ABRAVIBE
A MATLAB/Octave toolbox for Noise and Vibration Analysis and Teaching
Revision 2.0

Anders Brandt
Department of Technology and Innovation
University of Southern Denmark
abra@iti.sdu.dk

May 13, 2018

Contents

Chapter 1

Introduction

1.1 Contents of the Toolbox

This document presents some important features of the ABRAVIBE toolbox. This toolbox is intended as an accompanying toolbox for the book [1], to help understanding noise and vibration analysis as well as providing a powerful tool for those who need to teach students the topic of mechanical vibrations. Due to its powerful architecture you can also use it as an analysis tool. Please note, however, that **the toolbox is published 'AS IS', with no warranty whatsoever of correct performance**. See the license file for more information.

The ABRAVIBE toolbox can be downloaded by following the instructions at the book website at `http://www.abravibe.com`. Then follow the instructions in Section 2.2 of this manual to install the toolbox. **Note! After installing the toolbox, you should look in the directory `AbraVibe_Toolbox\ChapterExamples` in which you will find a number of examples illustrating each chapter of the book [1]!**.

On the website there will also be published a number of example scripts which illustrate many of the topics in the book and other useful resources for learning about vibrations and vibration analysis. These examples are divided into subdirectories for each chapter of the book.

The ABRAVIBE toolbox was written in MATLAB [1]. If you prefer free software, however, it has also been tested in GNU Octave, available from `http://www.octave.org`. There are, however, some important exceptions due to limitations in GNU Octave compared to MATLAB, see Section 2.7. This makes the use of the toolbox in Octave more and more limited, as ABRAVIBE develops. Particularly this includes some powerful graphical

[1]MATLAB is a registered trademark by The MathWorks Inc., www.mathworks.com

user interface (GUI) based features. I therefore no longer encourage the use of ABRAVIBE with Octave, but leave it up to you to decide.

The toolbox comprises a comprehensive set of functions for generation of test data, and for data analysis. This essentially means it includes functionality similar to an advanced multi-channel measurement and analysis system for noise and vibration analysis. In addition, the powerful method for generating forced response data of a mechanical system (described in Section 6.5 in [1]) is very attractive for teaching vibration analysis, as it gives you a tool to create simulated data for virtually any mechanical system. It is also important to validate a data analysis method you develop. The toolbox can thus be used for analyzing synthetically generated data as well as real, measured data, and it can cope with an unlimited number of measurement channels, much thanks to the included data storage formats (see Chapter 3).

The toolbox includes functions for noise and vibration analysis such as

- A data browser GUI (MATLAB only) which allows you to import universal files (format 58) and browse and plot data
- Time domain simulation of mechanical systems, useful for creating experimental data from known mechanical systems (either knowing the mass, damping and stiffness matrices, or from a modal model), by applying known forces to the structure model
- Synthesizing frequency responses of mechanical systems (either knowing the mass, damping and stiffness matrices, or from a modal model)
- Time series integration, differentiation, statistics analysis, and $1/n$-octave analysis
- Spectrum analysis by Welchs method and the smoothed periodogram method, time windows, etc.
- Frequency response estimation for single-input as well as multiple-input cases, including a GUI based software (if run in MATLAB) for impact testing, see Section 4.1 for the latter method
- Principal component and virtual coherence analysis, for noise source identification etc.
- Rotating machinery analysis, including synchronous resampling order tracking
- Operating deflection shapes analysis

- Advanced functions for experimental modal analysis, EMA, parameter extraction
- One function (similar to covariance-driven SSI) for operational modal analysis, OMA, parameter extraction
- Universal file import/export of record types 58 (time data), and 15, 55, and 82 for modal data
- Animation [2], also see Section 4.4

1.2 License

The toolbox, **except the `animate` command**, is published under the *GNU General Public License* (GNU GPL). For details of the legal information, see the file gpl_license_ver3_2007.txt. For details of copyright and license information for the `animate` command, please see inside the file **animation.html** in the **help** directory under the **animate** folder, or under the `animate -- Help -- Animation Help` inside the `animate` GUI.

As laid out by the license file, you can essentially use the toolbox for any purpose, commercial or noncommercial, as long as you keep the original toolbox file information, and, if you make any changes or additions, as long as you let your own software based on ABRAVIBE be published under GNU GPL. You can modify the files as you please, as long as you keep the original copyright information and clearly mark your changes. Good practice is to store modified files under new file names, to avoid confusion with the original.

If you find use of this toolbox outside the immediate scope (i.e. teaching yourself or others vibration analysis), I will be glad if you let me know by writing an email to abra@iti.sdu.dk. Also, if you use this toolbox for any scientific work that you publish, I'd really appreciate if you reference it by referring to this document [2].

1.3 What's New

The current revision, 2.0, is a major new version. It includes several new features, and particularly two new GUI based features. A number of bug

[2]The ANIMATE software was written by people at The Structural Dynamics Research Lab, SDRL, University of Cincinnati, and is provided by permission. Great thanks to Dave Brown for allowing me to use this software.

fixes have also been made. The most important new functionality is the GUI based impact testing software, and extended EMA and OMA support.

New or updated commands:

```
impactGui    Impact testing GUI (MATLAB only)
sdiagramGUI  GUI based stabilization diagram for
             EMA and OMA
ir2ptime     Time domain MDOF methods for EMA parameter
             extraction also, this function has a new
             option 'MMITD'
ir2pmitd     Modified Ibrahim time domain method for
             OMA parameter extraction
pv2modal     Convert separate variables to animate MODAL
             struct
modal2pv     Convert animate MODAL struct to separate
             variables
axcorrp      Scaled cross-correlation between x (input)
             and y (output)
time2corr    Process ABRAVIBE time data files into
             correlation function files
time2spec    Process ABRAVIBE time data files into
             spectra files
winsmooth    Smooth data with any smoothing window
acomac       Coordinate modal assurance criterion
noctfreqs    Compute the exact midband frequencies for
             a 1/n octave analysis
enhancefrf   Enhance FRF matrix by Singular Value
             Decomposition, SVD
wincomp      Compensate poles due to exponential
             window from impact testing
```

Changed commands:

```
frf2ptime    New EstType 'mmitd' implemented for modified
             multi-reference Ibrahim time domain parameter
             estimation. Also, the time domain execution is
             moved to the new, external command ir2ptime
timeint      Now includes an option 'FFT'
```

Commands no longer supported (although still available, but not in contents.m):

```
adaqhelp
agetdata
arecdata
apreview
impacthelp
impproc
impsetup
```

All these commands are removed due to changes in hardware support (for the data acquisition commands), or because they are obsolete once the ImpactGui is available. The files for these commands are, however, still available, for backward compatibility. Also, if you are using Octave, you can still use the files for impact testing.

1.3.1 Notes to GNU Octave Users

Unfortunately, I do not have the resources to test ABRAVIBE against new GNU Octave releases. If you find something not working, please send me an email, and I will do my best to make it work (unless it is due to an unsupported functionality in Octave). GUIs will only work in MATLAB.

Chapter 2

Getting Started

2.1 Prerequisites

The toolbox has been developed and tested in Windows, but it should most likely work also under Linux. The instructions below assume you are using Windows.

The requirements to run the toolbox is depending on whether you run it under MATLAB or under GNU Octave

- MATLAB (R14 or higher)

  Signal Processing Toolbox

  Data Aquisition Toolbox (*only if you want to do data acquisition*)

- GNU Octave (Note: I do no longer have the necessary time to test under GNU Octave. Version 3.4.3 of Octave is the latest release that was tested, and with the following packages)

  audio

  control

  struct

  miscellaneous package

  optim package

  signal package

  specfun package

Please see the documentation of MATLAB or GNU Octave for information on how to obtain and install those toolboxes/packages.

2.2 Installation Procedure

Installation of the ABRAVIBE toolbox is easy. Follow the steps below.

First, unpack the zip file containing the ABRAVIBE toolbox to a suitable location, with write permission to the directories. This gives you a directory structure

```
abravibe_toolbox
    AbraVibe
        abratest
        Docs
            html
    animate
    ChapterExamples
ImpactGui
```

The directory **AbraVibe** contains the actual m-files comprising the ABRAVIBE toolbox. In the **Docs** folder, there is a copy of the present manual, and the subdirectory html. The latter is an automatically generated directory which contains HTML files generated by the free toolbox M2HTML, available at MATLAB Central (see www.mathworks.com). To use the HTML files, open the file index.html with your favorite web browser.

To continue the installation process, paths need to be added to MATLAB or GNU Octave, depending on which software you are using. Follow one of the next subsections to continue.

2.2.1 Semi-Automatic Installation Procedure (MATLAB)

The most convenient way is to position MATLAB in the directory abravibe_toolbox. Then, execute the command

```
abrainst
```

This will add paths etc. and produce a working installation of the ABRAVIBE toolbox.

2.2.2 Manual Installation Procedure for MATLAB

To manually install the toolbox in MATLAB, execute the following steps

- Start MATLAB and go to **Set Path** and add paths to the subdirectories **AbraVibe**, **ImpactGui**, and **animate**

- Press **Save** and close the window

2.2.3 Manual Installation Procedure for GNU Octave

To manually install the toolbox in GNU Octave, execute the following steps

- In Octave, go to the directory $ABRAVIBEDIR and execute the matlab command

  ```
  abrainst
  ```

- Start a file browser and locate the file **octaverc** in the startup folder of the Octave installation. Open the file with, for example, WordPad, and add the following lines at the end of the file

  ```
  pkg load all
  addpath \$ABRAVIBEDIR\AbraVibe
  ```

  where $ABRAVIBEDIR is the full path to the directory where you installed the ABRAVIBE toolbox.

- Open a file browser and go to the directory **AbraVibe** (where all m-files are located). Copy the file contents.m to a new file named abravibe.m. This will ensure that the command 'help abravibe' will work also in Octave.

Especially note that you should not include paths to the subdirectories **abratest** and **abradocs**.

2.3 Testing That Everything Works

After installation of the toolbox following the procedure in the previous section, you can test that your installation is correct by typing the following at the prompt in either MATLAB or Octave

```
help abravibe
```

which should give you a formatted list of all functions in the ABRAVIBE toolbox.

Also see Section 2.4 for more information on how to find help for different functions and to find the documentation files for the toolbox.

2.3.1 Software Quality Checks

The toolbox has some included test functionality which can be run to ensure some of the functions operate consistently. The intention is to allow some checks that files that relate to each other are not changed so that erroneous results occur. Although there is no guarantee that all functions in the toolbox work properly, the available quality checks should hopefully help improve the likelihood that the toolbox commands perform as wished.

The check can be achieved by changing the working directory to $ABRAVIBEDIR\abravibe\abratest and there issue the command `abracheck`. This command will start a series of programs located in the same subdirectory, which checks various toolbox functions. The program will list documentation to the screen and report errors. You can optionally edit the file **abracheck.m** to output data to a file.

2.4 Getting Help

2.4.1 Documentation Files

Under the subdirectory $ABRAVIBEDIR\abravibe\Docs, the present document is available in PDF format in the file **AbravibeManual.pdf**. If you want easy access to this file, put a shortcut to the file on your desktop.

To get help on a particular command, type for example

```
help apsdw
```

which gives a description of the syntax together with an explanation of all input and output parameters.

By opening an m-file, for example **apsdw.m** you can often get further information by looking at the reference section, which is found just below the copyright lines. Many of the commands have a reference to the appropriate section of the book [1] where the theory of the command in question is located.

Further help can be found by reading through the example files found at the book website. These files are well documented to ensure they are informatory.

2.4.2 Bug Reports

No software is, unfortunately, completely free from bugs. Should you find a bug in the ABRAVIBE toolbox, I appreciate if you send and email to me at abra@iti.sdu.dk. I have limited ability to help with other, particularly application specific, problems. However, if you think you have a problem which is of general interest to other users of the toolbox, do not hesitate to contact me at the same email address.

2.5 Chapter Examples

Make sure you do not miss the many example scripts illustrating most of the concepts from each chapter of the book [1]! These examples are located under the installation directory of the toolbox, in the subdirectory `ChapterExamples`.

2.6 For New Users to MATLAB

If you are new to MATLAB, there is a vast amount of tutorials on Internet. The MathWorks Inc. have a large amount of getting started information both on their website, `http://www.mathworks.com`, and in the MATLAB Help function. If you use your favorite search engine to search for 'matlab tutorial' you will also find a large variety of documents. In the rest of this document I will assume you have some familiarity with basic MATLAB programming.

2.7 Functionality Not Available in GNU Octave

Due to restrictions in GNU Octave, the GUIs (graphical user interfaces) only work in MATLAB. If you have any idea of how to make wireframe animation work in Octave, you are welcome to send me an email at abra@iti.sdu.dk.

2.8 Finding Your Way Around ABRAVIBE

The easiest way to start using the toolbox is to go through the examples provided on the book website `http://www.abravibe.com` under the menu 'Learn More!'. Another way is to use the command list provided by typing

```
help abravibe
```

at the prompt in MATLAB or Octave. This list is sorted in groups of different topics, and should be easy to follow. For the more specialized functionality such as importing universal files or animating a wireframe model, see Sections 3.4 and 4.4, respectively. You should also regularly look at the website under 'Learn More!' since I sometimes publish videos describing certain parts of ABRAVIBE functionality. Furthermore, I have a YouTube channel called 'Anders Brandt ABRAVIBE'.

Chapter 3

Data Storage

To allow storage of noise and vibration data from real measurements, the following proposed file formats defined for measurement data can be used. The described formats here can easily be extended for your personal use. The proposed file format consists of three different types of files:

- Storage of general measurement data, in 'standard' MATLAB '*.mat' files, as defined in Section 3.1,
- A special data storage format for impact test time data, in files with extension '*.imptime', see Section 3.2, and
- A storage format for ODS and modal test data (either EMA or OMA) with geometry and mode shape information, see Section 3.3. These files have an extension '.mod'.

3.1 Measurement Data Format

There is some functionality for using experimental data with header information in the toolbox. This functionality is based on a particular file format which has some general characteristics.

- Data are stored in 'standard' MATLAB data files with extension '.mat'.
- Each file consists of data for *one channel of measurement data*. It is implemented in this way to allow for maximum amount of time data without producing out of memory results.
- Each file consists of *two or three variables*: **Data**, which for a normal file is a column vector containing the data, and **Header**, which for a

normal file is a structure containing the different header variables as fields (see Section 3.1.1), and optionally, **xAxis**, if the x axis does not have equidistant spacing.

- Data which belong together, for example from one measurement, should be stored in a common directory, in files with a simple naming convention consisting of a prefix and a number, for example the files abra1.mat, abra2.mat, ... In this way it is easy to make a loop for creating the file names and process each channel separately.

Data imported using the universal file interface, see Section 3.4, are automatically stored in this file format, with the header fields listed in Section 3.1.1. If you write your own file import routine, it is a good idea to store the data in the ABRAVIBE format, as it allows, for example, to use the data browser to scroll through and plot your data (provided you are using MATLAB).

3.1.1 Header Variables

The header information for ABRAVIBE consists of some mandatory header fields, and many more optional fields. It is also easy to add your own fields by just adding to the list. It is proposed that if you add your own fields, you make them subfields to a new field user, to make sure there is no conflict with possible new fields in future versions of the ABRAVIBE toolbox.

The mandatory fields for time data, and for any other one-channel functions such as spectra and correlation functions are listed in Table 3.1.

Additional mandatory fields have to be added for two-channel functions, such as frequency responses, where the function depends on a reference and a response measurement (typically being response/reference). In Table 3.2, these additional fields are listed.

Spectral densities should have some additional fields according to Table 3.3, to document how they were estimated.

In addition to the mandatory fields, particularly if you import data from a universal file, many more fields are added in the header. These optional fields are listed in Table 3.4. The fields are defined in the universal file format, see for example `http://sdrl.uc.edu`.

Table 3.4: Optional fields for a measurement (or analysis) file

Field Name	Type	Comment
Title2	String	Extra header line

Table 3.4 – continued from previous page

Field Name	Type	Comment
Date	String	Date and time
Title3	String	Extra header line
Title4	String	Extra header line
FuncId	Integer	Function identifier
SeqNo	Integer	Sequence number (usually channel number)
LoadCase	Integer	usually 0 for single-input, 1 for multiple-input data
RespId	String	Extra header line
Dof	Integer	Point number (of response if two-channel function)
Dir	String	Direction string
RefId	String	Point identifier (of reference for two-channel function)
RefDof	Integer	Point number (of reference for two-channel function)
RefDir	String	Direction string
OrdDataType	Integer	Ordinate data type
NoValues	Integer	Number of data values
xSpacing	Integer	x axis type; 0 if special x axis stored, 1 if only start value and increment
xStart	Float	First x axis value
xIncrement	Float	x axis increment
zValue	Float	Z axis value
xSpecDataType	Integer	x axis specific data type
xLUnitsExp	Integer	x axis length units exponent
xFUnitsExp	Integer	x axis force units exponent
xTUnitsExp	Integer	x axis time units exponent
xLabel	String	x axis label
xUnit	String	x axis unit label (typically 's' or 'Hz')
OrdSpecDataType	Integer	y axis (numerator) specific data type
OrdLUnitsExp	Integer	y axis (numerator) length units exponent
OrdFUnitsExp	Integer	y axis (numerator) force units exponent
OrdTUnitsExp	Integer	y axis (numerator) time units exponent
Label	String	y axis (numerator) label
Unit	String	y axis (numerator) unit label
DenSpecDataType	Integer	y axis (denominator) specific data type

Table 3.4 – continued from previous page

Field Name	Type	Comment
DenLUnitsExp	Integer	y axis (denominator) length units exponent
DenFUnitsExp	Integer	y axis (denominator) force units exponent
DenTUnitsExp	Integer	y axis (denominator) time units exponent
RefLabel	String	y axis (denominator) label
RefUnit	String	y axis (denominator) units label
zSpecDataType	Integer	z axis specific data type
zLUnitsExp	Integer	z axis length units exponent
zFUnitsExp	Integer	z axis force units exponent
zTUnitsExp	Integer	z axis time units exponent
zLabel	String	z axis label
zUnit	String	z axis units label

3.1.2 Function Types

The mandatory field `FunctionType` is an integer specifying the function in the variable `Data`, as defined by the universal file format. The meaning of the values of this variable are shown in Table 3.5.

Table 3.5: Meaning of values of the variable `FunctionType`

Field Value	Meaning
0	General or Unknown
1	Time Response
2	Auto Spectrum
3	Cross Spectrum
4	Frequency Response Function
5	Transmissibility
6	Coherence
7	Auto Correlation
8	Cross Correlation
9	Power Spectral Density (PSD)
10	Energy Spectral Density (ESD)
11	Probability Density Function
12	Spectrum

Table 3.5 – continued from previous page

Field Value	Meaning
13	Cumulative Frequency Distribution
14	Peaks Valley
15	Stress/Cycles
16	Strain/Cycles
17	Orbit
18	Mode Indicator Function
19	Force Pattern
20	Partial Power
21	Partial Coherence
22	Eigenvalue
23	Eigenvector
24	Shock Response Spectrum
25	Finite Impulse Response Filter
26	Multiple Coherence
27	Order Function
28	Phase Compensation

3.2 Impact Test Data Format

For impact testing with the time data processing suggested in Chapter 13 in [1], there is a special file format. This format is similar to the 'standard' format, but with the exceptions that

- The file extension should be **'.imptime'**
- The **Data** variable is a cell array with the force signal in the first cell, and then the time data of each response sensor (accelerometer) in subsequent cells. Each cell has to contain data in a column
- The **Header** variable is a structure with as many records as there are channels, and where each record contains the header variables for the corresponding cell in the **Data** cell array.

Let us look at an example for impact testing. Assume we have made an impact test with three accelerometers, and with measurements in 20 points where the impact hammer has been exciting the structure. We select a prefix, for example 'beam' suggesting it was a simple measurement on a beam. The series of measurements should then be stored in the files

Table 3.1: Mandatory header fields for a single-channel measurement (or analysis) file

Field Name	Type	Comment
FunctionType	Integer	See Section 3.1.2
NoValues	Integer	Number of data values
xStart	Integer	First x axis value, 0 if separate x axis
xIncrement	Float	x axis increment, 0 if separate x axis
Unit	String	Unit of measurement
Dof	Integer	Point number
Dir	String	Directions string, either 'X+', 'X-', etc., or 'RX+', 'RX-', etc.
Label	String	A free channel label
Title	String	A free title line (usually the same for all measurement channels)

Table 3.2: Mandatory header fields for an ABRAVIBE file containing a function which depends on two channels, for example a frequency response

Field Name	Type	Comment
RefDof	Integer	Dof number of reference
RefDir	String	Direction string, see Dir in Table 3.1

Table 3.3: Optional header fields for an ABRAVIBE file containing a spectral density function

Field Name	Type	Comment
Method	String	Either Welch, or SmoothedPer
NumberAverages	Integer	Number averages for spectrum
OverlapPerc	Double	Overlap Percentage (if Method is 'Welch')
SmoothLength	Integer	Smoothing window length (if Method is 'SmoothedPer')
SmoothWin	String	String with smoothing window ('boxcar','ahann'...)
ENBW	Double	Equivalent noise bandwidth of window used (see enwb command)

```
beam1.imptime
beam2.imptime
...
beam20.imptime
```

For processing, each of these files can be opened by using the syntax (given as an example here for the first file)

```
load beam1.imptime -mat
```

where the option `-mat` at the end tells MATLAB/Octave that the file is stored in MATLAB format despite its odd extension.

Once opened, if you type the command `whos`, you should get a list similar to

```
Name         Size                Bytes  Class     Attributes

Data         1x4               3760240  cell
Header       1x4                  3176  struct
```

To look at the header of, for example, the first channel you can use the command `Header(1)`, which could show

```
FunctionType: 1
    NoValues: 117500
      xStart: 0
  xIncrement: 3.9100e-004
        Unit: 'N'
         Dof: 1
         Dir: 'Z+'
       Label: 'Impact hammer'
       Title: 'Test on Beam'
```

which shows the fields and their values for this channel.

For further information on impact testing, see the commands listed under the header ‘Impact testing post processing’ in the ABRAVIBE help list, and Section 4.1 in the present document.

3.3 Modal Results File Format

For storing modal analysis, or operating deflection shape analysis, results, the following file format is used. The files should have an extension **.mod**, and should contain the variables `GEOMETRY`, `MODAL`, which are required by the

Table 3.6: Record fields for the GEOMETRY variable for modal data

Field Name	Type	Comment
node	Integer	Column vector with node labels
x	Float	Column vector with node X-axis data
y	Float	Column vector with node Y-axis data
z	Float	Column vector with node Z-axis data
conn	Integer	Row vector with node labels describing the wireframe. A line-break (pen-up) can use a NaN, 0, or -1 (double)

Table 3.7: Record fields for the MODAL variable for modal data . Each mode shape is stored in a separate record `MODAL(n)`

Field Name	Type	Comment
Freq	Float	Complex value for modal frequency in Hz.
Node	Integer	Column vector with node labels
X	Float	Column vector with node X-axis maximum deflection
Y	Float	Column vector with node Y-axis maximum deflection
Z	Float	Column vector with node Z-axis maximum deflection

`animate` command and described in Table 3.6 and Table 3.7, respectively, and an extra field `MHEADER` which is specific to the ABRAVIBE toolbox but not necessary for `animate` to work properly. See Section 4.4 for more information about the `animate` command.

The header information in the variable `MHEADER` can be used to store information about the modal analysis (or ODS) results, but is not required by the `animate` command, and thus the header variable is optional. Note, however, that some information in this header is necessary if data are to be exported in universal file format. In addition, if universal file import of modal data is used (see Section 3.4), `MHEADER` is automatically created by the `abra2modstr` command. The fields of the `MHEADER` command are listed in Table 3.8.

Table 3.8: Record fields for the MHEADER variable for modal data. Information relating to each mode shape is stored in a separate record `MHEADER(n)`. For more specific information, see documentation of the universal file format 55, for example at `http://sdrl.uc.edu/sdrl/referenceinfo/universalfileformats`

Field Name	**Type**	**Comment**
Title	String	A free title line
Title2	String	A free title line
Date	String	Date and time of analysis
Title3	String	A free title line
Title4	String	A free title line
ModelType	Integer	1 means structural model
AnalysisType	Integer	2=normal mode, 3=complex mode
DataChar	Integer	2=only translational DOFs, 3=translational and rotational DOFs
SpecDataType	Integer	8=displacement, 11=velocity, 12=acceleration
DataType	Integer	2=real, 5=complex
NumValuesPerNode	Integer	see unv55 format
NumIntDataValues	Integer	see unv55 format
NumRealDataValues	Integer	see unv55 format
LoadCase	Integer	see unv55 format
ModeNumber	Integer	Mode number
Eigenvalue	Float	Complex pole in radians/sec.
ModalA	Float	Modal A
ModalB	Float	Modal B

3.4 Universal File Import/Export

The ABRAVIBE toolbox contains a number of functions to import and export universal file test data (format 58), and modal data (formats 15, 55, and 82). The procedure to importing and exporting data is documented in this section and its subsections. The commands which should be called are documented in the help list under the section titled `Universal file import/export`.

3.4.1 Importing Universal File Test Data

For importing a universal file containing test data (i.e. measurement functions such as time data or spectra, frequency responses etc.), the command `unvread` produces one ABRAVIBE file for each record in the universal file, with a file name consisting of a prefix and a number. The ABRAVIBE file with extension **.mat** contains the standard variables **Data**, **Header** and, if the data need a special x axis due to the x axis increment not being constant, an x axis variable **xAxis**. The syntax for `unvread` is

```
UNVREAD   Read SDRC universal file into ABRAVIBE files

  LastNo = unvread(FileName,Prefix,FirstNo)

  LastNo      File number of last file, 0 if no files
              were saved

  FileName    Universal file name WITH extension
  Prefix      Prefix string for output file names
  FirstNo     File number for first file
  Type        Numeric unv type to 'filter' out (see below)
```

3.4.2 Importing Universal File Modal Data

If a universal file contains modal data (geometry and mode shape information), the command `unvread` stores the geometry and mode shape data into separate MATLAB files with extension **.mat**, one file for each universal file record. The command `abra2modstr` can then be used to convert the content in the MATLAB files into a single modal structure file with extension **.mod**, as suggested in Section 3.3.

3.4.3 Exporting to Universal Files

For creating universal files, either for test or for modal data, the command `unvwrite` can be used. For test data, it writes ABRAFILE files with data and header variables into a single universal file. For modal data, it assumes the variables `GEOMETRY`, `MODAL`, and, (optionally) `MHEADER` are available in the workspace. See the help text by typing `help unvwrite` for more information.

Chapter 4

Toolbox Functionality

4.1 Impact Testing

A special processing of impact testing data based on time data processing is outlined in Section 13.8.6 in [1]. In revision 2.0 of ABRAVIBE, this method has been implemented in a GUI (graphical user interface) software. In order to use this software, you need to store your impact testing data as time files with extension '.imptime', and where the force is located in the first array in data array Data{1}, and response measurements in subsequent arrays in Data{*}. The file format is described in Section 3.2.

To use the GUI based impact test GUI, start the software by typing the following command in the MATLAB Command Window

```
>>  ImpactGui
```

There is help inside the GUI.

4.2 Experimental Modal Analysis, EMA

There is now relatively powerful tools for experimental modal analysis (EMA) in the current version of ABRAVIBE. The functions have been implemented to work on either impulse responses (IRs), with the command

```
ir2ptime
```

which will directly produce a stabilization diagram, or by using frequency response functions, FRFs, by using the command

```
frf2ptime
```

in which cases the user is first asked to select the frequency range to be used.

The functionality for EMA includes two single-degree-of-freedom methods, and the following multiple-degree-of-freedom methods

- Prony's method, for one FRF or impulse response, or
- Least squares complex exponential (LSCE) method, for several FRFs or IRs, but for only one reference, or
- Polyreference time domain method (PTD), for several FRFs or IRs, and several references, or
- Modified multiple-reference Ibrahim time domain method, for several FRFs or IRs, and several references.

For each of these MDOF methods, a GUI based stabilization diagram is used, if used in MATLAB.

For mode shape estimation, there is one command which uses an MDOF least squares fit in frequency domain. In addition, there are commands for mode indicator functions, different mode scaling, mac matrix, etc. See the help text for ABRAVIBE for the section on EMA.

Some important information about the implementation is required. The curve-fitting algorithms implemented in ABRAVIBE assume measured data are in the format of **accelerance**, i.e. acceleration over force. In addition, all mode shapes are by default scaled for unity modal A (see Section 6.4.4 in [1]).

4.3 Operational modal analysis, OMA

In revision 2.0 of ABRAVIBE, there is a new command for OMA added, the command

```
ir2pmitd
```

which uses the multiple-reference Ibrahim time domain method for estimating poles and unscaled mode shapes. This algorithm is very similar to the well-known covariance-driven stochastic subspace identification (Cov–SSI) method. The main difference with this command, compared to the EMA command, is that it produces poles and mode shapes in one step. Since the mode shapes produces are uncscaled, it does not matter what kind of sensor data you supply to it. Furthermore, you can supply this command with either correlation functions, or free decays.

4.4 Animation

The full documentation of the `animate`[1] program is found by starting the program GUI by issuing the command `animate`, and then select `Help -- Animation Help` in the menu.

See Section 3.3 for documentation of the file format for modal data. `animate` can also directly import a universal file consisting of geometry and mode shape data.

[1]Note that the `animate` program is supplied together with the ABRAVIBE toolbox, but it is under special copyright. See the **animate.m** file for details

Bibliography

[1] A. Brandt. *Noise and Vibration Analysis – Signal Analysis and Experimental Procedures.* John Wiley and Sons, 2011.

[2] A. Brandt. *ABRAVIBE – A MATLAB toolbox for noise and vibration analysis and teaching.* Department of Technology and Innovation, University of Southern Denmark, http://www.abravibe.com, 2018.

Appendix

Command List

A list of all commands available in ABRAVIBE is provided here in the same format as results from the issue of

```
>> help abravibe
```

```
 About
 -------------------
   abrabout      - Type abrabout to print some information on screen

 Helpful functions
 -------------------
   checksw       - Check if running MATLAB or GNU/Octave
   makexaxis     - Create a time or frequency x axis
   makepulse     - Calculate a Gaussian or halfsine pulse

 Data storage info
 -------------------
   abra2imp      - Convert general AbraVibe time data files to
                   .imptime format
   abrahead      - Check if header structure is a valid AbraVibe
                   header
   datahelp      - Prints documentation for Abravibe data storage
                   (Data and Header variables)
   dir2nbr       - Convert Header direction string to number
   dofdir2n      - Convert dof number and dir string to numeric format
   functype      - Convert between numeric and string function type
   headpstr      - Create string out of Abravibe header Dof and Dir info
   makehead      - Make an empty AbraVibe header structure
   nbr2dir       - Convert numeric direction to string
   n2dofdir      - Convert dof numeric format to dof number and numeric dir

 Universal file import/export
 -------------------
   abra2modstr   - Convert AbraVibe modal information files into structure
   makemhead     - Make a default MODAL header
```

```
  unvread       - Read a standard UNV file
  unvwrite      - Write a universal file

Statistics analysis
-------------------
  apdf          - Calculate and plot probability density function, PDF
  acrest        - Calculate crest factor
  akurtosis     - Calculate kurtosis
  amoment       - Calculate statistical moment
  arms          - RMS calculation by 'analog' or linear integration
  askewness     - Calculate skewness
  framestat     - Calculate frame statistics of signal
  statchk       - Create standard statistics for time signal(s) in matrix
  statchkf      - STATCHK Create standard statistics for time signal(s)
                  in AbraVibe files
  teststat      - Hypothesis test for stationarity

Time data processing
-------------------
  afcoeff       - Compute filter coefficients for time domain filters
  apceps        - Calculate power cepstrum of single-sided autospectrum
  axcorr        - Scaled cross-correlation between x (input) and y (output)
  axcorrp       - Scaled cross-correlation between x (input) and y (output)
  smoothfilt    - Simple smoothing filter for time domain data
  timeavg       - Create synchronuous time average using blocksize N
  timediff      - Differentiate time signal
  timeint       - Integrate time signal
  timeweight    - Filter data with time weighting filter in time domain
  time2corr     - Process AbraVibe time data files into correlation
                  function files
  time2spec     - Process AbraVibe time data files into spectral
                  density function files
  winsmooth     - Smooth data with any smoothing window

Acoustics and octave bands
-------------------
  aweighf       - Calculate A-weighting curve for A-weighting a spectrum
  cweighf       - Calculate C-weighting curve for C-weighting a spectrum
  noctfilt      - Calculate filter coefficients for fractional octave band filter
  noctfreqs     - Compute the exact midband frequencies for a 1/n octave analysis
  noctlimits    - Calculate IEC/ANSI limits for fractional octave filter
  spec2noct     - Calculate 1/n octave spectrum from an FFT spectrum

Frequency Analysis
-------------------
  acsdsp        - Cross-Spectral Density, CSD, by smoothed periodogram method
  acsdw         - Calculate cross PSD from time data, Welch's method (standard)
  aenvspec      - Calculate envelope spectrum
  alinspec      - Calculate linear (rms) spectrum from time data
```

```
  alinspecp     - Calculate linear (rms) spectrum of time data, with phase
  apsdsp        - Power Spectral Density, PSD, by smoothed periodogram
  apsdw         - Calculate auto PSD from time data, Welch's method (standard)
  atranspec     - Calculate (linear) transient spectrum
  chkbsize      - Compare up to three blocksizes for PSD calculation
                  (Welch's method)
  fdiff         - Frequency differentiation by jw multiplication
  fint          - Frequency integration by jw division
  fnegfreq      - Create negative frequencies by mirroring positive
  nskipblocks   - Extract every NoSkip+1 blocks with size N from x
  time2xmtrx    - Calculate In-/Out cross spectral matrices for MIMO analysis
  time2xmtrxc   - Computes In-/Out cross spectral matrices from time data
                  by cyclic averaging
  welcherr      - Random error in PSD estimate with Welch's method

Time windows etc. (Also see impact testing post processing)
-------------------
  aflattop      - Calculate flattop window
  ahann         - Calculate a Hanning window
  hsinew        - Calculate half sine time window
  winacf        - Calculate amplitude correction factor of time window
  winenbw       - Calculate equivalent noise bandwidth of time window

Linear Systems Analysis
-------------------
  frf2ir        - Convert freqency response(s) to impulse response(s)
  frfim2re      - FRFRE2IM  Create FRF with Real part from Hilbert transform
                  of imaginary part
  frfre2im      - Create FRF with imaginary part from Hilbert transform of
                  real part
  mcoh          - Calculate multiple coherence
  xmtrx2frf     - FRF estimation (SISO, SIMO, MISO, or MIMO)

Principal components and virtual coherence
-------------------
  apcax         - Compute principal components and cumulated virtual coherences
  apcaxy        - Compute cumulated virtual coherences etc. based on two
                  sets of signals, x and y

System simulation and response analysis
-------------------
  asrs          - Acceleration shock response spectrum, SRS
  fz2poles      - Convert frequencies and damping to poles
  mck2frf       - Calculate FRF(s) from M, C, K matrices
  mck2modal     - Compute modal model (poles and mode shapes) from M,(C),K
  mkz2frf       - Calculate FRF(s) from M, K and modal damping, z
  mkz2modal     - Compute modal model (poles and mode shapes) from M,K, and z
  modal2frf     - Synthesize FRF(s) from modal parameters
  modal2frfh    - Synthesize FRF(s) from modal parameters with hysteretic damping
```

```
  poles2fz      - Convert poles to frequencies and relative damping
  timefresp     - Time domain forced response
  tunedamp      - Compute FRF of tuned damper with SDOF system
  uma2umm       - Rescale mode shapes from unity modal A to unity modal mass
  umm2uma       - Rescale mode shapes from unity modal mass to unity modal A

Create time data for simulation
-------------------
  abrand        - Create burst random time data blocks in column vector
  aprand        - Create pseudo random time data block in column vector
  psd2time      - Generate Gaussian random signal from PSD

Experimental modal analysis and operating deflection shape analysis
-------------------
  acomac        - AMAC  Calculate Coordinate Modal Assurance Critera matrix
                  C from two mode sets
  amac          - Calculate Modal Assurance Critera matrix M from two mode sets
  amif          - Calculate mode indicator function of (accelerance) FRFs
  data2hmtrx    - Import Abravibe FRF data into H matrix
  enhancefrf    - Enhance FRF matrix by Singular Value Decomposition
  frf2msdof     - Curve fit FRFs into poles and mode shapes, SDOF techniques
  frfp2modes    - Estimate mode shapes from FRFs and poles in frequency domain
  frf2ptime     - Time domain MDOF methods for parameter extraction
  frfsynt       - Synthesize FRF(s) after modal parameter extraction
  ir2ptime      - Time domain MDOF methods for parameter extraction from free
  listpoles     - List undamped natural frequencies and damping factors
  modal2pv      - Convert animate MODAL struct to separate variables
  modalchk      - Standard checks on FRF matrix for exp. modal analysis
  odspick       - Extract ODS shapes from phase spectrum
  plateanim     - Animate mode shapes on plate structures (1D mode shapes)
  plotmac       - Plot Manhattan type MAC matrix plot
  pv2modal      - Convert separate variables to animate MODAL struct
  sdiagramGUI   - MATLAB code for sdiagramGUI.fig. This is an internal function
  wincomp       - Compensate poles due to exponential window from Impact testing

Operational modal analysis
-------------------
  ir2pmitd      - ir2MITD Multi-reference Ibrahim Time domain MDOF method for
                  OMA parameter extraction

Order tracking (Rotating machinery analysis)
-------------------
Fix sampling frequency commands:
  map2order     - Extract orders from RPM map, fix fs or synchronuous sampling
  plotrpmmapc   - Plot a color map of RPM map data with fixed fs
  plotrpmmapwf  - Plot RPM map from fix fs in waterfall format
  rpmmap        - Compute rpm/frequency spectral map for order tracking, fixed fs
  tacho2rpm     - Extract rpm/time profile from tacho time signal
Synchronous sampling commands: (also use tacho2rpm and map2order above)
```

```
  plotrpmmapsc  - Plot a color map of RPM map data from synchronuous sampling
  plotrpmmapswf - Plot RPM map from synchronuous sampling in waterfall format
  rpmmaps       - Compute rpm/order spectral map for order tracking,
                  synchronuous sampling
  synchsampr    - Resample data synchronously with RPM, using rpm-time profile
  synchsampt    - Resample data synchronously with RPM, based on tacho signal

Data acquisition (using sound card or NI hardware)
-------------------
  adaqhelp      - Print some helpful info about data acquisition with AbraVibe
  agetdata      - Acquires time data from NI-card analog inputs into MATLAB
  arecdata      - Records data from NI-card to abravibe file format
  apreview      - View time data on one or more channels

Impact testing post processing
-------------------
  ImpactGui     - GUI based post processing
  aexpw         - Exponential window for impact testing
  aforcew       - Force window for impact testing

Plotting and animation
-------------------
  abrabrowse    - GUI-based data browser for AbraVibe toolbox data files
  angledeg      - Calculate angle in degrees for complex vector(s)
  animate       - GUI-based animation software
  anomograph    - Plot a displacement, velocity, acceleration nomograph
  db10          - Calculate dB for power data (units squared)
  db20          - Calculate dB for linear data (linear units)
  plotfile      - Plot data in ABRABIVE file(s) with default format
  plotmagph     - Plot complex data in magnitude/phase format
  plotreim      - Plot real and imaginary part of complex function
  plotscan      - Scan a vector with time data and plot blockwise
```