOrdered_∼.csv files .... The dataset ordered by modes, i.e. with modes identification
  - Modeshapes .. Directory where the modeshapes of each dataset tuple are saved as .csv files
  - ModesAnalysis .... Directory where modes identification data are saved
  - HyperParameters .... Directory where hyperparameters tuning data are saved
  - Neural Networks ..... Directory where the optimal architecture feedforward neural networks are saved
  - Results .. Directory where the estimated material parameters labeled per plate are saved
- paramsEstimations.xlsx .... Excel file with the material properties of 10 book-matched spruce tonewood plates estimated with *FRF2Params* along with the associated uncertanties
- referenceVals.csv .... center values of the input parameters of the dataset

The flow diagram shown in Fig. 2 is implemented by the *main_tag.m* programs and the FE model described above as shown in Fig. 3.

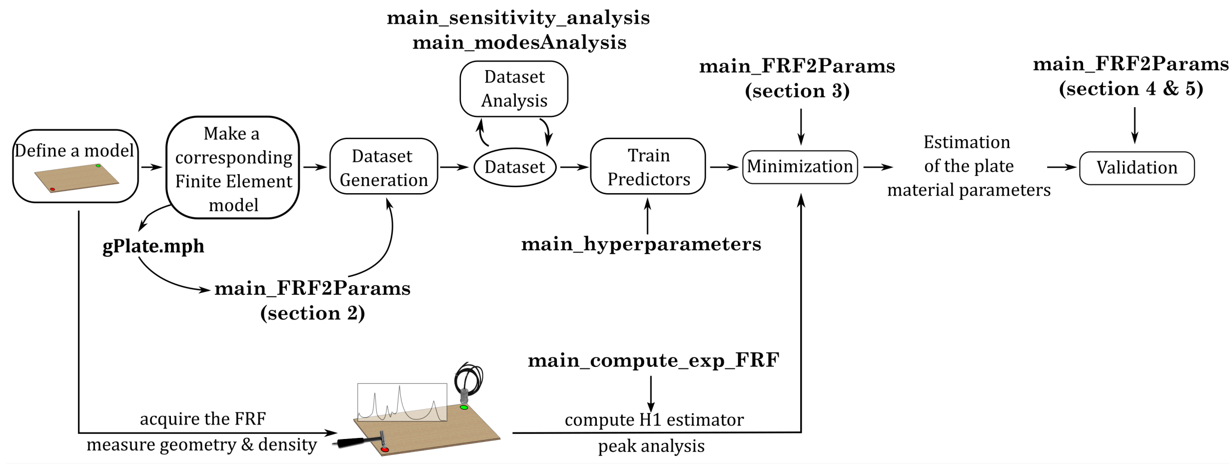


Figure 3: Practical implementation of the flow chart of *FRF2Params* shown in Fig. 2.

In the following we will provide an extended description for each element of the "FRF2Params" directory.

3.1 FRF2Params Directory Extended Description

Here you can find an extended description of each element you see in the directory "FRF2Params".

3.1.1 main_*.m files

Listing 1: `main_FRF2Params.m`

```
% APPLIES FRF2PARAMS TO ESTIMATE MATERIAL PROPERTIES OF PLATES.
% ------------------------------------------------------------------------
% tasks:
% A) Dataset generation
% B) FRF2Params application
% C) Validation with Comsol
% ------------------------------------------------------------------------
% summary:
% section 0)   initial setup, reference folders
% section 1)   retrieve measured FRFs + apply Caldersmith formulas
% section 2)   dataset generation
% section 3)   material properties estimation
% section 4)   validation on Comsol - eigenfrequency study
% section 5)   validation on Comsol - frequency domain study
% section 5.1) generate image of simulated (computed via section 5) and
%              experimental FRFs
```

This module applies *FRF2Params* on ten book matched spruce tonewood plates.

∘ **section 0) - init**: sets up the Matlab search path, declares flags and variables, reads geometry and mass measurements;
∘ **section 1) - FRFs + Caldersmith**: loads the measured FRFs from *measFRFs.mat* and applies Caldersmith formulas to later compare it with the results of FRF2Params;
∘ **section 2) - dataset generation**: generates the dataset with *Comsol Livelink for Matlab*. The dataset is stored in *csv_gPlates* with three .csv files, namely *inputs.csv*, *outputsAmp.csv* and *outputsEig.csv*.
∘ **section 3) - FRF2Params minimization**: applies FRF2Params minimization to estimate the mechanical parameters of the plates.
∘ **section 4) validation - eigenfrequencies** : uses *Comsol Livelink for Matlab* to compute the plate eigenfrequencies as its mechanical parameters are set to the estimates obtained with *FRF2Params*. The eigenfrequencies are then compared to the frequency values of the peaks of the plate FRF.
∘ **section 5) - validation FRFs, simulation**: uses *Comsol Livelink for Matlab* to compute the plate FRF over a user defined number of points as its mechanical parameters are set to the estimates obtained with *FRF2Params*.
∘ **section 5.1) - validation FRFs, postprocessing**: visually compares the simulated FRF and the experimentally acquired one with a figure. Computes metrics to assess the similarity between the two.

Listing 2: `main_compute_exp_FRF.m`

```
% COMPUTES THE EXPERIMENTAL FRFS WITH THE H1 ESTIMATOR
% STARTING FROM FORCE AND ACCELERATION MESUREMENTS
% -----------------------------------------------------------------------
% tasks:
% A) computation of the H1 estimator
% B) peak analysis on the estimated FRFs
% -----------------------------------------------------------------------
% summary:
% section 0) Preset directories, init variables
% section 1) calculation of the H1 estimator
% section 2) peak analysis and save FRFs
```

This module computes the H_1 estimator of the mobility (velocity/force) starting from force and acceleration measurements and performs peaks analysis on the estimated FRFs.

◦ **section 0) - init**: sets up the Matlab search path, declares flags and variables;
◦ **section 1) - $\mathbf{H_1}$ estimator**: computes the H_1 estimator of the mobility of the set of spruce tonewood plates, data are smoothed with an exponential filter;
◦ **section 2) - peak analysis**: performs peak analysis on the FRFs and saves the FRFs along with the frequency/amplitude pairs of their peaks in the file *measFRFs.mat*.

Listing 3: `main_hyperparameters.m`

```
% PERFORMS HYPERPARAMETERS TUNING ON FEEDFORWARD NEURAL NETWORKS
% -----------------------------------------------------------------------
% tasks:
% A) split dataset in train set and test set
% B) hyperparameters tuning
% C) optimized nns training
% -----------------------------------------------------------------------
% summary:
% section 0) initial setup, reference folders
% section 1) randomly split the dataset into train set and test set
% section 2) hyperparameters (HP) tuning
% section 3) get minimum and maximum R2 values (coefficient of ...
    determination) from HP tuning
% section 4) train optimal neural networks
% section 5) generate figures of HP tuning grid search
```

This module optimizes the topology, i.e. the number of layers and number of neurons per layer, of the feedforward neural networks employed to predict the plate eigenfrequencies and the corresponding FRF amplitude. This task is also known as hyperparameters tuning.

◦ **section 0) - init**: sets up the Matlab search path, declares flags and variables;
◦ **section 1) - split dataset into train and test sets**: randomly splits the dataset into train set and test set, saves them back in the directory *csv_gPlates/HyperParameters*;
◦ **section 2) - hyperparameters tuning**: performs hyperparameters tuning on both frequency and amplitude neural networks;
◦ **section 3) - max and min R2**: finds the best and the worst architectures for both frequency and amplitude neural networks;
◦ **section 4) - Train optimal NNs**: trains the neural networks with the optimal architecture and saves them in the directory *csv_gPlates/Neural Netorks* ;
◦ **section 5) - plot figures**: plots hyperparameters tuning data;

Listing 4: `main_modesAnalysis.m`

```
% PERFORMS MODES ANALYSIS ON A DATASET USING A SET OF REFERENCE MODE SHAPES
% ------------------------------------------------------------------------
% tasks:
% - A) resample the modeshapes of the dataset on a regular rectangular grid
% - B) compare the resampled modeshapes of the dataset with a reference set
%      of modeshapes
% - C) label modeshapes and order the dataset by modes
% ------------------------------------------------------------------------
% SUMMARY:
% section 0)   Init - preset directories, set variables, upload data [...]
% section 1)   Modeshapes Resampling
% section 2)   Compute reference set
% section 3)   Reference set resampling
% section 4)   Modeshapes Labeling --> NCC
% section 4.1) Code to modify the reference set
% section 5)   Postprocessing and outliers removal
% section 5.1) Remove poisson plates
% section 6)   See obtained modeshapes
% section 7)   Define dataset modes order
% section 8)   Generate and save ordered dataset
```

This module analyzes the modal shapes of the dataset generated with *Comsol Multiphysics livelink for Matlab*™

∘ **section 0) - init**: sets up the Matlab search path, declares flags and variables, fetches dataset;
∘ **section 1) - resample modeshapes**: resamples modeshapes from Comsol irregular grid to a regular user defined rectangular grid;
∘ **section 2) - compute reference set**: computes the reference set of modeshapes with *Comsol livelink with Matlab*™;
∘ **section 3) - resample reference modeshapes**: resamples the reference set of modeshapes from the irregular grid of Comsol to a user defined regular rectangular grid;
∘ **section 4.1) - see reference modeshapes**: plots the reference set of modeshapes;
∘ **section 5) - modes identification**: performs modes identification by computing the normalised cross correlation (NCC) between the modal shapes of each dataset tuple and the reference set. Each mode is identified with the best scoring reference mode name;
∘ **section 5) - modes identification, NCC computation**: performs modes identification by computing the normalised cross correlation (NCC) between the modal shapes of each dataset tuple and the reference set;
∘ **section 5.1) - modify reference set**: allows to modify the reference set of modal shapes;
∘ **section 6) - modes identification, postprocessing**: analyzes NCC data, labels each Labels each mode with the reference mode scoring the highest NCC, discards tules with either repeated modes or at least one mode with NCC < 0.9
∘ **section 6.1) - postprocessing**: removes Poisson plates from the dataset;
∘ **section 7) - plot modeshapes**: plots the identified modes for each dataset tuple;
∘ **section 8) - define modes order**: analyzes modes identification data to find the most common succession of the modes the frequency increases. This succession will define the order in which modes are listed in the dataset ordered by modes;
∘ **section 9) - generate and save ordered dataset**: orders the dataset by modes and saves it. The dataset columns are ordered with the ordering defined in the previous section.

Listing 5: `main_sensitivity_analysis.m`

```
% THIS PROGRAM ANALYZES THE INPUT/OUTPUT RELATION OF THE DATASET, ALLOWING
% US TO UNDERSTAND HOW MUCH THE PLATE EIGENFREQUENCIES AND CORRESPONDING
% FRF AMPLITUDES ARE SENSIBLE TO VARIATIONS OF ITS MECHANICAL PARAMETERS
% ------------------------------------------------------------------------
% tasks:
% A) Computation of the correlation between inputs and outputs of the
%    dataset, with or without modes ordering
% B) Representation of the correlation data in two images, one for
%    frequency and one for amplitude
% ------------------------------------------------------------------------
% summary:
% 0) init - set variables and paths
% 1) correlation computation and image representation
```

This module analyzes the input/output relationship of the dataset by computing the Pearson's correlation coefficient between each input and each output of the dataset. This allows us to understand how much sensible are the plate eigenfrequencies and the associate FRF amplitudes to the variation of each input parameter of the dataset.

◦ **section 0) - init**: sets up the Matlab search path, declares flags and variables;
◦ **section 1) - compute correlation and plot data**: fetches the dataset, computes the correlation between inputs and outputs and plots the correlation matrix as a chessboard plot.

3.1.2 csv_gPlates

FRF2Params

- csv_gPlates . directory with dataset, neural networks, results and modes analysis data
  - inputs.csv, outputsAmp.csv, outputsEig.csv . . . . The dataset without postprocessing. The dataset has 4500 tuples containing the first 18 plate eigenfrequencies and associated FRF amplitudes.
  - datasetOrdered_*.csv files . . . . The dataset ordered by modes, i.e. with modes identification
  - Modeshapes . . . . Directory where the modeshapes of each dataset tuple are saved
    - *modeshapes1.csv* to *modeshapes4500.csv* . . . . The modeshapes are not present in the repository. You can download them from this link. The same link is provided in the readMe of the repository
  - ModesAnalysis . . . . Directory where modes identification data are saved
    - modesNames.csv . . . . cell array of the dimensions of the outputs of the dataset (4500 x 18) with the label of the modeshape scoring the highest NCC
    - modesNCC.csv . . . . NCC value associated to modesNames.csv
    - modesOrdering.csv . . . . To read with modesNames.csv. Each cell of modesOrdering tells us how many times the corresponing mode in modesNames appears in the ascending order seen in modesNames. E.g. modesNames $= [f_{1,1}, f_{0,2} \rightarrow$ modesOrdering $= [2000, 2400]$ means $f_{1,1}$ is first 2000 times out of 4000, while $f_{0,2}$ is second 2400 times
    - modesPresence.csv says us how much times the mode is present in the dataset. Usually higher order modes are not always present in the dataset
    - reference.csv . . . . reference set of modal shapes
    - refModeNames.csv . . . . labels of the reference set of modal shapes
    - refModesOrder.csv . . . . most common ordering of modes in the dataset
  - HyperParameters . . . . Directory where hyperparameters tuning data are saved
  - Neural Networks . . . . . . Directory where the optimal architecture FFNNs are saved
    - optNN_15_Modes.mat frequency and amplitude neural networks trained on the dataset ordered by modes
    - optNN_15_Peaks.mat . frequency and amplitude neural networks trained on the dataset ordered by peaks, i.e. without postprocessing
  - Results . . . . Directory where the estimated material parameters are saved
    - Results_*_nR10_12.csv files containing the estimated mechanical parameters of each plate
    - FRF_Results_*_nR10_12beta_2e-06.csv . . . . . . simulated FRF associated to each results file

3.1.3 Other Files and Directories

FRF2Params

- geom_mass_measurements
  - FRF2Param_guitar_plates_density_error_prop.xlsx Excel file with the error propagation to estimate the uncertainty associated to the measurment of the density
  - geomParams.csv ..... measured geometry (length, width and thickness) of the set of plates
- gPlate.mph .... *Comsol Multiphysics*^{TM} FE model of the plate
- measFRFs.mat .... Matlab file with the acquired FRFs data
- paramsEstimations.xlsx ..results obtained with FRF2Params and associated uncertainties put in a single Excel file
- referenceVals.csv ....center values of the dataset inputs

4 Functions Directory → functions implementing FRF2Params

The "function" directory is structured as follows:

function
- baseline ... contains a baseline method to estimate the material properties of wooden thin plates, i.e. Caldersmith and McIntyre formulas
- comsol based . contains all *Comsol Multiphysics livelink for Matlab^{TM}* based functions
- FRF2Params minimization contains all functions employed in the minimization step of FRF2Params
- general .... contains general purpose functions
- generate figures .... contains functions to generate figures
- metrics ... contains functions implementing all the metrics computed in FRF2Params
- modeshapes analysis . contains all functions that perform modes identification on the dataset
- NN .... contains all functions used to train and test feedforward neural networks
- signal processing ...... contains all signal processing algorithms used in FRF2Params

In the following we will provide the documentation of each function, ordered per subdirectory.

4.1 Baseline

function
- baseline .... contains Caldersmith and MCIntyre formulas
  - caldersmith_formulas.m . computes elastic constants of the plate with Caldersmith formulas

Listing 6: caldersmith_formulas.m

```
caldersmith_formulas(f0,rho, geom, poissonRatio)
% ------------------------------------------------------------------------
  Computes the elastic constants of a plate with  caldersmith formulas
% ------------------------------------------------------------------------
% INPUTS:
%   f0    = 3x1 double - characteristic eigenfrequencies of the plate
%                      f0(1) = f11;  f0(2) = f02;  f0(3) = f20;
%   rho   = 1x1 double density of the plate
%   geom = 3x1 double - geometry of the plate
%           geom(1) = Length; geom(2) = Width; geom(3) = Thickness;
%   poissonRatio = 1x1 double between 0 and 1
% ------------------------------------------------------------------------
% OUTPUTS:
%   mechParams = 1x3 double --> [EL, ER, GLR]
%      mechParams(1) = longitudinal Young's modulus (EL);
%      mechParams(2) = radial Young's modulus (ER);
%      mechParams(3) = longitudinal to radial Shear modulus (GLR);
%   normParams = 1x3 double --> mechParams/EL
```

This function computes the mechanical parameters of a plate with Caldersmith formulas.

4.2 Comsol Based

function
- comsol based . contains all *Comsol Multiphysics livelink for Matlab^{TM}* based functions
  - comsol_Point_FRF.m . . . computes the plate FRF in Comsol for user defined input parameters
  - comsolRoutineFA_plate . . . . generates the dataset - randomly samples the input parameters, calculates the plate eigenfrequencies and corresponding FRF amplitudes, stores data
  - exportAllModesFromDataset.m . . . exports all the modes of a user defined comsol dataset
  - exportData.m . . . . exports data from a user defined comsol dataset
  - setParams.m . . . . sets the parameters of the comsol model to user defined values

Listing 7: `comsol_point_FRF.m`

```
comsol_point_FRF(model, resultsPath, resultsFilename, meshSize, fAx,
                 fBounds, nPointsAxis, dampingParams, dampingParams_idxs,
                 comsolDset, paramsIdxsModel, expression)
% -------------------------------------------------------------------------
  Simulates a point FRF on Comsol with Comsol livelink for Matlab
% -------------------------------------------------------------------------
% INPUTS:
% model             = comsol finite element model
% resultsPath       = string - path to the results directory
% resultsFilename   = string - name of the results file
% meshSize          = 1x1 double - mesh size
%                                 (9=extremely coarse / 1=extremely fine)
% fAx               = 1xfreqBins double - freq. axis of the measured FRFs
% fHigh             = 1x1 double - high bound for frequency axis
% nPointsAxis       = 1x1 double - number of points of the frequency
%                                 axis in the comsol simulation
% dampingParams       = 1x2 double - Rayleigh damping control variables
% dampingParams_idxs = 1x2 double - Rayleigh damping control variables
%                                   indexes as parameters of the comsol
%                                   model
% comsolDset = 1xnPoints cell - datasets from which comsol evaluates the
%                               FRF. Usually = {'cpt1'}, which evaluates
%                               the FRF for single point of the plate.
%                               e.g. 'cpt2' can be also set and it is
%                               another point (n.b. must be defined in
%                               the comsol model)
% paramsIdxsModel   = 1xM double - array with the indexes of the
%                         parameters to update in the model (for this
%              application the parameters are set in the model as follows
%              1:15 --> [density, mechParams(9), damping(2), geometry(3)])
% expression        = 1x1 cell - expression to compute, to have a FRF set
%                               it to {'solid.ut_Z'} or {'solid.vel'}
% -------------------------------------------------------------------------
% OUTPUTS:
% simulated_FRFs    = 1xnPointsAxis double - array of the FRF
% fAxisComsol = 1xnPointsAxis double - frequency axis associated to the FRF
```

This function simulates a point FRF in Comsol for user defined input parameters

Listing 8: `comsolRoutineFA_plate.m`

```
comsolRoutineFA_plate(model, nSim, nModes, dataset_center_values,
                      inputParamsNames, standardDev, mshapesPath,
                      datasetPath, writeNow, samplingMethod)
% -------------------------------------------------------------------------
  Function for dataset generation - randomly sample inputs, calculate
  eigenfrequencies and FRF amplitude at eigenfrequencies
% -------------------------------------------------------------------------
% INPUTS:
%   model            = comsol finite element model
%   nSim             = 1x1 double - number of simulations
%   nModes           = 1x1 double - number of modes calculated
%   dataset_center_values    = 1x15 double - nominal values of the dataset
%                              [density(1), elastic constants(9),
%                               Rayleigh constants (2), geometry (3)]
%   inputParamsNames = 1x15 cell array - names of the inputs of the dataset
%   standardDev      = 1x15 double - std of the dataset inputs
%   mshapesPath      = string - directory where modeshapes are stored
%   datasetPath      = string - directory where dataset is stored
%   writeNow         = boolean
%                       [true]  --> dataset computation starts from zero
%                       [false] --> dataset computation starts from the data
%                                   that already are in its directory
%   samplingMethod   = string - specifies the distribution of the dataset
%                               inputs --> 'uniform' or 'gaussian'
% -------------------------------------------------------------------------
% OUTPUTS:
%   Dataset_FA       = struct containing the raw Dataset fields --> inputs
%                               outputsEig, outputsAmp
```

This function generates the dataset - randomly samples the input parameters, calculates the plate eigenfrequencies and corresponding FRF amplitudes, stores data.

Listing 9: `exportAllModesFromDataset.m`

```
exportAllModesFromDataset(model,fileName,dirName, expression)
% -------------------------------------------------------------------------
  Exports all the modes of a comsol dataset
% -------------------------------------------------------------------------
% INPUTS:
%   model      = comsol model
%   fileName   = string - name of the without .txt
%   dirName    = string - name of the directory containing the file
%   expression = cell array - strings containing the expression of the
%                physical quantity exported - ex. solid.disp or solid.vel
%                (displacement and velocity)
% -------------------------------------------------------------------------
% OUTPUTS:
% -
% -------------------------------------------------------------------------
```

This function exports all the modes of a user defined Comsol dataset, e.g. "*surf1*"

Listing 10: `exportData.m`

```
exportData(model,dset, dirName, filename,dataExpression)
% ------------------------------------------------------------------------
  Exports data from the model
% ------------------------------------------------------------------------
% INPUTS:
%   model (Comsol model) = Comsol model from which export data
%   dset      (string)    = Dataset from which export (ex. 'dset1')
%   dirName   (string)    = Address of the directory where we export data
%   filename (string)     = Name of the file to export (ex. "vel")
%   dataExpression (cell array) = name of the data to export
%                                 (ex.{'solid.disp'})
% ------------------------------------------------------------------------
% OUTPUTS:
%   -
%-------------------------------------------------------------------------
```

This function exports user defined data from the Comsol model.

Listing 11: `setParams.m`

```
setParams(model,paramsNames, paramsVals)
% -----------------------------------------------------------------------
  This function set the params corresponding to *paramsNames* in the comsol
  model *model* to the values *paramsVals*
% ------------------------------------------------------------------------
% INPUTS
%   model = comsol.mph model
%   paramsNams = (cell array) cell array with the names of the parameters
%                             to set
%   paramsVals = array - values of the parameters
% ------------------------------------------------------------------------
% OUTPUTS:
% -
```

This function sets the parameters of the Comsol model from Matlab.

4.3 FRF2Params Minimization

functions
- FRF2Params minimization . . contains all functions employed in the minimization step
  - FRF2Params.m . . . calls the minimization procedure with user defined convergence criteria
  - error_FA.m . . . objective function of the minimization called in FRF2Params.m - predicts frequency and amplitude of FRF peaks for given material parameters and computes the loss function
  - lossFx_FA.m . computes the loss function of the minimization procedure. The loss function is defined in the frequency/amplitude (FA) space

Listing 12: `FRF2Params.m`

```
FRF2Params(options, fNet, aNet, f0, fAmps, NpeaksAxis, plotData,
           fixParamsVals, fixParamsIdxs,inputParsStart)
% -------------------------------------------------------------------------
  Calls the minimization procedure with @error_FA  set as objective
  function of the minimization
% -------------------------------------------------------------------------
% INPUTS:
%   options        = struct with options for minimization algorithm
%   fNet           = neural network object - neural network predicting
%                                            eigenfrequencies
%   aNet           = neural network object - neural network predicting
%                                            amplitudes
%   f0             = nPts x 1 cell - in each cell we have the frequencies
%                                    of the peaks of a single FRF
%   fAmps          = nPts x 1 cell - in each cell we have the amplitudes
%                                    of the peaks of a single FRF
%   rho            = 1x1 double - density of the plate
%   NpeaksAxis     = nPeaks x 1 double - axis with the FRF peaks considered
%                                        in the minimization
%   plotData       = boolean to decide whether to plot or not
%   fixParamsVals  = values for the material properties that are not
%                    optimized (at least density and geometry)
%   fixParamsIdxs  = indexes of the fixed params in the mechParams array
%   inputParsStart = first guess of material properties
% -------------------------------------------------------------------------
% OUTPUTS:
%   L2           = 1x1 double - loss function value
%   maps         = real FRF - estimation associations
```

This function calls the minimization procedure with objective function *error_FA.m* and user defined convergence criteria, number of peaks of the FRF taken into account, fixed parameters and starting point of the minimization (first guess).

Listing 13: errorFA.m

```
% errorFA(mechParams, fNet, aNet, f0, fAmps, NpeaksAxis, plotData,
%         fixParamsVals, fixParamsIdxs, nFRFs, figN)
% -----------------------------------------------------------------------
Objective function of the minimization - predicts frequency and amplitude
of FRF peaks for given material properties and computes the loss ...
    function;
also allows to evaluate multiple FRFs computed on multiple points
% -----------------------------------------------------------------------
% INPUTS:
%   mechParams    = array with mech params to be optimized (len ≤ 15)
%   fNet          = neural network object - neural network predicting
%                   eigenfrequencies
%   aNet          = neural network object - neural network predicting
%                   amplitudes
%   f0            = nPts x 1 cell - in each cell we have the frequencies
%                   of the peaks of a single FRF
%   fAmps         = nPts x 1 cell - in each cell we have the amplitudes of
%                   the peaks of a single FRF
%   rho           = 1x1 double - density of the plate
%   NpeaksAxis    = nPeaks x 1 double - axis with the FRF peaks considered
%                   in the minimization
%   plotData      = boolean to decide whether to plot or not
%   fixParamsVals = values for the material properties that are not
%                   optimized. (usually, density and geometry)
%   fixParamsIdxs = indexes of the fixed params in the mechParams array
%   nPts          = 1x1 double - number of point FRFs considered
%   figN          = 1x1 double - figure number
% -----------------------------------------------------------------------
% OUTPUTS:
%   L2                = 1x1 double - loss function value
%   mode_matching_map = mode matching btw FRF peaks and NNs
%                       eigenfrequencies
```

This function is the objective function of the minimization called in FRF2Params.m - predicts frequency and amplitude of FRF peaks for given material parameters and computes the loss function calling *lossFx_FA.m*. The loss function can optionally be computed on multiple points if the dataset is evaluated on multiple points.

Listing 14: `lossFx_FA.m`

```
lossFx_FA(fEst, aEst, fReal, aReal, NpeaksAxis, plotData, figN)
% ---------------------------------------------------------------------
  Computes the loss function of the minimization.
  Loss Fx is computed in 4 steps in the frequency/amplitude (FA) space:
  1) normalization btw R = (fReal, aReal) and E = (fEst, aEst)
  2) euclidean FA space distance btw each R and all E
  3) for each R select closest E
  4) Loss Fx = relative frequency difference btw R and closest E plus
               cost functional to avoid unwanted situations (read code)
  This because not all eigenfrequencies of the plate correspond to
  peaks in the FRF, we need to discard the "antiresonances"
% ---------------------------------------------------------------------
% INPUTS:
%   fEst = 12x1 double - frequencies estimated by fNetwork
%   aEst = 12x1 double - amplitudes estimated by aNetwork
%   fReal = nPeaks x 1 double - frequencies of the FRF peaks
%   aReal = nPeaks x 1 double - amplitudes of the FRF peaks
%   NpeaksAxis = 12x1 double  - axis with the number of peaks considered
%                               in the minimization
%   plotData = boolean        - says whether to plot images or not
%   figN     = 1x1 double     - number of the plotted figure
% ---------------------------------------------------------------------
% OUTPUTS:
%   L2   = 1x1  double - value of the loss function
%   map = 12x1 double - index of the prediction associated to a given
%         FRF peak. Some eigenfrequencies do not correspond to peaks
%         in the FRF --> discard them.
%         (ex. a map [1 2 4 3 5 6 8 7 10 11 12] indicates the index of
%              the NN eigenfrequencies associated to
%              [1 2 3 4 5 6 7 8 9 10 11 12] FRF peak.
```

This function computes the loss function of the minimization procedure. The loss function is defined in the frequency/amplitude (FA) space and is computed in four steps:

1) normalization between the frequency amplitude pairs characterizing the FRF (R) and the estimations of the neural networks (E);
2) computation of the euclidean frequency/amplitude distance between all combinations of R and E;
3) for each R select the ”closest” E,
4) loss function is the relative frequency difference between R and closest E plus a cost functional to avoid that two frequency amplitude pairs in R are associated to the same pair of E.

This strategy is chosen because not all eigenfrequencies correspond to peaks in the FRF. Since the eigenfrequencies not corresponding to peaks will be ”far” from the measured FRF peaks in the frequency/amplitude space, this is a suited algorithm to discard them.

4.4 General

function
- general .... contains general purpose functions
  - fetchDataset.m fetches the dataset with a user defined number of eigenfrequencies
  - readTuples.m .... reads a matrix from a .txt file
  - writeMat2File.m .... writes a matrix as a user defined file (e.g. .csv or .txt)

Listing 15: `fetchDataset.m`

```
fetchDataset(baseFolder, modesGet, getOrdered, csvName, saveData)
% ---------------------------------------------------------------------------
  Fetches the dataset with the number of columns specified by modesGet,
  if getOrdered = True --> fetches dataset ordered by modes, otherwise
  non processed one
% ---------------------------------------------------------------------------
% INPUTS:
%   baseFolder    = string  - directory of the basic folder - ex.
%                   FRF2Params/gPlates
%   modesGet      = 1x1 int - set how many modes are retrieved in the
%                             dataset
%   getOrdered    = boolean - if true get the dataset ordered by modes
%                             otherwise, get non processed (raw) dataset
%   csvName       = string  - name of the directory containing the dataset
% ---------------------------------------------------------------------------
% outputs:
%   Dataset_FA    = struct - contains the dataset
%   datasetPath   = string - complete path to dataset directory
%   HPFolder      = string - complete path to hyperparameters tuning folder
```

This function fetches the dataset with a user defined number of eigenfrequencies. The dataset can be either not postprocessed (raw, getOrdered = False) or ordered by modes (getOrdered = True).

Listing 16: `readTuples.m`

```
  readTuples(filename, rows, transpose)
% ---------------------------------------------------------------------------
  Allows to retrieve a nVals*nCols matrix from a .txt file
  nCols depends from the length of the file,
  When you call it be sure to be in the same directory where
  the source file is present.
%   -------------------------------------------------------------------------
% INPUTS:
%   filename  = string - name of the file to read without tag
%   rows      = int - number of rows to retrieve
%   transpose = bool - select wheter to transpose the matrix or not
% ---------------------------------------------------------------------------
% OUTPUTS:
%   fileData = read data in form of a matrix
```

This function reads a matrix from a .txt file.

Listing 17: `writeMat2File.m`

```
  writeMat2File(data,dstFileName, variablesName, nCols, singleTitles)
% -----------------------------------------------------------------------
  Custom function to write a .csv file with given variable names
  (columns names)
% -----------------------------------------------------------------------
% INPUTS:
%   data          = (nTuples x nCols) double - data to write in the file
%   dstFileName   = string - filename of the file to be written
%                             (contains tag ex. .csv)
%   variablesName = cell array - cell array with the labels of the columns
%   |---------------------------------------------------------------------|
%   | if singleTitles == 0 - variablesName has less entries than nCols ...
      |
%   | ex. variableName = {'f' 'g'} & singleTitles == 0 --> cols names ...
    are |
%   |     yields as columns names 'f1' 'g1' 'f2' 'g2' ..... and so on ...
       |
%   | if singleTitles == 1 - variableName must contain nCols entries ...
        |
%   |---------------------------------------------------------------------|
%   nCols         = int - number of columns of the data
%   singleTitles  = boolean - look at variableName description
% -----------------------------------------------------------------------
% OUTPUTS:
%   dataTable     = table - written data in the form of a table
```

This function writes a matrix as a .csv file with user defined columns and rows names.

4.5 Generate Figures

function
- generate figures .... contains functions to generate figures
  - defImg_comparison_FRFs.m ...... wraps the parameters of a comparison between experimental and simulated FRFs in two structs
  - defImg_matrix.m ... wraps the parameters of a chessboard representation of given entries of a matrix in a single struct
  - export_comparison_FRFs.m .... plots a figure comparing two FRFs
  - export_matrix.m plots a chessboard representation of a user defined submatrix of a matrix
  - minimization_finalFigures.m .... plots the figures generated at the end of a minimization procedure

Listing 18: `defImg_comparison_FRFs.m`

```
defImg_comparison_FRFs( xLengthImg, yLengthImg, imgN, xLabel, yLabel,
                            areaColor, axFontSize, areaAlpha, legenda,
                            lineWidth, cutHigh, cutLow, FRF, fAxis,
                            fAxisComsol, comsol_FRF,alpha, beta,
                            nRealizations, xyScale)
% -------------------------------------------------------------------------
  Defines the parameters of an image comparing experimental and simulated
  FRFs abd wraps them in two structs: ImgData and FRFData
% -------------------------------------------------------------------------
% INPUTS:
%  Imgdata entries:
%   xLengthImg = (float) - x length of the figure
%   yLengthImg = (float) - y length of the figure
%   imgN       = (int) - figure number
%   xyLabels   = (cell, len = 2) - cell with xLabel and yLabel strings
%   areaColor  = (1DArray, len = 3) - color of the area between the FRFs
%   areaAlpha  = (float) - transpacerncy of the area color, in [0,1]
%   axFontSize = (int) - font size of the axis
%   legenda    = (string) - legend of the figure
%   lineWidth  = (float) - width of the FRF line
%   xyscale    = (cell, len = 2) - {xScale, yScale} can be 'log' or ...
'linear'
%  FRFData entries:
%   cutHigh       = (float) - low bound of the frequency axis
%   cutLow        = (float) - high bound of the frequencye axis
%   FRF           = (1DArray) - measured FRF
%   fAxis         = (1DArray) - axis of the measured FRF
%   fAxisComsol   = (1DArray) - axis of simulated FRF
%   comsol_FRF    = (1DArray) - FRF simulated with Comsol Multiphysics
%   alpha         = (float) - Rayleigh damping parameter alpha value
%   beta          = (float) - Rayleigh damping parameter beta value
%                             n.b. you can annotate them in the figure
%   nRealizations = (int) - number of realizations with which the results
%                            of FRF2Params are computed
% ------------------------------------------------------------------------
% imgData : struct wrapping the parameters of the image
% FRFData : struct wrapping the data of experimental and simulated FRFs
```

This function wraps the parameters of a comparison between experimental and simulated FRFs in two structs.

Listing 19: `defImg_matrix.m`

```
defImg_matrix(xIdxs, yIdxs, xLengthImg, yLengthImg, imgN, xLabel,
              yLabel, colorMap, textFontSize, axFontSize,
              xTickLabels, yTickLabels, cbarLabel, displayCbar)
% ---------------------------------------------------------------------
  Defines the parameters of a chessboard plot of a matrix and wraps ...
     them in a single struct.
% ---------------------------------------------------------------------
% INPUTS:
%  xIdxs        = int array - indexes to plot in the x axis
%  yIdxs        = int array - indexes to plot in the y axis
%                 n.b. xIdxs & yIdxs allow to select a submatrix of the
%                 main matrix
%  xLength      = (int) img window xLength
%  yLength      = (int) img window yLength
%  imgN         = (int) img number
%  xLabel       = (string) x label string
%  yLabel       = (string) y label string
%  colorMap     = (double RGB matrix) ex. winter
%  textFontSize = (int) size of text inside matrix
%  axFontSize   = (int) size of axis text (x y labels, legends, etc)
%  xTyckLabels  = (cell) labels of the x ticks ex. {'E_L' 'E_R'}
%  yTickLabels  = (cell) labels of the y ticks ex. {'f1' 'f2' 'f3'}
%  cbarLabel    = (string) label of the co
%  displayCbar  = (bool) selects whether to display the colorbar
% ---------------------------------------------------------------------
% OUTPUTS:
%  imgData      = (struct) - struct with members equal to the inputs of ...
     the fx
```

This function wraps the parameters of chessboard plot in a single struct called *imgData*.

Listing 20: `export_comparison_FRFs.m`

```
export_comparison_FRFs(FRFData, imgData, imgMode, minPeakWidth,
                        doStem, Xref, Yref, subplotN)
% ---------------------------------------------------------------------------
 Plots a figure comparing the two FRFs, the area between the two FRFs is
 shaded. Computes the Frequency Response Assurcance Criterion between the
 FRFs, optionally annotates it in the image.
% ---------------------------------------------------------------------------
% INPUTS:
% FRFData = struct with the data of experimental and simulated FRFs
% imgData = struct with the data of the figure to be generated
% imgMode      = string - type 'db' to plot in dB, otherwise in linear
%                          magnitude
% minPeakWidth = 1x1 double - minimum peak width for findpeaks algorithm
% doStem       = boolean    - select whether to highlight peaks to stems
% Xref         = 1x1 double - position of the figure in X axis
% Yref         = 1x1 double - position of the figure in Y axis
% subplotN     = 1x1 double - number of the subplot if plotting in subplot
% ---------------------------------------------------------------------------
% OUTPUTS:
% img          = matlab figure
% FRAC         = 1x1 double - evaluation of the FRAC
% cut frequency axis
```

This function plots a figure comparing two FRFs and computes the Frequency Response Assurance Criterion (FRAC) metric between the two to assess their similarity.

Listing 21: `export_matrix.m`

```
export_matrixix(matrix,imgData, roundN, plotPercentage)
% -----------------------------------------------------------------------
  Generates a chessboard representation of a matrix the entries of the
  matrix are specified as text inside each square. Allows to choose a
  submatrix of the data and plot the data of in percentage their
  maximum
% -----------------------------------------------------------------------
% INPUTS:
%   matrix         = (nTuples x nCols) double - matrix to represent
%   imgData        = struct - struct with the various figure settings
%   roundN         = int - number of significant digits represented as
%                          text in the figure
%   plotPercentage = boolean - selects whether to represent the actual
%                              matrix values (plotPercentage == 0)
%                              or to plot them in percentage of the sum
%                              of the matrix row (plotPercentage == 1)
% -----------------------------------------------------------------------
% OUTPUTS:
% img              = matlab figure
```

This function plots a chessboard representation of a matrix. The entries of the matrix are written in each chess box.

Listing 22: `minimization_finalFigures.m`

```
minimization_finalFigures(f0, fNet, xpar,NpeaksAxis, plotData)
% ------------------------------------------------------------------------
  Generates a figure to see how the estimation is
  accurate under the frequency profile
% ------------------------------------------------------------------------
% INPUTS:
%   f0          = nPeaksx1 double - frequency of the peaks of one FRF
%   fNet        = neural network object - neural network for prediction of
%                 eigenfrequencies
%   xpar        = double - predicted parameters
%   NpeaksAxis = double - axis with FRF peaks considered
%   plotData    = boolean - whether or not to plot a image
% ------------------------------------------------------------------------
% OUTPUTS:
%   freqs       = predicted eigenfrequencies
```

This function plots the final figure for a single minimization run.

4.6 Metrics

function
- metrics ... contains functions implementing all the metrics computed in FRF2Params
  - computeR2.m .... computes the coefficient of determination (R2) between to observations
  - NCC.m .... computes the normalized cross correlation (NCC) between two signals
  - NMSE.m computes the normalized mean square error (NMSE) between two signals

Listing 23: `computeR2.m`

```
computeR2(observedData,predictedData)
% -------------------------------------------------------------------------
  Computes the coefficient (R2) of determination between observed data and
  predicted data.
  ------------------------- short explanation -----------------------------
    The R2 says us how much of the standard deviation of the observed data
    is represented in the predicted data, yielding a metric to measure the
                      accuracy of a predicting model.
% -------------------------------------------------------------------------
% INPUTS:
%   observedData  = nTuples x nCols double - array with observed data
%   predictedData = nTuples x nCols double - array with data predicted by
%                                            a regressor/predictor
% -------------------------------------------------------------------------
% OUTPUTS:
%   R2 = 1xnCols double - coefficient of determination, in [0,1]
```

This function computes the coefficient of determination (R^2) between two signals. Typically such signals are the observed data, $\hat{y}$, i.e. real data, and the data predicted by a predictor, y. The R^2 says us how much of the standard deviation of the observed data is represented in the predicted data and is thus a measure of the accuracy of a predicting model.

Listing 24: `NCC.m`

```
NCC(x,y)
% -------------------------------------------------------------------------
  Computes the normalized cross correlation function between two signals
  ------------------------- short explanation -----------------------------
       measures the degree of linear similarity between two signals
% -------------------------------------------------------------------------
% INPUTS:
%   x  (array) = signal, in our case simulated signal
%   y (array)  = signal, in our case measured signal
% -------------------------------------------------------------------------
% OUTPUTS:
%   NCC (float) = normalized cross correlation coefficient, in [0,1]
```

This function computes the normalised cross correlation (NCC) between two signals. The NCC is a metric $\in [0, 1]$ that assess the degree of linear similarity between two signals.

Listing 25: `NMSE.m`

```
NMSE(x,y)
% -------------------------------------------------------------------------
Calculates the normalised mean square error between two signals
  ------------------------ short explanation ----------------------------
    computes how different are two signals, ||x - y||^2 / ||x||^2
            similarly to a relative error, but with the L2 norm
% -------------------------------------------------------------------------
% INPUTS:
% x = signal, in our case measured signal
% y = signal, in our case simulated signal
% -------------------------------------------------------------------------
% NMSE = normalized mean squared error (NMSE) between x and y
```

This function computes the normalised mean squared error (NMSE) between two signals.

4.7 Modeshapes Analysis

function
- modeshapes analysis ..... contains all functions that analyze the dataset modeshapes
  - modeshapes_compute_reference_set.m ...... this function computes and saves the modeshapes for the dataset center values. This set of modes will be referred to as reference set
  - modeshapes_labeling.m .... this function labels the modeshapes of the dataset by comparing them with the reference set.
  - modeshapes_resampling.m resamples the modal shapes from Comsol irregular grid to a regular rectangular grid
  - modesOrderAnalysis.m ...... says us how many times a mode appears at a given position
  - saveRef.m .... saves the reference set to a .csv file
  - seeReferenceModes.m .... plots the reference set

Listing 26: modeshapes_compute_reference_set.m

```
modeshapes_compute_reference_set(model, nModesRef, csvPath, %
                                reference_parameters,...
                                reference_parameters_names, isWedge)
% --------------------------------------------------------------------------
  Computes and saves the modeshapes for user defined input parameters
  (material, damping, geometry)
% --------------------------------------------------------------------------
% INPUTS:
%   model                       = comsol model of gPlate.comsol
%   nModesRef                   = int - number of modes to compute
%   csvPath                     = string - path of the dataset directory
%   reference_parameters        = double array - input parameters values
%   reference_parameters_names  = cell array   - input parameters names
%   isWedge                     = bool - wether we analyze a plate or
%                                        wedge dataset
% --------------------------------------------------------------------------
% OUTPUTS:
%   refFilename = string - filename of a file with the generated reference
%                           modeshapes
% --------------------------------------------------------------------------
```

This function computes and saves the reference set of modal shapes with Comsol Multiphysics.

Listing 27: `modeshapes_labeling.m`

```
modeshapes_labeling(pX,pY, pxFFT, pyFFT, nModes, nTuples, plotFigures,
                    ref, refModesNames, compareType, printData)
% -------------------------------------------------------------------------
  Labels the modeshapes of the dataset by comparing
  them with the reference set.
% -------------------------------------------------------------------------
- n.b. this function works both with space domain modeshapes and
       Fourier space domain modeshapes
% -------------------------------------------------------------------------
% INPUTS:
%   pX      = int - number of points on the x axis of the rectangular
%             grid of modeshapes
%   pY      = int - number of points on the y axis of the rectangular
%             grid of modeshapes
%   pxFFT   = int - number of points on the x axis of the rectangular
%             grid of modeshapes in Fourier domain
%   pyFFT   = int - number of points on the y axis of the rectangular
%             grid of modeshapes in Fourier domain
%   nModes  = int - number of modeshapes to be labeled
%   nTuples = int - number of labeled dataset tuples
%   plotFigures   = boolean - select whether to plot figures while
%             labeling (way slower)
%   ref           = double array - reference set of modeshapes
%   refModesNames = cell array   - labels of the reference set of
%                   modeshapes
%   compareType   = string - can be "disp" or "fourier",
%                   selects the type of modeshapes to
%                   compare (fourier or displacement)
%   printData     = boolean - decide to print some messages while
%                   labeling (little bit slower)
% -------------------------------------------------------------------------
% OUTPUTS:
%   modesNames = cell array - labels of all the modeshapes of the dataset
%   modesNCC   = NCC scores associated to each label
```

This function labels the modeshapes of the dataset by comparing them with a reference set of modeshapes. For each dataset tuple it computes the Normalised Cross Correlation (NCC) and labels the given mode of the dataset with the label of the reference set modeshape scoring the highest NCC value.

Listing 28: `modeshapes_resampling.m`

```
%  modeshapes_resampling(pX,pY,nCols, nTuples, resampledFolder,
                             plotData)
% -----------------------------------------------------------------------
  Resamples modeshapes from Comsol irregular grid to a regular
  rectangular grid
% -----------------------------------------------------------------------
% INPUTS:
%   pX                 = int - number of points along the x axis of the
%                              rectangular grid
%   pY                 = int - number of points along the y axis of the
%                              rectangular grid
%   nCols              = int - number of dataset columns taken into ...
   account (how many eigenfrequencies)
%   nTuples            = int - number of dataset tuples
%   resampledFolder    = string - path where to save resampled modeshapes
%   plotData           = boolean - select wheter to plot figures while
%                                   resampling (slower)
% -----------------------------------------------------------------------
% OUTPUTS:
% ¬
```

This function resamples the modeshapes of the dataset generated via Comsol to a regular rectangular grid of points. This process is necessary because the grid from Comsol is irregular and variable along the dataset.

Listing 29: `modesOrderAnalysis.m`

```
modesOrderAnalysis(nModes, csvPath, modesFilename, nonOutliers)
% -----------------------------------------------------------------------
  Says us how many times a mode appears in a given ascending order.
  In other words, it says us how much times ex. f11 will be the first
  or second eigenfrequency in the dataset generated from comsol.
% -----------------------------------------------------------------------
% INPUTS:
%   nModes = int - number of modes taken into account;
%   csvPath = string - path of the dataset directory
%   modesFilename = string - fileName of the modeshapes file
%   nonOutliers    = boolean array - says us which tuples of the raw
%                    dataset are not outliers
%                    (repeated labels - NCC < 0.9 - Poisson plates)
% -----------------------------------------------------------------------
% OUTPUTS:
%   appears = int array - map saying us how much times a given mode
%                         appears on a given dataset column (i.e. how
%                         much f11 is the 1st peak or 2nd peak)
%   maxLoc   = int - the index of the modes Ordering most frequent in
%              the dataset
```

This function says us how many time a mode appears in a given ascending order along the dataset. Indeed, as the material properties of the plate change, also the order in which modes appear in the FRF changes. For example, in the dataset we have most of the time $f_{1,1}$ at the first peak of the FRF, but for given values of the material parameters, it becomes the second.

Listing 30: `saveRef.m`

```
saveRef(ref, refModesNames, modesAnalysisPath, isSum)
% ---------------------------------------------------------------------------
  Saves the reference set of modes
% ---------------------------------------------------------------------------
% INPUTS:
%  ref                = (nPts x nRefModes) 2DArray - reference set of
%                       modeshapes
%  refModesNames      = cell - labels (e.g. f_02) associated to the
%                       reference set
%  modesAnalysisPath = string - path to the modes analysis directory of
%                       csv_gPlates
%  isSum <--- deprecated, ignore it, the function works with three
%             parameters
% ---------------------------------------------------------------------------
% OUTPUTS:
% ¬
```

This function saves back the reference set of modeshapes.

Listing 31: `seeReferenceModes.m`

```
seeReferenceModes(modesAnalysisPath,pX, pY,subplotnRows, subplotnCols ...
    ,showFourier)
% ---------------------------------------------------------------------------
  Generates a figure of the reference modeshapes
% ---------------------------------------------------------------------------
% INPUTS:
%   modesAnalysisPath = string - path of the modesAnalysis directory
%   pX                = int - number of points on the x axis
%   pY                = int - number of points on the y axis
%   subplotnRows      = int - subplot of the figure, total number of
%                             rows
%   subplotnCols      = int - subplot of the figure, total number of
%                             columns
%   showFourier       = boolean - to show reference set in space
%                                 or space Fourier domain
% ---------------------------------------------------------------------------
% OUTPUTS:
% ¬
```

This function plots the reference set of modeshapes along with their labels, i.e. the name of the correspoonding eigenfrequency.

4.8 NN

- function
  - NN .... contains all functions used to train and test feedforward neural networks
    - NN_hyperparameters.m . performs hyperparameters tuning for both frequency and amplitude neural networks
    - NN_trainTest_extTest.m creates a feedforward neural network, trains it and tests it with external test set

Listing 32: `NN_hyperparameters.m`

```
NN_hyperparameters(nNeuronsVec, nLayersVec, nLaxis_f, nLaxis_a,
                   nModesGet, baseFolder, csvName, flags,
                   sets_filename, nRealizations, getOrdered)
% -------------------------------------------------------------------------
  Performs hyperparameters tuning (topology optimization) on both the
  frequency and amplitude feedforward neural networks
% -------------------------------------------------------------------------
% INPUTS:
%   nNeuronsVec    = double - array with the number of neurons of the
%                    grid search performed to tune the HPs of the NNs
%   nLayersVec     = double - array with the number of layers of the
%                    grid search performed to tune the HPs of the NNs
%   nLaxis_f       = double - array with the number of layers of the ...
    frequency NN
%   nLaxis_a       = double - array with the number of layers of the ...
    amplitude NN
%   nModesGet      = int - number of modes on which NNs are trained
%   baseFolder     = string - path of the base working directory
%   csvName        = string - name of the directory containing the dataset
%   flags          = array of 4 booleans
%                          a)           b)     c)      d)
%                    [writeNewFiles doFreq doAmp saveData]
%         a) write new hyperparameters tuning csv files or
%            load previous ones
%         b) perform hyperparameters tuning for NN that
%            predicts eigenfrequencies
%         c) perform hyperparameters tuning for NN that
%            predicts amplitudes
%         d) save all data from HP tuning
%   sets_filename  = string - filename of the csv file containing the
%                    results of HP tuning
%   nRealizations  = int - n times the grid search is carried on
%   getOrdered     = boolean - get sets ordered by modes or by the
%                    ascending value in Hz of the eigenfrequencies
% -------------------------------------------------------------------------
% OUTPUTS:
%   HPData = cell array containing two structs:
%            HPData_freq or HPData_amp, both have the members:
%              HP    --> matrix with the double averaged R2
%                         (coefficient of determination)
%                        averaged over all modes and over the
%                        number of realizations
%              nets  --> cell - trained neural networks
%              R2    --> R2 of each neural network
% -------------------------------------------------------------------------
```

This function performs hyperparameters tuning for both the frequency and amplitude neural networks. In particular, it optimizes the topology, i.e. the number of layers L and number of neurons per layer N, of the neural networks. It performs a grid search varying L and N evaluating the average coefficient of determination of the prediction. This process is carried for *nRealizations* times and then averaged once again. The best architecture is the one with highest score.

Listing 33: `NN_trainTest_extEst.m`

```
NN_trainTest_extTest(trainIn, trainOut, testIn, testOut, nNeurons,
                     nLayers, strTitle, plotData, figNumber)
% -------------------------------------------------------------------------
  Creates and trains a Multilayer Feedforward Neural Network (MFNN),
  evaluates coefficient of determination and saves training data
% -------------------------------------------------------------------------
% INPUTS:
%   trainIn   = nTuples train x nInputs double  - train set inputs
%   trainOut  = nTuples train x nOutputs double - train set outputs
%   testIn    = nTuples test x nInputs double   - test set inputs
%   testOut   = nTuples test x nOutputs double  - test set inputs
%   nNeurons  = int - number of neurons of the MFNN
%   nLayers   = int - number of layers of the MFNN
%   strTitle  = string  - title of a figure with a scatter plot of test ...
    results
%   plotData  = boolean - select whether to plot the scatter plot or not
%   figNumber = int - number of the plotted figure
% -------------------------------------------------------------------------
% OUTPUTS:
%   R2_NN     = coefficient of determination of the neural network ...
    (testing)
%   net       = trained MFNN
%   tr        = training data
% -------------------------------------------------------------------------
```

This function trains and tests a feedforward neural network with user defined topology.

4.9 Signal Processing

function
- signal processing ...... contains all signal processing algorithms used in FRF2Params
  - computeH1.m . computes the H_1 estimator of the mobility (velocity / force) from force and acceleration measurements
  - FFT.m .... computes the FFT of a signal

Listing 34: `computeH1.m`

```
computeH1(force, acceleration, nfft, window, Fs, lowHighcut, algorithm)
% -------------------------------------------------------------------------
  Computes the H1 estimator of the mobility (velocity/force)
  from force and acceleration measurements
% -------------------------------------------------------------------------
% INPUTS:
%   exc    = excitation              - force vector or matrix    (double)
%   resp   = response, acceleration  - response vector or matrix (double)
%   nfft   = number of fft points    - (int)
%   window = tapering window for fft - use hamm(floor(nfft/n))   (double)
%   Fs     = sampling frequency [Hz] -                           (int)
%   lowHighCut = 2x1 array - low and high cut in frequency of the H1
%                            estimator [lowCut, highCut]
%   algorithm  = (string)
%              = 'single' - to compute H1 for single vectors set to,
%              = 'multiple' averaging over multiple acquisitions (matrix)
% -------------------------------------------------------------------------
% outputs:
%   H1     = array - H1 estimator of the mobility
%   coh    = array - coherence associated to the H1 estimator
%   fAxis  = array - frequency axis associated to the H1 estimator
```

This function computes the H_1 estimator of the mobility (velocity / force) starting from force and acceleration measurements. The user can choose to compute the H_1 estimator for a single acquisition or to average over multiple acquisitions of the same FRF.

Listing 35: `FFT.m`

```
FFT(timeSignal)
% -------------------------------------------------------------------------
  Computes the DFT of a time series with the FFT algorithm. Returns the
  single sided spectrum.
-------------------------------------------------------------------------
% INPUTS:
%   timeSignal      (1DArray) = input time signal to transform
-------------------------------------------------------------------------
% OUTPUTS:
%   frequencySignal (1DArray) = spectrum of the signal
```

This function computes the DFT of a time series with the FFT algorithm and returns the single sided spectrum of the time series.