Microwave filters GUI

2.0.3

Generated by Doxygen 1.8.1.2

Thu Jun 13 2013 18:26:16

Contents

Chapter 1

LossyFilters software

Author

J.M.Rius, J.Mateu, J.M.Tamayo, C.Collado, A. Padilla and J.O'Callaghan,
Dpt. Signal Theory and Communications, Universitat Politècnica de Catalunya (UPC),

Contact:

lossyfilters@tsc.upc.edu
http://www.tsc.upc.edu/lossyfilters

Version

2.0.3

Date

June 11, 2013

Acknowledgement:

The software was developed in the frame of contract 21398/08/NL/GLC with the European Space Agency (E-SA). Technical Offer was Christoph Ernst. Further features were developed under contract UPC-C7767 with Thales Alenia Space España (TAS-E).

Contributions to the definition of the software functionality and testing have been made by Christoph Ernst, Mónica Martínez Mendoza and other ESA-ESTEC personnel, and Santiago Sobrino and Luis Roglá from TA-S-E.

1.1 Synthesis of microwave lossy filters

See Synthesis of microwave lossy filters for a brief introduction about synthesis of microwave lossy filters.

1.2 License

See License terms page.

1.3 Installation

See separate installation instructions in files README.txt, README.pdf or readme/html/README.html

1.4 Software description:

The LossyFilters software package has been written to synthesize filters following various forms of classical (no-loss considered in the synthesis), pre-distortion and prescribed insertion loss synthesis. This software obtains the coupling matrix of several network topologies for a given response and allows performing rotations on them to find the desired topology. Additionally, the software allows to evaluate the effect of loss in the networks resulting from the synthesis, even in those cases where the synthesis results in an ideal lossless network (i.e., classical and pre-distortion synthesis).

The software package has been divided in two parts:

- <B>Open-source GUI and free libraries:</B> The graphical user interface (GUI) MWfiltersGUI, the 'Free Fil
- <B>Non-free libraries:</B> The non-free libraries are not required but, when they are installed in the sy

1.4.1 Functionality of the free GUI and libraries

- The GUI is full-featured with complete functionality to open, edit, save and execute parameter files, as well as plot [S] parameters and manually edit coupling matrices:
  - Actions can be launched by menus, toolbar buttons or keyboard shortcuts.
  - There are tooltips in the parameters edit widgets.
  - Application output in main window log widget and status bar.
  - Modeless parameter edit dialog window with Accept / Compute / Discard buttons, to allow the computation with trial parameter values without closing the dialog.
  - Preventive data validation at the parameters edit widgets in order to prevent the user introducing invalid data.
  - Frequency parameters edit widgets allow the user to set THz, GHz, MHz, kHZ, Hz units.
  - The parameter edit window detects if there are changes in the parameters, so that the application can warn the user if he is going to destroy non-saved changes.
  - Plots S-parameters graph with phase or group delay in right-axis.
  - Plots support adding, moving and deleting markers.
  - Select which results to plot.
  - Zoom S-parameter plot.
  - Manual and autoscaling for each axis of S-parameter plot.
  - Print S-parameter plot to printer or PDF file.
  - Display coupling matrices and Q of resonators.
  - Edit coupling matrix topology for non-zero elements optimization.
  - Coupling matrices window: Manual rotation, node scaling, element anhilitation, add non-resonant nodes, edit matrix entries or Q values, file load/save, etc. There are facilities to undo and revert to any previous state.
  - Plot S-parameters computed from the current coupling matrix compared with S-parameters computed from the characteristic polynomials. Display the average error in dB.
  - Read [S] parameters from Touchstone format file.
  - Store, recover and compare results from different computations or read from Touchstone format file.
  - Specification mask read from file and plotted.
  - Sensitivity analysis.

  - User-selectable predistortion zeros.
  - Code fully documented with Doxygen.

- The free libraries have functionality to:
  - Read and parse parameters file.
  - Write parameters file.
  - Printing strings to:
    - Console, when called from the command line.
    - Main window log widget, when called from the GUI.
  - Catches syntax errors in the parameter file.
  - Correctly handles non-ASCII characters. Reads and writes UTF-8 format parameter files, accepts non-English characters in the GUI and in console output. When python reports that an output device is not able to print non-ASCII characters, the console interface replaces 8-bit characters by "?" in the output.
  - Filter synthesis: Butterworth, Chebyshev and Quasielpitic filters.
  - Generation of transversal coupling matrix (TCM) N+2.
  - Functions to generate rotation matrices and to find the angle that zeroes a given coupling matrix element.
  - Transformation from transversal coupling matrix (TCM) to folded coupling matrix (FCM) and to many other types of coupling matrices.
  - Calculation of the passband coupling matrix, which is obtained from the low-pass one by a scaling. The resulting passband coupling matrix can be directly related to design parameters used for the filter design, such as the coupling between resonators.
  - Optimize coupling matrix elements for a given topology and resonator Q, adjusting [S] parameters to the polynomials [S] parameters or to a specification mask.
  - Prescribed flatness for lossless-like filters with flat passband response. The quality factor of a filter to emulate is specified in order to append additional resonator losses.
  - Displays Q of resonators.
  - Compute energy stored at resonators and reactive couplings and power dissipated at resonators and resistive couplings.
  - Reports roundoff errors in polynomial roots and partial fraction expansion.
  - Reports error in S-parameters computed from transversal and folded coupling matrices compared with S-parameters computed from the characteristic polynomials.
  - Code fully documented with Doxygen.

1.4.2 Functionality of the non-free libraries

- The **"Library of Extra filters"** provides the following additional functions:
  - Generalized Chebyshev transfer function with arbitrary transmission zeros [from Cameron, chapter 6].
  - Linear phase equalization, optimizing the position of real and/or complex transmission zeros.
  - Code fully documented with Doxygen.

- The **"Library of Lossy filters"** provides the following additional functions:
  - Filter losses: predistortion (conventional or adaptive).
  - Filter losses: "Prescribed insertion loss" with non-Uniform Q case 1 (k21+k11+k22 asymmetrical), case 2 (k21) and case 3 (k21+zero/pole).
  - FCM matrix rotation to obtain uniform Q in resonators for Lossy Filter case-1 (kS11 + kS21), N=4 and N=6 and case-3 (kS21 + pole/zero), N=6. Automatic symmetrization of transmission zeros if necessary.
  - Code fully documented with Doxygen.

Chapter 2

License terms

Contact: lossyfilters@tsc.upc.edu

http://www.tsc.upc.edu/lossyfilters

2.1 Open-source GUI and libraries

2.2 Non-free libraries

The "Extra Filters Library" and the "Lossy Filters Library" are not free software and need a user license from UPC. The European Space Agency (ESA) has such user license with right to use and modify the present version and future updates of these libraries and to make copies of these libraries for use in multiple computers and for backup or archival purposes.

- "Extra Filters Library" for the computation of generalized Chebyshev filter characteristic polynomials is distributed by UPC either in the source file "libextrafilters.py", the corresponding bytecode file "libextrafilters.pyc" or bound into a binary executable file. License terms are described in the LICENSE.LIBEXTRAFILTERS file.
- "Lossy Filters Library" for the computation of characteristic polynomials including filter losses is distributed by UPC either in the source file "liblossyfilters.py", the corresponding bytecode file "liblossyfilters.pyc" or bound into a binary executable file. License terms are described in the LICENSE.LIBLOSSYFILTERS file.
- The "Extra Filters Library" and the "Lossy Filters Library" need the "License Check Library", which distributed toghether with them either in the executable Python bytecode file "libchecklicense.pyc" or bound into a binary executable file. The source file "libchecklicense.py" is not distributed. The "License Check Library" has been developed by UPC for checking the user's license to run software released by the AntennaLab research group at the Department of Signal Theory and Communications (TSC). The "License Check Library" is distributed and licensed by UPC under the same terms as the "Extra Filters Library" and the "Lossy Filters Library", with the following additional restrictions:
  - The source code is not distributed.
  - It is specially forbidden to modify the "License Check Library" source or binary code.
  - It is also forbidden to reverse engineer, decompile, or disassemble the "License Check Library" with the intent to violate license agreement of the "Extra Filters Library" or the "Lossy Filters Library".

Chapter 3

Synthesis of microwave lossy filters

The project goals at large are to investigate novel lossy filter synthesis techniques to obtain filters whose insertion loss flatness can be made equal to that of an (ideal) lossless filter, despite the limited Q of the resonators within the filter.

The classical synthesis procedure consists on obtaining a purely reactive (no dissipation effect) network that defines the resonant frequency of the resonators forming the filter and the way they are coupled. However, in practice, some dissipation always exist in the final filter implementation which can be evaluated afterwards by introducing the material losses (finite Q of the resonators) into the synthesized lossless network.

Among other effects, this approach leads to filters with minimum insertion loss in the passband at one frequency at the expense of additional passband rounding towards the band-edges, being the use of high Q resonators the only way to achieve filters with flat pass-band response by means of classical synthesis techniques. This may result in heavy and large filters which might be impractical in space systems, or in wireless and handset components with strong demand for low-cost and small size.

This can be overcome by the use of non standard filter synthesis techniques such as:

- **Pre-distortion synthesis:** synthesizes a lossless network whose pole placement is such that, after introducing the losses, produces the desired (improved) in-band flatness. However, this is at the expense of increased in-band signal reflections.

  R. J. Cameron, C. M. Kudsia, R.R. Mansour, Microwave Filters for Communication systems. Fundamentals, Design, and applications, John Wiley & Sons, 2007. Dissipation I movement de sigma

  Ming Yu et al., "Predistortion Technique for Cross-Coupled Filters and its application to satellite communication systems", IEEE MTT-51, pp. 2505-2515, Nov. 2003

- **Prescribed insertion loss synthesis:** In order to obtain both good insertion loss flatness and good return loss, the insertion loss flatness is obtained at the expense of an increase of the overall insertion loss, therefore the resulting filters are of use in those applications where the filters are placed after the low noise amplifier without affecting the noise figure, as it occurs in input multiplexers (IMUX) for satellite transponders.

  V. Miraftab, Ming Yu, "Advanced Coupling Matrix and Admittance Function Synthesis Techniques for Dissipative Microwave Filters," Microwave Theory and Techniques, IEEE Transactions on , vol.57, no.10, pp.2429-2438, Oct. 2009.

  I. C. Hunter, A. Guyette, and R. D. Pollard, "Passive Microwave Receive Filter Network Using Low-Q Resonators", IEEE microwave Magazine, September 2005.

  A.C. Guyette, I. C. Hunter, R. Pollard, "The design of microwave bandpass filters using resonators with nonuniform Q", IEEE MTT-54 (11), November 2006, pp.3914-3922.

Using these lossy filters synthesis techniques, very low resonator Q that can be employed and, therefore, low cost and compact filters (such as microstrip filters) can be realized.

3.1 References:

The main references to the work in this project are:

J.Mateu, C. Collado, J.M. O'Callaghan,
"Lossy Filter Synthesis and Study of Topology Solution Space",
ESA working paper 2366. TEC-ETM/2009.142/CE. First issued: September 2008.

J. Mateu, A. Padilla, C. Collado, M. Martinez-Mendoza, E. Rocas, C. Ernst, J M. O'Callaghan,
"Synthesis of 4th order Lossy Filters with uniform Q distribution",
Microwave Symposium Digest (MTT), *2010 IEEE MTT-S International*, pp.568-571, 23-28 May 2010.
doi: 10.1109/MWSYM.2010.5517741
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5517741&isnumber=5514662
URL: http://upcommons.upc.edu/e-prints/bitstream/2117/10412/1/Synthesis%20of%204th%20o
Lossy%20Filters%20with%20uniform%20Q%20distribution.pdf

A. Padilla, J. Mateu, C. Collado„ C. Ernst, J.M. Rius„ J.M. Tamayo, J.M. O'Callaghan,
"Comparison of lossy filters and predistorted filters using novel software",
Microwave Symposium Digest (MTT), *2010 IEEE MTT-S International*, pp.1720-1723, 23-28 May 2010
doi: 10.1109/MWSYM.2010.5517713
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5517713&isnumber=5514662
URL: http://upcommons.upc.edu/e-prints/bitstream/2117/10413/1/Comparison%20of%20lossy%
pdf

Chapter 4

Todo List

Class libcommonfunc.CouplingMatrices

Call numpy functions as methods of array class, whenever possible.

Member libcommonfunc.CouplingMatrices.transCouplingMatrix

The function "r21k, p21k, remainder = sc.signal.residue(Y21np,Yd)" does not give the correct remainder when it is complex but in the tested cases it does not afect. This function should be improved for the future.

Member libcommonfunc.CP

: See if MainWindow.cleanup() method can be used here. Close the [S] parameters plot and set all results to None

Member libcommonfunc.Sparameters.fromCouplingMatrix

Find out why sign of S11 from CM is opposite than from CP. The sign is changed below to force agreement with S11 from CP.

File mwfiltersgui.py

Linux executables created with cx_freeze report sys.stdout.encoding equal to None.

Check end-of-line in a Mac computer. It should work if platform.system() returns something different than 'Windows', 'Microsoft', 'Vista' or 'Linux'.

Possibility to change styles with a combobox. Not obvious in pyQt.

Member mwfiltersgui.sizeHint

Reimplment myTableWidget.resizeEvent() function to reduce size of table when there are no ScrollBars

Chapter 5

Namespace Index

5.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 6

Class Index

6.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 7

Class Index

7.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 8

File Index

8.1 File List

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

Chapter 9

Namespace Documentation

9.1 dbplot Namespace Reference

Plot data curves in multiple tabs.

Classes

- class DbPlot

  GUI dialog to contain plot.
- class AxisScalingDlg

  GUI dialog to contain axis scaling controls.
- class CurveFamily

  Data structure containing info about properties of a family of curves and the curve list.
- class EditCurvesDlg

  Dialog to select visible curves.
- class MyPicker

  Class derived from Qwt.QwtPlotPicker to allow reimplementation of Qwt.QwtPlotPicker.trackerText()
- class CanvasEventFilter

  Event filter for the canvas.
- class PlotMarker

  Class for markers.
- class AutoScaleZoomerBase

  Class that stores zoomer base.
- class PrintFilter

  Filter to transform color of some plot items and fontsize when printing.

Variables

- string MAC = "qt_mac_set_native_menubar"

  Global variable useful only for MacOSX.
- string applicationName = 'dBplot'

  Application name, to be displayed in 'About' dialog.

9.1.1 Detailed Description

Plot data curves in multiple tabs.

Author

J.M. Rius
Universitat Politècnica de Catalunya (UPC)

Version

1.0

Date

November 25, 2010

9.2 libcommonfunc Namespace Reference

Library of common functions for microwave filters synthesis.

Classes

- class parseError
  Parse error catch.
- class synthError
  Synthesis error catch.
- class licenseError
  License error catch.
- class Parameters
  Filter, synthesis and sweep parameters.
- class Sparameters
  [S] parameters class.
- class MatrixQ
  Class that contains a coupling matrix, its name, the number of non-resonant nodes, the Q of resonators and some operations.
- class CouplingMatrices

Coupling matrices class.

- class FrequencyTransformBP

  Frequency transformation fro band pass to low pass prototype.

Functions

- def setGlobalVariablesLibCommonFunc

  Function to allow the GUI set values to the global variables of libcomonfunc module.
- def readVerbose

  Returns the value of the global verbose variable.
- def pkgNameVersion

  Retrieves software package name and version from string.
- def myPrint

  Prints on standard output if console application.
- def printHeader

  Prints application header info.
- def readParamFile

  Read parameters file and return a Parameter class instance.
- def criticalErrorMsg

  Show critical error message.
- def warningMsg

  Show warning message.
- def informationMsg

  Show information message.
- def askQuestion

  Show dialog to ask question.
- def listStr

  Nicely convert list of floats to string, rounding to 'prec' significant digits.
- def complexStr

  Nicely convert complex number to string, rounding real and imaginary parts to 'prec' significant digits.
- def validate

List of Chebash parameters to ignore

chebashIgnoreList = ['n_ceros', 'n_frecpropias_eq', 'real', 'imag', 'tipo_transformacion', 'graf[1]', 'graf[2]', 'graf[3]', 'graf[5]', 'graf[6]', 'graf[7]', 'graf[8]', 'graf[9]', 'graf[10]', 'graf[11]', 'graf[12]', 'graf[13]', 'graf[14]', 'graf[15]', 'graf[16]', 'graf[17]', 'graf[18]']

- def copy

  Function to copy instances of the Parameters class (copies the self instance).
- def __str__

  Special method to print parameter class (prints the self instance).

Variables

- dictionary parameterDict

  Dictionary of parameter keywords and variable names.
- int saveSignificantDigits = 14

  Number of significant digits to save in files.
- int saveSignificantDigitsEnergy = 4

  Number of significant digits to save in energy and power results.

- mainWindow = None

  Module global variable equal either to None or to the GUI QMainWindow class instance, containing the application main window.
- verbose = False

  Module global variable equal to True if the GUI has been called with the -v flag command line option.
- nogui = False

  Module global variable equal to True if the GUI has been called with the –nogui flag command line option.
- string preGuiOutput = ”

  String containing console output generated before the GUI main window exists.
- CP = CMSPNone
- list nonSymmZeros = [ ]

  def normRealFreq(f): """ Auxiliary function """ return (f/self.f0 - self.f0/f)∗self.f0/self.BW

9.2.1 Detailed Description

Library of common functions for microwave filters synthesis.

Author

J.M. Rius, J. Mateu, J.M. Tamayo, C. Collado, A. Padilla and J. O'Callaghan

Dpt. Signal Tehory and Communications, Universitat Politècnica de Catalunya (UPC).

Version

2.0.3

Date

June 11, 2013

Acknowledgement: The software was developed in the frame of contract 21398/08/NL/GLC with the European Space Agency (ESA). Technical Offer was Christoph Ernst. Further features were developed under contract U-PC-C7767 with Thales Alenia Space España (TAS-E). Contributions to the definition of the software functionality and testing have been made by Christoph Ernst, Mónica Martínez Mendoza and other ESA-ESTEC personnel, and Santiago Sobrino and Luis Roglá from TAS-E.

Contact: lossyfilters@tsc.upc.edu

9.2.2 Function Documentation

9.2.2.1 def libcommonfunc.__str__ (*self*)

Special method to print parameter class (prints the self instance).

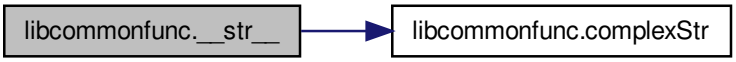
Floating point variables are printed as strings using str() because str() is inmune to roundoff errors in the floating point representation.

```
@return st = Human redable printing of the parameters (string).
```

Definition at line 883 of file libcommonfunc.py.

References complexStr().

Here is the call graph for this function:



9.2.2.2 def libcommonfunc.askQuestion (*st, buttons*)

Show dialog to ask question.

Parameters

st	= Information string. It can contain

and

html tags, that are discarded in console mode.

Parameters

buttons	= QMessageBox buttons to show. (e.g.: QMessageBox.Yes | QMessageBox.No)

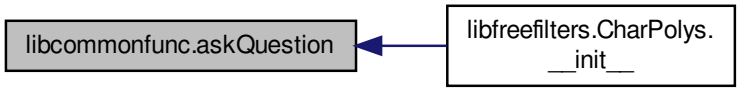
Returns

reply = button clicked ((e.g.: QMessageBox.Yes)

Definition at line 348 of file libcommonfunc.py.

Referenced by libfreefilters.CharPolys.__init__().

Here is the caller graph for this function:



9.2.2.3 def libcommonfunc.complexStr (*x, prec, format* = `None`, *minWidth* = `0`, *eng* = `False`, *listFormat* = `False`)

Nicely convert complex number to string, rounding real and imaginary parts to 'prec' significant digits.

Parameters

x	= Complex number.
prec	= Number of significant digits (int).
format	= None | 'DB' | 'MA' | 'RI' (string). Default is None.
minWidth	= Minimum field field (int). Default is 0.
eng	= Use engineering notation in formats None and 'RI'. Default is False. Formatted strings of the form 0.x are allowed for exponents 0 and 9.
listFormat	= Flag to return a list of strings instead of a string. The list contains [Magnitude, Angle] or [Real, imag]. Useful to save in .csv format.

The format is borrowed from TouchStone s2p [S] parameter file definition. None: Complex number, like 2.4-j5.3. DB: Decibel Magnitude/Degree Angle. MA: Linear Magnitude/Degree Angle (polar form). RI: Real Part/Imaginary Part (rectangular form).

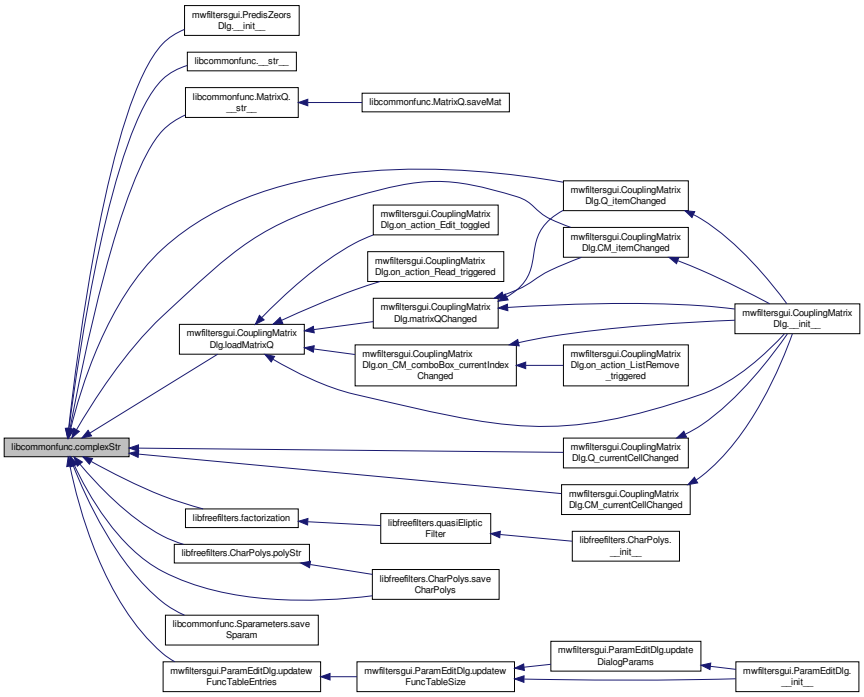
Returns

st = String representation of x.

Definition at line 386 of file libcommonfunc.py.

Referenced by mwfiltersgui.PredisZeorsDlg.__init__(), __str__(), libcommonfunc.MatrixQ.__str__(), mwfiltersgui.CouplingMatrixDlg.CM_currentCellChanged(), mwfiltersgui.CouplingMatrixDlg.CM_itemChanged(), libfreefilters.factorization(), mwfiltersgui.CouplingMatrixDlg.loadMatrixQ(), libfreefilters.CharPolys.polyStr(), mwfiltersgui.CouplingMatrixDlg.Q_currentCellChanged(), mwfiltersgui.CouplingMatrixDlg.Q_itemChanged(), libfreefilters.CharPolys.saveCharPolys(), libcommonfunc.Sparameters.saveSparam(), and mwfiltersgui.ParamEditDlg.updatewFuncTableEntries().

Here is the caller graph for this function:



9.2.2.4 def libcommonfunc.copy (*self*)

Function to copy instances of the Parameters class (copies the self instance).

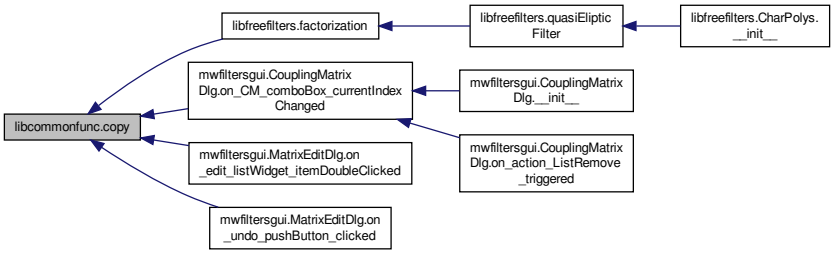
Returns

Pcopy = New instance of Parameters class copied from the self instance.

Definition at line 864 of file libcommonfunc.py.

Referenced by libfreefilters.factorization(), mwfiltersgui.CouplingMatrixDlg.on_CM_comboBox_currentIndex-Changed(), mwfiltersgui.MatrixEditDlg.on_edit_listWidget_itemDoubleClicked(), and mwfiltersgui.MatrixEditDlg.-on_undo_pushButton_clicked().

Here is the caller graph for this function:



9.2.2.5 def libcommonfunc.criticalErrorMsg (*st*)

Show critical error message.

Prints error and exists in console mode. Displays QMessageBox in GUI mode.

Parameters

st	= Error string. It can contain

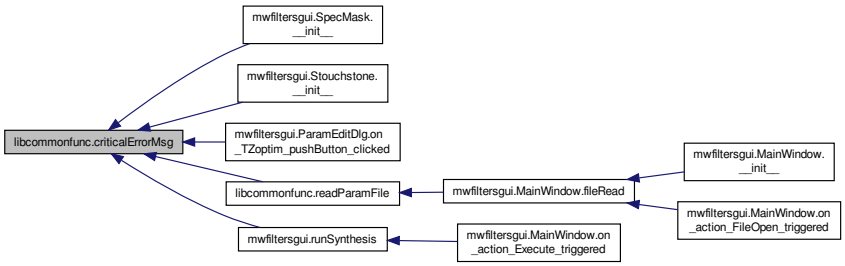
and

html tags, that are discarded in console mode.

Definition at line 290 of file libcommonfunc.py.

Referenced by mwfiltersgui.SpecMask.__init__(), mwfiltersgui.Stouchstone.__init__(), mwfiltersgui.ParamEditDlg.-on_TZoptim_pushButton_clicked(), readParamFile(), and mwfiltersgui.runSynthesis().

Here is the caller graph for this function:



9.2.2.6 def libcommonfunc.informationMsg (*st*)

Show information message.

Prints message in console mode. Displays QMessageBox in GUI mode.

Parameters

st	= Information string. It can contain

and

html tags, that are discarded in console mode.

Definition at line 332 of file libcommonfunc.py.

9.2.2.7 def libcommonfunc.listStr (*listX*, *prec* = 3)

Nicely convert list of floats to string, rounding to 'prec' significant digits.

Parameters

listX	= List of floats.
prec	= Number of significant digits (int). Default 3.

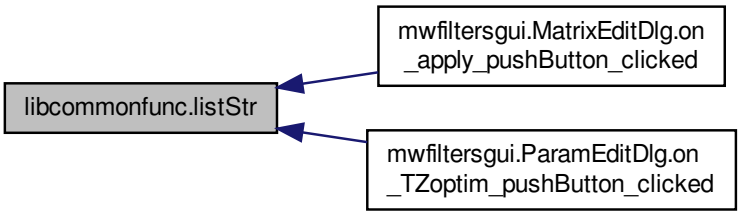
Returns

st = String representation of x.

Definition at line 362 of file libcommonfunc.py.

Referenced by mwfiltersgui.MatrixEditDlg.on_apply_pushButton_clicked(), and mwfiltersgui.ParamEditDlg.on_TZoptim_pushButton_clicked().

Here is the caller graph for this function:



9.2.2.8 def libcommonfunc.myPrint (*st*)

Prints on standard output if console application.

or on log window if GUI application.

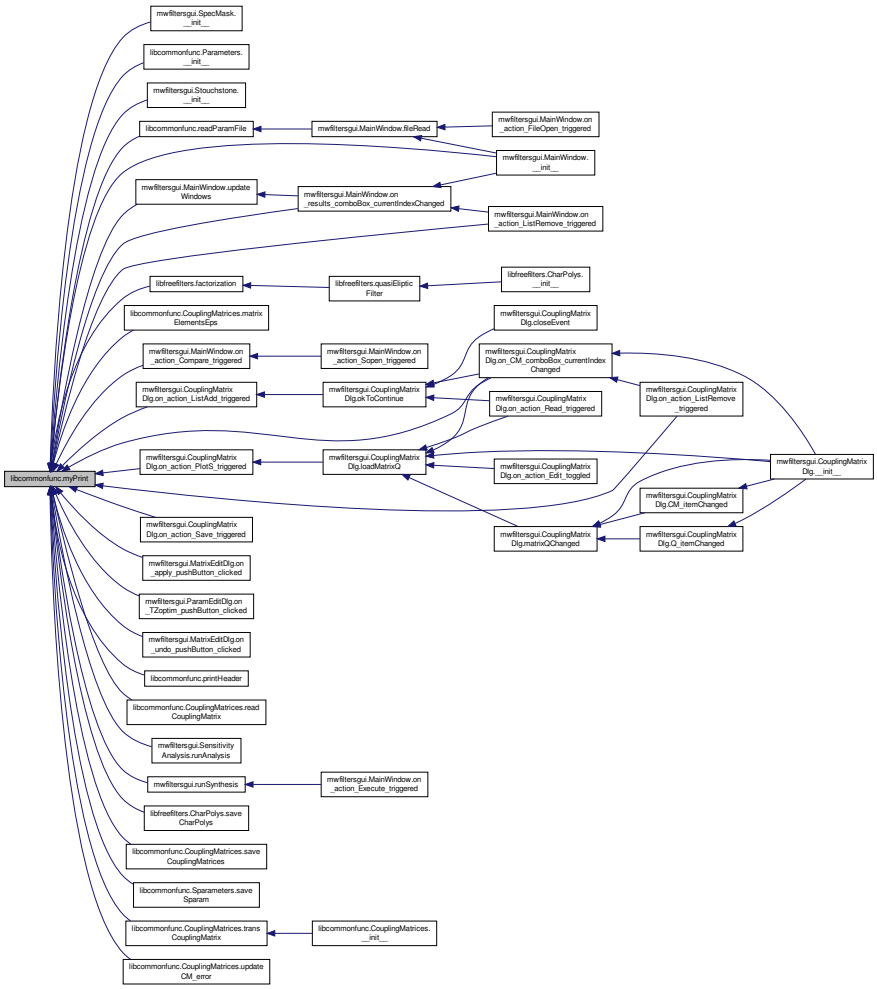
Parameters

st	= String.

Definition at line 192 of file libcommonfunc.py.

Referenced by mwfiltersgui.SpecMask.__init__(), libcommonfunc.Parameters.__init__(), mwfiltersgui.Stouchstone.__init__(), mwfiltersgui.MainWindow.__init__(), libfreefilters.factorization(), libcommonfunc.CouplingMatrices.matrixElementsEps(), mwfiltersgui.MainWindow.on_action_Compare_triggered(), mwfiltersgui.CouplingMatrixDlg.on_action_ListAdd_triggered(), mwfiltersgui.MainWindow.on_action_ListRemove_triggered(), mwfiltersgui.CouplingMatrixDlg.on_action_ListRemove_triggered(), mwfiltersgui.CouplingMatrixDlg.on_action_PlotS_triggered(), mwfiltersgui.CouplingMatrixDlg.on_action_Save_triggered(), mwfiltersgui.MatrixEditDlg.on_apply_pushButton_clicked(), mwfiltersgui.CouplingMatrixDlg.on_CM_comboBox_currentIndexChanged(), mwfiltersgui.MainWindow.on_results_comboBox_currentIndexChanged(), mwfiltersgui.ParamEditDlg.on_TZoptim_pushButton_clicked(), mwfiltersgui.MatrixEditDlg.on_undo_pushButton_clicked(), printHeader(), libcommonfunc.CouplingMatrices.readCouplingMatrix(), readParamFile(), mwfiltersgui.SensitivityAnalysis.runAnalysis(), mwfiltersgui.runSynthesis(), libfreefilters.CharPolys.saveCharPolys(), libcommonfunc.CouplingMatrices.saveCouplingMatrices(), libcommonfunc.Sparameters.saveSparam(), libcommonfunc.CouplingMatrices.transCouplingMatrix(), libcommonfunc.CouplingMatrices.updateCM_error(), and mwfiltersgui.MainWindow.updateWindows().

Here is the caller graph for this function:



9.2.2.9 def libcommonfunc.pkgNameVersion (*st*)

Retrieves software package name and version from string.

Parameters

st	= String to parse for package name and version. It must be the docstring (**doc**) of the calling module.

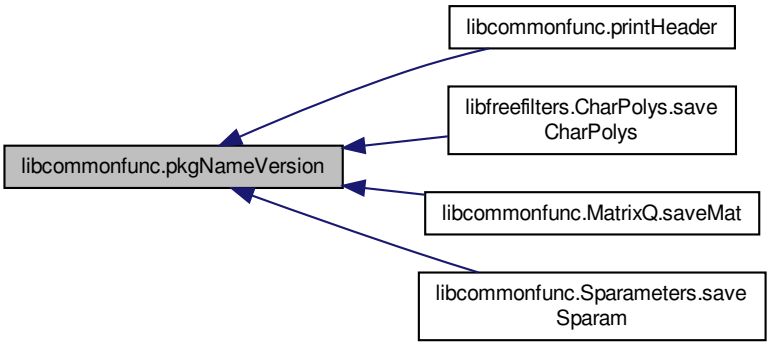
Returns

pkgname = Package name.
version = Package version.

Definition at line 180 of file libcommonfunc.py.

Referenced by printHeader(), libfreefilters.CharPolys.saveCharPolys(), libcommonfunc.MatrixQ.saveMat(), and libcommonfunc.Sparameters.saveSparam().

Here is the caller graph for this function:



9.2.2.10 def libcommonfunc.printHeader (*applicationName, st, importModule* = `False`)

Prints application header info.

The information shown is read from the file docstring.

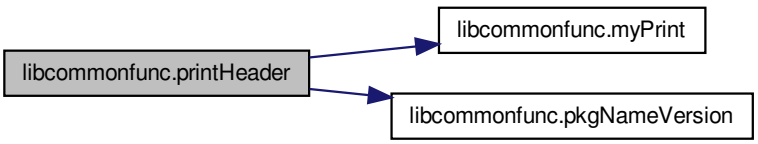
Parameters

applicationName	= Name of application, usually os.path.basename(sys.argv[0]).
st	= String to parse for package name and version. It must be the docstring (**doc**) of the calling module.
importModule	= Flag, set to True when this function is called to acknowledge successful module import.

Definition at line 223 of file libcommonfunc.py.

References myPrint(), and pkgNameVersion().

Here is the call graph for this function:



9.2.2.11 def libcommonfunc.readParamFile (*fileName*)

Read parameters file and return a Parameter class instance.

Parameters

fileName	= Parameter file name.

Returns

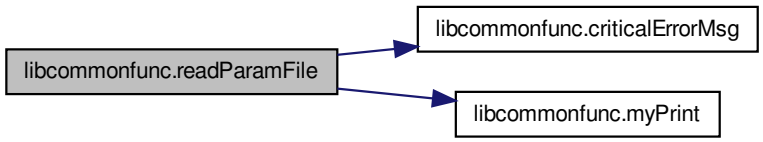
Parameter class instance.

Definition at line 252 of file libcommonfunc.py.

References criticalErrorMsg(), and myPrint().

Referenced by mwfiltersgui.MainWindow.fileRead().

Here is the call graph for this function:



Here is the caller graph for this function:

9.2.2.12 def libcommonfunc.readVerbose ( )

Returns the value of the global verbose variable.

Useful to get the value from other modules that were loaded before the value of verbose was updated

Definition at line 169 of file libcommonfunc.py.

Referenced by libfreefilters.factorization().

Here is the caller graph for this function:

9.2.2.13 def libcommonfunc.setGlobalVariablesLibCommonFunc (*variablesDict*)

Function to allow the GUI set values to the global variables of libcomonfunc module.

This function is used in order to avoid circular imports.

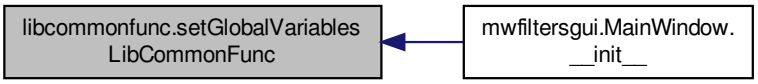
Parameters

variablesDict,:	Variable keyword argument list. Python creates a dictionary.

Definition at line 159 of file libcommonfunc.py.

Referenced by mwfiltersgui.MainWindow.__init__().

Here is the caller graph for this function:



9.2.2.14 def libcommonfunc.validate (*self*)

List of Chebash parameters to ignore

chebashIgnoreList = ['n_ceros', 'n_frecpropias_eq', 'real', 'imag', 'tipo_transformacion', 'graf[1]', 'graf[2]', 'graf[3]', 'graf[5]', 'graf[6]', 'graf[7]', 'graf[8]', 'graf[9]', 'graf[10]', 'graf[11]', 'graf[12]', 'graf[13]', 'graf[14]', 'graf[15]', 'graf[16]', 'graf[17]', 'graf[18]']

List of Chebash Mask parameters not to read

chebashMaskList = ['n_at', 'f_at', 'A_at', 't_at', 'offsetmba', 't_offsetmba', 'n_perd', 'f_abp', 'a_abp', 't_abp', 'offsetmbp', 't_offsetmbp', 'nr_retardo', 'f_retardo', 'r_retardo', 't_retardo', 'offsetrg', 't_offsetrg', 'nptos_pendht', 'f_pendht', 'd_pendht', 't_pendht', 'offsetpendht', 't_offsetpendht', 'nptos_pendre', 'f_pendre', 'd_pendre', 't_pendre', 'offsetpendre', 't_offsetpendre', 'nptos_gdst', 'f_gdst', 'v_gdst', 't_gdst', 'offsetgdst', 't_offsetgdst', 'nptos_gdsp', 'f_gdsp', 'v_gdsp', 't_gdsp', 'offsetgdsp', 't_offsetgdsp', 'n_frecpropias_eq', 'variacion_temp', 'margen_temp', 'modificar_mascaras', 'margen_gdsp'] processingComplex = False # Will be True when we start processing a 3-line complex parameter for (keyChebash, stValue) in paramList:

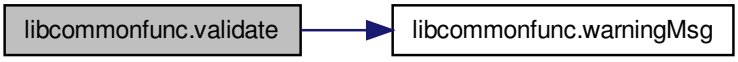
```
if not processingComplex:    # int, float, string or 1st line of a complex parameter
    try:
        (keyLossyFilters, type, scale) = chebashTranslateDict[keyChebash]
    except KeyError, err:
        if keyChebash not in chebashIgnoreList + chebashMaskList: myPrint('Unknown parameter key in Chebash fi
        continue

    if type in ['int', 'float']:

        # First process the special cases
        if keyChebash == 'graf[4]':
            fmin, fmax = tuple( stValue.split(',')[0:2] )
            paramListLF.append( ('minFreq', 'float', str(float(fmin)*scale) ) )
```

```
            paramListLF.append( ('maxFreq', 'float', str(float(fmax)*scale) ) )
            continue
        else:
            if stValue != 'X':  # In Chebash, 'X' indicates an empty parameter
                exec("value = [ scale* %s(n) for n in stValue.split(',') ]" % type)
                if len(value) == 1:     # Not a list
                    value = str(value[0])
                else:   # List
                    value = str(value).replace('[', '').replace(']', '')
            else:   # Empty parameter
                value = ' '

    elif type == 'string':
            value = stValue
    elif type == 'complex':
        processingComplex = True
        NComplex = int(stValue)
    else:
        raise parseError, 'Invalid data type'

else:   # 2nd and 3rd lines of a complex parameter
    if keyChebash == 'real':
        if NComplex > 0:
            complexRe = [ float(n)*scale for n in stValue.split(',') ]
    elif keyChebash == 'imag':
        if NComplex > 0:
            complexIm = [ float(n)*scale for n in stValue.split(',') ]
            value = str([ complex(nReal, nImag) for nReal, nImag in zip(complexRe, complexIm) ]).replace('[',
        else:
            value = ' '
        processingComplex = False

# Add lossyfilters parameter, for normal or for special case
paramListLF.append((keyLossyFilters, type, value))


    return paramListLF + chebashDefaultList

     Function to check if parameters have invalid values and make some basic parameter processing.
```

Definition at line 735 of file libcommonfunc.py.

References warningMsg().

Here is the call graph for this function:



9.2.2.15 def libcommonfunc.warningMsg (*st*)

Show warning message.

Prints warning in console mode. Displays QMessageBox in GUI mode.

Parameters

st	= Warning string. It can contain

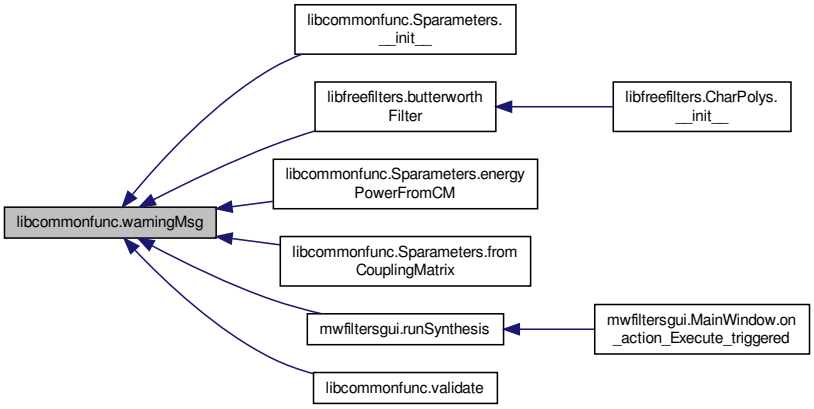
and

html tags, that are discarded in console mode.

Definition at line 316 of file libcommonfunc.py.

Referenced by libcommonfunc.Sparameters.__init__(), libfreefilters.butterworthFilter(), libcommonfunc.Sparameters.energyPowerFromCM(), libcommonfunc.Sparameters.fromCouplingMatrix(), mwfiltersgui.runSynthesis(), and validate().

Here is the caller graph for this function:



9.2.3 Variable Documentation

9.2.3.1 libcommonfunc.CP = CMSPNone

Todo : See if MainWindow.cleanup() method can be used here. Close the [S] parameters plot and set all results to None

Definition at line 302 of file libcommonfunc.py.

9.2.3.2 libcommonfunc.mainWindow = None

Module global variable equal either to None or to the GUI QMainWindow class instance, containing the application main window.

It is equal to None (default) when the calling program is the console interface (mwfiltersgui.py –nogui). When creating the main window, the GUI executes the libcommonfunc.setGlobalVariablesLibCommonFunc() function in order to set this variable. The GUI methods do not use this variable. They access the mainWindow class through ParamEditDlg.mainWindow and MainWindow 'self' attributes.

Definition at line 87 of file libcommonfunc.py.

9.2.3.3 libcommonfunc.nogui = False

Module global variable equal to True if the GUI has been called with the –nogui flag command line option.

Otherwise it is False.

Definition at line 95 of file libcommonfunc.py.

9.2.3.4 list libcommonfunc.nonSymmZeros = [ ]

def normRealFreq(f): """ Auxiliary function """ return (f/self.f0 - self.f0/f)∗self.f0/self.BW

for (keyChebash, stValue) in paramList: if paramList[9][0] == 'real': realEqZs = [normRealFreq(self.f0 + float(st)∗1e6) for st in paramList[9][1].split(',')]

Definition at line 649 of file libcommonfunc.py.

9.2.3.5 dictionary libcommonfunc.parameterDict

Initial value:

```
1 { 'outName': 'outName',  'outDir': 'outDir',  'outDirName': 'outDirName',  'N':
      'N',  'f0': 'f0',  'BW': 'BW',
2                          'filterTransferFunc': 'filterTransferFunc',  '
     chebyLr': 'chebyLr', 'qeZero': 'qeZero' ,  'qeLr':  'qeLr',
3                          'transImagZeros': 'transImagZeros',  '
     transComplexZeros': 'transComplexZeros', 'genChebyLr': 'genChebyLr',
4                          'genChebyLP': 'genChebyLP',  'LPalgor' : 'LPalgor',
      'LPmaxIter' : 'LPmaxIter', 'LPNsamples' : 'LPNsamples',
5                          'LPcomplexZeros': 'LPcomplexZeros',  'LPrealZeros':
      'LPrealZeros', 'LPfracBW': 'LPfracBW',  'LPripple': 'LPripple',
6                          'synthTech': 'synthTech',  'zerosArrang': '
     zerosArrang',  'adapPredis': 'adapPredis',  'Qef': 'Qef',  'wFunc': 'wFunc',
7                          'finiteQo': 'finiteQo',  'Qo': 'Qo',  'QoPredis': '
     QoPredis',
8                          'nuqK': 'nuqK',  'nuqK11c1': 'nuqK11c1',  'nuqK22c1
     ': 'nuqK22c1',  'nuqK21c1': 'nuqK21c1',  'nuqK21c2': 'nuqK21c2',  'nuqK21c3': '
     nuqK21c3',
9                          'nuqTech': 'nuqTech',  'nuqDelta': 'nuqDelta',
10                          'maxFreq': 'maxFreq',  'minFreq': 'minFreq',  '
     numFreq': 'numFreq' }
```

Dictionary of parameter keywords and variable names.

This dictionary allows the developer to change a variable name in the code without having to change the corresponding keyword in all the existing parameter files.

Definition at line 62 of file libcommonfunc.py.

9.2.3.6 int libcommonfunc.saveSignificantDigitsEnergy = 4

Number of significant digits to save in energy and power results.

Usually less than saveSignificantDigits, because not much precision is needed.

Definition at line 80 of file libcommonfunc.py.

9.2.3.7 libcommonfunc.verbose = False

Module global variable equal to True if the GUI has been called with the -v flag command line option.

Otherwise it is False.

Definition at line 91 of file libcommonfunc.py.

9.3 libfreefilters Namespace Reference

Free Filters Library to compute Butterworth, Chebyshev and Quasieliptic characteristic polynomials with minimum insertion loss.

Classes

- class CharPolys

Characteristic polynomials class.

Functions

- def setGlobalVariablesLibFreeFilters

  Function to allow the GUI set values to the global variables of libcomonfunc module.
- def polyOmega

  Returns $P_s(s)$ for the input polynomial $P_w(\omega)$, with $s = j\omega$ or $\omega = -js$.
- def polyCheby

  Compute Chebyshev polynomials from order 0 to N.
- def polyConj

  Returns the 'Paraconjugate' $P_c(s)$ of the input polynomial $P(s)$, with complex variable restricted to the imaginary axis, $s = j\omega$, and therefore $s^* = -s$.
- def findEps

  This function finds the maximum of the function $g(s) = |\frac{NUM(s)}{\varepsilon DEN(s)}|$ in the imaginary axis.
- def rootErrors

  Report errors in the roots position for a polynomial of the form SQF(s) = F(s)F(-s) with real coefficients.
- def factorization

  Factorization of polynomial SQF(s) of order 2N as a product of two polynomials, each one of order N.
- def vecSort

  Sort a vector of complex numbers so that their imaginary parts is in decreasing order.
- def butterworthFilter

  Compute characteristic polynomials and characteristic constants of a Butterwoth filter.
- def chebyshevFilter

  Compute characteristic polynomials and characteristic constants of a Chebyshev filter.
- def quasiElipticFilter

  Compute characteristic polynomials and characteristic constants of a Quasieliptic filter.

Variables

- mainWindow = None

  Module global variable equal either to None or to the GUI QMainWindow class instance, containing the application main window.
- importedExtraFilters = False

  Variable set to True if the non-free libextrafilters is available.

9.3.1 Detailed Description

Free Filters Library to compute Butterworth, Chebyshev and Quasieliptic characteristic polynomials with minimum insertion loss.

Author

J.M. Rius, J. Mateu, J.M. Tamayo, C. Collado, A. Padilla and J. O'Callaghan

Dpt. Signal Tehory and Communications, Universitat Politècnica de Catalunya (UPC),

Version

2.0.3

Date

June 11, 2013

Acknowledgement: The software was developed in the frame of contract 21398/08/NL/GLC with the European Space Agency (ESA). Technical Offer was Christoph Ernst. Further features were developed under contract U-PC-C7767 with Thales Alenia Space España (TAS-E). Contributions to the definition of the software functionality and testing have been made by Christoph Ernst, Mónica Martínez Mendoza and other ESA-ESTEC personnel, and Santiago Sobrino and Luis Roglá from TAS-E.

Contact: lossyfilters@tsc.upc.edu

9.3.2 Function Documentation

9.3.2.1 def libfreefilters.butterworthFilter (*P, FT, symmetrizeZeros* = None)

Compute characteristic polynomials and characteristic constants of a Butterwoth filter.

```
        @param P = Parameter class instance containing filter, synthesis and sweep parameters.
        @param  FT = Frequency transformation class instance.
        @param symmetrizeZeros = This parameter is ignored in this class of filter.
        @return Ps = Characteristic polynomial of the numerator of \form#25 over characteristic constant \form
        @return Es = Characteristic polynomial of the denominator of \form#25 and \form#23.
        @return Fs = Caracteristic polynomial of the numerator of \form#23 considering a lossy system.
        @return eps = Constant in the denominator of \form#25.
        @return epsR = Constant in the denominator of \form#23.
        @return rootsErr = Maximum distance between the theoretical position of SQF roots and the actual posit

    @section butterCharPolys Algorithm
        [TN 102.1 sec. 7.2].
    <ol>
    <li> The roots of \form#252 can be found as [TN 102.1 eq. (28)]:
```

$$s_k = \begin{cases} \exp\left(\frac{j\pi(2k-1)}{2N}\right) & N \text{ odd} \\ \exp\left(\frac{j\pi 2k}{2N}\right) & N \text{ even} \end{cases}$$

where $k = 1 \ldots 2N$ and N is the filter order.

The coeffcicients of $E(s)$ are obtained from the roots of $E(s)E(-s)$ that lie in the LHS of the s-plane using the numpy.poly1d methods.

$P(s) = 1$

$F(s) = s^N$

$\varepsilon = 1$

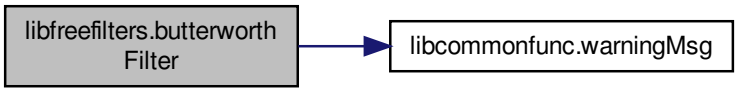
$\varepsilon_R = 1$

Definition at line 652 of file libfreefilters.py.

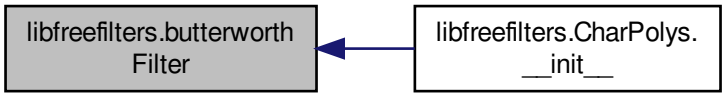
References libcommonfunc.warningMsg().

Referenced by libfreefilters.CharPolys.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.2.2 def libfreefilters.chebyshevFilter (*P, FT, symmetrizeZeros =* `None`)

Compute characteristic polynomials and characteristic constants of a Chebyshev filter.

```
        @param P = Parameter class instance containing filter, synthesis and sweep parameters.
        @param  FT = Frequency transformation class instance.
        @param symmetrizeZeros = This parameter is ignored in this class of filter.
        @return Ps = Characteristic polynomial of the numerator of \form#25 over characteristic constant \form
        @return Es = Characteristic polynomial of the denominator of \form#25 and \form#23.
        @return Fs = Caracteristic polynomial of the numerator of \form#23 considering a lossy system.
        @return eps = Constant in the denominator of \form#25.
        @return epsR = Constant in the denominator of \form#23.
        @return rootsErr = Maximum distance between the theoretical position of SQF roots and the actual posit

    @section chebyCharPolys Algorithm
    [TN 102.1 sec. 7.3]
    <ol>
    <li> \form#260,
    where \form#261 are the return losses (parameter "chebyLr") [TN102.1 eq.(25)].

    <li> \form#256

    <li> Since \form#255  has been obtained from its roots, the highest degree coeffcicient is equal to 1.
    In order to guarantee that
```

$$\max(S_{21}(\omega)) = \max\left(\frac{P(s)}{\varepsilon E(s)}\right) = 1$$

characteristic constant ε is recomputed with CharPolys.findEps() method.

$F(s) = T_N(-js)$ where T_N is the Chebyshev polynomial of order N [TN 102.1 eq. (32)].

The polyCheby() method is used to compute T_N, while polyOmega() makes the change of variable $\omega = -js$.

The roots of $E(s)$ are [TN 102.1 eq. (34)]:

$$\begin{aligned} s_k &= -\sinh(\mu)\sin(\theta) + j\cosh(\mu)\cos(\theta) \\ \mu &= \frac{1}{N}\sinh^{-1}\left(\frac{1}{\varepsilon}\right) \\ \theta &= \left(\frac{\pi}{2}\right)\left(\frac{2k-1}{N}\right) \\ k &= 1\ldots N \end{aligned}$$

The coeffcicients of $E(s)$ are obtained from the roots using the numpy.poly1d methods.

It is not necessary to define $k = 1\ldots 2N$ and pick up the roots in the LHS of the s-plane, as suggested in [TN 102.1 eq. (34)], since $\varepsilon > 0 \Rightarrow \mu > 0$ and $k = 1\ldots N \Rightarrow \sinh(\mu)\sin(\theta) > 0$.

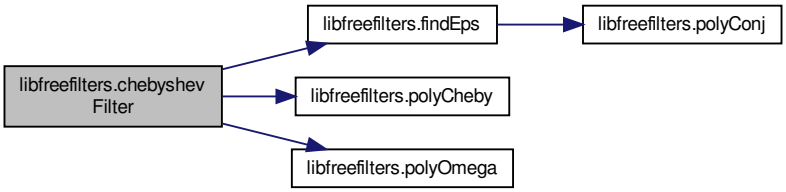
ε_R is the ratio of the highest degree coefficients of $F(s)$ and $E(s)$ in order to guarantee that $|S_{11}(\omega)| \to 1$ as $\omega \to \infty$.

Definition at line 734 of file libfreefilters.py.

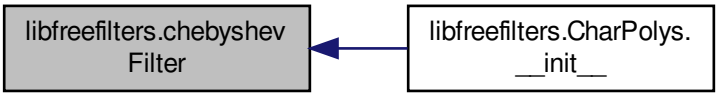
References findEps(), polyCheby(), and polyOmega().

Referenced by libfreefilters.CharPolys.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.2.3 def libfreefilters.factorization (*SQF, method*)

Factorization of polynomial SQF(s) of order 2N as a product of two polynomials, each one of order N.

```
Polynomials are of type numpy.poly1d.
```

```
@param SQF  = Polynomial to factorize as a product of two.
@param method = Either an integer value between 1 and 4 describing the method of choosing which roots
                <br>See documentation below.
@return Fs, F22s = The two polynomials in which the input polynomial SQF is factorized.
Fs and F22s are normalized to highest order coefficient equal to 1.
F22s is formed from the roots discarded in Fs, and should be equal to
```

$(-1)^N F(s)^* = (-1)^N F^*(-s)$ [Cameron pp. 208,212], where $F(s)^*$ is the 'Paraconjugate' polynomial of $F(s)$ (see function CharPolys.polyConj).

Returns

rootsErr = Maximum distance between the theoretical position of SQF roots and the actual position. Real or imaginary parts with abs() smaller than tolerance will be considered as a zero. A pair of roots separated less than this tolerance in s-plane will be cosidered as a double root.

9.3.3 Factorization

The input polynomial is factorized as:

$$SQF(s) = F(s)F_{22}(s)$$

so that when s is restricted to the imaginary axis, $s = j\omega$, it accomplishes the following:

$$|SQF(j\omega)| = F(j\omega)F^*(j\omega), |SQF(j\omega)| = F_{22}(j\omega)F_{22}^*(j\omega), \forall\omega \in \Re.$$

The roots of the polynomial SQF(s) must be symmetric about imaginary axis. Note that for pure imaginary roots, they must be of even multiplicity. This restriction is equivalent to the fact that SQF(s) can be factorized as:

$$SQF(s) = F(s)F(-s)$$

```
There are different methods to choose the roots of F(s) from the roots of SQF(s): <br>
<ol>
    <li>If 'method' argument is an integer:
        <ul>
        <li>method == 1 -> All zeros in the left s-plane (Hurwitz).
        <li>method == 2 -> All zeros in the right s-plane.
        <li>method == 3 -> The roots in right and left s-planes are ordered by decreasing imaginary part a
        <li>method == 4 -> The complementary of 3.
        </ul>
    <li>If 'method' argument is a list of Booleans:
        <ul>
            <li>If method[n] is True,  zeros_in_left_s-plane[n] go to F(s)  and the symmetrical zeros_in_r
            <li>If method[n] is False, zeros_in_left_s-plane[n] go to F(-s) and the symmetrical zeros_in_r
            <li>The zeros zero_in_right_s-plane and zero_in_left_s-plane are ordered by decreasing imagina
        </ul>
</ol>
F22(s) is composed of the remaining roots from SQF(s) to form the complementary function to F(s). Method 1
```

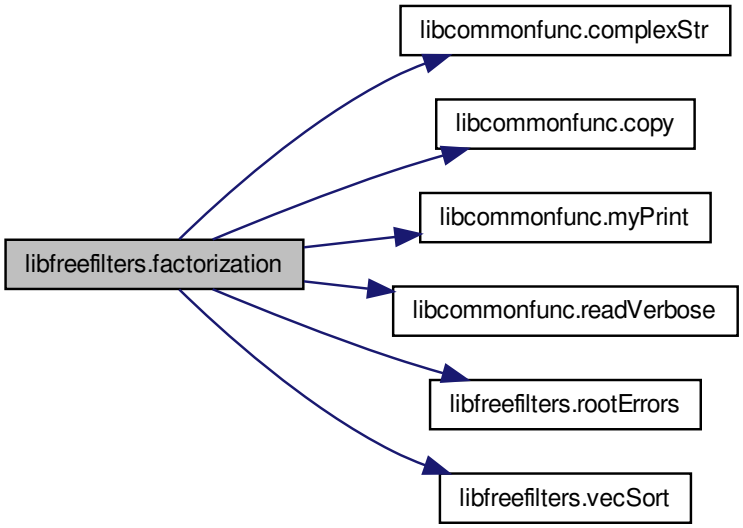
$$F^1(s) = F_{22}^2(s), F^2(s) = F_{22}^1(s), F^3(s) = F_{22}^4(s), F^4(s) = F_{22}^3(s).$$

Definition at line 515 of file libfreefilters.py.

References libcommonfunc.complexStr(), libcommonfunc.copy(), libcommonfunc.myPrint(), libcommonfunc.readVerbose(), rootErrors(), and vecSort().

Referenced by quasiElipticFilter().

Here is the call graph for this function:



Here is the caller graph for this function:

9.3.3.1 def libfreefilters.findEps (*NUM, DEN*)

This function finds the maximum of the function $g(s) = |\frac{NUM(s)}{\varepsilon DEN(s)}|$ in the imaginary axis.

Usually $NUM = Ps, Fs$ and $DEN = Es$. Polynomials are of type numpy.poly1d.

Parameters

NUM	= Numerator polynomial of the function to maximize.
DEN	= Denominator polynomial of the function to maximize.

Returns

eps = Maximum value of the function to maximize. It accomplishes that the maximum value of $g(s) = |\frac{NUM(s)}{\varepsilon DEN(s)}|$ in the imaginary axis is equal to 0 dB. To do the maximization we procede as follows: $SQNUM(s) = |NUM(s)|^2 = NUM(s)NUM^*(-s)$ for $s = j\omega$. $SQDEN(s) = |DEN(s)|^2 = DEN(s)DEN^*(-s)$ for $s = j\omega$. The function f to maximize is:

$$f(s) = \frac{SQNUM(s)}{SQDEN(s)} = \frac{|NUM(s)|^2}{|DEN(s)|^2}$$

$$f'(s) = \frac{SQNUM'(s)SQDEN(s) - SQNUM(s)SQDEN'(s)}{SQDEN(s)^2} = 0$$

The maximum of f is obtained in one of the roots of the numerator of $f'(s)$.

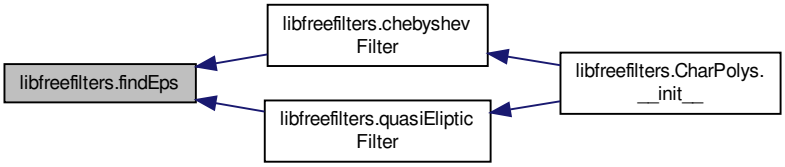
Definition at line 422 of file libfreefilters.py.

References polyConj().

Referenced by chebyshevFilter(), and quasiElipticFilter().

Here is the call graph for this function:

Here is the caller graph for this function:



9.3.3.2 def libfreefilters.polyCheby (*N*)

Compute Chebyshev polynomials from order 0 to N.

```
@param N = Maximum order of polynomials (int)
@return Tn = List containing Chebyshev polynomials. List elements of type numpy.poly1d.

@section alfPolyCheby Algorithm

The recursion formula for Chebyshev polynomials
```

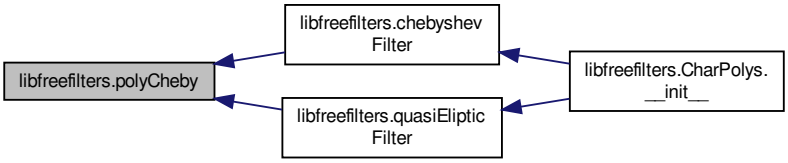
$$T_n(\omega) = 2\omega T_{n-1}(\omega) - T_{n-2}(\omega)$$

is used to find T_N after setting $T_0 = 1$ and $T_1 = \omega$.

Definition at line 379 of file libfreefilters.py.

Referenced by chebyshevFilter(), and quasiElipticFilter().

Here is the caller graph for this function:



9.3.3.3 def libfreefilters.polyConj (*P*)

Returns the 'Paraconjugate' $P_c(s)$ of the input polynomial $P(s)$, with complex variable restricted to the imaginary axis, $s = j\omega$, and therefore $s^* = -s$.

```
If \form#174 is the polynomial order, \form#230 can be expressed as:
```

$$P(s) = \sum_{n=0}^{N} c_n s^n$$

We define the paraconjugate $P_c(s)$ as [Cameron pp. 208, footnote].:

$$P_c(s) = P(s)^* = P^*(s^*) = P^*(-s) = \sum_{n=0}^{N} (-1)^n c_n^* s^n$$

When the frequency is evaluated for complex s not in the imaginary axis, $s = \sigma + j\omega$, you cannot compute S_{22} using the 'paraconjugate' polynomial of $F(s)$, and must complex-conjugate the array of values of $F(s)$ for each s .

Parameters

P	= Input polynomial $P(s)$.

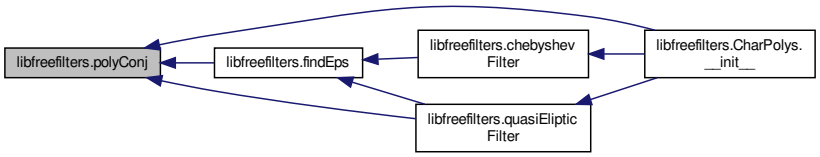
Returns

Pc = Paraconjugate polynomial $P_c(s)$.

Definition at line 401 of file libfreefilters.py.

Referenced by libfreefilters.CharPolys.__init__(), findEps(), and quasiElipticFilter().

Here is the caller graph for this function:



9.3.3.4 def libfreefilters.polyOmega (*Pw*)

Returns $P_s(s)$ for the input polynomial $P_w(\omega)$, with $s = j\omega$ or $\omega = -js$.

```
If \form#174 is the polynomial order, \form#220 and \form#216 can be expressed as:
```

$$P_w(w) = \sum_{n=0}^{N} c_n \omega^n$$

$$P_s(s) = [P_w(\omega)]|_{\omega=-js} = \sum_{n=0}^{N} (-j)^n c_n s^n$$

Parameters

Pw	= Input polynomial $P_w(\omega)$.

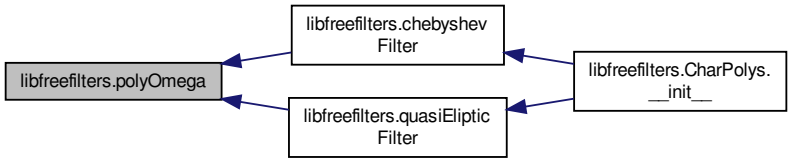
Returns

Ps = $P_s(s)$.

Definition at line 359 of file libfreefilters.py.

Referenced by chebyshevFilter(), and quasiElipticFilter().

Here is the caller graph for this function:



9.3.3.5 def libfreefilters.quasiElipticFilter (*P, FT, symmetrizeZeros* = None)

Compute characteristic polynomials and characteristic constants of a Quasieliptic filter.

```
        @param P = Parameter class instance containing filter, synthesis and sweep parameters.
        @param  FT = Frequency transformation class instance.
        @param symmetrizeZeros = This parameter is ignored in this class of filter.
        @return Ps = Characteristic polynomial of the numerator of \form#25 over characteristic constant \form
        @return Es = Characteristic polynomial of the denominator of \form#25 and \form#23.
        @return Fs = Caracteristic polynomial of the numerator of \form#23 considering a lossy system.
        @return eps = Constant in the denominator of \form#25.
        @return epsR = Constant in the denominator of \form#23.
        @return rootsErr = Maximum distance between the theoretical position of SQF roots and the actual posit
                                    Real or imaginary parts with abs() smaller than tolerance will be cons
                                    A pair of roots separated less than this tolerance in s-plane will be

    @section quasiElipticCharPolys Algorithm
    [TN 102.1 sec. 7.4]
    <ol>
    <li> \form#260,
    where \form#261 are the return losses (parameter "chebyLr") [TN102.1 eq.(25)].

    <li> \form#270, [TN 102.1, eq. (40)]
```

 In order to obtain \form#255, the first step is to compute the following linear combination of Chebys

$$\begin{aligned}\alpha &= \frac{(a+\sqrt{a^2-1})^2}{2}\\ \beta &= -a^2\\ \gamma &= \frac{(a-\sqrt{a^2-1})^2}{2}\\ K_N(\omega) &= \alpha\,\omega\,T_{N-1}(\omega)+\beta\,T_{N-2}(\omega)*+\gamma\,\omega\,T_{N-3}(\omega)\end{aligned}$$

Then, $E(j\omega)E(-j\omega)$ is [TN 102.1, eq. (41)]:

$$E(\omega)E(-\omega) = \left(a^2-\omega^2\right)^2+\varepsilon^2 K_N^2(\omega)$$

$E(s)E(-s)$ is found by replacing $\omega = -js$ above by the function polyOmega().

Finally, the roots of \form#255 are the roots of \form#252 that lie in the LHS of the s-plane.

 Since \form#255 has been obtained from its roots, the highest degree coeffcicient is equal to 1.
According to eq. (39) and (41) in TN102.1, the polynomial \form#274
is the denominator of \form#275 and therefore includes the constant \form#276.
Consequently, \form#10 must be set to the square root of the highest degree coeffcicient of \form#252.

 \form#215 follows from the energy conservation equation [TN 102.1, eq. (29)]:

$$|S_{11}(s)|^2+|S_{21}(s)|^2 = 1$$

$$\frac{|F(s)|^2}{\varepsilon_R^2} = |E(s)|^2 - \frac{|P(s)|^2}{\varepsilon^2}$$

$$\frac{F(s)F(-s)}{\varepsilon_R^2} = E(s)E(-s) - \frac{P(s)P(-s)}{\varepsilon^2}$$

where we assume $\varepsilon_R = 1$.

The roots of $F(s)$ are obtained from the roots of F(s)F(-s) by the CharPolys.factorization() method.

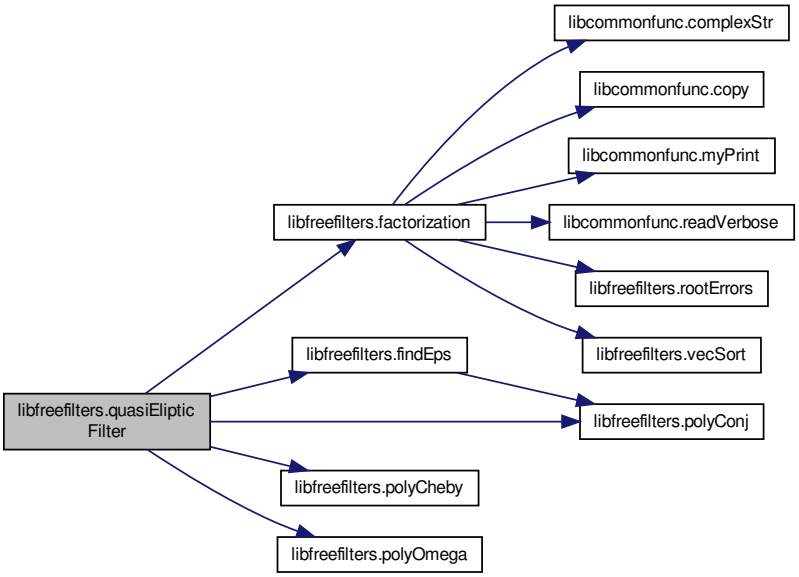
ε_R must be recomputed as the ratio of the highest degree coefficients of $F(s)$ and $E(s)$ in order to guarantee that $|S_{11}(\omega)| \to 1$ as $\omega \to \infty$.

Definition at line 833 of file libfreefilters.py.

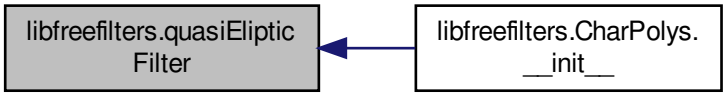
References factorization(), findEps(), polyCheby(), polyConj(), and polyOmega().

Referenced by libfreefilters.CharPolys.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



9.3.3.6 def libfreefilters.rootErrors (*roots_SQF*)

Report errors in the roots position for a polynomial of the form SQF(s) = F(s)F(-s) with real coefficients.

The roots of the polynomial SQF(s) must be symmetric about imaginary axis. Note that for pure imaginary roots, they must be of even multiplicity.

The error is computed as the maximum distance between the roots theoretical and actual positions.

Parameters

roots_SQF	= Array of roots. Type numpy.array.

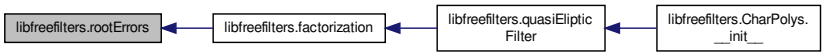
Returns

maxErr = Maximum distance between the roots theoretical and actual positions.

Definition at line 447 of file libfreefilters.py.

Referenced by factorization().

Here is the caller graph for this function:



9.3.3.7 def libfreefilters.setGlobalVariablesLibFreeFilters (*variablesDict*)

Function to allow the GUI set values to the global variables of libcomonfunc module.

This function is used in order to avoid circular imports.

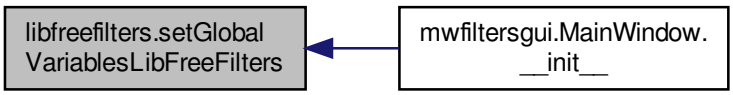
Parameters

variablesDict,:	Variable keyword argument list. Python creates a dictionary.

Definition at line 63 of file libfreefilters.py.

Referenced by mwfiltersgui.MainWindow.__init__().

Here is the caller graph for this function:



9.3.3.8 def libfreefilters.vecSort (*vec*)

Sort a vector of complex numbers so that their imaginary parts is in decreasing order.

Parameters

vec	= complex vector to sort.

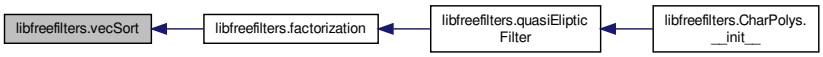
Returns

vecsort = sorted vector.

Definition at line 601 of file libfreefilters.py.

Referenced by factorization().

Here is the caller graph for this function:



9.3.4 Variable Documentation

9.3.4.1 libfreefilters.importedExtraFilters = False

Variable set to True if the non-free libextrafilters is available.

Otherwise set to False.

Definition at line 901 of file libfreefilters.py.

9.3.4.2 libfreefilters.mainWindow = None

Module global variable equal either to None or to the GUI QMainWindow class instance, containing the application main window.

It is equal to None (default) when the calling program is the console interface (mwfiltersgui.py –nogui). When creating the main window, the GUI executes the libfreefilters.setGlobalVariablesLibFreeFilters() function in order to set this variable. The GUI methods do not use this variable. They access the mainWindow class through ParamEditDlg.mainWindow and MainWindow 'self' attributes.

Definition at line 55 of file libfreefilters.py.

9.4 mwfiltersgui Namespace Reference

Graphical User Interface.

Classes

- class EnergyDbPlot

  Class derived from DbPlot to allow adding some widgets for plotting energy and power.
- class TempVarDlg

  Widget to ask the user for temperature variation parameters.
- class SpecMask

  Specification mask class.
- class Stouchstone

  [S] parameters class, as read from Touchstone file.
- class myTableWidget

  Subclass for:
- class MainWindow

GUI main window class.

- class HelpForm

  GUI dialog to display help.
- class SetCouplingDlg

  Dialog to set the type of a coupling.
- class MatrixEditDlg

  GUI dialog to edit coupling matrix.
- class PredisZeorsDlg

  Dialog to select predistortion zeros.
- class SensitivityAnalysis

  Class for sensitivity analysis.
- class CouplingMatrixDlg

  GUI dialog to show coupling matrices.
- class ParamEditDlg

  GUI dialog to edit synthesis parameters.

Functions

- def __init__

  Derived class constructor.
- def updateInputPower

  Update energy and power plots after inputPower value has been changed by the user.
- def updateExcitation

  Update energy and power plots after excitation value has been changed by the user.
- def appendReverseData

  Set YData for reverse excitation.
- def replaceReverseData

  Replace YData for reverse excitation.
- def sizeHint
- def runSynthesis

  Runs parameter validation, filters synthesis and results plots.
- def usage

  Print error message and command line help.

Variables

- string MAC = "qt_mac_set_native_menubar"

  Global variable useful only for Mac OS X.
- importedExtraFilters = False

  Variable set to True if the non-free libextrafilters is available.
- importedLossyFilters = False

  Variable set to True if the non-free liblossyfilters is available.
- string applicationName = "Microwave filters GUI"

  Program name as run by the user.
- list fileName = args[0]

  Name of the parameters file to process.
- tuple P = readParamFile(fileName)

  Parameters class instance containing filter and synthesis parameters read from the parameters file.
- tuple FT = FrequencyTransformBP(P)

  Create frequency transform instance.

- string stFloatPoint = r'((\d+\.?\d*|\d*\.?\d+)([Ee][-+]?\d+)?)'

  Create validator strings and regular expressions.
- tuple app = QApplication(QtArgs)

  QApplication class instance, containing our application.

9.4.1 Detailed Description

Graphical User Interface.

Author

J.M. Rius, J. Mateu, J.M. Tamayo, C. Collado, A. Padilla and J. O'Callaghan.

Dpt. Signal Tehory and Communications, Universitat Politècnica de Catalunya (UPC),

Version

2.0.3

Date

June 11, 2013

Acknowledgement: The software was developed in the frame of contract 21398/08/NL/GLC with the European Space Agency (ESA). Technical Offer was Christoph Ernst. Further features were developed under contract U-PC-C7767 with Thales Alenia Space España (TAS-E). Contributions to the definition of the software functionality and testing have been made by Christoph Ernst, Mónica Martínez Mendoza and other ESA-ESTEC personnel, and Santiago Sobrino and Luis Roglá from TAS-E.

Contact: lossyfilters@tsc.upc.edu

Lossyfilters software GUI uses the open-source dbplot.py module, which is based in part on the work of the Qwt project (http://qwt.sf.net) and has been released by UPC under the terms of the GNU GPL version 3.

9.4.2 Function Documentation

9.4.2.1 def mwfiltersgui.__init__ (*self, posPars, keyPars*)

Derived class constructor.

List of leftYData for reverse excitation. List for all Tabs. List of rightYData for reverse excitation. List for all Tabs.

Definition at line 101 of file mwfiltersgui.py.

9.4.2.2 def mwfiltersgui.appendReverseData (*self, leftYDataRev, rightYDataRev*)

Set YData for reverse excitation.

This method appends data to a list for all tabs, so it must be called for all tabs sequentially in tab order.

Parameters

leftYDataRev	= Left-y data for reverse excitation.
rightYDataRev	= Right-y data for reverse excitation.

Definition at line 182 of file mwfiltersgui.py.

9.4.2.3 def mwfiltersgui.replaceReverseData (*self, nTab, leftYDataRev, rightYDataRev*)

Replace YData for reverse excitation.

Parameters

nTab	= Tab index.
leftYDataRev	= Left-y data for reverse excitation.
rightYDataRev	= Right-y data for reverse excitation.

Definition at line 194 of file mwfiltersgui.py.

9.4.2.4 def mwfiltersgui.runSynthesis (*P, FT, symmetrizeZeros*)

Runs parameter validation, filters synthesis and results plots.

Parameters

P	= Parameters class instance containing the parameter values that will be used in the computation. May be modified during computation.
FT	= FrequencyTransform class instance.
symmetrizeZeros	= Automatically symmetrize Generalized Chebyshev zeros in Hz in order to compute a folded matrix with uniform Q in resonators. Equivalent to answering "Yes" in the GUI when it asks the user if he wants to symmetrize Generalized Chebyshev zeros in Hz.

Returns

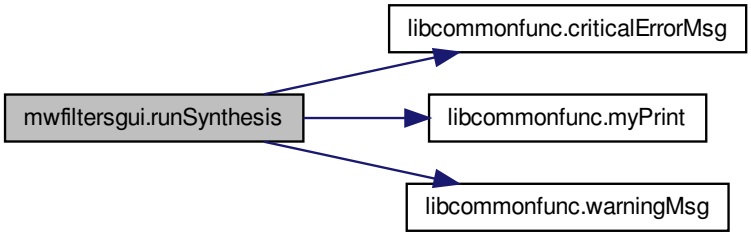
CP = CharPolys class instance, containing Characteristic Polynomials. Losses are included.
CM = CouplingMatrices class instance, containing Coupling Matrices. Losses are included.
SP = Sparameters class instance, containing frequency axis and [S] data. Losses are included.
done = True if Synthesis has been done without errors, False otherwise.

Definition at line 5445 of file mwfiltersgui.py.

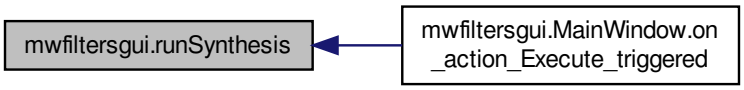
References libcommonfunc.criticalErrorMsg(), libcommonfunc.myPrint(), and libcommonfunc.warningMsg().

Referenced by mwfiltersgui.MainWindow.on_action_Execute_triggered().

Here is the call graph for this function:



Here is the caller graph for this function:



9.4.2.5 def mwfiltersgui.sizeHint (*self*)

Todo Reimplment myTableWidget.resizeEvent() function to reduce size of table when there are no ScrollBars

Definition at line 990 of file mwfiltersgui.py.

9.4.2.6 def mwfiltersgui.updateExcitation (*self, value*)

Update energy and power plots after excitation value has been changed by the user.

Y-axis are autoscaled after update.

Definition at line 161 of file mwfiltersgui.py.

9.4.2.7 def mwfiltersgui.updateInputPower (*self, value*)

Update energy and power plots after inputPower value has been changed by the user.

Y-axis are autoscaled after update.

Definition at line 146 of file mwfiltersgui.py.

9.4.2.8 def mwfiltersgui.usage (*st, exit_code*)

Print error message and command line help.

Parameters

st	= Error message string.
exit_code	= program exit return code.

Definition at line 5575 of file mwfiltersgui.py.

9.4.3 Variable Documentation

9.4.3.1 string mwfiltersgui.applicationName = "Microwave filters GUI"

Program name as run by the user.

Definition at line 5608 of file mwfiltersgui.py.

9.4.3.2 mwfiltersgui.importedExtraFilters = False

Variable set to True if the non-free libextrafilters is available.

Otherwise set to False.

Definition at line 74 of file mwfiltersgui.py.

9.4.3.3 mwfiltersgui.importedLossyFilters = False

Variable set to True if the non-free liblossyfilters is available.

Otherwise set to False.

Definition at line 86 of file mwfiltersgui.py.

9.4.3.4 tuple mwfiltersgui.P = readParamFile(fileName)

Parameters class instance containing filter and synthesis parameters read from the parameters file.

Definition at line 5677 of file mwfiltersgui.py.

9.4.3.5 string mwfiltersgui.stFloatPoint = r'((\d+\.?\d*|\d*\.?\d+)([Ee][-+]?\d+)?)'

Create validator strings and regular expressions.

The are global variables to be accessible from everywhere.

Definition at line 5697 of file mwfiltersgui.py.

Chapter 10

Class Documentation

10.1 dbplot.AutoScaleZoomerBase Class Reference

Class that stores zoomer base.

Public Member Functions

- def __init__

  Class constructor.
- def updateZoomerBase

  Given a set of max and min values for all axis, computes and returns the zoomer base rectangles.

10.1.1 Detailed Description

Class that stores zoomer base.

It is useul to autoscale to the zoomer base, not using the method setAxisAutoScale, which would scale accounting the masks.

Definition at line 2411 of file dbplot.py.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 def dbplot.AutoScaleZoomerBase.__init__ (*self, leftZoomerBaseRect, rightZoomerBaseRect*)

Class constructor.

Gets the data from dbplot zoomers base in nTab.

Parameters

leftZoomerBase-Rect	= Left zoomer base rectangle (QRectF).
rightZoomer-BaseRect	= Right zoomer base rectangle (QRectF). Left zoomer base. Right zoomer base for autoscale,. Zoomer base for autoscale, Xmin. Zoomer base for autoscale, Xmax. Zoomer base for autoscale, LeftYmin. Zoomer base for autoscale, LeftYmax. Zoomer base for autoscale, RightYmin. Zoomer base for autoscale, RightYmax.

Definition at line 2419 of file dbplot.py.

References dbplot.AutoScaleZoomerBase.leftYmax, dbplot.AutoScaleZoomerBase.leftYmin, dbplot.AutoScale-

ZoomerBase.leftZoomerBaseRect, dbplot.AutoScaleZoomerBase.rightYmax, dbplot.AutoScaleZoomerBase.rightYmin, dbplot.AutoScaleZoomerBase.rightZoomerBaseRect, dbplot.AutoScaleZoomerBase.Xmax, and dbplot.AutoScaleZoomerBase.Xmin.

10.1.3 Member Function Documentation

10.1.3.1 def dbplot.AutoScaleZoomerBase.updateZoomerBase (*self, Xmin, Xmax, leftYmin, leftYmax, rightYmin, rightYmax*)

Given a set of max and min values for all axis, computes and returns the zoomer base rectangles.

The attributes of AutoScaleZoomerBase class instance are updated accordingly.

Returns

leftZoomerBaseRect = Left zoomer base rectangle (QRectF).
rightZoomerBaseRect = Right zoomer base rectangle (QRectF).

Definition at line 2461 of file dbplot.py.

References dbplot.AutoScaleZoomerBase.leftYmax, dbplot.AutoScaleZoomerBase.leftYmin, dbplot.AutoScaleZoomerBase.leftZoomerBaseRect, dbplot.AutoScaleZoomerBase.rightYmax, dbplot.AutoScaleZoomerBase.rightYmin, dbplot.AutoScaleZoomerBase.rightZoomerBaseRect, dbplot.AutoScaleZoomerBase.Xmax, and dbplot.AutoScaleZoomerBase.Xmin.

The documentation for this class was generated from the following file:

- dbplot.py

10.2 dbplot.AxisScalingDlg Class Reference

GUI dialog to contain axis scaling controls.

Public Member Functions

- def __init__

  Constructor: Creates PyQt4.QtGui.Qdialog window from QtDesigner.
- def closeEvent

  Reimplementation of the window close function.

10.2.1 Detailed Description

GUI dialog to contain axis scaling controls.

Parameters

parent	= Parent widget, in this case is mainWindow. Default None.

Definition at line 1870 of file dbplot.py.

10.2.2 Member Function Documentation

10.2.2.1 def dbplot.AxisScalingDlg.closeEvent (*self, event*)

Reimplementation of the window close function.

Sets scalingDlgCurrent attribute to None and hide the window.

Definition at line 1885 of file dbplot.py.

The documentation for this class was generated from the following file:

- dbplot.py

10.3 dbplot.CanvasEventFilter Class Reference

Event filter for the canvas.

Public Member Functions

- def __init__
  Class constructor.
- def eventFilter
  The event filter.

10.3.1 Detailed Description

Event filter for the canvas.

This filter is necessary to pick up events that are not handled by the QwtPlotPicker class, like mouse dragging.

It would be enough (and better practice) to reimplement mousePressEvent() and keyPressEvent() of the QwtPlotCanvas class, but we if we create a derived class from QwtPlotCanvas it is not possible to set an instance of this class to the canvas of a QwtPlot object since there is not setPlotCanvas() method in QwtPlot class.

To handle left mouse clicks it is enough to create a mouseClicked() slot and connect to 'selected' signal of QwtPlotPicker class: self.connect(pickerLeft, SIGNAL('selected(QwtDoublePoint)'), self.mouseClicked) but since we have to implement this filter anyway in order to handle mouse dragging, we will handle also all the other events here.

Reimplementation of mousePressEvent() from QwtPlot does not work, because we need the mouse coordinates obtained from event.pos() to be relative to the QwtPlotCanvas widget, not to the QwtPlot, in order to get the graph coordinates using QwtPlot.invTransform().

Implementing this filter is essentially what the QwtPlotPicker class does, but unfortunately it does not support mouse dragging, only mouse selection.

Definition at line 2221 of file dbplot.py.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 def dbplot.CanvasEventFilter.__init__ (*self, dbplot, hPlot*)

Class constructor.

Parameters

dbplot	= DbPlot class instance, necessary to check the value of some dbPlot flags.
hPlot	= The parent widget. Must be a Qwt.QwtPlot class instance, since non-processed events will be sent to Qwt.QwtPlot.eventFilter(self, object, event).

Definition at line 2229 of file dbplot.py.

References dbplot.MyPicker.dbplot, and dbplot.CanvasEventFilter.dbplot.

10.3.3 Member Function Documentation

10.3.3.1 def dbplot.CanvasEventFilter.eventFilter (*self, object, event*)

The event filter.

When the event has been processed, for mouse movements the return value is not True, in order to allow the event to be processed also by other widgets like QwtPlotPicker.

Definition at line 2242 of file dbplot.py.

The documentation for this class was generated from the following file:

- dbplot.py

10.4 libfreefilters.CharPolys Class Reference

Characteristic polynomials class.

Public Member Functions

- def __init__

  Compute characteristic polynomials and characteristic constants.
- def symmetrizeTZ

  A list of transmission zeros in unnormalized frequency (Hz) is symmetrized in the normalized frequency axis.
- def checkUnormSymTZ

  Check if list of transmission zeros (in Hz) is symmetric with respect to center frequency f0.
- def checkNormSymTZ

  Check if list of normalized transmission zeros is symmetric with respect to zero.
- def polyStr

  Return string with polynomial coefficients separated by commas.
- def saveCharPolys

  Save Characteristic Polynomials and constants in output file.

10.4.1 Detailed Description

Characteristic polynomials class.

```
Characteristic polynomials and constants are defined as [TN 101 eq. (1)] :
```

$$S_{11}(s) = \left[\frac{F(s)}{\varepsilon_r E(s)}\right], S_{12}(s) = S_{21} = \left[\frac{P(s)}{\varepsilon E(s)}\right], S_{22}(s) = \left[\frac{(-1)^N F(s)^*}{\varepsilon_r E(s)}\right]$$

where $F(s)^* = F^*(-s)$ is the 'Paraconjugate' polynomial of $F(s)$ [Cameron pp. 208 footnote] and is computed using polyConj function.

Definition at line 83 of file libfreefilters.py.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 def libfreefilters.CharPolys.__init__ (*self, P, FT, symmetrizeZeros* = None)

Compute characteristic polynomials and characteristic constants.

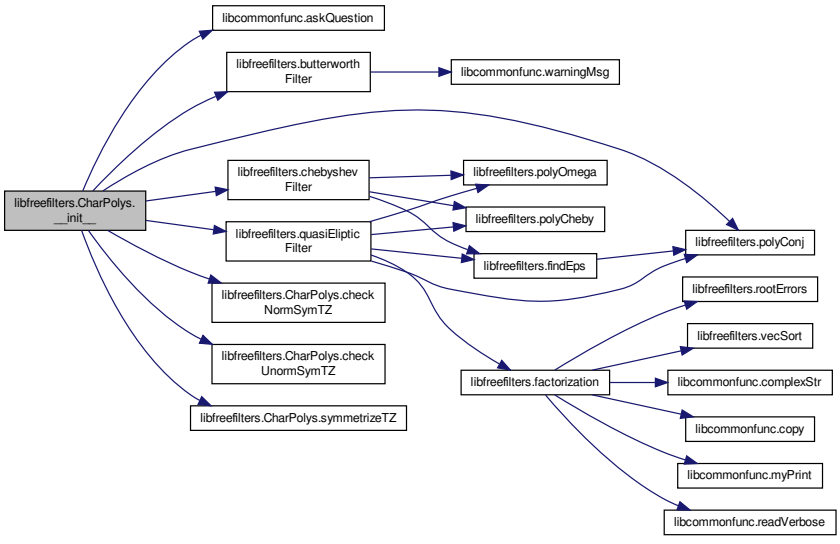
```
@param P = Parameter class instance containing filter, synthesis and sweep parameters.
@param  FT = Frequency transformation class instance.
@param symmetrizeZeros = Automatically symmetrize Generalized Chebyshev zeros in Hz in order to comput
                                                    Equivalent to answering "Yes" in the GUI when it asks the
@return CharPolys class instance.
```

Characteristic polynomial $P(s)$ Characteristic polynomial $E(s)$ Characteristic polynomial $F(s)$ Characteristic polynomial $F_{22}(s)$ Characteristic constant ε in the denominator of S_{12} and S_{21} Characteristic constant ε_R in the denominator of S_{11} Roundoff error estimation in the computation of lossless characteristic polynomials, given by the rootsErr return value in CharPolys.factorization method. Value 'a' of the (s-a)/(s+a) zero/pole added in lossy filters synthesis case 3.

Definition at line 93 of file libfreefilters.py.

References libcommonfunc.askQuestion(), libfreefilters.butterworthFilter(), libfreefilters.chebyshevFilter(), libfreefilters.CharPolys.checkNormSymTZ(), libfreefilters.CharPolys.checkUnormSymTZ(), libfreefilters.CharPolys.eps, libfreefilters.CharPolys.epsR, libfreefilters.CharPolys.Es, libfreefilters.CharPolys.F22s, libfreefilters.CharPolys.flagCase1, libfreefilters.CharPolys.flagCase2, libfreefilters.CharPolys.flagCase3, libfreefilters.CharPolys.flagComplexTZ, libfreefilters.CharPolys.flagExtraFilter, libfreefilters.CharPolys.flagSymTZ, libfreefilters.CharPolys.flagSymTZnorm, libfreefilters.CharPolys.flagSymTZunorm, libfreefilters.CharPolys.Fs, libfreefilters.CharPolys.nuqDelta, libfreefilters.polyConj(), libfreefilters.CharPolys.Ps, libfreefilters.quasiElipticFilter(), libfreefilters.CharPolys.rootsErr, and libfreefilters.CharPolys.symmetrizeTZ().

Here is the call graph for this function:



10.4.3 Member Function Documentation

10.4.3.1 def libfreefilters.CharPolys.checkNormSymTZ (*self, TZ*)

Check if list of normalized transmission zeros is symmetric with respect to zero.

Parameters

TZ	= List of normalized transmission zeros.

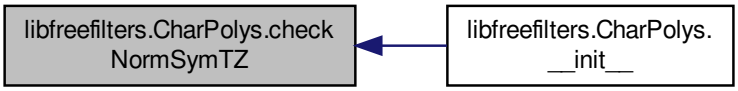
Returns

symmetricFlag = True is zeros are symmetric with relative error $<$ 1e-3, False otherwise.

Definition at line 282 of file libfreefilters.py.

Referenced by libfreefilters.CharPolys.__init__().

Here is the caller graph for this function:



10.4.3.2 def libfreefilters.CharPolys.checkUnormSymTZ (*self, TZ, FT*)

Check if list of transmission zeros (in Hz) is symmetric with respect to center frequency f0.

Parameters

TZ	= List of transmission zeros in Hz.
FT	= Frequency transform class instance. It will be used to get f0.

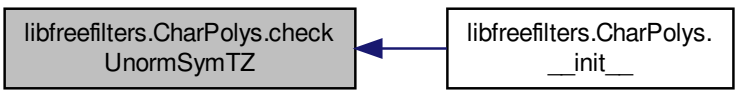
Returns

symmetricFlag = True is zeros are symmetric with relative error $<$ 1e-3, False otherwise.

Definition at line 264 of file libfreefilters.py.

Referenced by libfreefilters.CharPolys.__init__().

Here is the caller graph for this function:



10.4.3.3 def libfreefilters.CharPolys.polyStr (*self, pol, prec*)

Return string with polynomial coefficients separated by commas.

Rounds polynomial coefficients to given precision using complexStr() function.

Parameters

pol	= Polynomial (numpy.poly1d object).
prec	= Number of significant digits (int).

Returns

stcoef = String representation of pol.

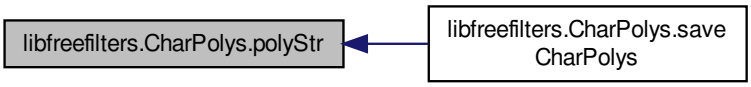
Definition at line 302 of file libfreefilters.py.

References libcommonfunc.complexStr().

Referenced by libfreefilters.CharPolys.saveCharPolys().

Here is the call graph for this function:

Here is the caller graph for this function:



10.4.3.4 def libfreefilters.CharPolys.saveCharPolys (*self, P, prec*)

Save Characteristic Polynomials and constants in output file.

Rounds polynomial coefficients to given precision using complexStr() function. Polynomials are of type numpy.poly1d.

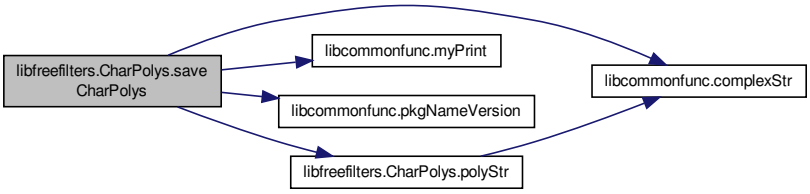
Parameters

P	= Parameter class instance.
prec	= Number of correct significant digits to print.

Definition at line 321 of file libfreefilters.py.

References libcommonfunc.complexStr(), libfreefilters.CharPolys.eps, libfreefilters.CharPolys.epsR, libfreefilters.CharPolys.Es, libfreefilters.CharPolys.Fs, libcommonfunc.myPrint(), libcommonfunc.pkgNameVersion(), libfreefilters.CharPolys.polyStr(), and libfreefilters.CharPolys.Ps.

Here is the call graph for this function:



10.4.3.5 def libfreefilters.CharPolys.symmetrizeTZ (*self, TZ, FT*)

A list of transmission zeros in unnormalized frequency (Hz) is symmetrized in the normalized frequency axis.

The positive zeros are averaged with the corresponding negative zeros in the normalized frequency axis.

Parameters

TZ	= List of transmission zeros. Unnormalized frequency (Hz).
FT	= Frequency transform class instance.

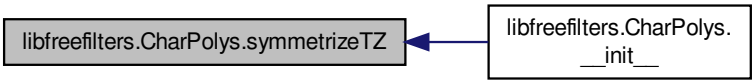
Returns

TZsym = List of transmission zeros, symmetrized n the normalized frequency axis. Unnormalized frequency (Hz).

Definition at line 245 of file libfreefilters.py.

Referenced by libfreefilters.CharPolys.__init__().

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- libfreefilters.py

10.5 libcommonfunc.CouplingMatrices Class Reference

Coupling matrices class.

Public Member Functions

- def __init__

Compute Coupling Matrices for all the selected topologies.

- def matrixElementsEps

  Convenience function that computes the threshold to determine if coupling matrix elements (or their real or imaginary parts) are null.

- def updateCM_error

  Convenience function that updates the error estimation in self.CM_error.

- def transCouplingMatrix

  Computation of the $N+2$ transversal coupling matrix of the filter defined by its caracteristic polynomials and constants.

- def polyExDiv

  Polynomial exact division.

- def tcm2fcm

  This function transforms the transversal $N+2$ coupling matrix M into the coupling matrix in Folded Canonical Form (FCM) $Mfcm$.

- def tcm2arrow

  This function transforms the transversal $N+2$ coupling matrix M into the coupling matrix in Arrow or Wheel Form $Marrow$.

- def arrow2triplet

  This function transforms the coupling matrix in Arrow or Wheel Form $Marrow$ into the coupling matrix in Trisections Form $Mtriplet$.

- def fcm2culdesac

  This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Cul de Sac coupling matrix $Mculdesac$.

- def fcm2cqs

  This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Cascaded Quartets coupling matrix $Mcqs$.

- def fcm2pfitzenmaier

  This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Pfitzenmaier coupling matrix $Mpfi$.

- def fcm2inlineAsymmetric

  This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Inline Asymmetric coupling matrix $Minasym$.

- def fcm2inlineSymmetric

  This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Inline Symmetric coupling matrix $Minsym$.

- def readCouplingMatrix

  Read coupling matrix from disk file.

- def saveCouplingMatrices

  Save coupling matrices in output file.

- def uniformQFCMOrder4

  This function transforms the Folded $N+2$ Canonical Form (FCM) coupling matrix of a filter of order 4 into an N+4 matrix so that all the resonant nodes have equal quality factor Q.

- def uniformQFCMOrder6

  This function transforms the Folded $N+2$ Canonical Form (FCM) coupling matrix of a filter of order 6 into an N+4 matrix so that all the resonant nodes have equal quality factor Q.

- def uniformQOrder6Optimize

  This function performs the hyperbolic rotations in TN102.3 sec.3.3.

- def uniformQFCMOrder6Case3

  This function transforms the Folded $N+2$ Canonical Form (FCM) coupling matrix of a filter of order 6 into an N+4 matrix so that all the resonant nodes have equal quality factor Q.

- def uniformQOrder6Case3Optimize

  This function performs the hyperbolic rotations in TN102.3.

10.5.1 Detailed Description

Coupling matrices class.

```
If \form#0 is the impedance matriz that relates the current at the network loops with the voltage sources with
the compupling matriz \form#1 is such that [TN 102.1, sec. 3.2]:
```

$$[Z] = [jM + sI]$$

Todo Call numpy functions as methods of array class, whenever possible.

Definition at line 2844 of file libcommonfunc.py.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 def libcommonfunc.CouplingMatrices.__init__ (*self, P, CP, Yo, FT, SP*)

Compute Coupling Matrices for all the selected topologies.

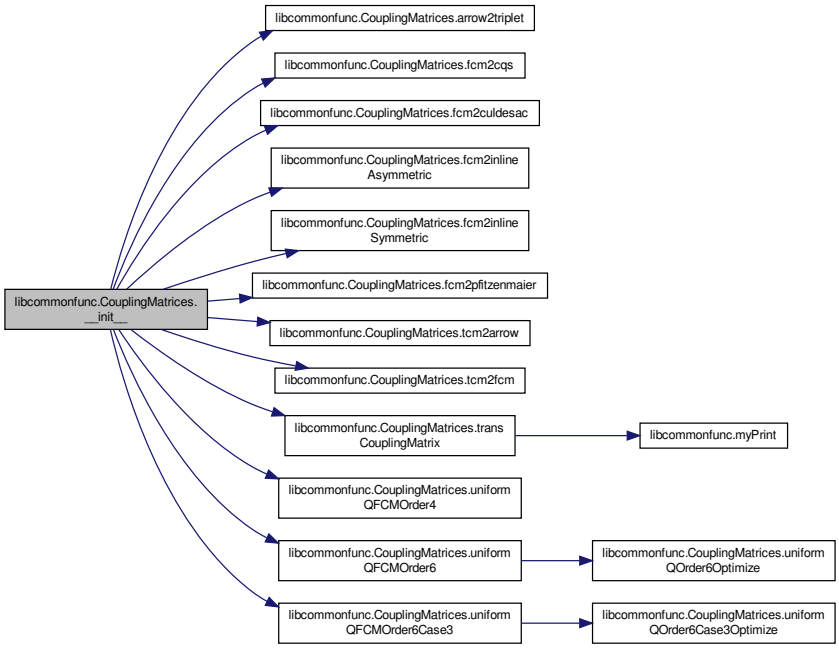
```
@param P  = Parameters class instance, containing filter, synthesis and sweep parameters.
@param CP = CharPolys class instance, containing Characteristic Polynomials Es, Ps, Fs and constants eps,
@param Yo = Termination admitance.
@param FT = Frequency transformation class instance.
@param SP = SParameters class instance.
@return CM = CouplingMatrices class instance.
```

Frequency transform used to compute these coupling matrices. It must be used when creating new matrices or reading matrices from disk. SParameters class instance containing the frequency sampling and SP.fromCoupling-Matrix() method. Will be used in the sensitivity analysis. Uniform Q folded coupling matrix N+4. List of available coupling matrices Last index of CM selection comboBox. Initially it is equal to zero(first matrix in list). Roundoff error in the computation of the Tranversal Coupling Matrix N+2. The error is obtained as the maximum of the largest of the remainders in the computation of $y_{22n}(s)$ and $y_d(s)$ and the constant term in the partial fraction expansion of $y_{21n}(s)/Y_d(s)$ Current coupling matrix, after rotations. Is the first in the list. Backup list of self.MatQ values, for undo. Must use deep copy, since self.MatQ will be modified by edit actions. Backup list of actions

Definition at line 2856 of file libcommonfunc.py.

References libcommonfunc.CouplingMatrices.arrow2triplet(), libcommonfunc.CouplingMatrices.bkpActions, libcommonfunc.CouplingMatrices.bkpMatQ, libcommonfunc.CouplingMatrices.CM_error, libcommonfunc.Coupling-Matrices.fcm2cqs(), libcommonfunc.CouplingMatrices.fcm2culdesac(), libcommonfunc.CouplingMatrices.-fcm2inlineAsymmetric(), libcommonfunc.CouplingMatrices.fcm2inlineSymmetric(), libcommonfunc.Coupling-Matrices.fcm2pfitzenmaier(), libcommonfunc.CouplingMatrices.fCM_uniQ, libcommonfunc.Sparameters.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.CouplingMatrices.FT, libcommonfunc.CouplingMatrices.indexC-M, libcommonfunc.CouplingMatrices.listM, libcommonfunc.CouplingMatrices.MatQ, libcommonfunc.Coupling-Matrices.SP, libcommonfunc.CouplingMatrices.tcm2arrow(), libcommonfunc.CouplingMatrices.tcm2fcm(), libcommonfunc.-CouplingMatrices.transCouplingMatrix(), libcommonfunc.CouplingMatrices.uniformQFCMOrder4(), libcommonfunc.-CouplingMatrices.uniformQFCMOrder6(), and libcommonfunc.CouplingMatrices.uniformQFCMOrder6Case3().

Here is the call graph for this function:



10.5.3 Member Function Documentation

10.5.3.1 def libcommonfunc.CouplingMatrices.arrow2triplet (*self, MatQarrow, TZ*)

This function transforms the coupling matrix in Arrow or Wheel Form $Marrow$ into the coupling matrix in Trisections Form $Mtriplet$.

Parameters

MatQarrow	= MatrixQ class instance, containing the Arrow topology coupling matrix.
TZ	= Transmission Zeros.

Returns

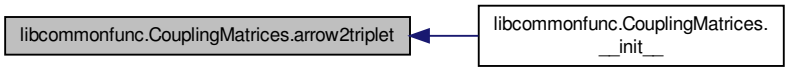
MatQtriplet = MatrixQ class instance, containing the Trisections topology coupling matrix.

[Cameron]

Definition at line 3431 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.3.2 def libcommonfunc.CouplingMatrices.fcm2cqs (*self, MatQfcm, r, UpDownNone*)

This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Cascaded Quartets coupling matrix $Mcqs$.

Parameters

MatQfcm	= MatrixQ class instance, containing the Folded Canonical Form (FCM) coupling matrix.
r	= If we want to shift the quartet, r indicates the position of the first resonator node of the quartet. It should be 0 if we do not want to shift it.
UpDownNone	= 'Up' : Shift the quartet above the diagonal. 'Down' : Shift the quartet below the diagonal. 'None' : No shifting.

Returns

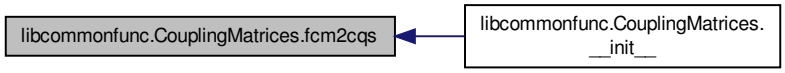
MatQcqs = MatrixQ class instance, containing the Cascaded Quartets coupling matrix.

[Cameron]

Definition at line 3507 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.3.3 def libcommonfunc.CouplingMatrices.fcm2culdesac (*self, MatQfcm, TZ*)

This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Cul de Sac coupling matrix $Mculdesac$.

Parameters

MatQfcm	= MatrixQ class instance, containing the Folded Canonical Form (FCM) coupling matrix.
TZ	= Transmission Zeros.

Returns

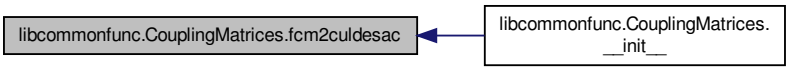
MatQculdesac = MatrixQ class instance, containing the Cul de Sac coupling matrix.

[Cameron]

Definition at line 3466 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.3.4 def libcommonfunc.CouplingMatrices.fcm2inlineAsymmetric (*self, MatQfcm*)

This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Inline Asymmetric coupling matrix $Minasym$.

Parameters

MatQfcm	= MatrixQ class instance, containing the Folded Canonical Form (FCM) coupling matrix.

Returns

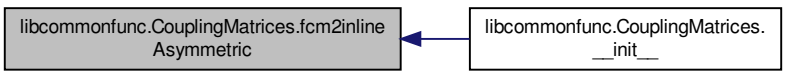
MatQinasym = MatrixQ class instance, containing the Inline Asymmetric coupling matrix.

[Cameron]

Definition at line 3601 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.3.5 def libcommonfunc.CouplingMatrices.fcm2inlineSymmetric (*self, MatQfcm, TZ*)

This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Inline Symmetric coupling matrix $Minsym$.

Parameters

MatQfcm	= MatrixQ class instance, containing the Folded Canonical Form (FCM) coupling matrix.
TZ	= Transmission Zeros.

Returns

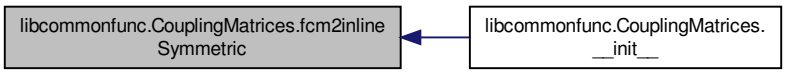
MatQinsym = MatrixQ class instance, containing the Inline Symmetric coupling matrix.

[Cameron]

Definition at line 3687 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.3.6 def libcommonfunc.CouplingMatrices.fcm2pfitzenmaier (*self, MatQfcm*)

This function transforms the coupling matrix in Folded Canonical Form (FCM) $Mfcm$ into the Pfitzenmaier coupling matrix $Mpfi$.

Parameters

MatQfcm	= MatrixQ class instance, containing the Folded Canonical Form (FCM) coupling matrix.

Returns

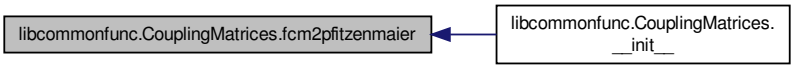
MatQpfi = MatrixQ class instance, containing the Pfitzenmaier coupling matrix.

[Cameron]

Definition at line 3574 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.3.7 def libcommonfunc.CouplingMatrices.matrixElementsEps (*self, st* = `None`)

Convenience function that computes the threshold to determine if coupling matrix elements (or their real or imaginary parts) are null.

Parameters

st	= String with relevant info. Default None.

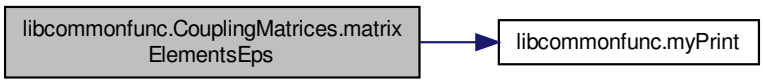
Returns

eps = Threshold.

Definition at line 2962 of file libcommonfunc.py.

References libcommonfunc.CouplingMatrices.CM_error, and libcommonfunc.myPrint().

Here is the call graph for this function:



10.5.3.8 def libcommonfunc.CouplingMatrices.polyExDiv (*self, num, den*)

Polynomial exact division.

Computes quot and remain polynomials so that num(s) = quot(s)∗den(s) + error(s).

Parameters

num	= Dividend.
den	= Divisor.

Returns

quot = Quotient.
error = Error in the computation of the quotient. Is equal to: num(s)-quot(s)∗den(s)

10.5.4 Algorithm

Since numpy.polydiv or the equivalent numpy.poly1d polynomial division fails for complex coefficients, at least in numpy v1.1.1, here we evaluate num(s)/den(s) for a finite number of points in the imaginary axis and use polyfit to build the polynomial that passes trhough this points. The remainder is the error.

This function does not work well with the example in "Prescribed_insertion_losses_K11+k21_asymmetrical.par" (the example in journal paper publication), while numpy.polydiv from numpy v1.3.0 works perfectly.

Definition at line 3318 of file libcommonfunc.py.

10.5.4.1 def libcommonfunc.CouplingMatrices.readCouplingMatrix (*self, fileName*)

Read coupling matrix from disk file.

Parameters

fileName	= File name (string).

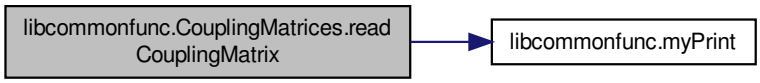
Returns

MatQ = Coupling matrix (MatrixQ class instance.)

Definition at line 3829 of file libcommonfunc.py.

References libcommonfunc.Sparameters.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.CouplingMatrices.FT, and libcommonfunc.myPrint().

Here is the call graph for this function:



10.5.4.2 def libcommonfunc.CouplingMatrices.saveCouplingMatrices (*self, P, SP, precCM, precEP*)

Save coupling matrices in output file.

The matrices are attributes of CouplingMatrices class, and of type numpy.array.

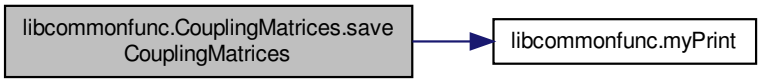
Parameters

P	= Parameter class instance.
SP	= SParameters class instance.
precCM	= Number of significant digits to save in coupling matrix and Q (int).
precEP	= Number of significant digits to save in energy and power (int).

Definition at line 3876 of file libcommonfunc.py.

References libcommonfunc.CouplingMatrices.listM, and libcommonfunc.myPrint().

Here is the call graph for this function:



10.5.4.3 def libcommonfunc.CouplingMatrices.tcm2arrow (*self, MatQ*)

This function transforms the transversal $N+2$ coupling matrix M into the coupling matrix in Arrow or Wheel Form $Marrow$.

Parameters

MatQ	= MatrixQ class instance, containing the transversal $N+2$ coupling matrix.

Returns

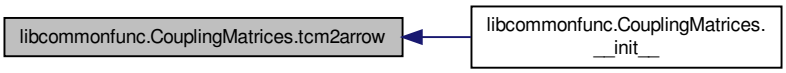
MatQarrow = MatrixQ class instance, containing the Arrow topology coupling matrix.

[Cameron]

Definition at line 3406 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.4.4 def libcommonfunc.CouplingMatrices.tcm2fcm (*self, MatQ*)

This function transforms the transversal $N+2$ coupling matrix M into the coupling matrix in Folded Canonical Form (FCM) $Mfcm$.

```
@param MatQ = MatrixQ class instance, containing the transversal \form#189 coupling matrix.
@return MatQfcm =  MatrixQ class instance, containing the Folded Canonical Form (FCM) coupling matrix.

[TN 102.2 sec. 9.2]
The procedure to obtain the matrix [FCM] consists in setting to zero step by step elements of the matrix [

-Step 1: We remove all the elements of the row 1 (In total there are nerk(1) elements).

-Step 2: We remove all the elements of the column 1 (In total there are neck(1) elements).

-Step 3: The same as in step 1 with the row 2 (In total there are nerk(2) elements).

-Step 4: The same as in step 2 with the column 2 (In total there are neck(2) elements).
```

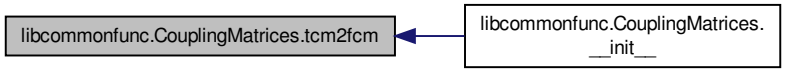
⋮

```
-Step numSteps-1:  We remove all the elements of the row numRows (In total there are nerk(numRows) element

-Step numSteps: We remove all the elements of the column numColumns (In total there are nerk(numColumns) e

The last two steps consist in removing the last remaining row and the last remaining column of the matrix
```

Definition at line 3352 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.4.5 def libcommonfunc.CouplingMatrices.transCouplingMatrix (*self, P, CP, Yo, FT*)

Computation of the $N+2$ transversal coupling matrix of the filter defined by its caracteristic polynomials and constants.

```
Polynomials are of type numpy.poly1d.
This function sets the self.CM_error attribute to the roundoff error in the computation of M,
given by the largest of the remainders in the computation of \form#128 and \form#129
and the constant term in the partial fraction expansion of \form#130
@param P  = Parameters class instance, containing filter, synthesis and sweep parameters.
@param CP = CharPolys class instance, containing Characteristic Polynomials Es, Ps, Fs and constants e
@param Yo = Termination admitance.
@param FT = Frequency transformation class instance.
@return MatQ = MatrixQ class instance, containing the N+2 transversal coupling matrix.

@section Smat Scattering matrix [S]

The first step to find the \form#131 transversal coupling matrix of a two-port network is finding the assc

By definition, the scattering matrix of a two-port network is  [TN 102.1, eq (1)]:
```

$$[S] = \begin{bmatrix} S_{11}(s) & S_{12}(s) \\ S_{21}(s) & S_{22}(s) \end{bmatrix} = \frac{1}{S_d(s)} \begin{bmatrix} S_{11n}(s) & S_{12n}(s) \\ S_{21n}(s) & S_{22n}(s) \end{bmatrix}$$

```
The [S] parameters are computed from the polinomials as:
```

$$S_{11} = \frac{1}{\varepsilon_R} \frac{F(s)}{E(s)}$$

$$S_{21} = S_{12} = \frac{1}{\varepsilon} \frac{P(s)}{E(s)}$$

$$S_{22} = \frac{1}{\varepsilon_R} \frac{F_{22}(s)}{E(s)} = (-1)^N \frac{1}{\varepsilon_R} \frac{F(s)^*}{E(s)}$$

except for asymmetrical "Prescribed Insertion Loss technique" case 1 (kS21+kS11), where [S] parameters are computed from the lossless case above as:

$$S_{11} = k_{11} S_{11\text{lossless}}$$

$$S_{21} = S_{12} = k_{21} S_{21\text{lossless}}$$

$$S_{22} = k_{22} \left(1 - \left(\frac{k_{21}}{k_{11}}\right)^2\right) + \left(\frac{k_{21}}{k_{11}}\right)^2 S_{11\text{lossless}}$$

```
and hence the numerators and denominator of the \form#137 rational polynomials are:
```

$$S_{11n}(s) = \frac{F(s)}{\varepsilon_R}$$

$$S_{12n}(s) = S_{21n} = \frac{P(s)}{\varepsilon}$$

$$S_d(s) = E(s)$$

$$S_{22n}(s) = \frac{F_{22}(s)}{\varepsilon_R}$$

except for asymmetrical "Prescribed Insertion Loss technique" case 1 (kS21+kS11), where they are computed as

$$S_{11n}(s) = k_{11} \frac{F(s)}{\varepsilon_R}$$

$$S_{12n}(s) = S_{21n} = k_{21} \frac{P(s)}{\varepsilon}$$

$$S_d(s) = E(s)$$

$$S_{22n}(s) = k_{22} \left(1 - \left(\frac{k_{21}}{k_{11}}\right)^2\right) S_d(s) + \left(\frac{k_{21}}{k_{11}}\right)^2 S_{11n}(s)$$

In practice, the software divides ε_R by k_{11} and ε by k_{21} , and stores the updated values to automatically apply the k_{11} and k_{21} whenever computing S_{11n} and S_{21n} .

10.5.5 Admitance matrix [Y]

The next step is the computation of $y_{21}(s)$ and $y_{22}(s)$ of the admitance matrix $[Y]$. [Y] is related to [S] as [TN 102.1, eq (12)]:

$$[Y] = \frac{1}{y_d(s)} \begin{bmatrix} y_{11n}(s) & y_{12n}(s) \\ y_{21n}(s) & y_{22n}(s) \end{bmatrix}$$

Where:

$$y_{21n}(s) = -2S_{21n}(s)$$

$$y_{22n}(s) = \frac{(S_d(s)+S_{11n}(s))(S_d(s)-S_{22n}(s))+S_{12n}S_{21n}}{S_d(s)}$$

$$y_{11n}(s) = \frac{(S_d(s)-S_{11n}(s))(S_d(s)+S_{22n}(s))+S_{12n}S_{21n}}{S_d(s)}$$

$$y_d(s) = \frac{(S_d(s)+S_{11n}(s))(S_d(s)+S_{22n}(s))-S_{12n}S_{21n}}{Y_0(s)S_d(s)}$$

Y_0 = Load admitance at the end of the network.

```
@section Mmat N+2 Coupling Matrix
A partial fraction expansion of \form#156 yields [Cameron, eq. (8.41)] [TN 101.1, eq (12)]:
```

$$[Y] = \frac{1}{y_d(s)} \begin{bmatrix} y_{11n}(s) & y_{12n}(s) \\ y_{21n}(s) & y_{22n}(s) \end{bmatrix} = j \begin{bmatrix} 0 & K_\infty \\ K_\infty & 0 \end{bmatrix} + \sum_{k=1}^{N} \frac{1}{s-j\lambda_k} \cdot \begin{bmatrix} r_{11k}(s) & r_{12k}(s) \\ r_{21k}(s) & r_{22k}(s) \end{bmatrix} + \begin{bmatrix} KK_{11} & 0 \\ 0 & KK_{22} \end{bmatrix}$$

```
<ul>

<li> Computation of the residuals \form#158 and \form#159:

The residuals \form#158 and \form#159 can be found as [Cameron, pp. 294, footnote]:
```

$$r_{22k}(s) = \left.\frac{y_{22n}(s)}{y'_d(s)}\right|_{s=j\lambda_k} \qquad r_{11k}(s) = \left.\frac{y_{11n}(s)}{y'_d(s)}\right|_{s=j\lambda_k} \qquad r_{12k}(s) = r_{21k}(s) = \left.\frac{y_{21n}(s)}{y'_d(s)}\right|_{s=j\lambda_k}$$

```
where:
    <ul>
    <li> \form#161 is the first derivative of \form#162. It is variable Ydp in the code.
    <li> \form#163 are the roots of \form#161.
    </ul>
The residuals are actually computed using scipy.signal.residue() function in order to obtain the constant
    <ul>
    <li>In \form#164 the constant term is the source-load coupling term \form#165 :
```

$$\frac{y_{21n}(s)}{y_d(s)} = \sum_{k=1}^{N} \frac{r_{21k}}{s-j\lambda_k} + jK_\infty$$

In $[Y_{22}]$ and $[Y_{11}]$, "Prescribed Insertion Loss technique" case 1, there is a constant term $KK \neq 0$ that must be included in the N+2 transversal coupling matrix [TN 102.1, eq (13)]:

$$\frac{y_{22n}(s)}{y_d(s)} = \sum_{k=1}^{N} \frac{r_{22k}}{s-j\lambda_k} + KK_{22}$$

$$\frac{y_{11n}(s)}{y_d(s)} = \sum_{k=1}^{N} \frac{r_{11k}}{s-j\lambda_k} + KK_{11}$$

Computation of K_∞:

K_∞ is a real constant equal to 0 except for fully canonical filters where the number of finite-position transmission zeros n_{fz} is equal to the filter degree N. For lossless filters it can be computed as [Cameron, eq (8.43a)]:

$$K_\infty = \frac{\varepsilon_R}{\varepsilon} \frac{1}{(\varepsilon_R+1)}$$

However, this formula is not valid for lossy synthesis, so we set K_∞ equal to the imaginary part of the constant term in the polynomial expansion of $y'_{21n}(s)/y_d(s)$.

Computation of the eigenvectors T_{1k} and T_{Nk}:

The eigenvectors T_{Nk} and T_{1k} can be found as [Cameron eq. (8.55)]:

$$T_{Nk} = \sqrt{r_{22k}}$$

$$T_{1k} = r_{21k}/\sqrt{r_{22k}}$$

```
<li> Build the \form#131 transversal coupling matrix \form#79:

The \form#131 transversal coupling matrix is built from the orthogonal vectors \form#178 and \form#177
and from the eigenvalues \form#181, as follows [Cameron, fig 8.18]:
```

$$[M] = \begin{bmatrix} -jKK_{11} & \cdots & T_{1k} & \cdots & K_\infty \\ \vdots & \ddots & & 0 & \vdots \\ T_{1k} & & -\lambda_k & & T_{Nk} \\ \vdots & 0 & & \ddots & \vdots \\ K_\infty & \cdots & T_{Nk} & \cdots & -jKK_{22} \end{bmatrix}$$

The steps to build the matrix are the following:

1. We put the vector T1k at the first row (between the first and the last column not-included)
2. We put the value of Kinf at the last column of the first row
3. We put the vector TNk at the last column (between the first and the last row not-included)
4. As the matrix M is symmetric, we copy the upper triangle to the lower
5. We put the lambdak (with opposite sign) at the main diagonal (between the first and the last element not-included)
6. If there is a remaining constants KK_{11} and KK_{22} in the partial fraction expansion of Y_{11n}/Y_d and Y_{22n}/Y_d, set M[0,0] to $-jKK_{11}$ and M[N+1,N+1] to $-jKK_{22}$.

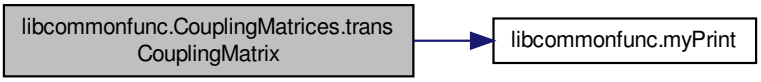
Todo The function "r21k, p21k, remainder = sc.signal.residue(Y21np,Yd)" does not give the correct remainder when it is complex but in the tested cases it does not afect. This function should be improved for the future.

Definition at line 3168 of file libcommonfunc.py.

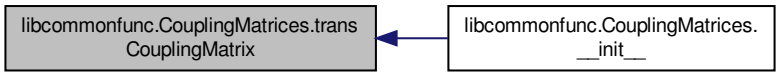
References libcommonfunc.CouplingMatrices.CM_error, and libcommonfunc.myPrint().

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.5.1 def libcommonfunc.CouplingMatrices.uniformQFCMOrder4 (*self, MatQfcm2*)

This function transforms the Folded $N+2$ Canonical Form (FCM) coupling matrix of a filter of order 4 into an N+4 matrix so that all the resonant nodes have equal quality factor Q.

```
(Valid for Butterworth, Chevyshev, Quasieliptic and Generalized Chebyshev with symmetric transmission zero
@param MatQfcm2 = MatrixQ class instance, containing the Folded \form#189 Canonical Form (FCM) coupling ma
@return MatQfcm4 = MatrixQ class instance, containing the Folded \form#200 Canonical Form (FCM) coupling m

For a better description see TN102.3 sec. 3.2.

First of all we add two non-resonant nodes, what means converting the N+2 folded matrix into the N+4. Ther
```

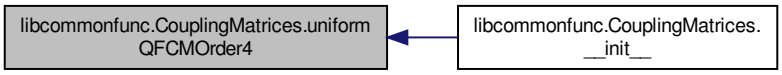
$$\alpha = \frac{1}{2} ln \left[\frac{\frac{M_{01}^2}{G} \pm \sqrt{4G_1 \frac{M_{01}^2}{G} - 4G_1^2 + 16M_{12}^2}}{2G_1 + \frac{M_{01}^2}{G} + 4M_{12}} \right]$$

Finally, we need to re-scale with a value h so that the non-resonant nodes have a sum of imaginary parts equal to 0.

Definition at line 3904 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the caller graph for this function:



10.5.5.2 def libcommonfunc.CouplingMatrices.uniformQFCMOrder6 (*self, MatQfcm2*)

This function transforms the Folded $N+2$ Canonical Form (FCM) coupling matrix of a filter of order 6 into an N+4 matrix so that all the resonant nodes have equal quality factor Q.

(Valid for Butterworth, Chevyshev, Quasieliptic and Generalized Chebyshev with symmetric transmission zeros).

Parameters

MatQfcm2	= MatrixQ class instance, containing the Folded $N+2$ Canonical Form (FCM) coupling matrix.

Returns

MatQfcm4 = MatrixQ class instance, containing the Folded $N+4$ Canonical Form (FCM) coupling matrix with uniform Q.

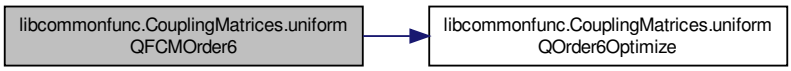
See TN102.3 sec. 3.3.

Definition at line 3945 of file libcommonfunc.py.

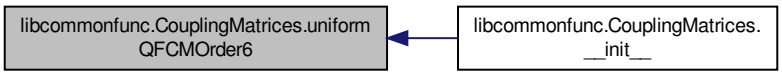
References libcommonfunc.CouplingMatrices.uniformQOrder6Optimize().

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.5.3 def libcommonfunc.CouplingMatrices.uniformQFCMOrder6Case3 (*self, MatQfcm2*)

This function transforms the Folded $N+2$ Canonical Form (FCM) coupling matrix of a filter of order 6 into an N+4 matrix so that all the resonant nodes have equal quality factor Q.

(Valid for Butterworth, Chevyshev, Quasieliptic and Generalized Chebyshev with symmetric transmission zeros).

Parameters

MatQfcm2	= MatrixQ class instance, containing the Folded $N+2$ Canonical Form (FCM) coupling matrix.

Returns

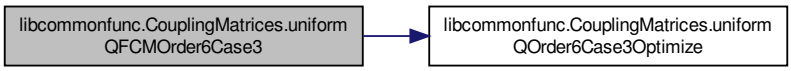
MatQfcm4 = MatrixQ class instance, containing the Folded $N+4$ Canonical Form (FCM) coupling matrix with uniform Q.

Definition at line 4008 of file libcommonfunc.py.

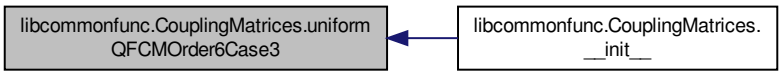
References libcommonfunc.CouplingMatrices.uniformQOrder6Case3Optimize().

Referenced by libcommonfunc.CouplingMatrices.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



10.5.5.4 def libcommonfunc.CouplingMatrices.uniformQOrder6Case3Optimize (*self, angRot, MatQfcm2a, flag*)

This function performs the hyperbolic rotations in TN102.3.

Given four angles angRot it performs the hyperbolic rotations with those angles. and sees the differences between the quality factors of the different nodes [TN102.3].

Parameters

angRot	= Vector containing two angles. The first angle corresponds with α in TN102.3 eq. (14) and the second one with α_1
MatQfcm2a	= MatrixQ class instance, containing the Folded $N+2$ Canonical Form (FCM) coupling matrix.
flag	= If False, return goal; if True, return MatQfcm2.

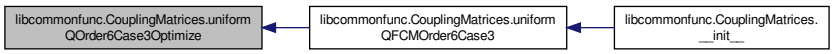
Returns

goal = Function we are minimizing. It contains the summation of the absolute values of the diferences in losses in the different nodes.
MatQfcm2 = MatrixQ class instance, containing the transformed Folded $N+2$ Canonical Form (FCM) coupling matrix.

Definition at line 4038 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.uniformQFCMOrder6Case3().

Here is the caller graph for this function:



10.5.5.5 def libcommonfunc.CouplingMatrices.uniformQOrder6Optimize (*self, angRot, MatQfcm2a, flag*)

This function performs the hyperbolic rotations in TN102.3 sec.3.3.

Given two angles angRot it performs the hyperbolic rotations with those angles and sees the differences between the quality factors of the different nodes [TN102.3 sec.3.3].

Parameters

angRot	= Vector containing two angles. The first angle corresponds with α in TN102.3 eq. (14) and the second one with α_1
MatQfcm2a	= MatrixQ class instance, containing the Folded $N+2$ Canonical Form (FCM) coupling matrix.
flag	= If False, return goal; if True, return MatQfcm2.

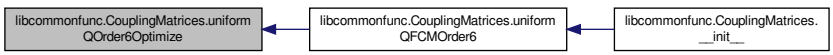
Returns

goal = Function we are minimizing. It contains the summation of the absolute values of the diferences in losses in the different nodes.
MatQfcm2 = MatrixQ class instance, containing the transformed Folded $N+2$ Canonical Form (FCM) coupling matrix.

Definition at line 3975 of file libcommonfunc.py.

Referenced by libcommonfunc.CouplingMatrices.uniformQFCMOrder6().

Here is the caller graph for this function:



10.5.5.6 def libcommonfunc.CouplingMatrices.updateCM_error (*self, CM_error, st* = `None`)

Convenience function that updates the error estimation in self.CM_error.

Parameters

CM_error	= New error estimation.
st	= String with relevant info. Default None.

Definition at line 2979 of file libcommonfunc.py.

References libcommonfunc.CouplingMatrices.CM_error, and libcommonfunc.myPrint().

Here is the call graph for this function:

The documentation for this class was generated from the following file:

- libcommonfunc.py

10.6 mwfiltersgui.CouplingMatrixDlg Class Reference

GUI dialog to show coupling matrices.

Public Member Functions

- def __init__

  Constructor: Creates dialog window with a table to display coupling matrices and buttons to save or store to disk, select matrix from list, plot [S] parameters or edit current matrix.
- def closeEvent

  Reimplementation of the window close function.
- def loadMatrixQ

  Load a N+2 coupling matrix into a QTableWidget.
- def okToContinue

  This function checks if matrix has been modified and not saved to list.

Functions automatically excuted when the user interacts with the GUI

- def CM_currentCellChanged

  Slot that is executed whenever current CM cell is changed.
- def Q_currentCellChanged

  Slot that is executed whenever current Q cell is changed.
- def matrixQChanged

  Slot that is executed whenever CM.MatQ is changed by a matrix edit action.
- def CM_itemChanged

  Update the contents of self.CM.MatQ after the user has edited the CM table widget.
- def Q_itemChanged

  Update the contents of self.CM.MatQ after the user has edited the Q table widget.
- def on_runSensitivity_pushButton_clicked

  Run sensitivity analysis This function is automatically executed by the GUI when the user pushes the 'Run' button in the sensitivity groupBox.

- def on_action_Edit_toggled
  Enable or disable matrix edit.
- def on_sensitivity_groupBox_toggled
  Enable or disable sensitivity analysis.
- def on_action_Read_triggered
  Load current matrix from disk.
- def on_action_SaveAll_triggered
  Save all matrices to disk, using default file names and extensions.
- def on_action_Save_triggered
  Save current matrix in disk.
- def on_action_ListAdd_triggered
  Store current matrix in self.CM_comboBox list.
- def on_action_ListRemove_triggered
  Remove current matrix from self.CM_comboBox and CM.listM lists.
- def on_action_PlotEnergy_triggered
  Plot stored energy and dissipated power computed from current coupling matrix.
- def on_action_PlotS_triggered
  Plot [S] parameters computed from current coupling matrix.
- def on_CM_comboBox_currentIndexChanged
  Load a new coupling matrix into the QTableWidget.
- def **pushEditAction**

10.6.1 Detailed Description

GUI dialog to show coupling matrices.

Definition at line 3048 of file mwfiltersgui.py.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 def mwfiltersgui.CouplingMatrixDlg.__init__ (*self, CM, parent* = None)

Constructor: Creates dialog window with a table to display coupling matrices and buttons to save or store to disk, select matrix from list, plot [S] parameters or edit current matrix.

```
@param CM = CouplingMatrix class instance.
@param parent = Parent widget, in this case is mainWindow.
```

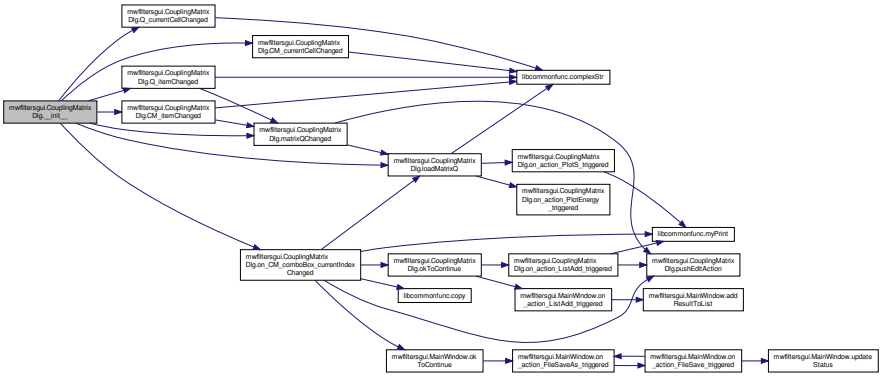
Copy of the mainWindow class instance. It is necessary to store a copy since it must be accessed by some member functions. Matrix edit dialog. It will be a Modeless Live dialog, that is only hidden when closed. Sensitivity analysis dialog. It will be a Modeless Live dialog, that is only hidden when closed. Shallow copy of the mainWindow CM class instance, for easier access. Attribute that is set to True when CM.MatQ has been modified by user edit actions and not saved in list. Attribute that is set to True when changes to matrix must be recorded. It is disabled, for example, when the user does and incorrect matrix edit and the program automatically restores the old matrix value. Attribute that is set to True before automatic calls to on_action_CM_comboBox_currentIndexChanged() to avoid loading a matrix. SensitivityAnalisys class instance.

Definition at line 3056 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.CM, mwfiltersgui.CouplingMatrixDlg.CM, mwfiltersgui.CouplingMatrixDlg.CM_comboBox, mwfiltersgui.CouplingMatrixDlg.CM_currentCellChanged(), mwfiltersgui.CouplingMatrixDlg.CM_itemChanged(), mwfiltersgui.CouplingMatrixDlg.CM_name, mwfiltersgui.CouplingMatrixDlg.CM_nameLabel, mwfiltersgui.CouplingMatrixDlg.CM_prec, mwfiltersgui.CouplingMatrixDlg.CM_prec_label, mwfiltersgui.CouplingMatrixDlg.CM_tableWidget, mwfiltersgui.CouplingMatrixDlg.CM_tooltip, mwfiltersgui.CouplingMatrixDlg.CMbp_name, mwfiltersgui.CouplingMatrixDlg.CMbp_nameLabel, mwfiltersgui.CouplingMatrixDlg.CMbp_prec, mwfiltersgui.CouplingMatrixDlg.CMbp_prec_label, mwfiltersgui.CouplingMatrixDlg.CMbp_tableWidget,

mwfiltersgui.CouplingMatrixDlg.CMbp_tooltip, mwfiltersgui.MainWindow.dirty, mwfiltersgui.CouplingMatrixDlg.dirty, mwfiltersgui.CouplingMatrixDlg.horizontalLayout_2, mwfiltersgui.CouplingMatrixDlg.horizontalLayout_3, mwfiltersgui.CouplingMatrixDlg.horizontalLayout_3bp, mwfiltersgui.CouplingMatrixDlg.horizontalLayout_4, mwfiltersgui.CouplingMatrixDlg.horizontalLayout_4bp, mwfiltersgui.CouplingMatrixDlg.loadMatrixQ(), dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.SetCouplingDlg.mainWindow, mwfiltersgui.MatrixEditDlg.mainWindow, mwfiltersgui.PredisZeorsDlg.mainWindow, mwfiltersgui.SensitivityAnalysis.mainWindow, mwfiltersgui.CouplingMatrixDlg.mainWindow, mwfiltersgui.CouplingMatrixDlg.matrixEditDlg, mwfiltersgui.CouplingMatrixDlg.matrixQChanged(), mwfiltersgui.CouplingMatrixDlg.noLoadCM, mwfiltersgui.CouplingMatrixDlg.on_CM_comboBox_currentIndexChanged(), mwfiltersgui.CouplingMatrixDlg.Q_currentCellChanged(), mwfiltersgui.CouplingMatrixDlg.Q_itemChanged(), mwfiltersgui.CouplingMatrixDlg.Q_prec, mwfiltersgui.CouplingMatrixDlg.Q_prec_label, mwfiltersgui.CouplingMatrixDlg.Q_tableWidget, mwfiltersgui.CouplingMatrixDlg.Q_tooltip, mwfiltersgui.CouplingMatrixDlg.Qprec_Layout, mwfiltersgui.CouplingMatrixDlg.recordChanges, mwfiltersgui.CouplingMatrixDlg.SA, mwfiltersgui.CouplingMatrixDlg.sensitivityDlg, mwfiltersgui.CouplingMatrixDlg.verticalLayout_CM, and mwfiltersgui.CouplingMatrixDlg.verticalLayout_CMbp.

Here is the call graph for this function:



10.6.3 Member Function Documentation

10.6.3.1 def mwfiltersgui.CouplingMatrixDlg.closeEvent (*self, event*)

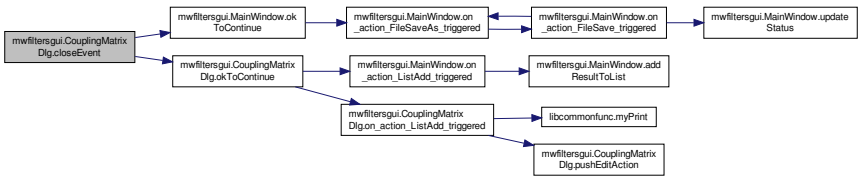
Reimplementation of the window close function.

Sets mainWindow.CMdialog to None.

Definition at line 3295 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.okToContinue(), and mwfiltersgui.CouplingMatrixDlg.okToContinue().

Here is the call graph for this function:



10.6.3.2 def mwfiltersgui.CouplingMatrixDlg.loadMatrixQ (*self, MatQ, loadMatrix* = `True`, *loadQ* = `True`)

Load a N+2 coupling matrix into a QTableWidget.

All actions that modify the coupling matrix, end up executing this method, so the [S] error plot is updated if it exists.

Parameters

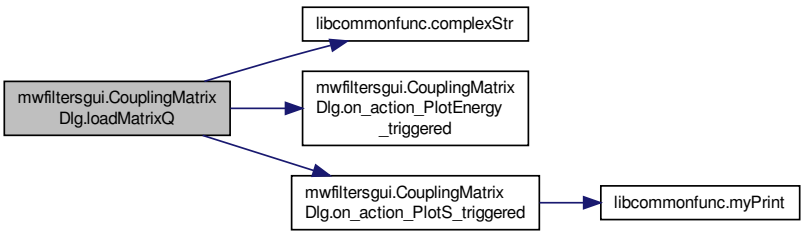
MatQ	= MatrixQ class instance.
loadMatrix	= Load contents of coupling matrix. Default True.
loadQ	= Load contents of Q vector. Default True.

Definition at line 3328 of file mwfiltersgui.py.

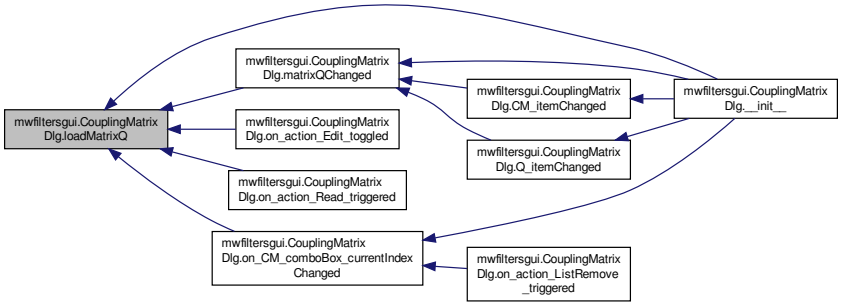
References mwfiltersgui.CouplingMatrixDlg.CM_tableWidget, mwfiltersgui.CouplingMatrixDlg.CMbp_tableWidget, libcommonfunc.complexStr(), mwfiltersgui.CouplingMatrixDlg.matrixEditDlg, mwfiltersgui.CouplingMatrixDlg.on_action_PlotEnergy_triggered(), mwfiltersgui.CouplingMatrixDlg.on_action_PlotS_triggered(), mwfiltersgui.CouplingMatrixDlg.Q_tableWidget, libcommonfunc.MatrixQ.rowIndices, and mwfiltersgui.CouplingMatrixDlg.rowIndices.

Referenced by mwfiltersgui.CouplingMatrixDlg.__init__(), mwfiltersgui.CouplingMatrixDlg.matrixQChanged(), mwfiltersgui.CouplingMatrixDlg.on_action_Edit_toggled(), mwfiltersgui.CouplingMatrixDlg.on_action_Read_triggered(), and mwfiltersgui.CouplingMatrixDlg.on_CM_comboBox_currentIndexChanged().

Here is the call graph for this function:



Here is the caller graph for this function:



10.6.3.3 def mwfiltersgui.CouplingMatrixDlg.matrixQChanged (*self*)

Slot that is executed whenever CM.MatQ is changed by a matrix edit action.

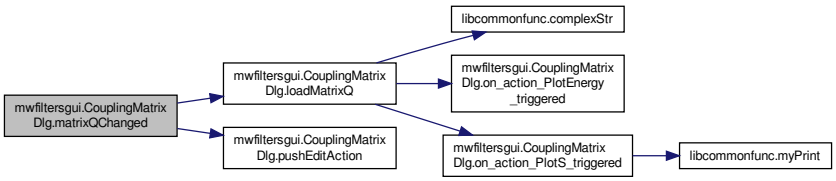
The self.dirty attribute is set to True.

Definition at line 3511 of file mwfiltersgui.py.

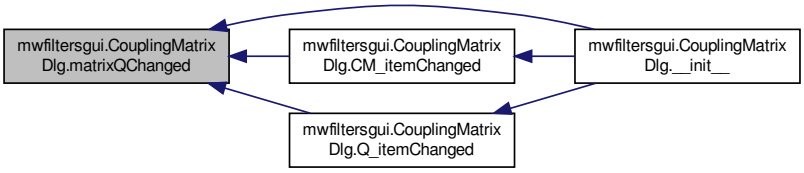
References mwfiltersgui.CouplingMatrixDlg.CM_name, mwfiltersgui.MainWindow.dirty, mwfiltersgui.CouplingMatrixDlg.dirty, mwfiltersgui.CouplingMatrixDlg.loadMatrixQ(), mwfiltersgui.CouplingMatrixDlg.pushEditAction(), and mwfiltersgui.CouplingMatrixDlg.recordChanges.

Referenced by mwfiltersgui.CouplingMatrixDlg.__init__(), mwfiltersgui.CouplingMatrixDlg.CM_itemChanged(), and mwfiltersgui.CouplingMatrixDlg.Q_itemChanged().

Here is the call graph for this function:



Here is the caller graph for this function:



10.6.3.4 def mwfiltersgui.CouplingMatrixDlg.okToContinue (*self*)

This function checks if matrix has been modified and not saved to list.

If it has been modified, asks the user to save the matrix. The function is executed by the GUI when there is an action that will destroy CM.MatQ: Load matrix from list or read matrix from disk.

Returns

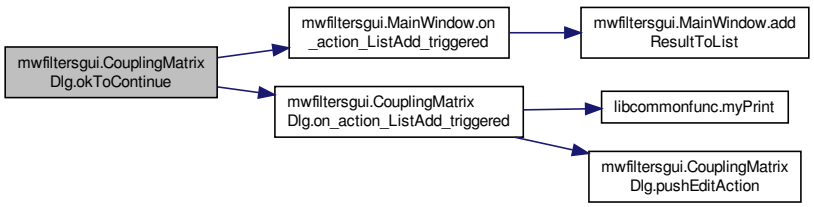
ok (True / False) = If True the use has agreed to continue, if False the user cancels the operation.

Definition at line 3439 of file mwfiltersgui.py.

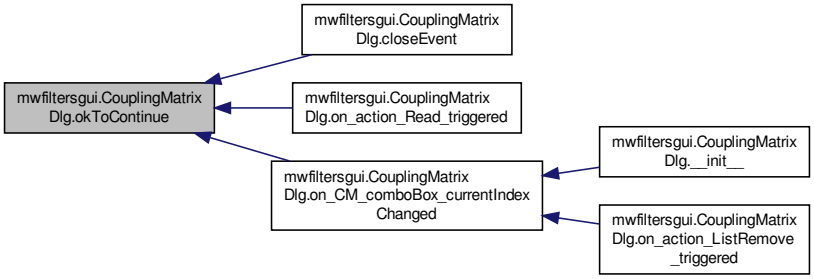
References mwfiltersgui.MainWindow.dirty, mwfiltersgui.CouplingMatrixDlg.dirty, mwfiltersgui.MainWindow.on_action_ListAdd_triggered(), and mwfiltersgui.CouplingMatrixDlg.on_action_ListAdd_triggered().

Referenced by mwfiltersgui.CouplingMatrixDlg.closeEvent(), mwfiltersgui.CouplingMatrixDlg.on_action_Read_triggered(), and mwfiltersgui.CouplingMatrixDlg.on_CM_comboBox_currentIndexChanged().

Here is the call graph for this function:



Here is the caller graph for this function:



10.6.3.5 def mwfiltersgui.CouplingMatrixDlg.on_action_Edit_toggled (*self, checked*)

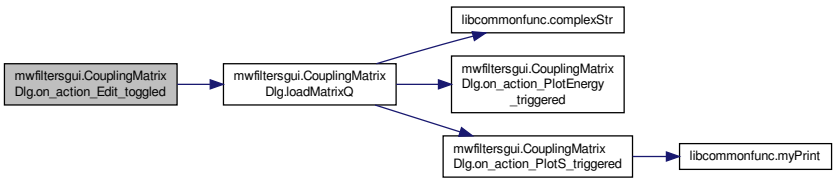
Enable or disable matrix edit.

This function is automatically executed by the GUI when the user toggles the 'Edit' button.

Definition at line 3687 of file mwfiltersgui.py.

References mwfiltersgui.CouplingMatrixDlg.CM_tooltip, mwfiltersgui.CouplingMatrixDlg.loadMatrixQ(), mwfiltersgui.CouplingMatrixDlg.Q_tooltip, and mwfiltersgui.CouplingMatrixDlg.recordChanges.

Here is the call graph for this function:



10.6.3.6 def mwfiltersgui.CouplingMatrixDlg.on_action_ListAdd_triggered (*self, checked, programmed* = `False`)

Store current matrix in self.CM_comboBox list.

Parameters

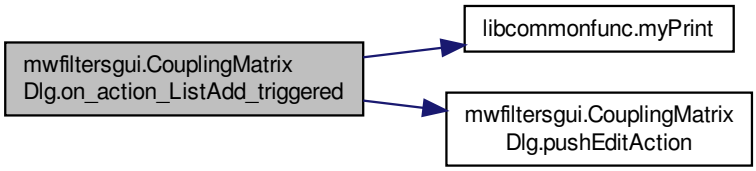
checked	= SIGNAL/SLOT parameter.
programmed	= Flag that is true when this function is executed by the program, not the user, to avoid setting CM_comboBox CurrentIndex. Default False.

Definition at line 3815 of file mwfiltersgui.py.

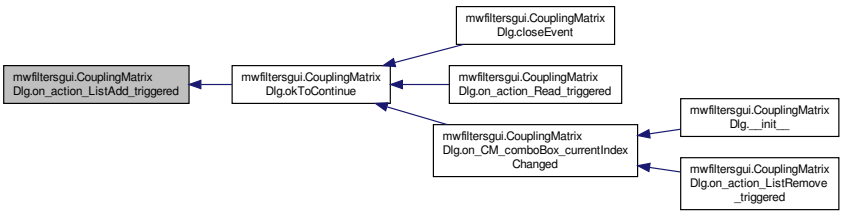
References mwfiltersgui.MainWindow.dirty, mwfiltersgui.CouplingMatrixDlg.dirty, mwfiltersgui.CouplingMatrixDlg.matrixEditDlg, libcommonfunc.myPrint(), mwfiltersgui.CouplingMatrixDlg.pushEditAction(), and mwfiltersgui.CouplingMatrixDlg.recordChanges.

Referenced by mwfiltersgui.CouplingMatrixDlg.okToContinue().

Here is the call graph for this function:



Here is the caller graph for this function:



10.6.3.7 def mwfiltersgui.CouplingMatrixDlg.on_action_PlotS_triggered (*self, checked*)

Plot [S] parameters computed from current coupling matrix.

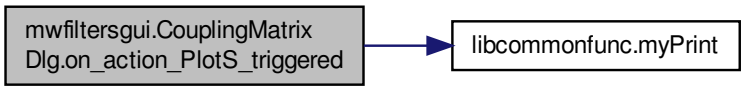
S21 and S11 are plotted on top of [S] parameters computed from Characteristic Polynomials.

Definition at line 4018 of file mwfiltersgui.py.

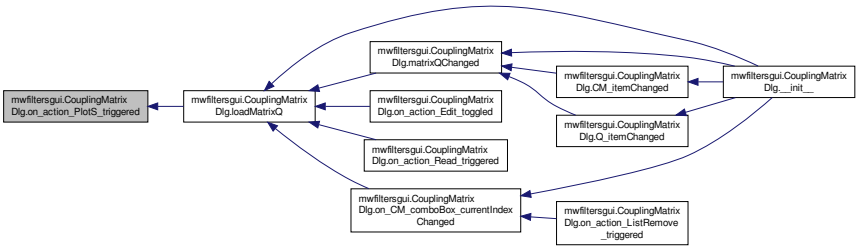
References dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.SetCouplingDlg.mainWindow, mwfiltersgui.MatrixEditDlg.mainWindow, mwfiltersgui.PredisZeorsDlg.mainWindow, mwfiltersgui.SensitivityAnalysis.mainWindow, mwfiltersgui.CouplingMatrixDlg.mainWindow, and libcommonfunc.myPrint().

Referenced by mwfiltersgui.CouplingMatrixDlg.loadMatrixQ().

Here is the call graph for this function:



Here is the caller graph for this function:



10.6.3.8 def mwfiltersgui.CouplingMatrixDlg.on_CM_comboBox_currentIndexChanged (*self, index*)

Load a new coupling matrix into the QTableWidget.

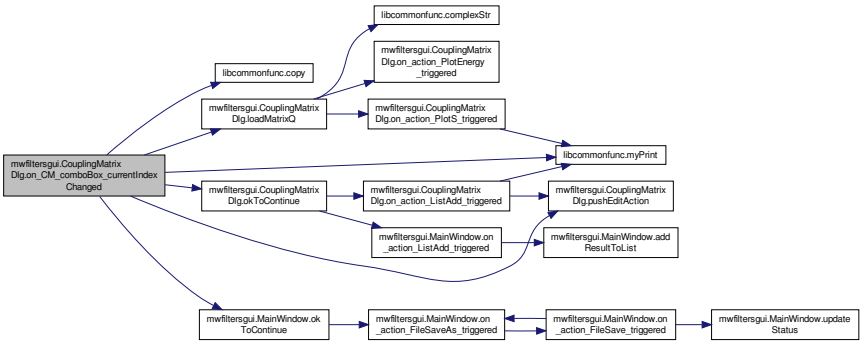
This function is automatically executed by the GUI when the user changes the coupling matrix type comboBox.

Definition at line 4105 of file mwfiltersgui.py.

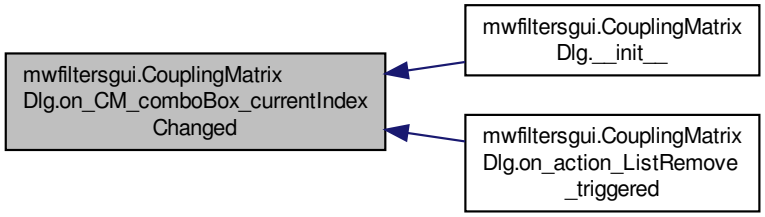
References libcommonfunc.copy(), mwfiltersgui.MainWindow.dirty, mwfiltersgui.CouplingMatrixDlg.dirty, mwfiltersgui.CouplingMatrixDlg.loadMatrixQ(), mwfiltersgui.CouplingMatrixDlg.matrixEditDlg, libcommonfunc.myPrint(), mwfiltersgui.CouplingMatrixDlg.noLoadCM, mwfiltersgui.MainWindow.okToContinue(), mwfiltersgui.CouplingMatrixDlg.okToContinue(), mwfiltersgui.CouplingMatrixDlg.pushEditAction(), and mwfiltersgui.CouplingMatrixDlg.recordChanges.

Referenced by mwfiltersgui.CouplingMatrixDlg.__init__(), and mwfiltersgui.CouplingMatrixDlg.on_action_ListRemove_triggered().

Here is the call graph for this function:



Here is the caller graph for this function:



10.6.3.9 def mwfiltersgui.CouplingMatrixDlg.on_sensitivity_groupBox_toggled (*self, checked*)

Enable or disable sensitivity analysis.

This function is automatically executed by the GUI when the user toggles the 'Sensitivity analysis' groupBox.

Definition at line 3741 of file mwfiltersgui.py.

10.6.3.10 def mwfiltersgui.CouplingMatrixDlg.Q_itemChanged (*self, item*)

Update the contents of self.CM.MatQ after the user has edited the Q table widget.

```
For each node \form#47, the quality factor is
```

$$Q_i = \frac{1}{G_i \mathrm{FBW}}$$

where the loss conductance of the unloaded resonator is:

$$G_i = -\sum_{j=1}^{N} \Im M_{ji} = -\Im M_{ii} - \sum_{i \neq j} \Im M_{ji}$$

```
If we change the \form#43 of a resonator, the new value of \form#292 is:
```

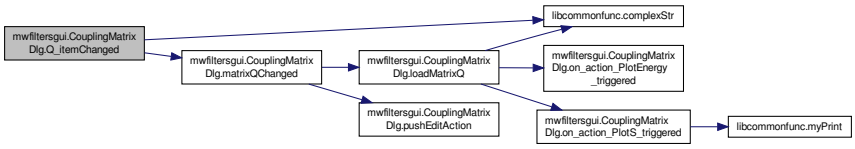
$$\Im M_{ii} = -G_i - \sum_{i \neq j} \Im M_{ji} = \frac{-1}{Q_i \,\mathrm{FBW}} - \sum_{i \neq j} \Im M_{ji}$$

Definition at line 3619 of file mwfiltersgui.py.

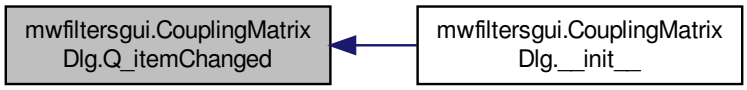
References libcommonfunc.complexStr(), mwfiltersgui.CouplingMatrixDlg.matrixQChanged(), and mwfiltersgui.CouplingMatrixDlg.recordChanges.

Referenced by mwfiltersgui.CouplingMatrixDlg.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.7 dbplot.CurveFamily Class Reference

Data structure containing info about properties of a family of curves and the curve list.

Public Member Functions

- def __init__

  Class constructor.
- def newCurves

  Create new list of curves.
- def deleteCurves

  Delete all curves.
- def updateCurves

  Update all curves with new data.
- def setPen

  Set QPen for all curves.
- def setVisible

Set visibility of all curves.

- def setSymbol

  Set symbol for all curves.

- def setLegendItems

  After creating a legend for the QWtPlot, this method will assign the QwtLegendItem of the first curve to self.legendItem, and remove the other QwtLegendItem's from the QwtLegend layout.

10.7.1 Detailed Description

Data structure containing info about properties of a family of curves and the curve list.

A curve family is a list of curves sharing the same properties. The Y data for a curve family is defined in the columns of a 2-D numpy array.

The class attributes and methods will not be documented, because the names are self-explanatory and the code is trivial.

Definition at line 1901 of file dbplot.py.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 def dbplot.CurveFamily.__init__ (*self, hPlot, name, xdata, ydata, axis, visible, color, width*)

Class constructor.

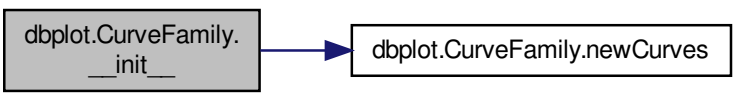
Parameters

hPlot	
name	
xdata	
ydata	
axis	
visible	
color	
width	

Definition at line 1915 of file dbplot.py.

References dbplot.CurveFamily.axis, dbplot.DbPlot.hPlot, dbplot.CurveFamily.hPlot, dbplot.CurveFamily.legend, dbplot.CurveFamily.legendItem, dbplot.CurveFamily.name, dbplot.CurveFamily.Ncurves, dbplot.CurveFamily.newCurves(), dbplot.CurveFamily.pen, and dbplot.CurveFamily.visible.

Here is the call graph for this function:



10.7.3 Member Function Documentation

10.7.3.1 def dbplot.CurveFamily.newCurves (*self, xdata, ydata*)

Create new list of curves.

Parameters

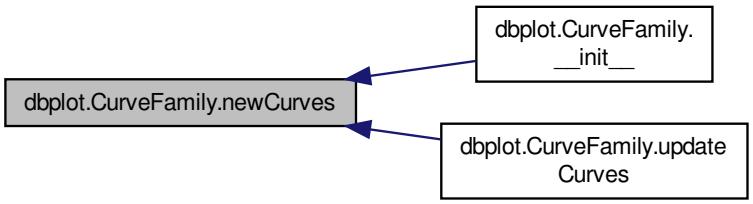
xdata	
ydata	

Definition at line 1936 of file dbplot.py.

References dbplot.CurveFamily.axis, dbplot.CurveFamily.curveList, dbplot.DbPlot.hPlot, dbplot.CurveFamily.hPlot, dbplot.CurveFamily.name, dbplot.CurveFamily.Ncurves, dbplot.CurveFamily.pen, and dbplot.CurveFamily.visible.

Referenced by dbplot.CurveFamily.__init__(), and dbplot.CurveFamily.updateCurves().

Here is the caller graph for this function:



10.7.3.2 def dbplot.CurveFamily.setLegendItems (*self, legend* = `None`, *updateLegend* = `False`)

After creating a legend for the QWtPlot, this method will assign the QwtLegendItem of the first curve to self.legendItem, and remove the other QwtLegendItem's from the QwtLegend layout.

Parameters

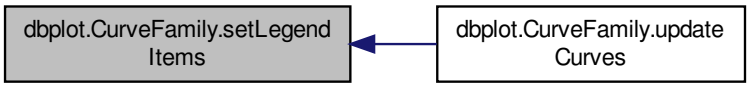
legend	= QwtLegend in QWtPlot. If equal to None, the curves are being updated and the QwtLegend handle is already stored in self.legend . Default None.
updateLegend	= Call curve.updateLegend() for the 1st curve. Default False.

Definition at line 2040 of file dbplot.py.

References dbplot.CurveFamily.curveList, dbplot.CurveFamily.legend, dbplot.CurveFamily.legendItem, and dbplot.CurveFamily.visible.

Referenced by dbplot.CurveFamily.updateCurves().

Here is the caller graph for this function:



10.7.3.3 def dbplot.CurveFamily.setPen (*self, pen*)

Set QPen for all curves.

Parameters

pen	

Definition at line 2007 of file dbplot.py.

References dbplot.CurveFamily.curveList, and dbplot.CurveFamily.pen.

10.7.3.4 def dbplot.CurveFamily.setSymbol (*self, symbol*)

Set symbol for all curves.

Parameters

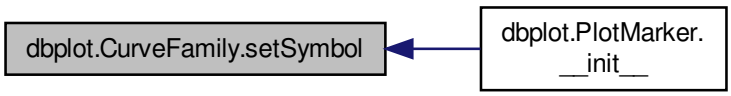
symbol	

Definition at line 2027 of file dbplot.py.

References dbplot.CurveFamily.curveList, and dbplot.CurveFamily.symbol.

Referenced by dbplot.PlotMarker.__init__().

Here is the caller graph for this function:



10.7.3.5 def dbplot.CurveFamily.setVisible (*self, visible*)

Set visibility of all curves.

Parameters

visible	= Visibility flag (Bool)

Definition at line 2017 of file dbplot.py.

References dbplot.CurveFamily.curveList, and dbplot.CurveFamily.visible.

10.7.3.6 def dbplot.CurveFamily.updateCurves (*self, xdata, ydata*)

Update all curves with new data.

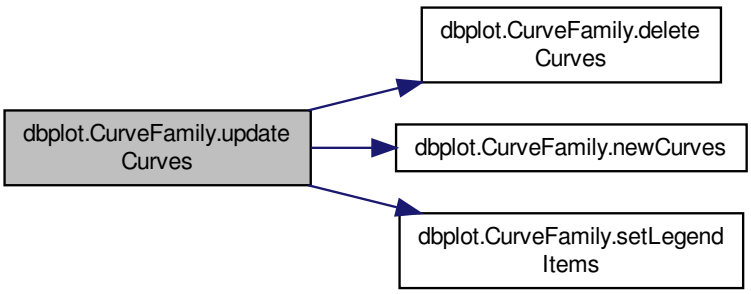
Parameters

xdata	
ydata	

Definition at line 1973 of file dbplot.py.

References dbplot.CurveFamily.curveList, dbplot.CurveFamily.deleteCurves(), dbplot.CurveFamily.Ncurves, dbplot.CurveFamily.newCurves(), and dbplot.CurveFamily.setLegendItems().

Here is the call graph for this function:



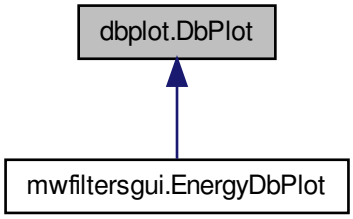
The documentation for this class was generated from the following file:

- dbplot.py

10.8 dbplot.DbPlot Class Reference

GUI dialog to contain plot.

Inheritance diagram for dbplot.DbPlot:



Public Member Functions

- def __init__

  Constructor: Creates PyQt4.QtGui.Qdialog window containing PyQt4.Qwt5.QwtPlot widget.
- def addTab

  Adds new tab with plot.
- def addMask

  Add specification mask.
- def deleteMasks

  Delete all masks.
- def update

  Update an existing plot with new data.
- def deleteLastMarker

  Deletes the last marker appended to current tab.
- def selectMarker

  Returns marker closer to the mouse cursor in screen coordinates, if distance is smaller than marker radius.
- def deleteMarker

  Deletes marker if a marker is selected.
- def selectAndHighlightMarker

  Selects a marker, highlights it and enables marker movement.
- def hightlightMarker

  Highlights marker and enables marker movement.
- def deselectMarker

  Deselect marker and mouse dragging.
- def shiftMarker

  Shifts marker if a marker is selected.
- def dragMarker

  Drags marker if a marker is selected.
- def addMarker

  Add marker at the curve point with is closer to the mouse cursor, if distance is smaller than marker radius.
- def closeEvent

  Reimplementation of the window close function.
- def getScaleMaxMin

  Get actual max and min values of scaling for a given axis.
- def getManualScale

Get max and min manual scaling values for a given axis.

- def getPrecision

  Compute the required number of significant digits of markers labels, depending on axis scaling.
- def updateMoveKnobRange

  Update markerMove Knob limits according to actual xBottom scale.
- def changeCurveVisibility

  Change visibility of curves of an existing plot.
- def autoScaleSomeAxis

  Autoscale some axis.

Functions automatically executed when the user interacts with the GUI

- def on_moveKnob_valueChanged

  This function is automatically executed when the user moves the moveKnob.
- def setCurveVisibility

  Change curve visibility by using the EditCurvesDlg dialog.
- def setCurveWidth

  Change curve pen width by using the EditCurvesDlg dialog.
- def setCurveColor

  Change curve pen color by using the EditCurvesDlg dialog.
- def showCurve

  Change curve visibility by checking legend items.
- def updateManualAxisScaling

  Change scaling of a given axis according to manual scaling control values.
- def setManualScaling

  Set manaul/auto axis scaling.
- def setAutoScaling

  Axis auto scake.
- def on_tabWidget_currentChanged

  Disable main window actions actionZoom and actionCurves.
- def on_actionPrint_triggered

  Print [S] parameters plot.
- def on_actionAbout_triggered

  Display About dialog.
- def on_actionPDF_triggered

  Save [S] parameters plot to PDF file.
- def on_actionZoom_toggled

  Enable or disable zoom.
- def on_actionNewMarker_toggled

  Allow adding markers with mouse click.
- def on_actionDeleteMarker_toggled

  Delete markers with mouse click.
- def on_actionMoveMarker_toggled

  Move marker.
- def on_actionCurves_toggled

  Set curves visibility.
- def on_actionScaleAxis_toggled

  Axis scaling.

10.8.1 Detailed Description

GUI dialog to contain plot.

Uses pyqwt5, the python binding of qwt5 C++ library for Qt applications.

Definition at line 52 of file dbplot.py.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 def dbplot.DbPlot.__init__ (*self, XData, leftYData, rightXData* = None, *rightYData* = None, *leftWidth* = None, *rightWidth* = None, *leftYmin* = None, *leftYmax* = None, *rightYmin* = None, *rightYmax* = None, *Xmin* = None, *Xmax* = None, *windowTitle* = None, *tabTitle* = None, *title* = None, *leftYNames* = None, *rightYNames* = None, *leftYVisible* = None, *rightYVisible* = None, *XLabel* = None, *leftYLabel* = None, *rightYLabel* = None, *leftYColors* = None, *rightYColors* = None, *textLabel* = None, *Xunits* = None, *leftYunits* = None, *rightYunits* = None, *parent* = None)

Constructor: Creates PyQt4.QtGui.Qdialog window containing PyQt4.Qwt5.QwtPlot widget.

```
     Uses pyqwt5, the python binding of qwt5 C++ library for Qt applications.

@param XData        = List of X-axis data (list of numpy arrays).
                Each element in the list is the x-axis data for the corresponding element in leftYData.
                When there is only one curve, it is not necessary to create a list of curves: XData argument c
@param leftYData    = List of curve-data to plot in left y-axis (list of numpy arrays).
                Curve-data can be a list of int or float or a numpy array.
                If a list element is a 2-D numpy array, it defines a curve family sharing the same curve prope
                Curves in a curve family have Ydata in the columns of the 2-D numpy array.
                When there is only one curve, it is not necessary to create a list of curves: leftYData argume
@param rightXData   = List of X-axis data (list of numpy arrays).
                Each element in the list is the x-axis data for the corresponding element in rightYData.
                When there is only one curve, it is not necessary to create a list of curves: rightXData argum
                If equal to None, XData is used both for left-y and right-y axis data.
                Default None.
@param rightYData    = List of curve-data to plot in right y-axis (list of numpy arrays).
                Curve-data can be a list of int or float or a numpy array.
                If a list element is a 2-D numpy array, it defines a curve family sharing the same curve prope
                Curves in a curve family have Ydata in the columns of the 2-D numpy array.
                When there is only one curve, it is not necessary to create a list of curves: rightYData argum
                If it is equal to None, there is no data in the right-Y axis. Default None.
@param leftYmin  = Set manual scaling for left-y axis with this minimum value.
@param leftYmax  = Set manual scaling for left-y axis with this maximum value.
@param rightYmin = Set manual scaling for right-y axis with this minimum value.
@param rightYmax = Set manual scaling for right-y axis with this maximum value.
@param Xmin      = Set manual scaling for x axis with this minimum value.
@param Xmax      = Set manual scaling for x axis with this maximum value.
@param windowTitle  = Window title (string). Default None.
@param tabTitle     = Tab title (string). Default None.
@param title        = Plot title (string). Default None.
@param leftYNames   = List of names corresponding to leftYData (list of strings).
                If equal to None, no legend is created. Default None.

@param rightYNames   = List of names corresponding to rightYData (list of strings).
                If equal to None, no legend is created. Default None.
@param leftYVisible   = List of visibility flags corresponding to leftYData (list of bool).
                If equal to None, all curves are visible. Default None.
@param rightYVisible   = List of visibility flags corresponding to rightYData (list of bool).
                If equal to None, all curves are visible. Default None.
@param XLabel       = X-axis label (string). Default None.
@param leftYLabel   = Left y-axis label (string). Default None.
@param rightYLabel   = Right y-axis label (string). Default None.
@param leftYColors  = List of colors corresponding to leftYData (list of Qt.QColor instances, like Qt.red or Q
                If equal to None, default colors are used. Default None.
@param rightYColors  = List of colors corresponding to rightYData (list of Qt.QColor instances, like Qt.red or
                If equal to None, default colors are used. Default None.
@param Xunits = X-axis units for markers (string). If equal to None, no units are displayed. Default None.
@param leftYunits = Ledt Y-axis units for markers and tracker (string). If equal to None, no units are display
@param rightYunits = Right Y-axis units for markers and tracker (string). If equal to None, no units are displ
@param textLabel = Text to display as a label (string). Can be multiline. Default None.
@param parent = Parent widget, in this case is mainWindow. Default None.
```

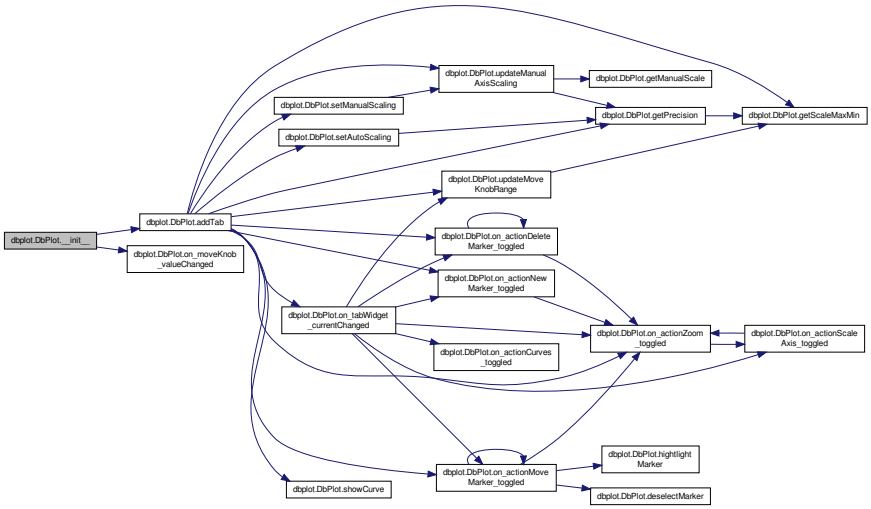
Move marker knob Number of TabWidgets List of QwtPlot class instances Left axis zoomer. List for all Tabs. Right axis zoomer. List for all Tabs. Left axis picker. List for all Tabs. Axis scaling dialog. List for all Tabs. Current instance of scalingDlg class, if the scalingDlg window is open, None if it is closed. Dialog for setting curves visibility. List for all Tabs. Current instance of EditCurvesDlg class, if the setCurves window is open, None if it is closed. List of curves in left y-axis. List for all Tabs. List of curves in right y-axis. List for all Tabs. List of specification masks. List

for all Tabs. List of markers. List for all Tabs. List of XData. List for all Tabs. List of leftYData. List for all Tabs. List of rightXData. List for all Tabs. List of rightYData. List for all Tabs. List of left y-axis curve names. List for all Tabs. List of right y-axis curve names. List for all Tabs. List of mask names. List for all Tabs. Legend for each plot. List for all Tabs. Zoomer base for autoscaling. List for all Tabs. X-axis units for markers. List for all Tabs. Left Y-axis units for markers. List for all Tabs. Right Y-axis units for markers. List for all Tabs. Marker to move Size of curve symbols, when curve points are visible for adding new marker. Number of x-axis significant digits required to display markers and tracker labels, depending on axis scaling. List for all Tabs. Number of left- y axis significant digits required to display markers and tracker labels, depending on axis scaling. List for all Tabs. Flag that is True when the program processes mouse dragging. If True, LeftButton clicks are discarded. Link to the mainWindow. Flag that will be set to true when the window is closed, in order to allow reconstruction of the class instead of updating the plot when execute or compute buttons are clicked in other windows.

Definition at line 109 of file dbplot.py.

References dbplot.DbPlot.addTab(), dbplot.DbPlot.autoScaleBase, dbplot.DbPlot.closed, dbplot.DbPlot.curveSymbolSize, dbplot.DbPlot.draggingMode, dbplot.DbPlot.hPlot, dbplot.DbPlot.label, dbplot.DbPlot.leftYCurves, dbplot.DbPlot.leftYData, dbplot.DbPlot.leftYNames, dbplot.DbPlot.leftYunits, dbplot.DbPlot.legends, dbplot.DbPlot.mainWindow, dbplot.DbPlot.markerMove, dbplot.DbPlot.markers, dbplot.DbPlot.maskNames, dbplot.DbPlot.masks, dbplot.DbPlot.moveKnob, dbplot.DbPlot.Ntabs, dbplot.DbPlot.on_moveKnob_valueChanged(), dbplot.DbPlot.pickerLeft, dbplot.DbPlot.precX, dbplot.DbPlot.precY, dbplot.DbPlot.rightXData, dbplot.DbPlot.rightYCurves, dbplot.DbPlot.rightYData, dbplot.DbPlot.rightYNames, dbplot.DbPlot.rightYunits, dbplot.DbPlot.scalingDlg, dbplot.DbPlot.scalingDlgCurrent, dbplot.DbPlot.setCurves, dbplot.DbPlot.setCurvesCurrent, dbplot.DbPlot.XData, dbplot.DbPlot.Xunits, dbplot.DbPlot.zoomerLeft, and dbplot.DbPlot.zoomerRight.

Here is the call graph for this function:



10.8.3 Member Function Documentation

10.8.3.1 def dbplot.DbPlot.addMarker (*self, point*)

Add marker at the curve point with is closer to the mouse cursor, if distance is smaller than marker radius.

Both left-y and right-y axis curves are processed.

Parameters

point	= Point coordinates (QPoint)

Definition at line 1160 of file dbplot.py.

References dbplot.DbPlot.curveSymbolSize, dbplot.DbPlot.hPlot, dbplot.DbPlot.leftYCurves, dbplot.DbPlot.markers, and dbplot.DbPlot.rightYCurves.

10.8.3.2 def dbplot.DbPlot.addMask (*self, nTab, XData, YData, axis, name, color, type, visible =* `True`, *transpar =* `64`)

Add specification mask.

In order to split the mask into separated parts, use numpy NaN in y-axis (numpy.nan), as show in the example at the end of this file.

Parameters

nTab	= TabWidget page index.
XData	= x data (float numpy array).
YData	= y data (float numpy array).
axis	= 'left' or 'right' (string). Y-axis to which the YData values correspond.
name	= String to display in legend
color	= Qt.QColor instance, like Qt.red or Qt.blue
type	= 'upper' or 'lower' (string).
visible	= Visibility flag. If True the mask curve is initially visible, otherwise it is hidden. Default True.
transpar	= Alpha value for transparency (255=opaque, 0=transparent). Default 64.

Definition at line 724 of file dbplot.py.

References dbplot.DbPlot.hPlot, dbplot.DbPlot.legends, dbplot.DbPlot.maskNames, and dbplot.DbPlot.masks.

10.8.3.3 def dbplot.DbPlot.addTab (*self, XData, leftYData, rightXData =* `None`, *rightYData =* `None`, *leftWidth =* `None`, *rightWidth =* `None`, *leftYmin =* `None`, *leftYmax =* `None`, *rightYmin =* `None`, *rightYmax =* `None`, *Xmin =* `None`, *Xmax =* `None`, *tabTitle =* `None`, *title =* `None`, *leftYNames =* `None`, *rightYNames =* `None`, *leftYVisible =* `None`, *rightYVisible =* `None`, *XLabel =* `None`, *leftYLabel =* `None`, *rightYLabel =* `None`, *leftYColors =* `None`, *rightYColors =* `None`, *Xunits =* `None`, *leftYunits =* `None`, *rightYunits =* `None`, *replaceTab =* `None`)

Adds new tab with plot.

Parameters

XData	= List of X-axis data (list of numpy arrays). Each element in the list is the x-axis data for the corresponding element in leftYData. When there is only one curve, it is not necessary to create a list of curves: XData argument can be the curve x-data alone.
leftYData	= List of curve-data to plot in left y-axis (list of numpy arrays). Curve-data can be a list of int or float or a numpy array. If a list element is a 2-D numpy array, it defines a curve family sharing the same curve properties. Curves in a curve family have Ydata in the columns of the 2-D numpy array. When there is only one curve, it is not necessary to create a list of curves: leftYData argument can be the curve-data alone.
rightXData	= List of X-axis data (list of numpy arrays). Each element in the list is the x-axis data for the corresponding element in rightYData. When there is only one curve, it is not necessary to create a list of curves: rightXData argument can be the curve x-data alone. If equal to None, XData is used both for left-y and right-y axis data. Default None.
rightYData	= List of curve-data to plot in right y-axis (list of numpy arrays). Curve-data can be a list of int or float or a numpy array. If a list element is a 2-D numpy array, it defines a curve family sharing the same curve properties. Curves in a curve family have Ydata in the columns of the 2-D numpy array. When there is only one curve, it is not necessary to create a list of curves: rightYData argument can be the curve-data alone. If it is equal to None, there is no data in the right-Y axis. Default None.
leftYmin	= Set manual scaling for left-y axis with this minimum value.
leftYmax	= Set manual scaling for left-y axis with this maximum value.
rightYmin	= Set manual scaling for right-y axis with this minimum value.

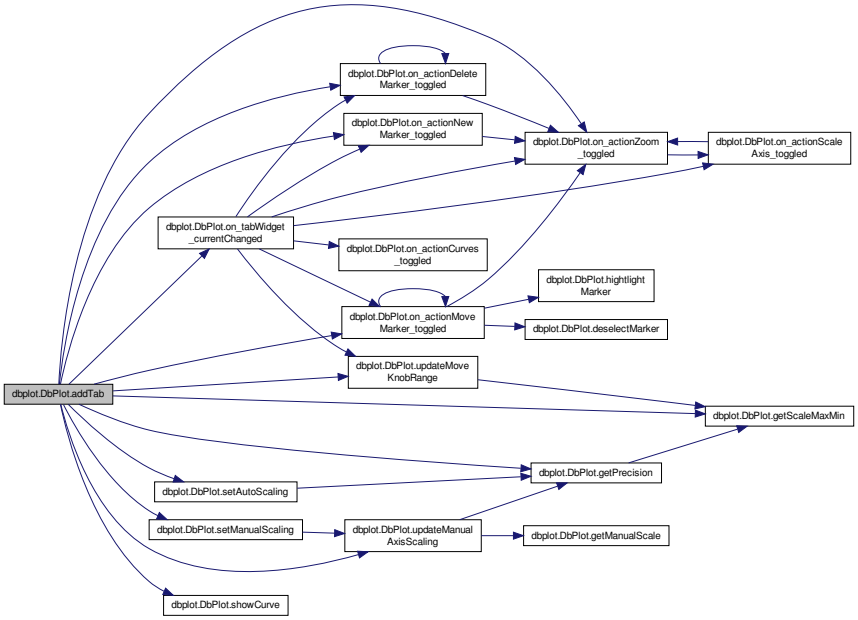
rightYmax	= Set manual scaling for right-y axis with this maximum value.
Xmin	= Set manual scaling for x axis with this minimum value.
Xmax	= Set manual scaling for x axis with this maximum value.
tabTitle	= Tab title (string). Default None.
title	= Plot title (string). Default None.
leftYNames	= List of names corresponding to leftYData (list of strings). If equal to None, no legend is created. Default None.
rightYNames	= List of names corresponding to rightYData (list of strings). If equal to None, no legend is created. Default None.
leftYVisible	= List of visibility flags corresponding to leftYData (list of bool). If equal to None, all curves are visible. Default None.
rightYVisible	= List of visibility flags corresponding to rightYData (list of bool). If equal to None, all curves are visible. Default None.
XLabel	= X-axis label (string). Default None.
leftYLabel	= Left y-axis label (string). Default None.
rightYLabel	= Right y-axis label (string). Default None.
leftYColors	= List of colors corresponding to leftYData (list of Qt.QColor instances, like Qt.red or Qt.blue). If equal to None, default colors are used. Default None.
rightYColors	= List of colors corresponding to rightYData (list of Qt.QColor instances, like Qt.red or Qt.blue). If equal to None, default colors are used. Default None.
Xunits	= X-axis units for markers (string). If equal to None, no units are displayed. Default None.
leftYunits	= Left Y-axis units for markers and tracker (string). If equal to None, no units are displayed. Default None.
rightYunits	= Right Y-axis units for markers and tracker (string). If equal to None, no units are displayed. Default None.
replaceTab	= If not None, it is the index of an already existing Tab and all the CurveFamilies have to be replaced with data from the function arguments. Default False.

Definition at line 342 of file dbplot.py.

References dbplot.DbPlot.autoScaleBase, dbplot.DbPlot.getPrecision(), dbplot.DbPlot.getScaleMaxMin(), dbplot.DbPlot.hPlot, dbplot.DbPlot.leftYCurves, dbplot.DbPlot.leftYData, dbplot.DbPlot.leftYunits, dbplot.DbPlot.legends, dbplot.DbPlot.markers, dbplot.DbPlot.masks, dbplot.DbPlot.Ntabs, dbplot.DbPlot.on_actionDeleteMarker_toggled(), dbplot.DbPlot.on_actionMoveMarker_toggled(), dbplot.DbPlot.on_actionNewMarker_toggled(), dbplot.DbPlot.on_actionZoom_toggled(), dbplot.DbPlot.on_tabWidget_currentChanged(), dbplot.DbPlot.pickerLeft, dbplot.DbPlot.precX, dbplot.DbPlot.precY, dbplot.DbPlot.rightXData, dbplot.DbPlot.rightYCurves, dbplot.DbPlot.rightYData, dbplot.DbPlot.rightYunits, dbplot.DbPlot.scalingDlg, dbplot.DbPlot.setAutoScaling(), dbplot.DbPlot.setCurves, dbplot.DbPlot.setManualScaling(), dbplot.DbPlot.showCurve(), dbplot.DbPlot.updateManualAxisScaling(), dbplot.DbPlot.updateMoveKnobRange(), dbplot.DbPlot.XData, dbplot.DbPlot.Xunits, dbplot.DbPlot.zoomerLeft, and dbplot.DbPlot.zoomerRight.

Referenced by dbplot.DbPlot.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:

10.8.3.4 def dbplot.DbPlot.autoScaleSomeAxis (*self, nTab, autoScaleBottomX, autoScaleLeftY, autoScaleRightY*)

Autoscale some axis.

This function is intended to be called after calling self.update when the programmer wishes to autoscale axis after update.

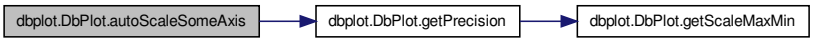
Parameters

nTab	= Tab index where to autoscale axis.
autoScaleX	= Flag to indicate if bottom X axis must be autoscaled (True | False).
autoScaleLeftY	= Flag to indicate if left Y axis must be autoscaled (True | False).
autoScaleRightY	= Flag to indicate if right Y axis must be autoscaled (True | False).

Definition at line 1386 of file dbplot.py.

References dbplot.DbPlot.autoScaleBase, dbplot.DbPlot.getPrecision(), dbplot.DbPlot.hPlot, and dbplot.DbPlot.-scalingDlg.

Here is the call graph for this function:



10.8.3.5 def dbplot.DbPlot.changeCurveVisibility (*self, nTab, leftYVisible* = `None`, *rightYVisible* = `None`, *masksVisible* = `None`)

Change visibility of curves of an existing plot.

Parameters

nTab	= Tab index
leftYVisible	= List of visibility flags corresponding to leftYData curves (list of bool or None values). Instead of a list of flags, it can be a dictionary of the form { curveName: flag, ... }. Curves corresponding to a missing dictionary key are not changed. If a flag is None, the visibility of the corresponding curve is not changed. If the parameter is equal to None, leftYData curves visibility is not changed. Default None.
rightYVisible	= List of visibility flags corresponding to rightYData curves (list of bool or None values). Instead of a list of flags, it can be a dictionary of the form { curveName: flag, ... }. Curves corresponding to a missing dictionary key are not changed. If a flag is None, the visibility of the corresponding curve is not changed. If the parameter is equal to None, rightYData curves visibility is not changed. Default None.
masksVisible	= List of visibility flags corresponding to mask curves (list of bool or None values). Instead of a list of flags, it can be a dictionary of the form { curveName: flag, ... }. Curves corresponding to a missing dictionary key are not changed. If a flag is None, the visibility of the corresponding curve is not changed. If the parameter is equal to None, mask curves visibility is not changed. Default None.

Definition at line 1320 of file dbplot.py.

References dbplot.DbPlot.hPlot, dbplot.DbPlot.leftYCurves, dbplot.DbPlot.leftYNames, dbplot.DbPlot.legends, dbplot.DbPlot.maskNames, dbplot.DbPlot.masks, dbplot.DbPlot.rightYCurves, dbplot.DbPlot.rightYNames, and dbplot.DbPlot.setCurves.

10.8.3.6 def dbplot.DbPlot.closeEvent (*self, event*)

Reimplementation of the window close function.

Sets closed attribute to True.

Definition at line 1218 of file dbplot.py.

References dbplot.DbPlot.closed.

10.8.3.7 def dbplot.DbPlot.deleteMarker (*self, point*)

Deletes marker if a marker is selected.

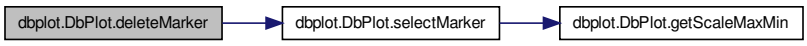
Parameters

point	= Point coordinates (QPoint)

Definition at line 1014 of file dbplot.py.

References dbplot.DbPlot.hPlot, dbplot.DbPlot.markers, and dbplot.DbPlot.selectMarker().

Here is the call graph for this function:



10.8.3.8 def dbplot.DbPlot.deleteMasks (*self*)

Delete all masks.

Does nothing if there are no masks.

Definition at line 777 of file dbplot.py.

References dbplot.DbPlot.hPlot, dbplot.DbPlot.maskNames, dbplot.DbPlot.masks, and dbplot.DbPlot.Ntabs.

10.8.3.9 def dbplot.DbPlot.dragMarker (*self, point*)

Drags marker if a marker is selected.

Parameters

point	= Point coordinates (QPoint)

Definition at line 1129 of file dbplot.py.

References dbplot.DbPlot.hPlot, and dbplot.DbPlot.markerMove.

10.8.3.10 def dbplot.DbPlot.getManualScale (*self, nTab, axis*)

Get max and min manual scaling values for a given axis.

Parameters

nTab	= tab index
axis	= QwtPlot axis (Qwt.QwtPlot.yLeft, Qwt.QwtPlot.yRight, Qwt.QwtPlot.xBottom).

Returns

maxScale = Maximum scale value.
minScale = Minimum scale value.

Definition at line 1252 of file dbplot.py.

References dbplot.DbPlot.scalingDlg.

Referenced by dbplot.DbPlot.updateManualAxisScaling().

Here is the caller graph for this function:

10.8.3.11 def dbplot.DbPlot.getPrecision (*self, nTab*)

Compute the required number of significant digits of markers labels, depending on axis scaling.

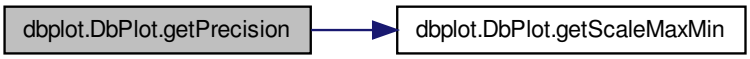
param nTab = Tab widget index.

Definition at line 1271 of file dbplot.py.

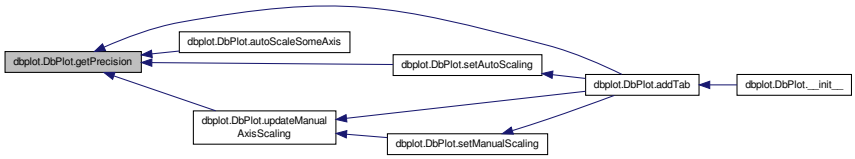
References dbplot.DbPlot.getScaleMaxMin(), dbplot.DbPlot.markers, dbplot.DbPlot.precX, and dbplot.DbPlot.precY.

Referenced by dbplot.DbPlot.addTab(), dbplot.DbPlot.autoScaleSomeAxis(), dbplot.DbPlot.setAutoScaling(), and dbplot.DbPlot.updateManualAxisScaling().

Here is the call graph for this function:



Here is the caller graph for this function:



10.8.3.12 def dbplot.DbPlot.getScaleMaxMin (*self, nTab, axis*)

Get actual max and min values of scaling for a given axis.

It is a separate function to encapsulate the different method names for qwt5.1 and qwt5.2

Parameters

nTab	= Tab index
axis	= QwtPlot axis (Qwt.QwtPlot.yLeft, Qwt.QwtPlot.yRight, Qwt.QwtPlot.xBottom).

Returns

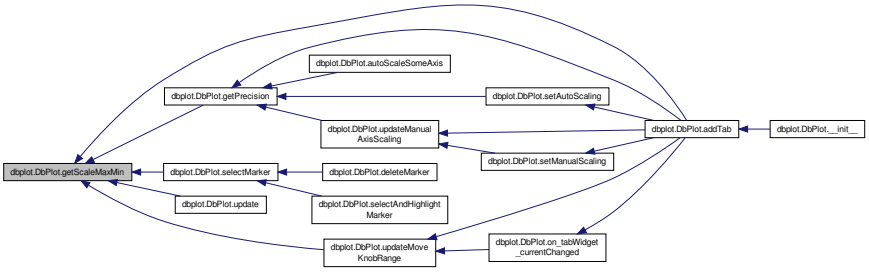
maxScale = Maximum scale value.
minScale = Minimum scale value.

Definition at line 1232 of file dbplot.py.

References dbplot.DbPlot.hPlot.

Referenced by dbplot.DbPlot.addTab(), dbplot.DbPlot.getPrecision(), dbplot.DbPlot.selectMarker(), dbplot.DbPlot.update(), and dbplot.DbPlot.updateMoveKnobRange().

Here is the caller graph for this function:



10.8.3.13 def dbplot.DbPlot.hightlightMarker (*self, nTab, markerMove*)

Highlights marker and enables marker movement.

Parameters

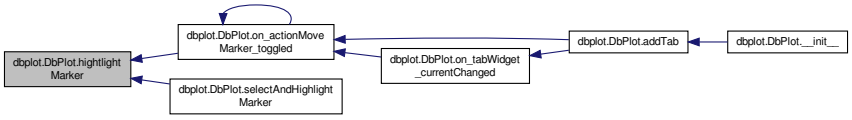
nTab	= Tab index
markerMove	= Marker to move (PlotMarker)

Definition at line 1061 of file dbplot.py.

References dbplot.DbPlot.draggingMode, dbplot.DbPlot.hPlot, and dbplot.DbPlot.markerMove.

Referenced by dbplot.DbPlot.on_actionMoveMarker_toggled(), and dbplot.DbPlot.selectAndHighlightMarker().

Here is the caller graph for this function:



10.8.3.14 def dbplot.DbPlot.on_actionAbout_triggered (*self, checked*)

Display About dialog.

This function is automatically executed by the GUI when the user triggers the 'About' action. The information shown is read from the file docstring.

Definition at line 1632 of file dbplot.py.

10.8.3.15 def dbplot.DbPlot.on_actionCurves_toggled (*self, checked*)

Set curves visibility.

This function is automatically executed by the GUI when the user toggles the 'Set curves visibility' button.

Definition at line 1827 of file dbplot.py.

References dbplot.DbPlot.hPlot, dbplot.DbPlot.setCurves, and dbplot.DbPlot.setCurvesCurrent.

Referenced by dbplot.DbPlot.on_tabWidget_currentChanged().

Here is the caller graph for this function:

10.8.3.16 def dbplot.DbPlot.on_actionDeleteMarker_toggled (*self, checked*)

Delete markers with mouse click.

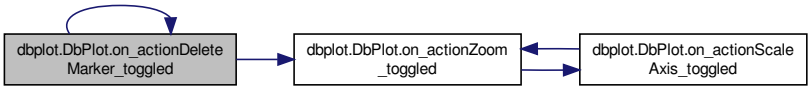
This function is automatically executed by the GUI when the user toggles the 'Delete Marker' button.

Definition at line 1756 of file dbplot.py.

References dbplot.DbPlot.hPlot, dbplot.DbPlot.markers, dbplot.DbPlot.on_actionDeleteMarker_toggled(), dbplot.DbPlot.on_actionZoom_toggled(), and dbplot.DbPlot.pickerLeft.

Referenced by dbplot.DbPlot.addTab(), dbplot.DbPlot.on_actionDeleteMarker_toggled(), and dbplot.DbPlot.on_tabWidget_currentChanged().

Here is the call graph for this function:



Here is the caller graph for this function:

10.8.3.17 def dbplot.DbPlot.on_actionMoveMarker_toggled (*self, checked*)

Move marker.

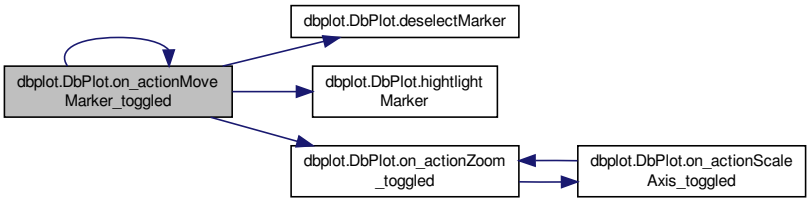
This function is automatically executed by the GUI when the user toggles the 'Move Marker' button.

Definition at line 1788 of file dbplot.py.

References dbplot.DbPlot.deselectMarker(), dbplot.DbPlot.hightlightMarker(), dbplot.DbPlot.hPlot, dbplot.DbPlot.markers, dbplot.DbPlot.on_actionMoveMarker_toggled(), dbplot.DbPlot.on_actionZoom_toggled(), and dbplot.DbPlot.pickerLeft.

Referenced by dbplot.DbPlot.addTab(), dbplot.DbPlot.on_actionMoveMarker_toggled(), and dbplot.DbPlot.on_tabWidget_currentChanged().

Here is the call graph for this function:



Here is the caller graph for this function:

10.8.3.18 def dbplot.DbPlot.on_actionNewMarker_toggled (*self, checked*)

Allow adding markers with mouse click.

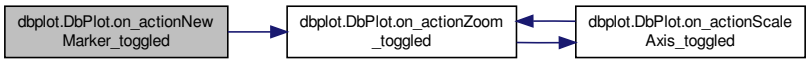
This function is automatically executed by the GUI when the user toggles the 'New Marker' button.

Definition at line 1711 of file dbplot.py.

References dbplot.DbPlot.curveSymbolSize, dbplot.DbPlot.hPlot, dbplot.DbPlot.leftYCurves, dbplot.DbPlot.on_actionZoom_toggled(), dbplot.DbPlot.pickerLeft, and dbplot.DbPlot.rightYCurves.

Referenced by dbplot.DbPlot.addTab(), and dbplot.DbPlot.on_tabWidget_currentChanged().

Here is the call graph for this function:



Here is the caller graph for this function:

10.8.3.19 def dbplot.DbPlot.on_actionPDF_triggered (*self, checked*)

Save [S] parameters plot to PDF file.

This function is automatically executed by the GUI when the user triggers the 'PDF' action.

Definition at line 1653 of file dbplot.py.

References dbplot.DbPlot.hPlot.

10.8.3.20 def dbplot.DbPlot.on_actionPrint_triggered (*self, checked*)

Print [S] parameters plot.

This function is automatically executed by the GUI when the user triggers the 'Print' action.

Definition at line 1609 of file dbplot.py.

References dbplot.DbPlot.hPlot.

10.8.3.21 def dbplot.DbPlot.on_actionScaleAxis_toggled (*self, checked*)

Axis scaling.

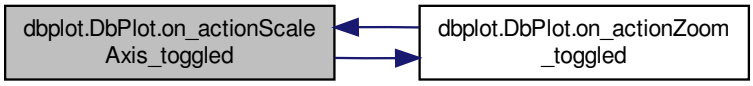
This function is automatically executed by the GUI when the user toggles the 'Axis scale' button.

Definition at line 1846 of file dbplot.py.

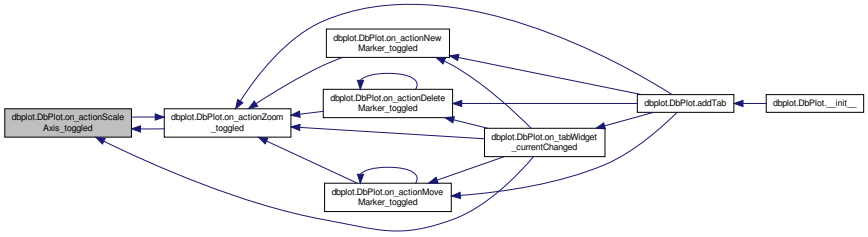
References dbplot.DbPlot.hPlot, dbplot.DbPlot.on_actionZoom_toggled(), dbplot.DbPlot.scalingDlg, and dbplot.DbPlot.scalingDlgCurrent.

Referenced by dbplot.DbPlot.on_actionZoom_toggled(), and dbplot.DbPlot.on_tabWidget_currentChanged().

Here is the call graph for this function:



Here is the caller graph for this function:



10.8.3.22 def dbplot.DbPlot.on_actionZoom_toggled (*self, checked*)

Enable or disable zoom.

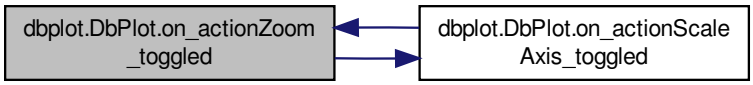
This function is automatically executed by the GUI when the user toggles the 'Zoom' button.

Definition at line 1675 of file dbplot.py.

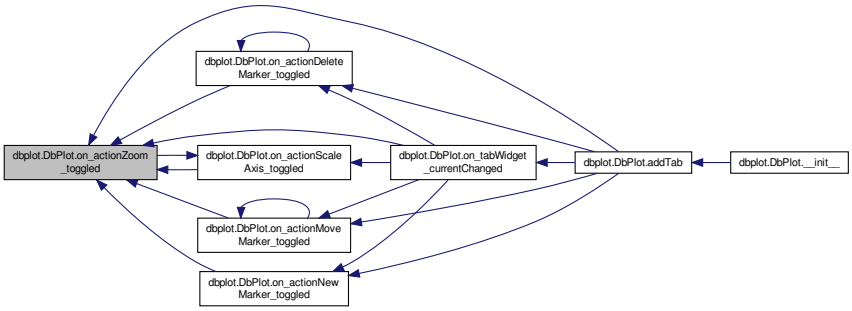
References dbplot.DbPlot.hPlot, dbplot.DbPlot.on_actionScaleAxis_toggled(), dbplot.DbPlot.scalingDlg, dbplot.DbPlot.zoomerLeft, and dbplot.DbPlot.zoomerRight.

Referenced by dbplot.DbPlot.addTab(), dbplot.DbPlot.on_actionDeleteMarker_toggled(), dbplot.DbPlot.on_actionMoveMarker_toggled(), dbplot.DbPlot.on_actionNewMarker_toggled(), dbplot.DbPlot.on_actionScaleAxis_toggled(), and dbplot.DbPlot.on_tabWidget_currentChanged().

Here is the call graph for this function:



Here is the caller graph for this function:



10.8.3.23 def dbplot.DbPlot.on_moveKnob_valueChanged (*self, value*)

This function is automatically executed when the user moves the moveKnob.

This slot must be manually connected to valueChanged, since the QwtKnob widget was not created in QtDesigner.

Parameters

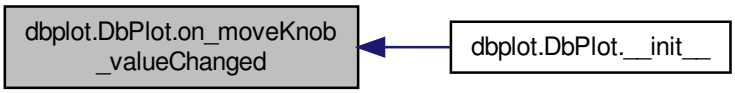
value	= Knob value (double)

Definition at line 1421 of file dbplot.py.

References dbplot.DbPlot.hPlot.

Referenced by dbplot.DbPlot.__init__().

Here is the caller graph for this function:



10.8.3.24 def dbplot.DbPlot.on_tabWidget_currentChanged (*self, index*)

Disable main window actions actionZoom and actionCurves.

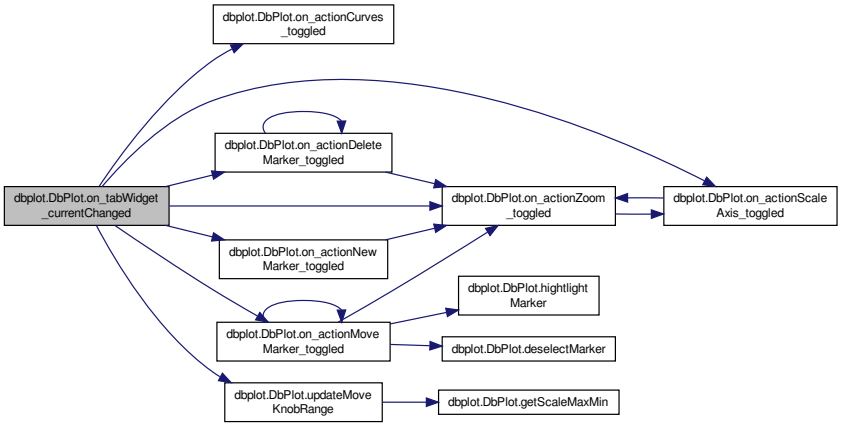
This function is automatically executed by the GUI when the user changes the current tab page.

Definition at line 1572 of file dbplot.py.

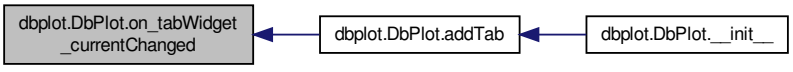
References dbplot.DbPlot.hPlot, dbplot.DbPlot.on_actionCurves_toggled(), dbplot.DbPlot.on_actionDeleteMarker-_toggled(), dbplot.DbPlot.on_actionMoveMarker_toggled(), dbplot.DbPlot.on_actionNewMarker_toggled(), dbplot.-DbPlot.on_actionScaleAxis_toggled(), dbplot.DbPlot.on_actionZoom_toggled(), dbplot.DbPlot.scalingDlgCurrent, dbplot.DbPlot.setCurves, dbplot.DbPlot.setCurvesCurrent, and dbplot.DbPlot.updateMoveKnobRange().

Referenced by dbplot.DbPlot.addTab().

Here is the call graph for this function:



Here is the caller graph for this function:



10.8.3.25 def dbplot.DbPlot.selectAndHighlightMarker (*self, point*)

Selects a marker, highlights it and enables marker movement.

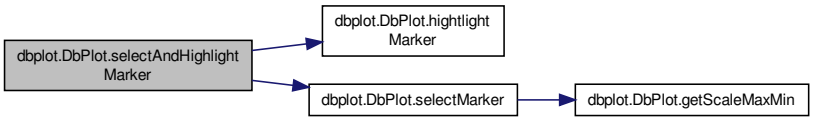
Parameters

point	= Point coordinates (QPoint)

Definition at line 1047 of file dbplot.py.

References dbplot.DbPlot.hightlightMarker(), and dbplot.DbPlot.selectMarker().

Here is the call graph for this function:



10.8.3.26 def dbplot.DbPlot.selectMarker (*self, nTab, point*)

Returns marker closer to the mouse cursor in screen coordinates, if distance is smaller than marker radius.

Parameters

nTab	= Tab index
point	= Point coordinates (QPoint)

Returns

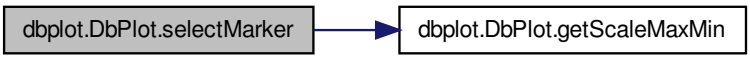
markerSelected = QwtPlotMarker instance if marker is selected, None otherwise.

Definition at line 962 of file dbplot.py.

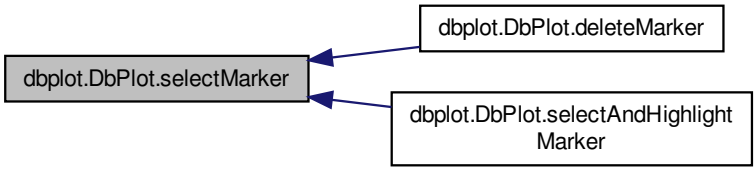
References dbplot.DbPlot.getScaleMaxMin(), dbplot.DbPlot.hPlot, and dbplot.DbPlot.markers.

Referenced by dbplot.DbPlot.deleteMarker(), and dbplot.DbPlot.selectAndHighlightMarker().

Here is the call graph for this function:



Here is the caller graph for this function:



10.8.3.27 def dbplot.DbPlot.setAutoScaling (*self, axis, groupBox*)

Axis auto scake.

This function is automatically executed by the GUI when the user presses the 'Auto' buttom of some axis.

Parameters

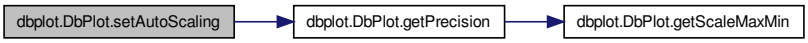
axis	= QwtPlot axis (Qwt.QwtPlot.yLeft, Qwt.QwtPlot.yRight, Qwt.QwtPlot.xBottom)
groupBox	= Manual scaling groupBox, that will be unchecked.

Definition at line 1552 of file dbplot.py.

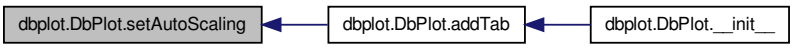
References dbplot.DbPlot.autoScaleBase, dbplot.DbPlot.getPrecision(), and dbplot.DbPlot.hPlot.

Referenced by dbplot.DbPlot.addTab().

Here is the call graph for this function:



Here is the caller graph for this function:



10.8.3.28 def dbplot.DbPlot.setCurveColor (*self, checked*)

Change curve pen color by using the EditCurvesDlg dialog.

self.setCurves.curvesPushButtonsDict is used to find the QwtPlotCurve corresponding to the self.setCurves push-Button that sent the signal.

Parameters

checked	= Check state of the pushButton that sent the signal.

Definition at line 1485 of file dbplot.py.

References dbplot.DbPlot.hPlot, and dbplot.DbPlot.setCurves.

10.8.3.29 def dbplot.DbPlot.setCurveVisibility (*self, value*)

Change curve visibility by using the EditCurvesDlg dialog.

self.setCurves.curvesCheckBoxesDict is used to find the QwtPlotCurve corresponding to the self.setCurves checkBox that sent the signal.

Parameters

value	= State of the checkBox that sent the signal.

Definition at line 1447 of file dbplot.py.

References dbplot.DbPlot.hPlot, dbplot.DbPlot.markers, and dbplot.DbPlot.setCurves.

10.8.3.30 def dbplot.DbPlot.setCurveWidth (*self, value*)

Change curve pen width by using the EditCurvesDlg dialog.

self.setCurves.curvesCheckBoxesDict is used to find the QwtPlotCurve corresponding to the self.setCurves spinBox that sent the signal.

Parameters

value	= Value of the spinBox that sent the signal.

Definition at line 1470 of file dbplot.py.

References dbplot.DbPlot.hPlot, and dbplot.DbPlot.setCurves.

10.8.3.31 def dbplot.DbPlot.setManualScaling (*self, value, axis*)

Set manaul/auto axis scaling.

Parameters

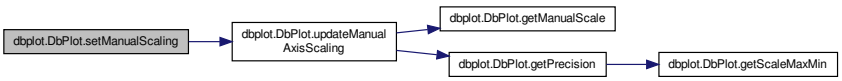
value	= Value of axis manual scaling group CheckBox (int). 0 -> False (auto), 2 -> True (manual).
axis	= QwtPlot axis (Qwt.QwtPlot.yLeft, Qwt.QwtPlot.yRight, Qwt.QwtPlot.xBottom).

Definition at line 1533 of file dbplot.py.

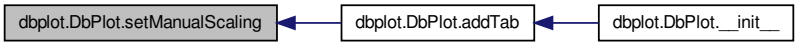
References dbplot.DbPlot.hPlot, and dbplot.DbPlot.updateManualAxisScaling().

Referenced by dbplot.DbPlot.addTab().

Here is the call graph for this function:



Here is the caller graph for this function:



10.8.3.32 def dbplot.DbPlot.shiftMarker (*self, increment*)

Shifts marker if a marker is selected.

Parameters

increment	= Number of curve points to shift (int)

Definition at line 1107 of file dbplot.py.

References dbplot.DbPlot.markerMove.

10.8.3.33 def dbplot.DbPlot.showCurve (*self, item, on*)

Change curve visibility by checking legend items.

Parameters

value	= State of the legendItem that sent the signal.

Definition at line 1503 of file dbplot.py.

References dbplot.DbPlot.hPlot, and dbplot.DbPlot.setCurves.

Referenced by dbplot.DbPlot.addTab().

Here is the caller graph for this function:

10.8.3.34 def dbplot.DbPlot.update (*self, nTab, XData, leftYData, rightXData* = `None`, *rightYData* = `None`, *textLabel* = `None`, *autoScaleBottomX* = `False`, *autoScaleLeftY* = `False`, *autoScaleRightY* = `False`)

Update an existing plot with new data.

The number of curves in the leftY and rightY axis must be the same as in the existing plot before update. A change in the number of curves will result in an assertion error.

Parameters

nTab	= TabWidget page index.
XData	= x-axis data (float numpy array or list of floats).

leftYData	= List of arrays to plot in left y-axis (list of numpy arrays or list of lists of floats).
rightXData	= x-axis data for right-axis curves (float numpy array or list of floats). If equal to None, XData is used both for left-y and right-y axis data. Default None.
rightYData	= List of arrays to plot in right y-axis (list of numpy arrays or list of lists of floats). Default None.
textLabel	= Text to display as a label (string). Can be multiline. Default None.

Definition at line 800 of file dbplot.py.

References dbplot.DbPlot.autoScaleBase, dbplot.DbPlot.getScaleMaxMin(), dbplot.DbPlot.hPlot, dbplot.DbPlot.label, dbplot.DbPlot.leftYCurves, dbplot.DbPlot.legends, dbplot.DbPlot.markers, dbplot.DbPlot.masks, dbplot.DbPlot.rightYCurves, dbplot.DbPlot.scalingDlg, dbplot.DbPlot.zoomerLeft, and dbplot.DbPlot.zoomerRight.

Here is the call graph for this function:

10.8.3.35 def dbplot.DbPlot.updateManualAxisScaling (*self, axis*)

Change scaling of a given axis according to manual scaling control values.

Parameters

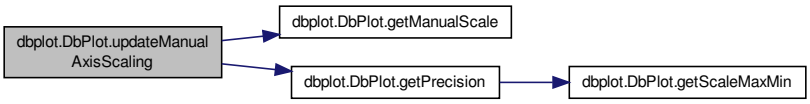
axis	= QwtPlot axis (Qwt.QwtPlot.yLeft, Qwt.QwtPlot.yRight, Qwt.QwtPlot.xBottom).

Definition at line 1519 of file dbplot.py.

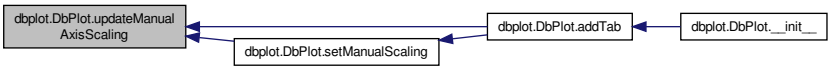
References dbplot.DbPlot.getManualScale(), dbplot.DbPlot.getPrecision(), and dbplot.DbPlot.hPlot.

Referenced by dbplot.DbPlot.addTab(), and dbplot.DbPlot.setManualScaling().

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- dbplot.py

10.9 dbplot.EditCurvesDlg Class Reference

Dialog to select visible curves.

Public Member Functions

- def __init__

  Create dialog to select visible curves.
- def closeEvent

  Reimplementation of the dialog close function.

10.9.1 Detailed Description

Dialog to select visible curves.

Definition at line 2066 of file dbplot.py.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 def dbplot.EditCurvesDlg.__init__ (*self, leftYCurves, rightYCurves, parent* = None)

Create dialog to select visible curves.

```
@param leftYCurves  = List of QwtPlotCurves in left y-axis.
@param rightYCurves = List of QwtPlotCurves in right y-axis.
@param parent       = Parent window (default None).
```

Handle to the dbplot main window. Dictionary with curves corresponding to each spinBox widget Dictionary with curves corresponding to each pushBotton widget

Definition at line 2075 of file dbplot.py.

References dbplot.EditCurvesDlg.curvesCheckBoxesDict, dbplot.EditCurvesDlg.curvesPushButtonsDict, dbplot.EditCurvesDlg.curvesSpinBoxesDict, dbplot.DbPlot.mainWindow, dbplot.AxisScalingDlg.mainWindow, and dbplot.EditCurvesDlg.mainWindow.

10.9.3 Member Function Documentation

10.9.3.1 def dbplot.EditCurvesDlg.closeEvent (*self, event*)

Reimplementation of the dialog close function.

Uncheck the toolbar action.

Definition at line 2166 of file dbplot.py.

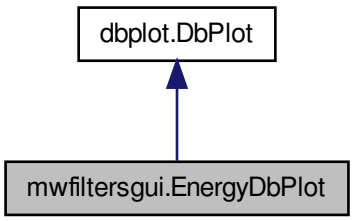
The documentation for this class was generated from the following file:

- dbplot.py

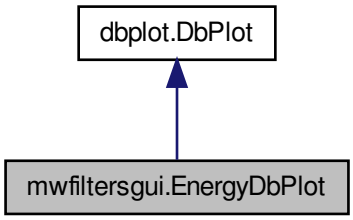
10.10 mwfiltersgui.EnergyDbPlot Class Reference

Class derived from DbPlot to allow adding some widgets for plotting energy and power.

Inheritance diagram for mwfiltersgui.EnergyDbPlot:



Collaboration diagram for mwfiltersgui.EnergyDbPlot:



Additional Inherited Members

10.10.1 Detailed Description

Class derived from DbPlot to allow adding some widgets for plotting energy and power.

Definition at line 95 of file mwfiltersgui.py.

The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.11 libcommonfunc.FrequencyTransformBP Class Reference

Frequency transformation fro band pass to low pass prototype.

Public Member Functions

- def __init__

  Create a frequency transformation instance.
- def normFreq

  Normalize frequency to lowpass prototype.
- def unormFreq

  Unnormalize frequency from lowpass prototype.
- def unormGroupDelay

  Convert lowpass prototype group delay to unnormalized group delay.
- def normGroupDelay

  Convert unnormalized group delay to lowpass prototype group delay.

10.11.1 Detailed Description

Frequency transformation fro band pass to low pass prototype.

Definition at line 4071 of file libcommonfunc.py.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 def libcommonfunc.FrequencyTransformBP.__init__ (*self, P*)

Create a frequency transformation instance.

Fractional bandwidth.

Definition at line 4076 of file libcommonfunc.py.

References libcommonfunc.FrequencyTransformBP.BW, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, libcommonfunc.Sparameters.FBW, and libcommonfunc.FrequencyTransformBP.FBW.

10.11.3 Member Function Documentation

10.11.3.1 def libcommonfunc.FrequencyTransformBP.normFreq (*self, freq*)

Normalize frequency to lowpass prototype.

```
@param freq = Vector with frequency values to convert (in Hz). Numpy array.
@return Vector with lowpass prototype normalized frequency values. Numpy array.

[Cameron eq. (3.113)]:
```

$$\omega' = \frac{f_0}{\Delta f}\left(\frac{f}{f_0} - \frac{f_0}{f}\right)$$

Definition at line 4095 of file libcommonfunc.py.

References libcommonfunc.FrequencyTransformBP.BW, libcommonfunc.Sparameters.f0, and libcommonfunc.FrequencyTransformBP.f0.

10.11.3.2 def libcommonfunc.FrequencyTransformBP.normGroupDelay (*self, groupDelay*)

Convert unnormalized group delay to lowpass prototype group delay.

```
@param groupDelay = Double with unnormalized group delay (in seconds) value to convert.
@return groupDelay = normalized filter group delay value.

[Cameron, eq. (3.91)]:
```

$\tau_0 = \frac{2}{\Delta\omega}\tau'_0$

Definition at line 4157 of file libcommonfunc.py.

References libcommonfunc.FrequencyTransformBP.BW.

10.11.3.3 def libcommonfunc.FrequencyTransformBP.unormFreq (*self, w*)

Unnormalize frequency from lowpass prototype.

```
@param w = Vector with normalized frequency values to convert. Numpy array.
@return Vector with unnormalized frequency values (in Hz). Numpy array.
```

$$\omega' = \frac{f_0}{\Delta f}\left(\frac{f}{f_0} - \frac{f_0}{f}\right)$$

$$f^2 - \omega'\Delta f f - f_0^2 = 0$$

$$f = \frac{\omega'\Delta f + \sqrt{(\omega'\Delta f)^2 + 4f_0^2}}{2}$$

with $\omega'' = \omega'\Delta f / f_0$:

$$f = \frac{f_0}{2}\left(\omega'' + \sqrt{\omega''^2 + 4}\right)$$

Definition at line 4126 of file libcommonfunc.py.

References libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, libcommonfunc.Sparameters.-FBW, and libcommonfunc.FrequencyTransformBP.FBW.

10.11.3.4 def libcommonfunc.FrequencyTransformBP.unormGroupDelay (*self, freq, groupDelay*)

Convert lowpass prototype group delay to unnormalized group delay.

```
@param freq = Vector with unnormalized frequency values (in Hz). Numpy array.
@param groupDelay = Vector with lowpass prototype group delay values to convert. Numpy array.
@return Vector with unnormalized filter group delay values (in seconds). Numpy array.

[Cameron, eq. (3.90)]:
```

$\tau_{BPF} = \frac{\omega_0}{\Delta\omega}\left(\frac{1}{\omega_0} - \frac{\omega_0}{\omega^2}\right)\tau' = \frac{1}{2\pi\Delta f}\left(1 + \left(\frac{f_0}{f}\right)^2\right)\tau'$

Definition at line 4144 of file libcommonfunc.py.

References libcommonfunc.FrequencyTransformBP.BW, libcommonfunc.Sparameters.f0, and libcommonfunc.-FrequencyTransformBP.f0.

The documentation for this class was generated from the following file:

- libcommonfunc.py

10.12 mwfiltersgui.HelpForm Class Reference

GUI dialog to display help.

Public Member Functions

- def __init__

  Constructor: Creates dialog window to display html help file.

10.12.1 Detailed Description

GUI dialog to display help.

Definition at line 2021 of file mwfiltersgui.py.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 def mwfiltersgui.HelpForm.__init__ (*self, page, parent* = None)

Constructor: Creates dialog window to display html help file.

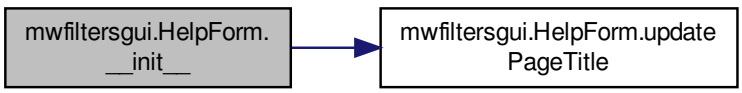
```
@param page   = html file name, relative to the program folder.
@param parent = Parent widget, in this case is coupling matrix mainWindow.
```

Copy of the mainWindow class instance. It is necessary to store a copy since it must be used to access actions and resources.

Definition at line 2029 of file mwfiltersgui.py.

References dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.HelpForm.pageLabel, mwfiltersgui.HelpForm.textBrowser, and mwfiltersgui.HelpForm.updatePageTitle().

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.13 libcommonfunc.licenseError Class Reference

License error catch.

10.13.1 Detailed Description

License error catch.

Definition at line 125 of file libcommonfunc.py.

The documentation for this class was generated from the following file:

- libcommonfunc.py

10.14 mwfiltersgui.MainWindow Class Reference

GUI main window class.

Public Member Functions

- def __init__

  Constructor to initialize main window.
- def okToContinue

  This function checks if there are unsaved changes and, if there are, aks the user to save them.
- def updateStatus

  Print message and update window title.
- def fileRead

  Reads parameters from a file.
- def closeSPlotMagPhaseWindow

  Close SPlotMagPhase plot window, if it exists.
- def closeResultsWindow

  Close SPlotCompare plot windows, if they exist.
- def closeCMWindows

  Close Coupling Matrix window and CM error plot window, if they exist.
- def cleanup

  Clean up previous results and any open plot or coupling matrix window.
- def plotSParameters

  Plot [S] parameters magnitude and phase graph, if they are available.
- def plotMasks

  Add masks to [S] plots.
- def updateWindows

  Update all plot windows and CM window if they exist.

Functions automatically excuted when the user interacts with the GUI

- def closeEvent

  Reimplementation of the window close function.
- def on_action_HelpAbout_triggered

  Show "HelpAbout"" dialog from help menu.
- def on_action_HelpHelp_triggered

  Show "HelpHelp"" dialog from help menu.
- def on_action_FileExit_triggered

  Close application.
- def on_action_Execute_triggered

  Execute synthesis computation.
- def on_action_CouplingMatrix_triggered

  Opens dialog to display and rotate coupling matrices.
- def on_action_Edit_triggered

  Opens dialog to edit parameters.
- def on_action_FileSave_triggered

  Save parameters to file.
- def on_action_FileSaveAs_triggered

  Ask for a file name and save parameters to file.
- def on_action_FileOpen_triggered

Ask for a file name and load parameters from this file.

- def on_action_Mask_triggered

  If checked: Ask for a file name and load specification mask from this file.
- def on_action_Sopen_triggered

  Ask for a file name and load [S] from this file in Touchstone format.
- def on_action_FileNew_triggered

  Creates an empty instance of Parameters Class and opens dialog to edit parameters.
- def on_action_ListAdd_triggered

  Store current result in self.results_comboBox list and self.listResults.
- def on_action_ListRemove_triggered

  Remove result matrix from self.results_comboBox and self.listResults lists.
- def on_results_comboBox_currentIndexChanged

  Load a new result.
- def on_action_Compare_triggered

  Compare results in list.

10.14.1 Detailed Description

GUI main window class.

Definition at line 998 of file mwfiltersgui.py.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 def mwfiltersgui.MainWindow.__init__ (*self, fileName, parent* = None)

Constructor to initialize main window.

```
@param fileName = Name of file to automatically open after creating the window. If equal to None, no f
@param parent = Parent widget, in this case is None.
```

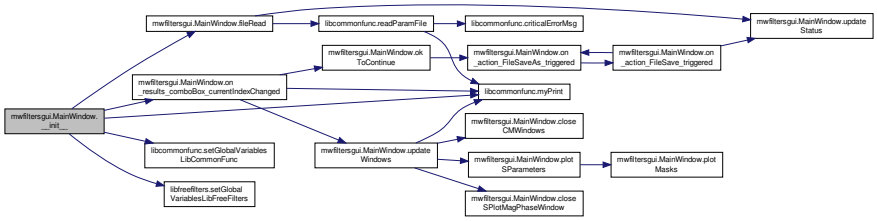
Variable to store the modified or unmodifed status of parameters class instance P. Variable that will avoid cleanup if set to True. It is used when a method that calls self.clenaup() is called by the GUI and not by the user. The user may want to avoid cleanup in some situations. Name of the parameters file to process. When main Window is created, self.fileName is a copy of the file name given in the command line when calling the application. Later self.fileName can be modified when opening a parameter file with the GUI. Application name, to be accesible outside this module. Last open directory name Parameters class instance containing filter and synthesis parameters. Read from the parameters file and -maybe- modified with the Parameter Edit Dialog. Copy of the CharPolys class instance containing characteristic polynomials. Copy of the CouplingMatrices class instance containing coupling matrices. Copy of the SParameters class instance containing [S] parameters. Copy of the FrequencyTransform class instance containing the frequency transform. Specification Mask class instance. Flag that is True when a successful computation has been done. When the flag is False, no results are plotted. QMainWindow that contains the parameter edit window. QwtPlot that contains the Magnitude and Phase S-parameter plot. QwtPlot that compares S-parameters obtained from CP from those obtained from CM. QwtPlot that compares S-parameters for different results. QwtPlot that contains stored energy and dissipated power QwtPlot that contains S-parameters for sensitivity analysis QMainWindow that contains the coupling matrix table. List of results to compare Last index of results selection comboBox Flag that controls if frequency axis must be autoscaled. It will become true after changing the sweep min and max frequencies. Store the mainWindow class instance into global variable 'mainWindow' defined in the libcommonfunc.py and libfreefilters.py in order to let the console interface functions in libcommonfunc.py and libfreefilters.py access the GUI mainWindow. The GUI methods access the mainWindow class through ParamEditDlg.mainWindow and MainWindow self attributes.

Definition at line 1006 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.applicationName, mwfiltersgui.MainWindow.autoScaleFreq, mwfiltersgui.MainWindow.CM, mwfiltersgui.MainWindow.CMdialog, mwfiltersgui.MainWindow.computed, mwfiltersgui.MainWindow.CP, mwfiltersgui.MainWindow.dirty, mwfiltersgui.MainWindow.EnergyPlot, mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, mwfiltersgui.MainWindow.fileRead(), libcommonfunc.Sparameters.FT,

mwfiltersgui.MainWindow.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.CouplingMatrices.FT, mwfiltersgui.MainWindow.indexResults, mwfiltersgui.MainWindow.lastDir, mwfiltersgui.MainWindow.listResults, libcommonfunc.myPrint(), mwfiltersgui.MainWindow.noCleanup, mwfiltersgui.MainWindow.on_results_comboBox_currentIndexChanged(), mwfiltersgui.MainWindow.P, mwfiltersgui.MainWindow.paramEdit, mwfiltersgui.MainWindow.results_comboBox, libcommonfunc.setGlobalVariablesLibCommonFunc(), libfreefilters.setGlobalVariablesLibFreeFilters(), mwfiltersgui.MainWindow.settings, mwfiltersgui.MainWindow.SM, mwfiltersgui.MainWindow.SP, libcommonfunc.CouplingMatrices.SP, mwfiltersgui.MainWindow.SPlotCompare, mwfiltersgui.MainWindow.SPlotError, mwfiltersgui.MainWindow.SPlotMagPhase, and mwfiltersgui.MainWindow.SPlotSensitivity.

Here is the call graph for this function:



10.14.3 Member Function Documentation

10.14.3.1 def mwfiltersgui.MainWindow.cleanup (*self*)

Clean up previous results and any open plot or coupling matrix window.

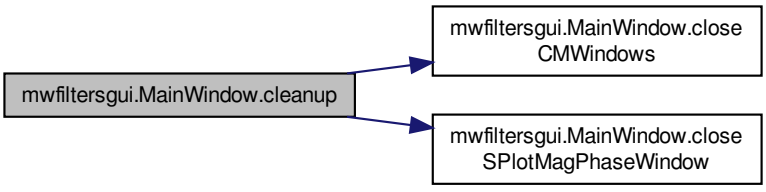
Clean up is disabled if self.nocleanup is True, in that case this method does nothing. Sets self.dirty to False.

Definition at line 1275 of file mwfiltersgui.py.

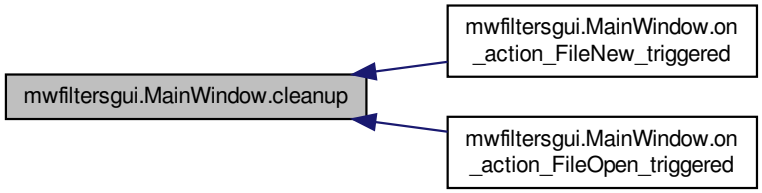
References mwfiltersgui.MainWindow.closeCMWindows(), mwfiltersgui.MainWindow.closeSPlotMagPhaseWindow(), mwfiltersgui.MainWindow.CM, mwfiltersgui.MainWindow.CP, libcommonfunc.Sparameters.FT, mwfiltersgui.MainWindow.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.CouplingMatrices.FT, mwfiltersgui.MainWindow.noCleanup, mwfiltersgui.MainWindow.P, mwfiltersgui.MainWindow.SP, and libcommonfunc.CouplingMatrices.SP.

Referenced by mwfiltersgui.MainWindow.on_action_FileNew_triggered(), and mwfiltersgui.MainWindow.on_action_FileOpen_triggered().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.3.2 def mwfiltersgui.MainWindow.fileRead (*self, fname*)

Reads parameters from a file.

Parameters

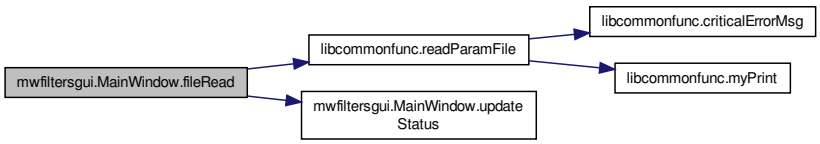
fname	= File name.

Definition at line 1194 of file mwfiltersgui.py.

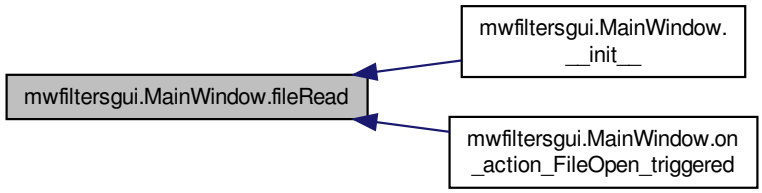
References mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, libcommonfunc.Sparameters.FT, mwfiltersgui.MainWindow.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.CouplingMatrices.FT, mwfiltersgui.MainWindow.lastDir, mwfiltersgui.MainWindow.P, mwfiltersgui.MainWindow.paramEdit, libcommonfunc.readParamFile(), and mwfiltersgui.MainWindow.updateStatus().

Referenced by mwfiltersgui.MainWindow.__init__(), and mwfiltersgui.MainWindow.on_action_FileOpen_triggered().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.3.3 def mwfiltersgui.MainWindow.okToContinue (*self*)

This function checks if there are unsaved changes and, if there are, aks the user to save them.

The function is executed by the GUI when there is an action that will destroy the parameter class P: Close application, Open file, New file, etc.

Returns

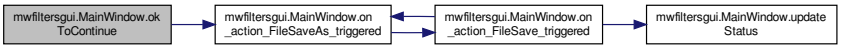
ok (True / False) = If True the use has agreed to continue, if False the user cancels the operation.

Definition at line 1155 of file mwfiltersgui.py.

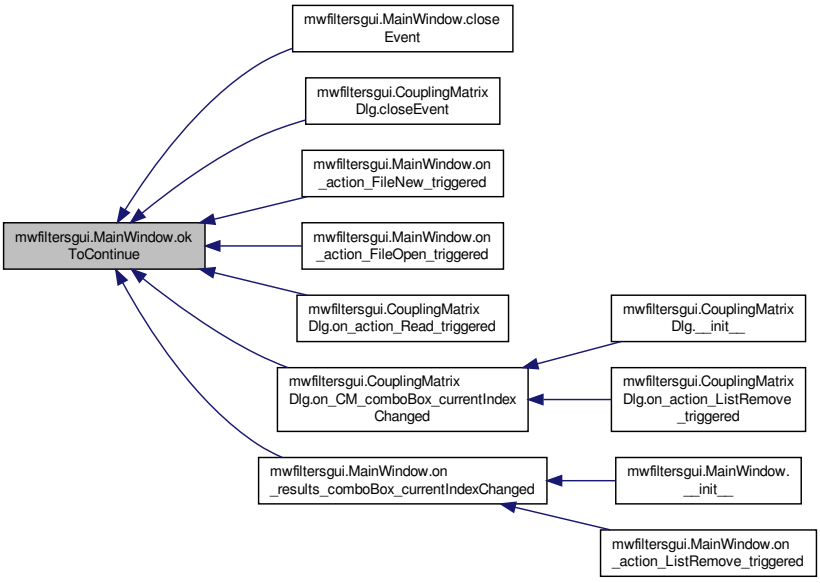
References mwfiltersgui.MainWindow.dirty, mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, and mwfiltersgui.MainWindow.on_action_FileSaveAs_triggered().

Referenced by mwfiltersgui.MainWindow.closeEvent(), mwfiltersgui.CouplingMatrixDlg.closeEvent(), mwfiltersgui.MainWindow.on_action_FileNew_triggered(), mwfiltersgui.MainWindow.on_action_FileOpen_triggered(), mwfiltersgui.CouplingMatrixDlg.on_action_Read_triggered(), mwfiltersgui.CouplingMatrixDlg.on_CM_comboBox_currentIndexChanged(), and mwfiltersgui.MainWindow.on_results_comboBox_currentIndexChanged().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.3.4 def mwfiltersgui.MainWindow.on_action_Compare_triggered (*self, checked*)

Compare results in list.

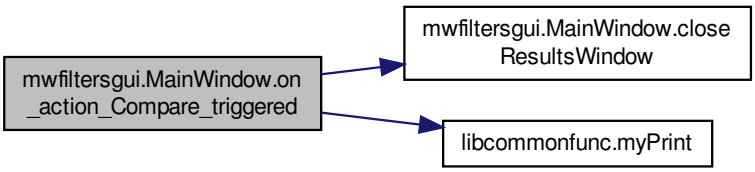
This function is automatically executed by the GUI when the user toggles the 'Compare results' button.

Definition at line 1949 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.closeResultsWindow(), mwfiltersgui.MainWindow.listResults, libcommonfunc.myPrint(), and mwfiltersgui.MainWindow.SPlotCompare.

Referenced by mwfiltersgui.MainWindow.on_action_Sopen_triggered().

Here is the call graph for this function:



Here is the caller graph for this function:

10.14.3.5 def mwfiltersgui.MainWindow.on_action_CouplingMatrix_triggered (*self, checked*)

Opens dialog to display and rotate coupling matrices.

This function is automatically executed by the GUI when the user triggers the 'Coupling Matrix' action.

Definition at line 1636 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.CM, and mwfiltersgui.MainWindow.CMdialog.

10.14.3.6 def mwfiltersgui.MainWindow.on_action_Edit_triggered (*self, checked*)

Opens dialog to edit parameters.

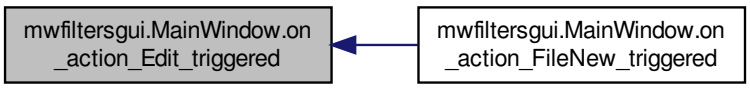
This function is automatically executed by the GUI when the user triggers the Edit action.

Definition at line 1651 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.P, and mwfiltersgui.MainWindow.paramEdit.

Referenced by mwfiltersgui.MainWindow.on_action_FileNew_triggered().

Here is the caller graph for this function:



10.14.3.7 def mwfiltersgui.MainWindow.on_action_Execute_triggered (*self, checked*)

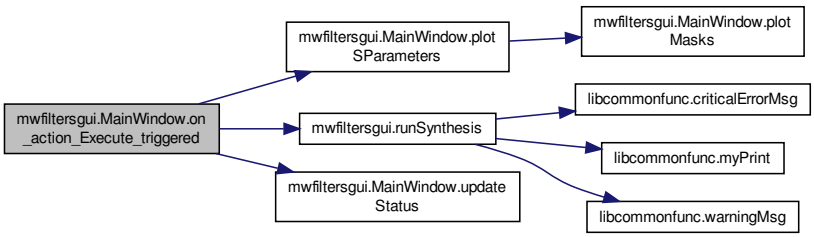
Execute synthesis computation.

This function is automatically executed by the GUI when the user triggers the Execute action.

Definition at line 1580 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.CM, mwfiltersgui.MainWindow.CMdialog, mwfiltersgui.MainWindow.computed, mwfiltersgui.MainWindow.CP, mwfiltersgui.MainWindow.EnergyPlot, libcommonfunc.Sparameters.FT, mwfiltersgui.MainWindow.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.CouplingMatrices.FT, mwfiltersgui.MainWindow.P, mwfiltersgui.MainWindow.paramEdit, mwfiltersgui.MainWindow.plotSParameters(), mwfiltersgui.runSynthesis(), mwfiltersgui.MainWindow.SP, libcommonfunc.CouplingMatrices.SP, mwfiltersgui.MainWindow.SPlotError, and mwfiltersgui.MainWindow.updateStatus().

Here is the call graph for this function:



10.14.3.8 def mwfiltersgui.MainWindow.on_action_FileExit_triggered (*self, checked*)

Close application.

This function is automatically executed by the GUI when the user triggers the FileExit action.

Definition at line 1570 of file mwfiltersgui.py.

10.14.3.9 def mwfiltersgui.MainWindow.on_action_FileNew_triggered (*self, checked*)

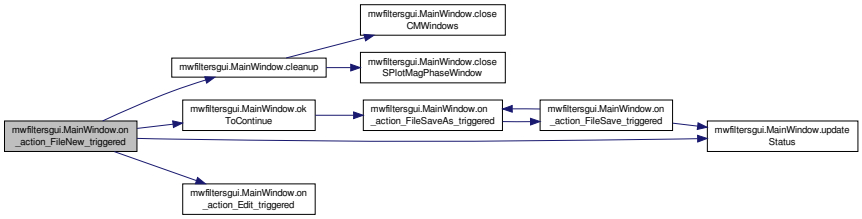
Creates an empty instance of Parameters Class and opens dialog to edit parameters.

This function is automatically executed by the GUI when the user triggers the FileNew action.

Definition at line 1821 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.cleanup(), mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, mwfiltersgui.MainWindow.okToContinue(), mwfiltersgui.MainWindow.on_action_Edit_triggered(), mwfiltersgui.MainWindow.P, mwfiltersgui.MainWindow.paramEdit, and mwfiltersgui.MainWindow.updateStatus().

Here is the call graph for this function:



10.14.3.10 def mwfiltersgui.MainWindow.on_action_FileOpen_triggered (*self, checked*)

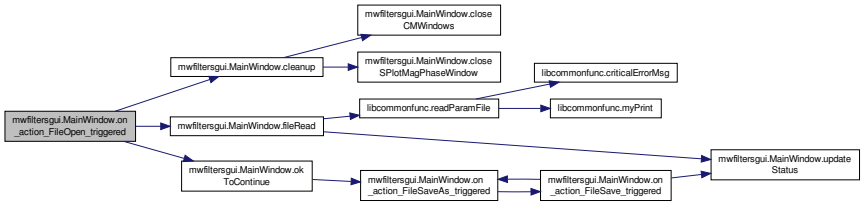
Ask for a file name and load parameters from this file.

This function is automatically executed by the GUI when the user triggers the FileOpen action.

Definition at line 1718 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.cleanup(), mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, mwfiltersgui.MainWindow.fileRead(), mwfiltersgui.MainWindow.lastDir, mwfiltersgui.MainWindow.okToContinue(), and mwfiltersgui.MainWindow.paramEdit.

Here is the call graph for this function:



10.14.3.11 def mwfiltersgui.MainWindow.on_action_FileSave_triggered (*self, checked*)

Save parameters to file.

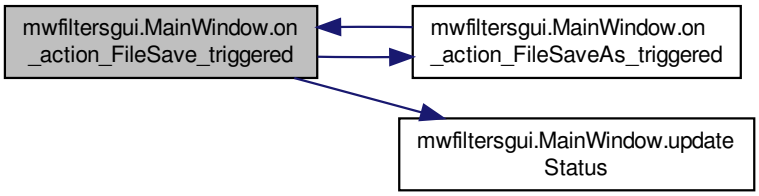
This function is automatically executed by the GUI when the user triggers the FileSave action.

Definition at line 1671 of file mwfiltersgui.py.

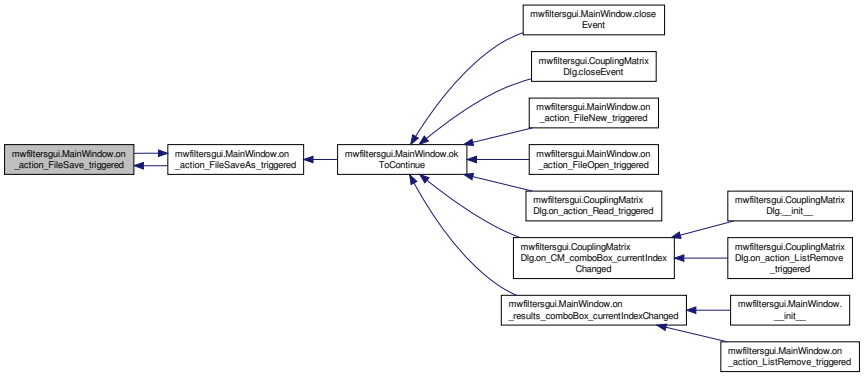
References mwfiltersgui.MainWindow.dirty, mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, mwfiltersgui.MainWindow.on_action_FileSaveAs_triggered(), mwfiltersgui.MainWindow.P, and mwfiltersgui.MainWindow.updateStatus().

Referenced by mwfiltersgui.MainWindow.on_action_FileSaveAs_triggered().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.3.12 def mwfiltersgui.MainWindow.on_action_FileSaveAs_triggered (*self, checked*)

Ask for a file name and save parameters to file.

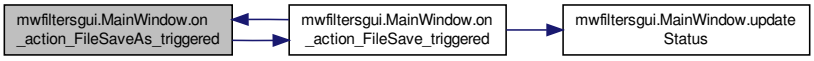
This function is automatically executed by the GUI when the user triggers the FileSaveAs action.

Definition at line 1692 of file mwfiltersgui.py.

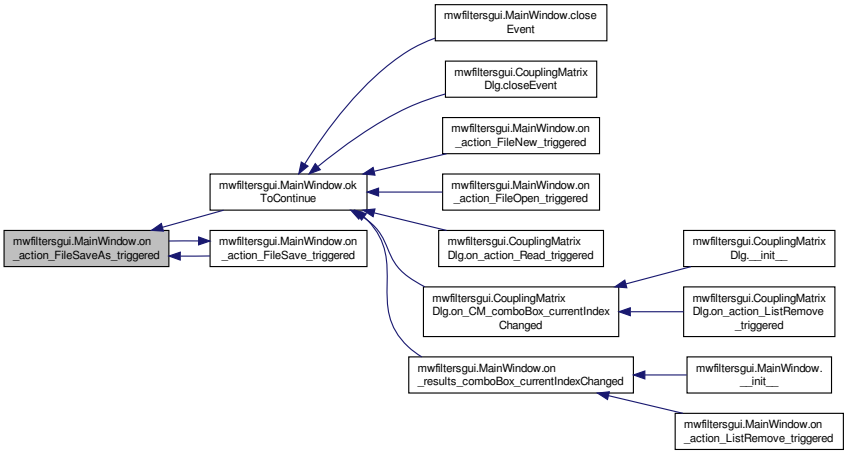
References mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, mwfiltersgui.MainWindow.on_-action_FileSave_triggered(), and mwfiltersgui.MainWindow.P.

Referenced by mwfiltersgui.MainWindow.okToContinue(), and mwfiltersgui.MainWindow.on_action_FileSave_-triggered().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.3.13 def mwfiltersgui.MainWindow.on_action_HelpAbout_triggered (*self, checked*)

Show "HelpAbout"" dialog from help menu.

This function is automatically executed by the GUI when the user selects the menu Help/About. The information shown is read from the file docstring.

Definition at line 1536 of file mwfiltersgui.py.

10.14.3.14 def mwfiltersgui.MainWindow.on_action_HelpHelp_triggered (*self*)

Show "HelpHelp"" dialog from help menu.

This function is automatically executed by the GUI when the user selects the menu Help/Help. The information shown is read file Help/GUI_help.html.

Definition at line 1558 of file mwfiltersgui.py.

10.14.3.15 def mwfiltersgui.MainWindow.on_action_ListAdd_triggered (*self, checked*)

Store current result in self.results_comboBox list and self.listResults.

Parameters

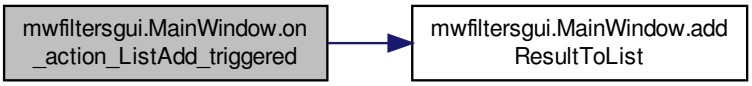
checked	= SIGNAL/SLOT parameter.

Definition at line 1844 of file mwfiltersgui.py.

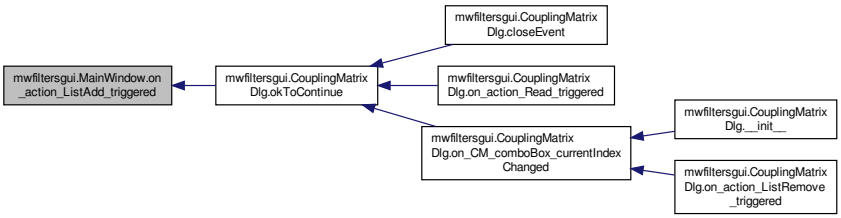
References mwfiltersgui.MainWindow.addResultToList(), mwfiltersgui.MainWindow.CM, mwfiltersgui.MainWindow.CP, libcommonfunc.Sparameters.FT, mwfiltersgui.MainWindow.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.CouplingMatrices.FT, mwfiltersgui.MainWindow.P, mwfiltersgui.MainWindow.paramEdit, mwfiltersgui.MainWindow.SP, and libcommonfunc.CouplingMatrices.SP.

Referenced by mwfiltersgui.CouplingMatrixDlg.okToContinue().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.3.16 def mwfiltersgui.MainWindow.on_action_Mask_triggered (*self, checked*)

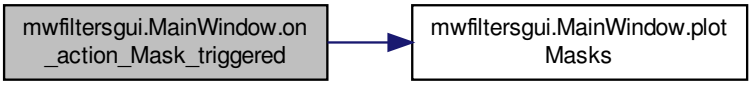
If checked: Ask for a file name and load specification mask from this file.

Plot the mask. If unchecked: Delete the mask data from memory and delete the mask curve from the plots. This function is automatically executed by the GUI when the user triggers the Mask action.

Definition at line 1747 of file mwfiltersgui.py.

References mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, mwfiltersgui.MainWindow.lastDir, mwfiltersgui.MainWindow.plotMasks(), mwfiltersgui.MainWindow.SM, mwfiltersgui.MainWindow.SPlotError, mwfiltersgui.MainWindow.SPlotMagPhase, and mwfiltersgui.MainWindow.SPlotSensitivity.

Here is the call graph for this function:



10.14.3.17 def mwfiltersgui.MainWindow.on_action_Sopen_triggered (*self, checked*)

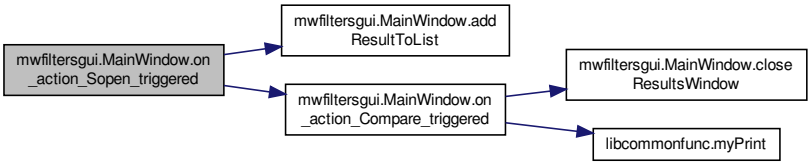
Ask for a file name and load [S] from this file in Touchstone format.

Plot the [S] parameters in the results comparison window. This function is automatically executed by the GUI when the user triggers the Sopen action.

Definition at line 1799 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.addResultToList(), mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, mwfiltersgui.MainWindow.lastDir, and mwfiltersgui.MainWindow.on_action_Compare_triggered().

Here is the call graph for this function:



10.14.3.18 def mwfiltersgui.MainWindow.on_results_comboBox_currentIndexChanged (*self, index*)

Load a new result.

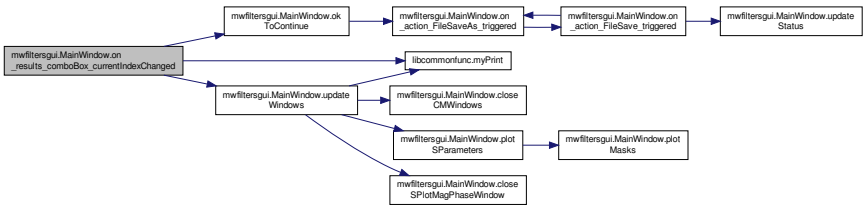
This function is automatically executed by the GUI when the user changes the results list comboBox. It does nothing if self.noCleanup is True.

Definition at line 1902 of file mwfiltersgui.py.

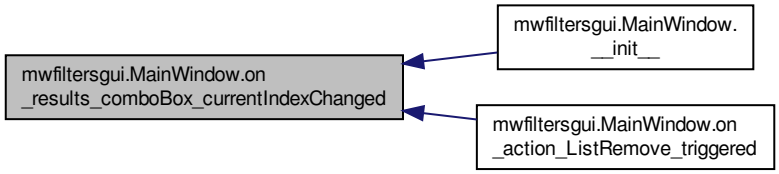
References mwfiltersgui.MainWindow.CM, mwfiltersgui.MainWindow.CP, libcommonfunc.Sparameters.FT, mwfiltersgui.MainWindow.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.CouplingMatrices.FT, mwfiltersgui.MainWindow.indexResults, mwfiltersgui.MainWindow.listResults, libcommonfunc.myPrint(), mwfiltersgui.MainWindow.noCleanup, mwfiltersgui.MainWindow.okToContinue(), mwfiltersgui.MainWindow.P, mwfiltersgui.MainWindow.SP, libcommonfunc.CouplingMatrices.SP, and mwfiltersgui.MainWindow.updateWindows().

Referenced by mwfiltersgui.MainWindow.__init__(), and mwfiltersgui.MainWindow.on_action_ListRemove_triggered().

Here is the call graph for this function:



Here is the caller graph for this function:



10.14.3.19 def mwfiltersgui.MainWindow.updateStatus (*self, message*)

Print message and update window title.

Prints message in the log widget at the main window background and also at the status bar for 5 sec. Updates main window title with open file name and modified state.

Parameters

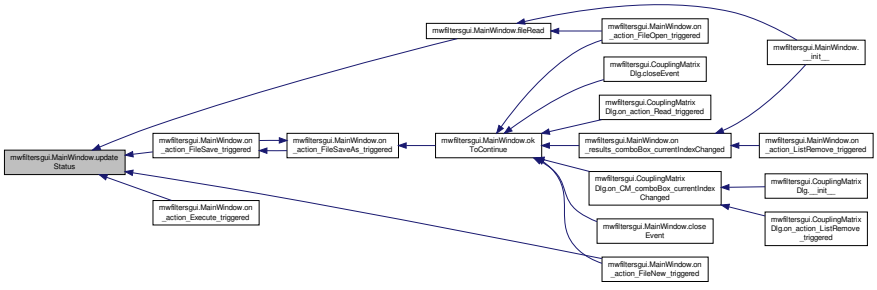
message	= Message to print (string).

Definition at line 1176 of file mwfiltersgui.py.

References mwfiltersgui.MainWindow.dirty, mwfiltersgui.SpecMask.fileName, mwfiltersgui.MainWindow.fileName, and mwfiltersgui.MainWindow.P.

Referenced by mwfiltersgui.MainWindow.fileRead(), mwfiltersgui.MainWindow.on_action_Execute_triggered(), mwfiltersgui.MainWindow.on_action_FileNew_triggered(), and mwfiltersgui.MainWindow.on_action_FileSave_-triggered().

Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.15 mwfiltersgui.MatrixEditDlg Class Reference

GUI dialog to edit coupling matrix.

Public Member Functions

- def __init__

  Constructor: Creates dialog window to edit current coupling matrix.
- def udpateTopology

  Fill in Matrix topology table from values in self.mainWindow.CM.MatQ.
- def getTopologyFlags

  Return matrices of flags that describe the topology of the real part and the imaginary part of the coupling matrix, based on the entries in the topology table.
- def topologyItemClicked

  Update the contents of self.topology_tableWidget after the user has clicked an item.
- def setMatrixSize

  Set maximum row and column indices in matrixEditDlg spinBoxes, according to size of self.mainWindow.CM.MatQ.M.

Functions automatically excuted when the user interacts with the GUI

- def dialogChanged

  Custom slot in Qt designer connected to widgets "changed" signal.
- def on_edit_listWidget_itemDoubleClicked

  Load coupling matrix saved in backup list.
- def on_undo_pushButton_clicked

  Undo last edit action from edit actions list.
- def on_clear_pushButton_clicked

  Clear edit actions list.
- def on_apply_pushButton_clicked

  Apply selected action to the matrix.
- def closeEvent

  Reimplementation of the window close function.

10.15.1 Detailed Description

GUI dialog to edit coupling matrix.

Definition at line 2161 of file mwfiltersgui.py.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 def mwfiltersgui.MatrixEditDlg.__init__ (*self, parent*)

Constructor: Creates dialog window to edit current coupling matrix.

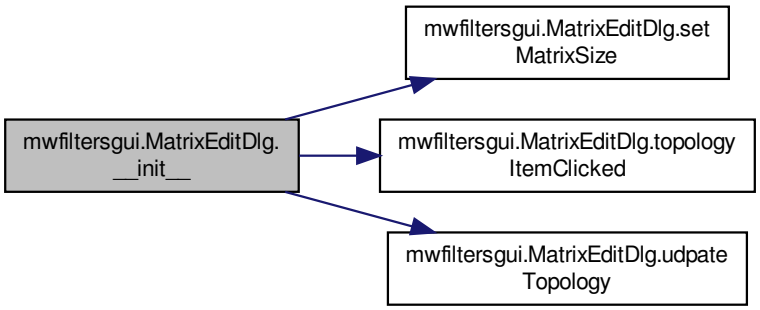
```
@param parent = Parent widget, in this case is coupling matrix mainWindow.
```

Copy of the coupling matrix mainWindow class instance. It is necessary to store a copy since it must be accessed by some member functions. String that explains the last edit action. This is what is pushed into the edit action list after edit actions. Shallow copy of the CM.bkpMatQ list, for easier access. Shallow copy of the CM.bkpActionslist, for easier access. Variable to store a list of widgets with bad input. Only when this list is empty, he "Accept" and "Compute" buttons are enabled. Text corresponding to couplings type in topology table Colors corresponding to couplings type in topology table Variable to store the modified/unmodifed status of the dialog widgets. The dialogChanged() method is automatically called when the user modifies the edit widgets contents and sets this variable to True. This variable must be set to False at the end of the **init** dialog constructor, since dialogChanged() is called when the constructor sets the edit widgets contents. This attribute is defined in this class only for compatibility with the dialogChanged() method, which has been copied literally (except for the name of pushButtons) from ParamEditDlg class.

Definition at line 2168 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.badInput, mwfiltersgui.MatrixEditDlg.bkpActions, libcommonfunc.CouplingMatrices.bkpActions, mwfiltersgui.MatrixEditDlg.bkpMatQ, libcommonfunc.CouplingMatrices.bkpMatQ, mwfiltersgui.MatrixEditDlg.dialogModified, mwfiltersgui.MatrixEditDlg.lastEditAction, dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.SetCouplingDlg.mainWindow, mwfiltersgui.MatrixEditDlg.mainWindow, mwfiltersgui.MatrixEditDlg.setMatrixSize(), mwfiltersgui.SetCouplingDlg.topolColors, mwfiltersgui.MatrixEditDlg.topolColors, mwfiltersgui.MatrixEditDlg.topology_tableWidget, mwfiltersgui.MatrixEditDlg.topologyItemClicked(), mwfiltersgui.SetCouplingDlg.topolText, mwfiltersgui.MatrixEditDlg.topolText, and mwfiltersgui.MatrixEditDlg.udpateTopology().

Here is the call graph for this function:



10.15.3 Member Function Documentation

10.15.3.1 def mwfiltersgui.MatrixEditDlg.closeEvent (*self, event*)

Reimplementation of the window close function.

Sets couplingMatrixDlg toolbar Edit action to unchecked.

Definition at line 2648 of file mwfiltersgui.py.

10.15.3.2 def mwfiltersgui.MatrixEditDlg.dialogChanged (*self*)

Custom slot in Qt designer connected to widgets "changed" signal.

This function is automatically executed by the GUI every time the user changes dialog values, and sets the self.dialogModified variable to True.

Definition at line 2433 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.badInput, and mwfiltersgui.MatrixEditDlg.dialogModified.

10.15.3.3 def mwfiltersgui.MatrixEditDlg.getTopologyFlags (*self*)

Return matrices of flags that describe the topology of the real part and the imaginary part of the coupling matrix, based on the entries in the topology table.

Returns

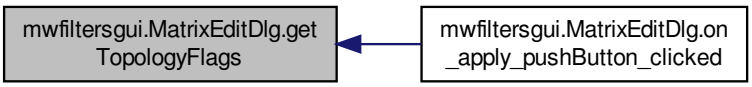
topologyReal = Numpy matrix of Bool type that has True values for elements with non-zero real part.
topologyImag = Numpy matrix of Bool type that has True values for elements with non-zero imag part.
topologySignReal = Int numpy array having -1,1 or 0 values at the position of coupling matrix entries with negative, positive or any real part, respectively.
topologySignImag = Int numpy array having -1,1 or 0 values at the position of coupling matrix entries with negative, positive or any imaginary part, respectively.

Definition at line 2341 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.topology_tableWidget.

Referenced by mwfiltersgui.MatrixEditDlg.on_apply_pushButton_clicked().

Here is the caller graph for this function:



10.15.3.4 def mwfiltersgui.MatrixEditDlg.on_apply_pushButton_clicked (*self, checked*)

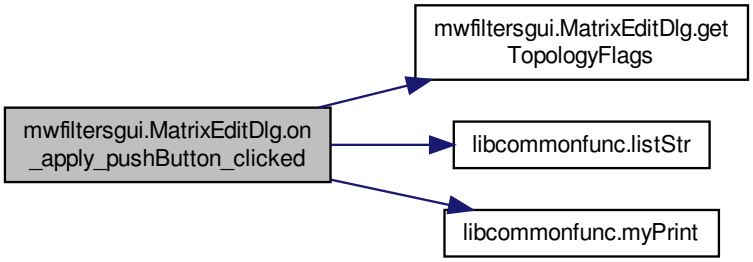
Apply selected action to the matrix.

The coupling matrix self.dirty attribute is set to True.

Definition at line 2528 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.getTopologyFlags(), mwfiltersgui.MatrixEditDlg.lastEditAction, libcommonfunc.listStr(), and libcommonfunc.myPrint().

Here is the call graph for this function:



10.15.3.5 def mwfiltersgui.MatrixEditDlg.on_clear_pushButton_clicked (*self, checked*)

Clear edit actions list.

Clears edit action list and the backup Qmatrices list.

Definition at line 2503 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.bkpActions, libcommonfunc.CouplingMatrices.bkpActions, mwfiltersgui.MatrixEditDlg.bkpMatQ, and libcommonfunc.CouplingMatrices.bkpMatQ.

10.15.3.6 def mwfiltersgui.MatrixEditDlg.on_undo_pushButton_clicked (*self, checked*)

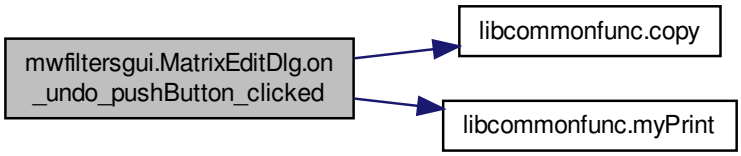
Undo last edit action from edit actions list.

Pops the edit action list and the backup Qmatrices list.

Definition at line 2479 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.bkpMatQ, libcommonfunc.CouplingMatrices.bkpMatQ, libcommonfunc.copy(), and libcommonfunc.myPrint().

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.16 libcommonfunc.MatrixQ Class Reference

Class that contains a coupling matrix, its name, the number of non-resonant nodes, the Q of resonators and some operations.

Public Member Functions

- def __init__

  Initialize MatrixQ class.
- def copy

  Function to copy instances of the MatrixQ class (copies the self instance).
- def setName

  Function to set name field of the instance of the MatrixQ class.
- def setFileExt

  Function to set file extension field of the instance of the MatrixQ class.
- def Qresonators

  Compute the quality factors of each resonant node of coupling matrix self.M.
- def lp2bp2cbw

  Compute bandpass and coupling bandwidth versions of the matrix.

- def rotateMatrix

  Rotate the coupling matrix self.M.
- def inflateMatrix

  Add external non-resonant nodes to the coupling matrix self.M at the source or load sides, or both.
- def scaleNode

  Scale a node of the coupling matrix self.M, by multiplying row [i] and column [i] by a factor.
- def rotAngleEliminate

  Compute the rotation angle such that, after applying a rotation of type rotationType, the element $M[m,n]$ of the matrix self.M is equal to zero.
- def addLossesQeff

  Add losses to resonators by substracting a factor 1/(Qeff∗FBW) to the diagonal of the coupling matrix.
- def setupOptimTopology

  Prepare optimization parameters depending on matrix topology and forced signs.
- def newMatQFromNonZero

  Return a new MatrixQ object that is a copy of self, except for the M matrix which is created from the NonZeroValues, NonZeroReal, NonZeroImag and Q parameters.
- def uniformQ

  Optimize the coupling matrix to achieve a prescribed uniform Q and [S] parameters as close as possible to those computed from the characteristic polynomials.
- def uniformQOptimize

  Optimization objective function for uniformQ function.
- def uniformQMask

  Optimize the coupling matrix to achieve a prescribed uniform Q and [S] parameters that comply with a specification mask.
- def uniformQOptimizeMask

  Optimization objective function for uniformQMask function.
- def saveMat

  Save this MatrixQ instance to an ascii file, including energy and power.
- def __str__

  Convert numpy matrix to string representation.

10.16.1 Detailed Description

Class that contains a coupling matrix, its name, the number of non-resonant nodes, the Q of resonators and some operations.

Definition at line 1566 of file libcommonfunc.py.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 def libcommonfunc.MatrixQ.__init__ (*self, parent, M, name, extraNodesS, extraNodesL, FT, fileExt* = `None`, *nodeScalingFactor* = `None`, *flagRotateAllowed* = `True`)

Initialize MatrixQ class.

Computes Q of resonators and stores the vector of Q values in self.Q .

Parameters

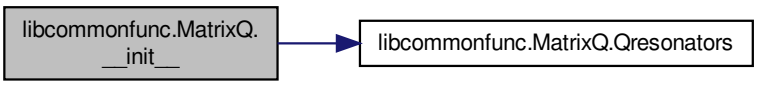
parent	= CouplingMatrices class instance to which this coupling matrix belongs.
M	= Coupling matrix.
name	= Matrix name.
extraNodesS	= Number of extra rows and columns in current matrix M at the source side (non-resonant nodes).

extraNodesL	= Number of extra rows and columns in current matrix M at the load side (non-resonant nodes).
FT	= Frequency transformation class instance.
fileExt	= Extension of file name, if this matrix is to be automatically saved in a file (with the output file name in the parameters class). If equal to None, this matrix is not automatically saved. Anyway the matrix can be manually saved. Default None.
nodeScaling-Factor	= Vector containing in each node position the factor of the node scaling applied to that node. Default None.
flagRotate-Allowed	= Flag that indicates if it is allowed to do a matrix rotation (Impossible after a resonant node scaling). Default True. Coupling matrix Matrix order Number of extra rows and columns in current matrix M at the source side (non-resonant nodes). Number of extra rows and columns in current matrix M at the load side (non-resonant nodes). Matrix name. It is converted to unicode to allow safe comparison with matrix names entered by the user in Matrix Edit Gui. Frequency transform to use when computing Q of resonators. Extension of file name, if this matrix is to be saved in a file. Otherwise None. CouplingMatrices class instance to which this coupling matrix belongs. Band pass version of the matrix Coupling bandwidth version of the matrix Coupling bandwidth with transmission lines version of the matrix Labels for rows and columns. Will be used for saving energy and power. Characteristic impedance of resonators Characteristic impedance of source Characteristic impedance of load

Definition at line 1582 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, libcommonfunc.MatrixQ.fileExt, libcommonfunc.MatrixQ.flagRotateAllowed, libcommonfunc.Sparameters.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.MatrixQ.M, libcommonfunc.MatrixQ.Mbp, libcommonfunc.MatrixQ.Mcbw, libcommonfunc.MatrixQ.McbwTL, libcommonfunc.MatrixQ.name, dbplot.CurveFamily.name, libcommonfunc.MatrixQ.nodeScalingFactor, libcommonfunc.MatrixQ.order, libcommonfunc.MatrixQ.parent, libcommonfunc.MatrixQ.Qresonators(), libcommonfunc.MatrixQ.rowIndices, libcommonfunc.MatrixQ.Zo, libcommonfunc.MatrixQ.ZoL, and libcommonfunc.MatrixQ.ZoS.

Here is the call graph for this function:



10.16.3 Member Function Documentation

10.16.3.1 def libcommonfunc.MatrixQ.__str__ (*self, prec* = 5)

Convert numpy matrix to string representation.

Rounds coupling matrix elements to given precision using complexStr() function.

Parameters

prec	= Number of significant digits (int). Default 5.

Returns

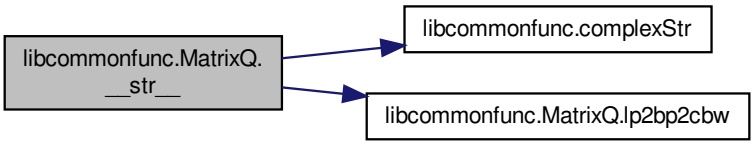
st = String representation of the numpy matrix self.M.

Definition at line 2791 of file libcommonfunc.py.

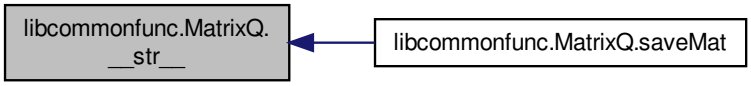
References libcommonfunc.complexStr(), libcommonfunc.MatrixQ.lp2bp2cbw(), libcommonfunc.MatrixQ.M, libcommonfunc.MatrixQ.Mcbw, libcommonfunc.MatrixQ.McbwTL, libcommonfunc.MatrixQ.name, and dbplot.CurveFamily.name.

Referenced by libcommonfunc.MatrixQ.saveMat().

Here is the call graph for this function:



Here is the caller graph for this function:



10.16.3.2 def libcommonfunc.MatrixQ.addLossesQeff (*self, Qeff*)

Add losses to resonators by substracting a factor 1/(Qeff∗FBW) to the diagonal of the coupling matrix.

The typical application is to set a prescribed flatness in the ideal response of a lossy filter synthesis by emulating the non-flat response of a lossy filter of quality factor equal to Qeff. Qeff = Quality factor of the resonators to emulate.

Definition at line 2216 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, and libcommonfunc.MatrixQ.M.

10.16.3.3 def libcommonfunc.MatrixQ.copy (*self*)

Function to copy instances of the MatrixQ class (copies the self instance).

Returns

New instance of MatrixQ class copied from the self instance.

Definition at line 1662 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, libcommonfunc.MatrixQ.fileExt, libcommonfunc.MatrixQ.flagRotateAllowed, libcommonfunc.Sparameters.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.MatrixQ.M, libcommonfunc.MatrixQ.name, dbplot.CurveFamily.name, libcommonfunc.MatrixQ.nodeScalingFactor, and libcommonfunc.MatrixQ.parent.

10.16.3.4 def libcommonfunc.MatrixQ.inflateMatrix (*self, position*)

Add external non-resonant nodes to the coupling matrix self.M at the source or load sides, or both.

```
The resulting self.M matrix will have one or two columns and rows more than the input one.
@param position = Position where we want to inflate the matrix (string). It can be 'source', 'load' or 'bo

If position == 'source'
```

$$Minflate = \begin{bmatrix} 0 & k_1 & 0 & \dots \\ k_1 & & & \\ 0 & & M & \\ \vdots & & & \end{bmatrix}$$

```
If position == 'load'
```

$$Minflate = \begin{bmatrix} & & & \vdots \\ & M & & 0 \\ & & & k_2 \\ \dots & 0 & k_2 & 0 \end{bmatrix}$$

```
If position == 'both'
```

$$Minflate = \begin{bmatrix} 0 & k_1 & 0 & \dots & 0 \\ k_1 & & & & \vdots \\ 0 & & M & & 0 \\ \vdots & & & & k_2 \\ 0 & \dots & 0 & k_2 & 0 \end{bmatrix}$$

```
where \form#117 and \form#118 are usually equal to 1 unless the first or last nodes of the matrix have bee
If this is the case, the value of \form#117 and \form#118 for that nodes are retrieved from the self.nodeS
```

Definition at line 2058 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, libcommonfunc.MatrixQ.-M, and libcommonfunc.MatrixQ.nodeScalingFactor.

10.16.3.5 def libcommonfunc.MatrixQ.lp2bp2cbw (*self*)

Compute bandpass and coupling bandwidth versions of the matrix.

```
<b>Coupling bandwidth matrix</b>

The diagonal elements of the coupling bandwidth matrix \form#84 is obtained as:
```

$$\Delta f_{ii} = f_i$$

where the unnormalised resonant frequencies (Hz) of the resonators are:

$$f_i = \frac{f_0}{2}\left(\omega'' + \sqrt{\omega''^2 + 4}\right)$$

with $\omega'' = -B_i \Delta f / f_0$ and $B_i = \Re M_{ii}$. The factor $\Delta f / f_0$ is the fractional bandwidth.

For non-resonant nodes $f_i = f_0$.

The couplings with source and load are:

$$\Delta f_{Si} = M_{Si}^2 \Delta f \qquad \Delta f_{Li} = M_{Li}^2 \Delta f$$

$$\Delta f_{SS} = \Delta f_{LL} = 0$$

where M_{ij} are the normalised low-pass coupling matrix elements and Δf is the bandwidth (MHz).

The couplings between inner nodes (resonators or non-resonant nodes) are:

$$\Delta f_{ij} = M_{ij}\Delta f$$

Coupling bandwidth matrix with transmission lines

For each matrix element \form#96, the scaling factor for the pass-band coupling matrix is

$$K_{ij} = \alpha_{ij} M_{ij}$$

where $[M]$ is the low-pass normalised matrix and $[K]$ is the pass-band coupling matrix.

$$\alpha_{ij} = \frac{\pi}{2}\frac{\mathrm{FBW}}{Z_0} \qquad \text{if } i \text{ and } j \neq S \text{ or } L$$

$$\alpha_{ij} = \sqrt{\frac{\pi}{2}\frac{\mathrm{FBW}}{Z_0 Z_{S,L}}} \qquad \text{if } i \text{ or } j = S \text{ or } L$$

$$\alpha_{SS} = \alpha_{SL} = \alpha_{LS} = \alpha_{LL} = 1$$

$$Z_0 = Z_S = Z_L = 50\Omega$$

The diagonal elements of the coupling bandwidth matrix \form#84 is obtained as:

$$\Delta f_{ii} = f_i$$

where the unnormalised resonant frequencies (Hz) of the resonators are:

$$f_i = \frac{f_0}{2}\left(\omega'' + \sqrt{\omega''^2 + 4}\right)$$

with $\omega'' = -B_i \Delta f / f_0$ and $B_i = \Re M_{ii}$. The factor $\Delta f / f_0$ is the fractional bandwidth.

For non-resonant nodes $f_i = f_0$.

The couplings with source and load are:

$$\Delta f_{Si} = \frac{f0}{K_{Si}} \qquad \Delta f_{Li} = \frac{f0}{K_{Li}}$$

$$\Delta f_{SS} = \Delta f_{LL} = 0$$

The couplings between inner nodes (resonators or non-resonant nodes) are:

$$\Delta f_{ij} = f_0 \sqrt{\left(\frac{K_{ij}}{A}\right)^2 + C}$$

with

$$A = \frac{1}{2}\left(\frac{f_j}{f_i} + \frac{f_i}{f_j}\right)$$

$$C = \left|\frac{f_j^2 - f_i^2}{f_j^2 + f_i^2}\right|^2$$

For syncronous nodes, \form#90 and therefore

$$A = 1 \qquad C = 0$$

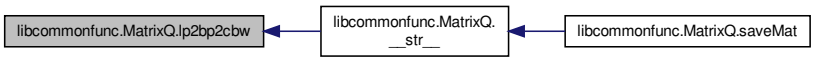
$$\Delta f_{ij} = K_{ij} f_0$$

Definition at line 1851 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.M, libcommonfunc.MatrixQ.Mbp, libcommonfunc.MatrixQ.Mcbw, libcommonfunc.MatrixQ.McbwTL, libcommonfunc.MatrixQ.Zo, libcommonfunc.MatrixQ.ZoL, and libcommonfunc.MatrixQ.ZoS.

Referenced by libcommonfunc.MatrixQ.__str__().

Here is the caller graph for this function:



10.16.3.6 def libcommonfunc.MatrixQ.newMatQFromNonZero (*self, NonZeroValues, NonZeroReal, NonZeroImag, numReal, Q*)

Return a new MatrixQ object that is a copy of self, except for the M matrix which is created from the NonZeroValues, NonZeroReal, NonZeroImag and Q parameters.

Parameters

NonZeroValues	= Double numpy vector with the values of the matrix entries which are different from zero. It is a concatenation of the non-zero real elements and the non-zero imaginary elements.
NonZeroReal	= Bool numpy array having True values at the position of coupling matrix entries with non-zero real part in the upper-triangular part, including the diagonal, of the matrix.
NonZeroImag	= Bool numpy array having True values at the position of coupling matrix entries with non-zero imaginary part in the upper-triangular part, including the diagonal, of the matrix. The entries corresponding to resonators are False, since the imaginary part will be obtained from the prescribed Q.
numReal	= Number of entries in the upper triangular part of the matrix, including the diagonal, with non-zero real part.
Q	= Prescribed uniform Q for the resonators.

Returns

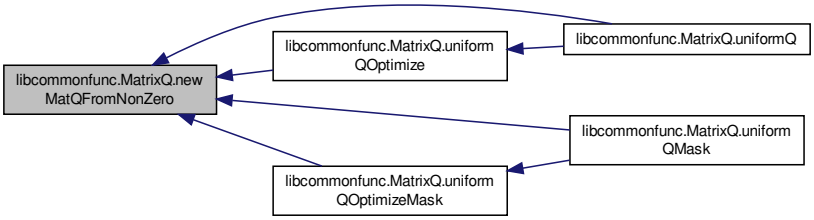
newMatQ = New MatrixQ object

Definition at line 2314 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, libcommonfunc.MatrixQ.fileExt, libcommonfunc.MatrixQ.flagRotateAllowed, libcommonfunc.Sparameters.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.MatrixQ.name, dbplot.CurveFamily.name, libcommonfunc.MatrixQ.nodeScalingFactor, libcommonfunc.MatrixQ.order, and libcommonfunc.MatrixQ.parent.

Referenced by libcommonfunc.MatrixQ.uniformQ(), libcommonfunc.MatrixQ.uniformQMask(), libcommonfunc.MatrixQ.uniformQOptimize(), and libcommonfunc.MatrixQ.uniformQOptimizeMask().

Here is the caller graph for this function:



10.16.3.7 def libcommonfunc.MatrixQ.Qresonators (*self*)

Compute the quality factors of each resonant node of coupling matrix self.M.

```
Frequency transform is self.FT and output Q is stored in self.Q.

For each node \form#62, the quality factor is
```

$$Q_n = \frac{1}{G_n \, \mathrm{FBW}}$$

where the loss conductance of the unloaded resonator is:

$$G_n = -\sum_k \Im M_{kn}$$

```
<b> Proof: </b>

In the parallel N+2 normalized lowpass prototype equivalent circuit, at each node we have current excitati
the current that flows away from the node through the circuit elements and couplings:
```

$$I_n = (G_n + jB_n + sC_n)\,V_n + \sum_{k \neq n} jM'_{nk} V_k + \sum_{k \neq n} G_{nk}(V_n - V_k)$$

where M'_{kn} are the real couplings, G_n is the unloaded resonator conductance (losses), G_{kn} is the conductance of resistive couplings between resonators and, obviously, the excitation is $I_n = 0$ for $n \neq S$. For $n = S$ we have $I_n = I_S$ (source) .

If we define the mutual admittances between nodes as:

$$Y_{nm} = \left. \frac{I_n}{V_m} \right|_{V_{k \neq m} = 0}$$

we have

$$Y_{nn} = G_n + jB_n + sC_n + \sum_{k \neq n} G_{kn}$$

and

$$Y_{nm} = jM'_{nm} - G_{nm}$$

In matrix form, with $C_n = 1$:

$$[Y] = j[M] + s[I]$$

where the elements of the coupling matrix $[M]$ are:

$$M_{nn} = \frac{Y_{nn} - s}{j} = B_n - jG_n - j\sum_{k \neq n} G_{kn}$$

and

$$M_{nm} = \frac{Y_{nm}}{j} = M'_{nm} + jG_{nm}$$

Therefore:

$$\Im M_{nn} = -G_n - \sum_{k \neq n} G_{kn} = -G_n - \sum_{k \neq n} \Im M_{kn}$$

and we can compute the resonator unloaded conductance as:

$$G_n = -\Im M_{nn} - \sum_{k \neq n} \Im M_{kn} = -\sum_{k} \Im M_{kn}$$

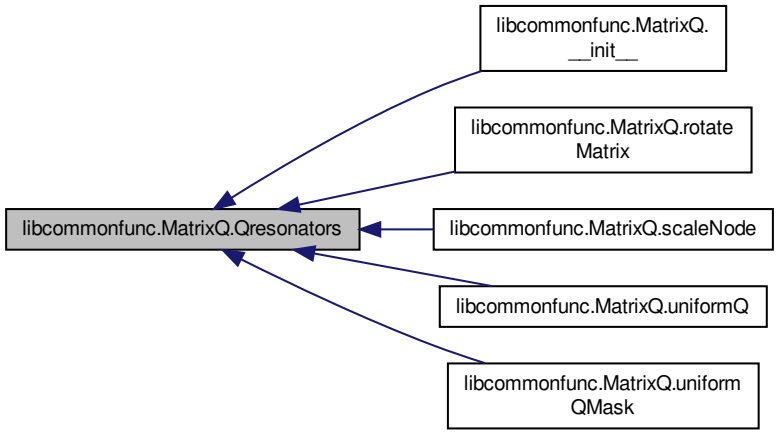
Q of resonators

Definition at line 1741 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, libcommonfunc.MatrixQ.-M, and libcommonfunc.MatrixQ.Q.

Referenced by libcommonfunc.MatrixQ.__init__(), libcommonfunc.MatrixQ.rotateMatrix(), libcommonfunc.MatrixQ.-scaleNode(), libcommonfunc.MatrixQ.uniformQ(), and libcommonfunc.MatrixQ.uniformQMask().

Here is the caller graph for this function:



10.16.3.8 def libcommonfunc.MatrixQ.rotAngleEliminate (*self, rotationType, i, j, m, n*)

Compute the rotation angle such that, after applying a rotation of type rotationType, the element $M[m,n]$ of the matrix self.M is equal to zero.

```
The element to set to zero [m,n] must be in the row or column of the rotation pivot [i,j] or its transpose
@param rotationType = Type of rotation to perform. The possible values are: {'trigonometric', 'hyperbolic'
@param i = Row index of the pivot that will be used in the rotation.
@param j = Column index of the pivot that will be used in the rotation.
@param m = Row index of the element that will become zero after rotation.
@param n = Column index of the element that will become zero after rotation.
@return rotAng = Rotation angle to remove the element \form#119.

First of all, and following the rules in the next table [TN 102.2 tab. A.1], we need to compute a value \f
on the pivot \form#122:
```

	Element to eliminate	i	j	k	val
1	M_{mn}	m	$\neq m,n$	n	$\frac{M_{ik}}{M_{jk}}$
2	M_{mn}	$\neq m,n$	m	n	$-\frac{M_{jk}}{M_{ik}}$
3	M_{mn}	n	$\neq m,n$	m	$\frac{M_{ki}}{M_{kj}}$
4	M_{mn}	$\neq m,n$	n	m	$-\frac{M_{kj}}{M_{ki}}$
5	M_{mm}	m	$\neq m$	-	$\frac{-M_{ij}+\sqrt{M_{ij}^2-M_{ii}M_{jj}}}{M_{jj}}$
6	M_{mm}	$\neq m$	m	-	$\frac{M_{ij}+\sqrt{M_{ij}^2-M_{ii}M_{jj}}}{M_{ii}}$
7	M_{mn}	m	n	-	$\frac{2M_{ij}}{M_{jj}-M_{ii}}$
8	M_{mn}	n	m	-	$\frac{2M_{ij}}{M_{jj}-M_{ii}}$

We define $f_l = 1$ for the cases {1,2,3,4,5,6} and $f_l = 0.5$ for cases {7,8}. Then, the rotation angle which sets the element $[m,n]$ to zero after applying the rotation is:

$$\begin{aligned} rotAng &= f_l tan^{-1}(val), && \text{if rotationType = 'trigonometric'} \\ rotAng &= -jf_l tan^{-1}(val), && \text{if rotationType = 'hyperbolic'} \end{aligned}$$

Definition at line 2156 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.M.

10.16.3.9 def libcommonfunc.MatrixQ.rotateMatrix (*self, rotationType, i, j, rotAng, flagQ =* `True`)

Rotate the coupling matrix self.M.

The resulting self.M matrix maintains the eigenvalues of the original matrix and his reflection and transmission properties.

Parameters

rotationType	= Type of rotation to perform. The possible values are: {'trigonometric', 'hyperbolic'}.
i	= Row index of the pivot that will be used in the rotation.
j	= Column index of the pivot that will be used in the rotation.
rotAng	= Rotation angle (rad).
flagQ	= Flag that determines if the quality factor attribute self.Q is updated or not (True / False).

[TN 102.2 sec. 9.1] By definition, a rotation matrix is a matrix where every value in the diagonal is 1 except the two values $R_{ii} = R_{jj} = c_r$ and the rest of the values of the matrix are zero except the two values $R_{ji} = -R_{ij} = s_r$, where [TN 102.2 eq.(A.2) and (A.3)]:

$$\begin{aligned} [i,j] &= \text{rotation pivot} \\ c_r &= cos(rotAng), && \text{if rotationType = 'trigonometric'} \\ & \quad cosh(rotAng), && \text{if rotationType = 'hyperbolic'} \\ s_r &= sin(rotAng), && \text{if rotationType = 'trigonometric'} \\ & \quad jsinh(rotAng), && \text{if rotationType = 'hyperbolic'} \end{aligned}$$

```
To apply the rotation we just need to do the product [TN 102.2 eq.(A.1)]:
```

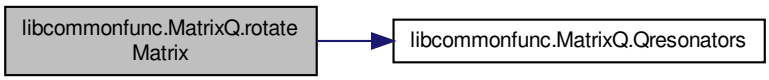
$$M_{rot} = RMR^T$$

where R is the rotation matrix described above.

Definition at line 1930 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, libcommonfunc.MatrixQ.-flagRotateAllowed, libcommonfunc.MatrixQ.M, and libcommonfunc.MatrixQ.Qresonators().

Here is the call graph for this function:



10.16.3.10 def libcommonfunc.MatrixQ.saveMat (*self, fileName, P, SP, precCM, precEP*)

Save this MatrixQ instance to an ascii file, including energy and power.

Uses self.__str__() to generate the ascii representation of the numpy matrix self.M.

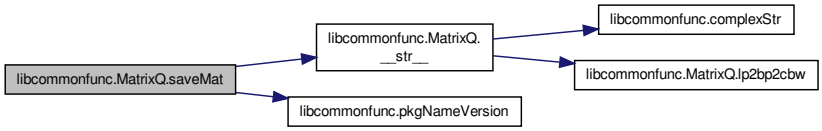
Parameters

fileName	= Name of file, including extension (string).
P	= Parameter class instance.
SP	= SParameters class instance.
precCM	= Number of significant digits to save in coupling matrix and Q (int).
precEP	= Number of significant digits to save in energy and power (int).

Definition at line 2675 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.__str__(), libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, libcommonfunc.MatrixQ.flagRotateAllowed, libcommonfunc.MatrixQ.name, dbplot.CurveFamily.name, libcommonfunc.MatrixQ.nodeScalingFactor, libcommonfunc.pkgNameVersion(), and libcommonfunc.MatrixQ.Q.

Here is the call graph for this function:



10.16.3.11 def libcommonfunc.MatrixQ.scaleNode (*self, i, b, flagQ* = `True`)

Scale a node of the coupling matrix self.M, by multiplying row [i] and column [i] by a factor.

Matrix rotations are not allowed after node scaling of a resonant node.

Parameters

i	= Index of row and column which we want to scale.
b	= Factor to use to scale the ith row and column.
flagQ	= Flag that determines if the quality factor attribute self.Q is updated or not (True / False).

Definition at line 2100 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, libcommonfunc.MatrixQ.-flagRotateAllowed, libcommonfunc.MatrixQ.M, libcommonfunc.MatrixQ.nodeScalingFactor, and libcommonfunc.-MatrixQ.Qresonators().

Here is the call graph for this function:

10.16.3.12 def libcommonfunc.MatrixQ.setFileExt (*self, fileExt*)

Function to set file extension field of the instance of the MatrixQ class.

Parameters

fileExt	= Extension of file name, if this matrix is to be saved in a file. Otherwise None.

Definition at line 1680 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.fileExt.

10.16.3.13 def libcommonfunc.MatrixQ.setName (*self, name*)

Function to set name field of the instance of the MatrixQ class.

Parameters

name	= Matrix name.

Definition at line 1671 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.name, and dbplot.CurveFamily.name.

10.16.3.14 def libcommonfunc.MatrixQ.setupOptimTopology (*self, topologyReal, topologyImag, topologySignReal, topologySignImag*)

Prepare optimization parameters depending on matrix topology and forced signs.

Parameters

topologyReal	= Bool numpy array having True values at the position of coupling matrix entries with non-zero real part.
topologyImag	= Bool numpy array having True values at the position of coupling matrix entries with non-zero imaginary part.
topologySign-Real	= Int numpy array having -1,1 or 0 values at the position of coupling matrix entries with negative, positive or any real part, respectively.
topologySign-Imag	= Int numpy array having -1,1 or 0 values at the position of coupling matrix entries with negative, positive or any imaginary part, respectively.

Returns

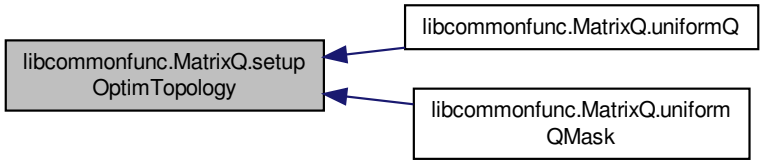
x0 = Initial guess. Double numpy vector with the values of the matrix entries which are different from zero. It is a concatenation of the non-zero real elements and the non-zero imaginary elements.
NonZeroReal = Bool numpy array having True values at the position of coupling matrix entries with non-zero real part in the upper-triangular part, including the diagonal, of the matrix.
NonZeroImag = Bool numpy array having True values at the position of coupling matrix entries with non-zero imaginary part in the upper-triangular part, including the diagonal, of the matrix. The entries corresponding to resonators are False
numReal = Number of entries in the upper triangular part of the matrix, including the diagonal, with non-zero real part.
bounds = List of tuples (Min,Max) with bounds for all variables to optimize. The maximum abs of any matrix element is set to 4 times the largest coupling, with a minimum of 2. This bound also affects the real part of diagonal elements (normalized resonant frequencies).

Definition at line 2245 of file libcommonfunc.py.

References libcommonfunc.MatrixQ.extraNodesL, libcommonfunc.MatrixQ.extraNodesS, and libcommonfunc.MatrixQ.M.

Referenced by libcommonfunc.MatrixQ.uniformQ(), and libcommonfunc.MatrixQ.uniformQMask().

Here is the caller graph for this function:



10.16.3.15 def libcommonfunc.MatrixQ.uniformQ (*self, P, CP, weights, Q, algor, maxIter, Nsamples, S11DR, S21DR, topologyReal, topologyImag, topologySignReal, topologySignImag*)

Optimize the coupling matrix to achieve a prescribed uniform Q and [S] parameters as close as possible to those computed from the characteristic polynomials.

The non-zero coupling matrix entries to optimize are specified by topologyReal and topologyImag parameters, which are set by the matrix topology widget in the GUI.

Parameters

P	= Parameters class instance, containing filter, synthesis and sweep parameters.
CP	= CharPolys class instance, containing Characteristic Polynomials Es, Ps, Fs and constants eps, epsR.
weights	= Optimization weights for the difference in S11 and S21 between the computation from the coupling matrix and from polynomials: List [weightS11, weightS22, weightS21, weightGD]
Q	= Prescribed uniform Q for the resonators.
algor	= Constrained optimization algorithm. String equal to 'L-BFG-S', 'TNC' or 'SLSQP'. See below.
maxIter	= Maximum number of iterations, or function evaluations, depending on algorithm.
Nsamples	= Number of samples to test. Half of them go to the pass band and the other half to the rejection band, between P.minFreq and P.maxFreq.
S11DR	= Dynamic range of S11 and S22 to test.
S21DR	= Dynamic range of S21 to test.

topologyReal	= Bool numpy array having True values at the position of coupling matrix entries with non-zero real part.
topologyImag	= Bool numpy array having True values at the position of coupling matrix entries with non-zero imaginary part.
topologySign-Real	= Int numpy array having -1,1 or 0 values at the position of coupling matrix entries with negative, positive or any real part, respectively.
topologySign-Imag	= Int numpy array having -1,1 or 0 values at the position of coupling matrix entries with negative, positive or any imaginary part, respectively.

10.16.4 Constrained optimization algorithms:

- http://en.wikipedia.org/wiki/Optimization
- ttp://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html

In optimization, quasi-Newton methods (also known as variable metric methods) are algorithms for finding local maxima and minima of functions. Quasi-Newton methods are based on Newton's method to find the stationary point of a function, where the gradient is 0. Newton's method assumes that the function can be locally approximated as a quadratic in the region around the optimum, and use the first and second derivatives (gradient and Hessian) to find the stationary point. In Quasi-Newton methods the Hessian matrix of second derivatives of the function to be minimized does not need to be computed. The Hessian is updated by analyzing successive gradient vectors instead (or approximate gradient evaluations computed using finite differences). Quasi-Newton methods are a generalization of the secant method to find the root of the first derivative for multidimensional problems. In multi-dimensions the secant equation is under-determined (has not a unique solution), and quasi-Newton methods differ in how they constrain the solution, typically by adding a simple low-rank update to the current estimate of the Hessian.

A necessary condition for optimality is that the gradient be zero. Quasi-Newton's method need not converge unless the function has a quadratic Taylor expansion near an optimum.

- http://en.wikipedia.org/wiki/Quasi-Newton_methods
- http://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

10.16.4.1 Limited memory variation Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS-B)

Limited memory variation of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) update to approximate the inverse Hessian matrix.

The BFGS method is one of the most popular quasi-Newton optimization methods. L-BFGS is a limited-memory version of BFGS that is particularly suited to problems with very large numbers of variables.

References:

- http://en.wikipedia.org/wiki/L-BFGS
- http://en.wikipedia.org/wiki/BFGS_method
- Wright, and Nocedal 'Numerical Optimization', 1999, pg. 198.
- R. H. Byrd, P. Lu and J. Nocedal. A Limited Memory Algorithm for Bound Constrained Optimization, (1995), SIAM Journal on Scientific and Statistical Computing , 16, 5, pp. 1190-1208.
- C. Zhu, R. H. Byrd and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization (1997), ACM Transactions on Mathematical Software, Vol 23, Num. 4, pp. 550 - 560.

10.16.4.2 Sequential Least SQuares Programming (SLSQP)

SLSQP optimizer is a sequential least squares programming algorithm which uses the Han–Powell quasi–Newton method with a BFGS update of the B–matrix and an L1–test function in the step–length algorithm. The optimizer uses a slightly modified version of Lawson and Hanson's NNLS nonlinear least-squares solver.

Sequential quadratic programming (SQP) is one of the most popular and robust algorithms for nonlinear continuous optimization. The method is based on solving a series of subproblems designed to minimize a quadratic model of the objective subject to a linearization of the constraints. If the problem is unconstrained, then the method reduces to Newton's method for finding a point where the gradient of the objective vanishes. If the problem has only equality constraints, then the method is equivalent to applying Newton's method to the first-order optimality conditions, or Karush–Kuhn–Tucker conditions, of the problem. A number of packages (including NPSOL, NLPQL, OPSYC, OPTIMA, MATLAB, and SQP) are founded on this approach.

References:

- http://en.wikipedia.org/wiki/Sequential_quadratic_programming
- Kraft D (1988) A software package for sequential quadratic programming. Tech. Rep. DFVLR-FB 88-28, DLR German Aerospace Center — Institute for Flight Mechanics, Koln, Germany.

10.16.4.3 Bound-constrained truncated newton algorithm (TNC)

Bound-constrained truncated newton algorithm using gradient information.

Truncated-Newton methods are a family of methods suitable for solving large nonlinear optimization problems. At each iteration, the current estimate of the solution is updated (i.e., a step is computed) by approximately solving the Newton equations using an iterative algorithm. This results in a doubly iterative method: an outer iteration for the nonlinear optimization problem, and an inner iteration for the Newton equations. The inner iteration is typically stopped or truncated before the solution to the Newton equations is obtained. Over the past two decades, a solid convergence theory has been derived for the methods. In addition, many algorithmic enhancements have been developed and studied, resulting in a number of publicly-available software packages. The result has been a collection of powerful, flexible, and adaptable tools for large-scale nonlinear optimization.

References:

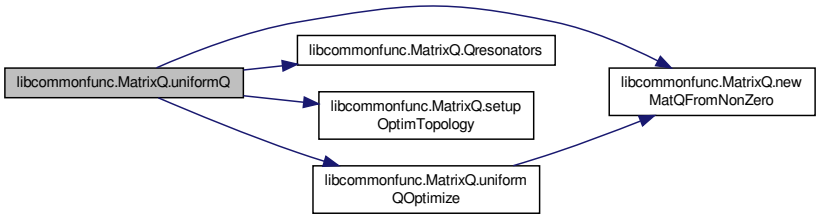
- http://js2007.free.fr/code/index.html#TNC
- http://iris.gmu.edu/~snash/nash/assets/TN_Survey/tn_survey.html
- Stephen G. Nash, "A Survey of Truncated-Newton Methods", Journal of Computational and Applied Mathematics, 124 (2000), pp. 45-59. http://iris.gmu.edu/~snash/papers/nash_3_99.ps

```
Return a SParameters class instance with sampling frequencies to optimize.
S11, S22, S21 and groupDelay attributes are initialized with those computed from the polynomials.

@param P  = Parameters class instance, containing filter, synthesis and sweep parameters.
@param CP = CharPolys class instance, containing Characteristic Polynomials Es, Ps, Fs and constants eps,
@param FT = FrequencyTransform class instance, containing f0 and BW parameters.
@param minFreq = Minimum frequency to optimize.
@param maxFreq = Maximum frequency to optimize.
@param Nsamples = Number of samples to test. Half of them go to the pass band and the other half to the r

@return SPref = SParameters class instance with sampling frequencies to optimize and [S] parameters compu
```

Definition at line 2432 of file libcommonfunc.py.

References libcommonfunc.Sparameters.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.MatrixQ.M, libcommonfunc.MatrixQ.newMatQFromNonZero(), libcommonfunc.MatrixQ.Qresonators(), libcommonfunc.MatrixQ.setupOptimTopology(), and libcommonfunc.MatrixQ.uniformQOptimize().

Here is the call graph for this function:



10.16.4.4 def libcommonfunc.MatrixQ.uniformQMask (*self, P, CP, weights, Q, algor, maxIter, Nsamples, SM, topologyReal, topologyImag, topologySignReal, topologySignImag*)

Optimize the coupling matrix to achieve a prescribed uniform Q and [S] parameters that comply with a specification mask.

The non-zero coupling matrix entries to optimize are specified by topologyReal and topologyImag parameters, which are set by the matrix topology widget in the GUI. Only the magnitude of [S] is optimized to comply with the S11 and S21 masks. Group delay in the mask file, if any, is not used in optimization,

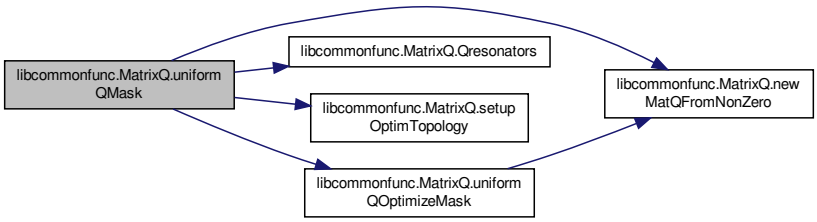
Parameters

P	= Parameters class instance, containing filter, synthesis and sweep parameters.
CP	= CharPolys class instance, containing Characteristic Polynomials Es, Ps, Fs and constants eps, epsR.
weights	= Optimization weights for the difference in S11 and S21 between the computation from the coupling matrix and from polynomials: List [weightS11, weightS22, weightS21, weightGD]. The last element, weightGD, is not used.
Q	= Prescribed uniform Q for the resonators.
algor	= Constrained optimization algorithm. String equal to 'L-BFG-S', 'TNC' or 'SLSQP'. See algorithms description MatrixQ.uniformQ method documentation.
maxIter	= Maximum number of iterations, or function evaluations, depending on algorithm.
Nsamples	= Number of samples to test. Half of them go to the pass band and the other half to the rejection band, between P.minFreq and P.maxFreq.
SM	= SpecMask class instance, with the mask to compare with.
topologyReal	= Bool numpy array having True values at the position of coupling matrix entries with non-zero real part.
topologyImag	= Bool numpy array having True values at the position of coupling matrix entries with non-zero imaginary part.
topologySignReal	= Int numpy array having -1,1 or 0 values at the position of coupling matrix entries with negative, positive or any real part, respectively.
topologySignImag	= Int numpy array having -1,1 or 0 values at the position of coupling matrix entries with negative, positive or any imaginary part, respectively.

Definition at line 2577 of file libcommonfunc.py.

References libcommonfunc.Sparameters.FT, libcommonfunc.MatrixQ.FT, libcommonfunc.MatrixQ.M, libcommonfunc.MatrixQ.newMatQFromNonZero(), libcommonfunc.MatrixQ.Qresonators(), libcommonfunc.MatrixQ.setupOptimTopology(), and libcommonfunc.MatrixQ.uniformQOptimizeMask().

Here is the call graph for this function:



10.16.4.5 def libcommonfunc.MatrixQ.uniformQOptimize (*self, NonZeroValues, SPref, S11min, S21min, NonZeroReal, NonZeroImag, numReal, weights, Q*)

Optimization objective function for uniformQ function.

The output matrix is forced symmetric, with positive imaginary part out of the diagonal and with uniform Q in all the resonators. The imaginary part of the diagonal elements is determined so that the Q is constant,so it is not a variable to optimize.

Parameters

NonZeroValues	= Double numpy vector with the values of the matrix entries which are different from zero. It is a concatenation of the non-zero real elements and the non-zero imaginary elements.
SPref	= SParameters class instance containing [S] computed from the characteristic polynomials. It is used as the reference to compare.
S11min	= Minimum value of S11 and S22 to test.
S21min	= Minimum value of S21 outside to test.
NonZeroReal	= Bool numpy array having True values at the position of coupling matrix entries with non-zero real part in the upper-triangular part, including the diagonal, of the matrix.
NonZeroImag	= Bool numpy array having True values at the position of coupling matrix entries with non-zero imaginary part in the upper-triangular part, including the diagonal, of the matrix. The entries corresponding to resonators are False, since the imaginary part will be obtained from the prescribed Q.
numReal	= Number of entries in the upper triangular part of the matrix, including the diagonal, with non-zero real part.
weights	= Optimization weights for the difference in S11(dB), S21(dB) and GD(inband) between the computation from the coupling matrix and from polynomials: List [weightS11, weightS22, weightS21, weightGD]
Q	= Prescribed uniform Q for the resonators.

Returns

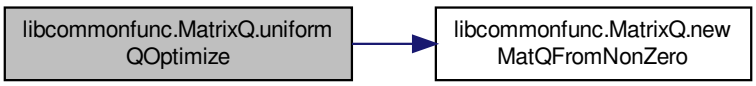
goal = Objective function value to minimize.

Definition at line 2513 of file libcommonfunc.py.

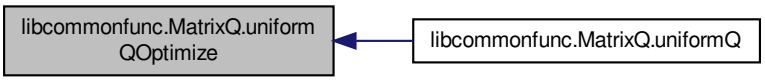
References libcommonfunc.MatrixQ.newMatQFromNonZero().

Referenced by libcommonfunc.MatrixQ.uniformQ().

Here is the call graph for this function:



Here is the caller graph for this function:



10.16.4.6 def libcommonfunc.MatrixQ.uniformQOptimizeMask (*self, NonZeroValues, SPref, SM, NonZeroReal, NonZeroImag, numReal, weights, Q*)

Optimization objective function for uniformQMask function.

The output matrix is forced symmetric, with positive imaginary part out of the diagonal and with uniform Q in all the resonators. The imaginary part of the diagonal elements is determined so that the Q is constant,so it is not a variable to optimize. Only the magnitude of [S] is optimized to comply with the S11 and S21 masks. Group delay in the mask file, if any, is not used in optimization,

Parameters

NonZeroValues	= Double numpy vector with the values of the matrix entries which are different from zero. It is a concatenation of the non-zero real elements and the non-zero imaginary elements.
SPref	= SParameters class instance containing method and storage to compute [S] from CM.
SM	= SpecMask class instance, with the mask to compare with.
NonZeroReal	= Bool numpy array having True values at the position of coupling matrix entries with non-zero real part in the upper-triangular part, including the diagonal, of the matrix.
NonZeroImag	= Bool numpy array having True values at the position of coupling matrix entries with non-zero imaginary part in the upper-triangular part, including the diagonal, of the matrix. The entries corresponding to resonators are False, since the imaginary part will be obtained from the prescribed Q.
numReal	= Number of entries in the upper triangular part of the matrix, including the diagonal, with non-zero real part.
weights	= Optimization weights for the difference in S11(dB), S21(dB) and GD(inband) between the computation from the coupling matrix and from polynomials: List [weightS11, weightS22, weightS21, weightGD]. The last element, weightGD, is not used.

Q	= Prescribed uniform Q for the resonators.

Returns

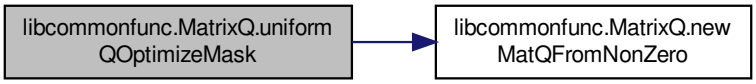
goal = Objective function value to minimize.

Definition at line 2624 of file libcommonfunc.py.

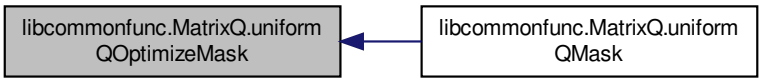
References libcommonfunc.MatrixQ.newMatQFromNonZero().

Referenced by libcommonfunc.MatrixQ.uniformQMask().

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- libcommonfunc.py

10.17 dbplot.MyPicker Class Reference

Class derived from Qwt.QwtPlotPicker to allow reimplementation of Qwt.QwtPlotPicker.trackerText()

Public Member Functions

- def __init__
  Derived class constructor.
- def trackerText
  Reimplementation of tracker text function, to customize tracker appereance.

10.17.1 Detailed Description

Class derived from Qwt.QwtPlotPicker to allow reimplementation of Qwt.QwtPlotPicker.trackerText()

Definition at line 2175 of file dbplot.py.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 def dbplot.MyPicker.__init__ (*self, dbplot, nTab, args*)

Derived class constructor.

Saves dbplot and nTab as class attributes.

Parameters

dbplot	= DbPlot class instance containing the Tab where this QwtPlotPicker exists.
nTab	= Tab widget index containing this QwtPlotPicker.

Definition at line 2183 of file dbplot.py.

References dbplot.MyPicker.dbplot, and dbplot.MyPicker.nTab.

The documentation for this class was generated from the following file:

- dbplot.py

10.18 mwfiltersgui.myTableWidget Class Reference

Subclass for:

Public Member Functions

- def __init__

  Table name.
- def setUnits

  Store units in self.untis, to use in TZ table headers.
- def setf0

  Set the value in Center Frequency widget as self.f0.
- def updateTableSize

  Resize rows and columns to fit contents.
- def tableCellChanged

  Table cell has been modified by user, take necessary actions.
- def addRow

  Append new row for a new zero.

10.18.1 Detailed Description

Subclass for:

- reimplementation of SinzeHint and MinimumSizeHint
- Methods for tables of transmission zeros

Definition at line 813 of file mwfiltersgui.py.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 def mwfiltersgui.myTableWidget.__init__ (*self, name =* None*, mainDlg =* None*, parent =* None)

Table name.

Main dialog to which this table belongs, used only where applicable. Frequency units, used only where applicable. Filter center frequency. It will be used for computation of 'f-f0' column and for autoupdate of 'Imag' and 'f-f0' colums when one of them is edited. Flag to inform that changes in table cells are due to user interaction

Definition at line 815 of file mwfiltersgui.py.

References mwfiltersgui.SpecMask.f0, mwfiltersgui.myTableWidget.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, mwfiltersgui.myTableWidget.mainDlg, mwfiltersgui.myTableWidget.name, libcommonfunc.MatrixQ.name, dbplot.CurveFamily.name, mwfiltersgui.myTableWidget.units, and mwfiltersgui.myTableWidget.updatedByUser.

10.18.3 Member Function Documentation

10.18.3.1 def mwfiltersgui.myTableWidget.setf0 (*self, updateImag =* True)

Set the value in Center Frequency widget as self.f0.

It will be used for computation of 'f-f0' column and for autoupdate of 'Imag' and 'f-f0' colums when one of them is edited.

Parameters

updateImag,:	If updateImag is True, then the 'Imag' column of the whole table is updated according to f0 and to the f-f0 column. If updateImag is False, f0 is only stored in self.f0 and the 'Imag' column is not updated.

Definition at line 855 of file mwfiltersgui.py.

References mwfiltersgui.SpecMask.f0, mwfiltersgui.myTableWidget.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, and mwfiltersgui.SetCouplingDlg.item.

10.18.3.2 def mwfiltersgui.myTableWidget.setUnits (*self, units*)

Store units in self.untis, to use in TZ table headers.

Parameters

units	= Units string.

Definition at line 844 of file mwfiltersgui.py.

References mwfiltersgui.myTableWidget.units.

10.18.3.3 def mwfiltersgui.myTableWidget.tableCellChanged (*self, row, col*)

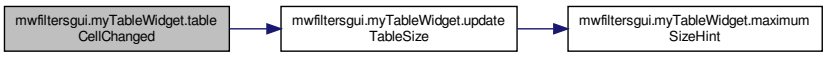
Table cell has been modified by user, take necessary actions.

If Imag or Imag-f0 columns have been modified, updated the companion cell

Definition at line 892 of file mwfiltersgui.py.

References mwfiltersgui.SpecMask.f0, mwfiltersgui.myTableWidget.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, mwfiltersgui.myTableWidget.updatedByUser, and mwfiltersgui.myTableWidget.updateTableSize().

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.19 mwfiltersgui.ParamEditDlg Class Reference

GUI dialog to edit synthesis parameters.

Public Member Functions

- def __init__

  Constructor: Creates dialog window containing values of parameters.
- def updateDialogParams

  Updates paramEditDlg widgets with contents of local copy of P (self.P).
- def updatewFuncTableEntries

  Update predistorsion weights table Widget with list.
- def readwFuncTableEntries

  Read list from predistorsion weights table Widget entries.
- def readTZsTableEntries

  Read list from TZs table Widget entries.
- def updateTZsTable

  Updates a TZs tableWidget with contents of list, and makes the number of rows equal to the number of zeros.
- def frequencyUnits

  Returns frequency expressed in GHz, MHz, 'kHz' or Hz.
- def st2floatInf

  Convert string to float.
- def float2stInf

  Convert float to string.
- def readDialogParams

  Reads parameter values from dialog and store in self.P (dialog working copy of parameters).

Functions automatically excuted when the user interacts with the GUI

- def updatePredisZerosArrang

  Update the size of the predistortion zeros arrangement list after the user changes the filter order.
- def updatewFuncTableSize

  Make the size of the adaptive predistorsion weights table equal to the order of the filter.
- def updateTotaltransmissionZeros

  Count the number of Generalized Chebyshev transmission zeros set in the GUI widgets and update the lcdNumber widget.
- def updateDlgHeight

  Update dialog size to fit table.
- def dialogChanged

Custom slot in Qt designer conected to widgets "changed" signal.

- def on_outDir_pushButton_clicked

  Opens getExistingDirectory Qt dialog in order to set P.outDir variable.
- def on_TZoptim_pushButton_clicked

  Run imaginary TZ and bandwidth optimization to comply specification mask.
- def on_compute_pushButton_clicked

  Run synthesis.
- def on_TZS_newZero_pushButton_clicked

  Add new zero (blank) to transmission zeros table.
- def on_LP_newZero_pushButton_clicked

  Add new zero (blank) to LP complex zeros table.
- def on_TZs_symm_pushButton_clicked

  Add new zero with symmetrical imaginary part (in normalized frequency) to transmission zeros table.
- def on_LP_symm_pushButton_clicked

  Add new zero with symmetrical imaginary part (in normalized frequency) to transmission zeros table.
- def closeEvent

  Save current dialog parameters to mainWindow.P and close dialog.
- def on_discard_pushButton_clicked

  Reject current dialog parameters and close dialog.
- def on_f0_comboBox_currentIndexChanged

  Update lineEdit value when the units ComboBox is changed.
- def on_BW_comboBox_currentIndexChanged

  Update lineEdit value when the units ComboBox is changed.
- def on_qeZero_comboBox_currentIndexChanged

  Update lineEdit value when the units ComboBox is changed.
- def on_transmissionZeros_comboBox_currentIndexChanged

  Update lineEdit value when the units ComboBox is changed.
- def on_minFreq_comboBox_currentIndexChanged

  Update lineEdit value when the units ComboBox is changed.
- def on_maxFreq_comboBox_currentIndexChanged

  Update lineEdit value when the units ComboBox is changed.

10.19.1 Detailed Description

GUI dialog to edit synthesis parameters.

Definition at line 4155 of file mwfiltersgui.py.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 def mwfiltersgui.ParamEditDlg.__init__ (*self*, *parent* = `None`)

Constructor: Creates dialog window containing values of parameters.

```
Modifies MainWindow.P , using a shallow copy for easier access.
Saves a backup copy of the mainWindow Parameters and synthesis results, that will be restored if the u
@param parent = Parent widget, in this case is mainWindow.
```

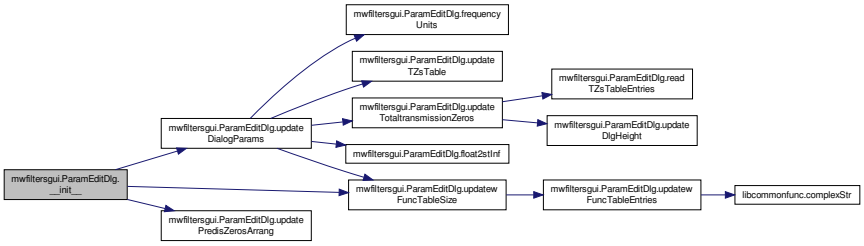
Index of the Center Frequency Units comboBox. It is used to check if the user has changed the Frequency Units in the GUI and to update the Frequency value accordingly. Index of the Bandwidth Units comboBox. It is used to check if the user has changed the Bandwidth Units in the GUI and to update the Bandwidth value accordingly. Index of the Quasieliptic zero frequency units comboBox. It is used to check if the user has changed the Frequency Units in the GUI and to update the Frequency value accordingly. Index of the Generalized Chebyshev zeros frequency units comboBox. It is used to check if the user has changed the Frequency Units in the GUI and to update the Frequency value accordingly. Index of the Minimum Frequency Units comboBox. It is used to check if the user has changed the Frequency Units in the GUI and to update the Frequency value accordingly. Index of the Maximum Frequency Units comboBox. It is used to check if the user has changed the Frequency Units in the GUI and to

update the Frequency value accordingly. Variable that becomes true when the user clicks the 'Discard' button to reject the dialog. Its is necessary to use this variable, since now the dialog is a QMainWindow instead of a QDialog, and QMainWindow does not have a reject() slot. Variable that is False when it is necessary to recompute synthesis because Parameters have been modified. Variable to store a list of widgets with bad input. Only when this list is empty, he "Accept" and "Compute" buttons are enabled. Variable to store sweep value in order to check if it has been changed by the user, and then update frequency axis. Tuple (fmin, fmax) Copy of the mainWindow class instance. It is necessary to store a copy since it must be accessed by some member functions. Shallow copy just for easier access. Backup copy of the mainWindow Parameters and synthesis results, that will be restored if the user Discards the dialog. Initial dialog size as set in Qt designer Variable to store the modified/unmodifed status of the dialog working copy of parameters self.P. The dialogChanged() method is automatically called when the user modifies the edit widgets contents and sets this variable to True. This variable must be set to False at the end of the **init** dialog constructor, since dialogChanged() is called when the constructor sets the edit widgets contents.

Definition at line 4164 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.badInput, mwfiltersgui.ParamEditDlg.badInput, mwfiltersgui.ParamEditDlg.BWUnitsIndex, mwfiltersgui.MatrixEditDlg.dialogModified, mwfiltersgui.ParamEditDlg.dialogModified, mwfiltersgui.ParamEditDlg.dlgInitSize, mwfiltersgui.ParamEditDlg.f0UnitsIndex, mwfiltersgui.ParamEditDlg.genChebyTZs_horizontalLayout, mwfiltersgui.ParamEditDlg.genChebyTZs_tableWidget, mwfiltersgui.ParamEditDlg.genChebyTZs_tooltip, mwfiltersgui.ParamEditDlg.LPcomplexZeros_horizontalLayout, mwfiltersgui.ParamEditDlg.LPcomplexZeros_tableWidget, mwfiltersgui.ParamEditDlg.LPcomplexZeros_tooltip, dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.SetCouplingDlg.mainWindow, mwfiltersgui.MatrixEditDlg.mainWindow, mwfiltersgui.PredisZeorsDlg.mainWindow, mwfiltersgui.SensitivityAnalysis.mainWindow, mwfiltersgui.CouplingMatrixDlg.mainWindow, mwfiltersgui.ParamEditDlg.mainWindow, mwfiltersgui.ParamEditDlg.maxFreqUnitsIndex, mwfiltersgui.ParamEditDlg.minFreqUnitsIndex, mwfiltersgui.MainWindow.P, mwfiltersgui.ParamEditDlg.P, mwfiltersgui.ParamEditDlg.qeZeroUnitsIndex, mwfiltersgui.ParamEditDlg.recomputed, mwfiltersgui.ParamEditDlg.rejected, mwfiltersgui.ParamEditDlg.resultBkp, mwfiltersgui.ParamEditDlg.transmissionZerosUnitsIndex, mwfiltersgui.ParamEditDlg.updateDialogParams(), mwfiltersgui.ParamEditDlg.updatePredisZerosArrang(), mwfiltersgui.ParamEditDlg.updatewFuncTableSize(), mwfiltersgui.ParamEditDlg.wFunc_horizontalLayout, mwfiltersgui.ParamEditDlg.wFunc_tableWidget, and mwfiltersgui.ParamEditDlg.wFunc_tooltip.

Here is the call graph for this function:



10.19.3 Member Function Documentation

10.19.3.1 def mwfiltersgui.ParamEditDlg.closeEvent (*self, event*)

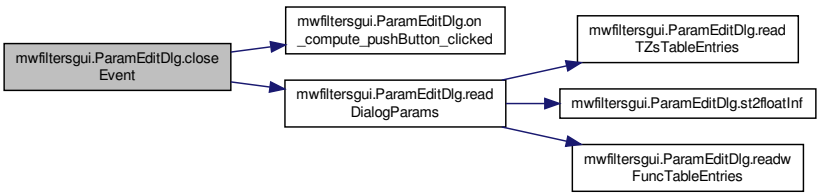
Save current dialog parameters to mainWindow.P and close dialog.

This function is automatically executed by the GUI when the user clickes the 'Accept' button or closes the dialog. Closing the dialog by killing the window is equivalent to clicking the 'Accept' button.

Definition at line 5280 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.dialogModified, mwfiltersgui.ParamEditDlg.dialogModified, mwfiltersgui.ParamEditDlg.on_compute_pushButton_clicked(), mwfiltersgui.ParamEditDlg.readDialogParams(), mwfiltersgui.ParamEditDlg.recomputed, and mwfiltersgui.ParamEditDlg.rejected.

Here is the call graph for this function:



10.19.3.2 def mwfiltersgui.ParamEditDlg.dialogChanged (*self*)

Custom slot in Qt designer conected to widgets "changed" signal.

This function is automatically executed by the GUI every time the user changes dialog values, and sets the self.-dialogModified variable to True and self.recomputed to False.

Definition at line 5056 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.badInput, mwfiltersgui.ParamEditDlg.badInput, mwfiltersgui.MatrixEditDlg.-dialogModified, mwfiltersgui.ParamEditDlg.dialogModified, and mwfiltersgui.ParamEditDlg.recomputed.

10.19.3.3 def mwfiltersgui.ParamEditDlg.float2stInf (*self, value*)

Convert float to string.

numpy.inf is converted to 'Inf'.

Parameters

value	= Float value to convert.

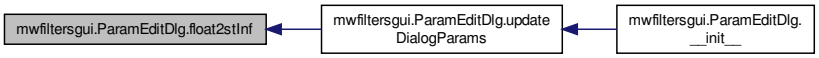
Returns

st = String.

Definition at line 4757 of file mwfiltersgui.py.

Referenced by mwfiltersgui.ParamEditDlg.updateDialogParams().

Here is the caller graph for this function:



10.19.3.4 def mwfiltersgui.ParamEditDlg.frequencyUnits (*self, freq*)

Returns frequency expressed in GHz, MHz, 'kHz' or Hz.

Parameters

freq	= Frequency in Hz. Can be a float or a list of floats.

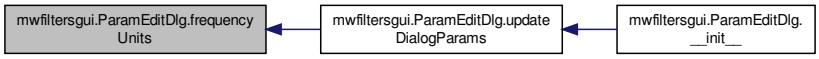
Returns

(newFreq, units).
newFreq = Frequency expressed in units.
units = 'GHz', 'MHz', 'kHz' or 'Hz'.

Definition at line 4714 of file mwfiltersgui.py.

Referenced by mwfiltersgui.ParamEditDlg.updateDialogParams().

Here is the caller graph for this function:



10.19.3.5 def mwfiltersgui.ParamEditDlg.on_BW_comboBox_currentIndexChanged (*self, value*)

Update lineEdit value when the units ComboBox is changed.

This function is automatically executed by the GUI when the user changes the filter bandwidht units ComboBox.

Definition at line 5356 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.BWUnitsIndex.

10.19.3.6 def mwfiltersgui.ParamEditDlg.on_compute_pushButton_clicked (*self, checked*)

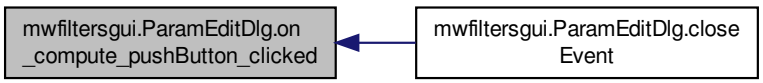
Run synthesis.

If parameters are valid, self.mainWindow.on_action_Execute_triggered() will set self.recomputed to True.

Definition at line 5176 of file mwfiltersgui.py.

Referenced by mwfiltersgui.ParamEditDlg.closeEvent().

Here is the caller graph for this function:



10.19.3.7 def mwfiltersgui.ParamEditDlg.on_discard_pushButton_clicked (*self, checked*)

Reject current dialog parameters and close dialog.

A backup copy of parameters and synthesis results is recovered. This function is automatically executed by the GUI when the user clickes the 'Discard' button.

Definition at line 5315 of file mwfiltersgui.py.

References mwfiltersgui.MatrixEditDlg.dialogModified, mwfiltersgui.ParamEditDlg.dialogModified, mwfiltersgui.ParamEditDlg.recomputed, mwfiltersgui.ParamEditDlg.rejected, and mwfiltersgui.ParamEditDlg.resultBkp.

10.19.3.8 def mwfiltersgui.ParamEditDlg.on_f0_comboBox_currentIndexChanged (*self, value*)

Update lineEdit value when the units ComboBox is changed.

This function is automatically executed by the GUI when the user changes the filter frequency units ComboBox.

Definition at line 5341 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.f0UnitsIndex.

10.19.3.9 def mwfiltersgui.ParamEditDlg.on_maxFreq_comboBox_currentIndexChanged (*self, value*)

Update lineEdit value when the units ComboBox is changed.

This function is automatically executed by the GUI when the user changes the maximum sweep frequency units ComboBox.

Definition at line 5417 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.maxFreqUnitsIndex.

10.19.3.10 def mwfiltersgui.ParamEditDlg.on_minFreq_comboBox_currentIndexChanged (*self, value*)

Update lineEdit value when the units ComboBox is changed.

This function is automatically executed by the GUI when the user changes the minimum sweep frequency units ComboBox.

Definition at line 5402 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.minFreqUnitsIndex.

10.19.3.11 def mwfiltersgui.ParamEditDlg.on_qeZero_comboBox_currentIndexChanged (*self, value*)

Update lineEdit value when the units ComboBox is changed.

This function is automatically executed by the GUI when the user changes the Quasieliptic zeros frequency units ComboBox.

Definition at line 5371 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.qeZeroUnitsIndex.

10.19.3.12 def mwfiltersgui.ParamEditDlg.on_transmissionZeros_comboBox_currentIndexChanged (*self, value*)

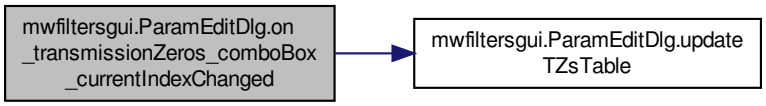
Update lineEdit value when the units ComboBox is changed.

This function is automatically executed by the GUI when the user changes the Generalized Chebyshev zeros frequency units ComboBox.

Definition at line 5386 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.genChebyTZs_tableWidget, mwfiltersgui.ParamEditDlg.LPcomplexZeros_tableWidget, mwfiltersgui.ParamEditDlg.transmissionZerosUnitsIndex, and mwfiltersgui.ParamEditDlg.updateTZsTable().

Here is the call graph for this function:



10.19.3.13 def mwfiltersgui.ParamEditDlg.readDialogParams (*self*)

Reads parameter values from dialog and store in self.P (dialog working copy of parameters).

```
Error control only for invalid parameter values that would crash the parser or the GUI.
Empty lineEdit boxes are allowed in parameters that will not be used in the computation.
More error control in lossyfilters.Parameters.validate() .
    @return ok (True / False) = If True, parameters have been read correctly, if False, there has been an
```

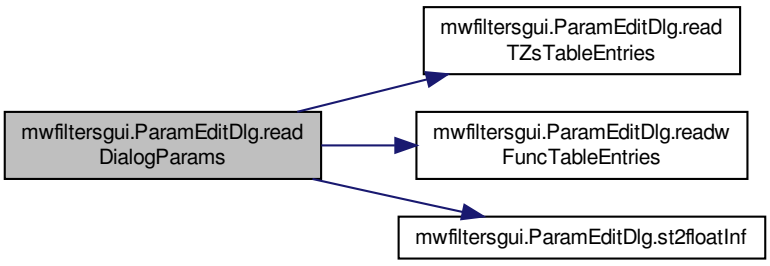
key is used to store parameter name for error catching

Definition at line 4769 of file mwfiltersgui.py.

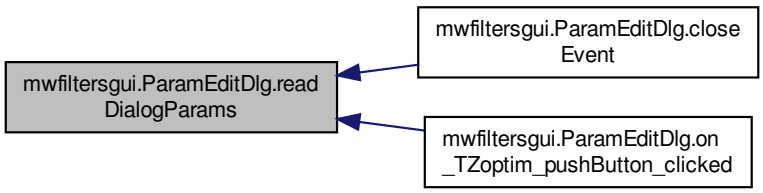
References mwfiltersgui.ParamEditDlg.BWUnitsIndex, mwfiltersgui.ParamEditDlg.f0UnitsIndex, mwfiltersgui.ParamEditDlg.genChebyTZs_tableWidget, mwfiltersgui.ParamEditDlg.LPcomplexZeros_tableWidget, mwfiltersgui.ParamEditDlg.maxFreqUnitsIndex, mwfiltersgui.ParamEditDlg.minFreqUnitsIndex, mwfiltersgui.ParamEditDlg.oldSweep, mwfiltersgui.MainWindow.P, mwfiltersgui.ParamEditDlg.P, mwfiltersgui.ParamEditDlg.qeZeroUnitsIndex, mwfiltersgui.ParamEditDlg.readTZsTableEntries(), mwfiltersgui.ParamEditDlg.readwFuncTableEntries(), and mwfiltersgui.ParamEditDlg.st2floatInf().

Referenced by mwfiltersgui.ParamEditDlg.closeEvent(), and mwfiltersgui.ParamEditDlg.on_TZoptim_pushButton_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



10.19.3.14 def mwfiltersgui.ParamEditDlg.readTZsTableEntries (*self, table*)

Read list from TZs table Widget entries.

Parameters

table	= tableWidget to read

Returns

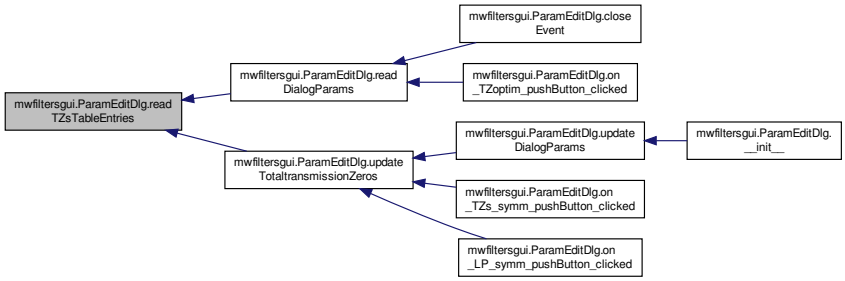
listImag = list containing purely imaginary zeros entries in tableWidget
listComplex = list containing complex zeros entries in tableWidget
listReal = list containing purely real zeros entries in tableWidget

Definition at line 4611 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.transmissionZerosUnitsIndex.

Referenced by mwfiltersgui.ParamEditDlg.readDialogParams(), and mwfiltersgui.ParamEditDlg.updateTotaltransmissionZeros().

Here is the caller graph for this function:



10.19.3.15 def mwfiltersgui.ParamEditDlg.readwFuncTableEntries (*self*)

Read list from predistorsion weights table Widget entries.

If row or column count is equal to 1, the 2D list is collapsed to a 1D list.

Returns

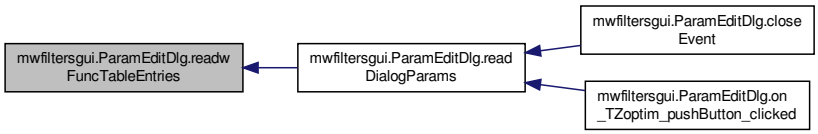
list = list containing entries in tableWidget

Definition at line 4585 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.wFunc_tableWidget.

Referenced by mwfiltersgui.ParamEditDlg.readDialogParams().

Here is the caller graph for this function:



10.19.3.16 def mwfiltersgui.ParamEditDlg.st2floatInf (*self, st*)

Convert string to float.

'INF', 'Inf', 'inf' are converted to numpy.inf.

Parameters

st	= String to convert.

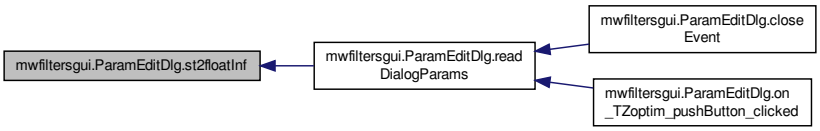
Returns

value = Numerical value of string (float).

Definition at line 4747 of file mwfiltersgui.py.

Referenced by mwfiltersgui.ParamEditDlg.readDialogParams().

Here is the caller graph for this function:



10.19.3.17 def mwfiltersgui.ParamEditDlg.updateDialogParams (*self, autoFreqUnits* = `True`)

Updates paramEditDlg widgets with contents of local copy of P (self.P).

Parameters

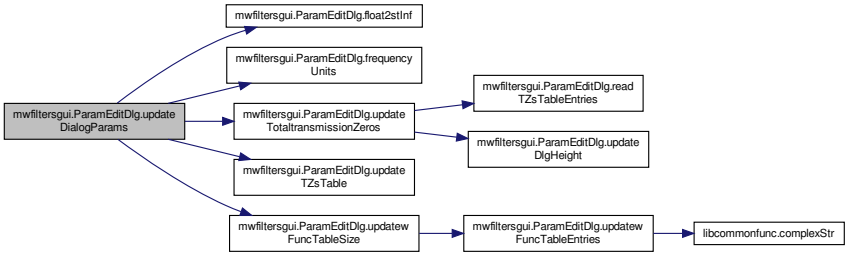
autoFreqUnits	= If True, automatically update frequency units in ComboBoxes. It is True by default.

Definition at line 4347 of file mwfiltersgui.py.

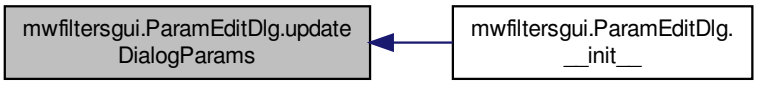
References mwfiltersgui.ParamEditDlg.BWUnitsIndex, mwfiltersgui.ParamEditDlg.f0UnitsIndex, mwfiltersgui.ParamEditDlg.float2stInf(), mwfiltersgui.ParamEditDlg.frequencyUnits(), mwfiltersgui.ParamEditDlg.genChebyTZs_tableWidget, mwfiltersgui.ParamEditDlg.LPcomplexZeros_tableWidget, mwfiltersgui.ParamEditDlg.maxFreqUnitsIndex, mwfiltersgui.ParamEditDlg.minFreqUnitsIndex, mwfiltersgui.ParamEditDlg.oldSweep, mwfiltersgui.MainWindow.P, mwfiltersgui.ParamEditDlg.P, mwfiltersgui.ParamEditDlg.qeZeroUnitsIndex, mwfiltersgui.ParamEditDlg.transmissionZerosUnitsIndex, mwfiltersgui.ParamEditDlg.updateTotaltransmissionZeros(), mwfiltersgui.ParamEditDlg.updateTZsTable(), and mwfiltersgui.ParamEditDlg.updatewFuncTableSize().

Referenced by mwfiltersgui.ParamEditDlg.__init__().

Here is the call graph for this function:



Here is the caller graph for this function:



10.19.3.18 def mwfiltersgui.ParamEditDlg.updatePredisZerosArrang (*self, N*)

Update the size of the predistortion zeros arrangement list after the user changes the filter order.

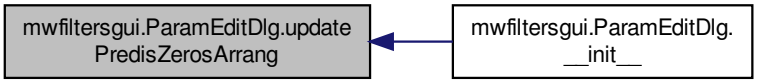
Parameters

N	= Filter order as in self.N_spinBox.value()

Definition at line 4968 of file mwfiltersgui.py.

Referenced by mwfiltersgui.ParamEditDlg.__init__().

Here is the caller graph for this function:



10.19.3.19 def mwfiltersgui.ParamEditDlg.updateTotaltransmissionZeros (*self*)

Count the number of Generalized Chebyshev transmission zeros set in the GUI widgets and update the lcdNumber widget.

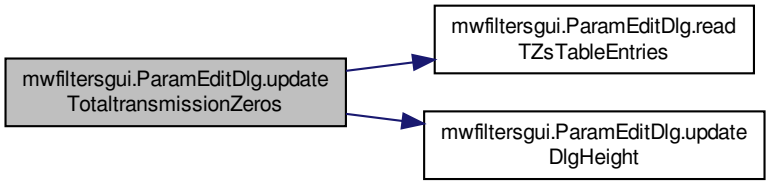
The dialog window is automatically resized to fit the rows corresponding to TZs. This function is automatically executed by the GUI every time the user changes one of the zeros lists in the GUI.

Definition at line 5012 of file mwfiltersgui.py.

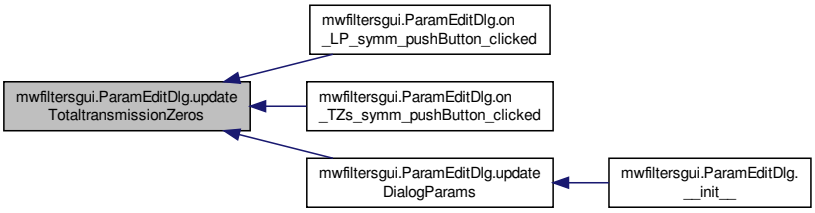
References mwfiltersgui.ParamEditDlg.genChebyTZs_tableWidget, mwfiltersgui.ParamEditDlg.LPcomplexZeros_-tableWidget, mwfiltersgui.ParamEditDlg.readTZsTableEntries(), and mwfiltersgui.ParamEditDlg.updateDlgHeight().

Referenced by mwfiltersgui.ParamEditDlg.on_LP_symm_pushButton_clicked(), mwfiltersgui.ParamEditDlg.on_T-Zs_symm_pushButton_clicked(), and mwfiltersgui.ParamEditDlg.updateDialogParams().

Here is the call graph for this function:



Here is the caller graph for this function:



10.19.3.20 def mwfiltersgui.ParamEditDlg.updateTZsTable (*self, table, list*)

Updates a TZs tableWidget with contents of list, and makes the number of rows equal to the number of zeros.

Adjusts widget size to table contents. This function is called by the self.updateDialogParams() function.

Parameters

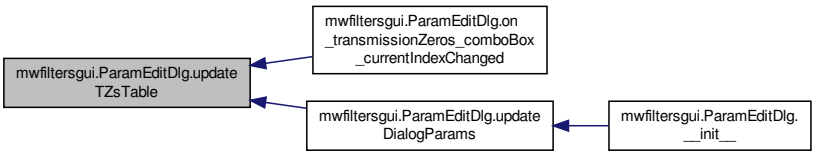
table	= tableWidget to update
list	= list containing entries to set in tableWidget

Definition at line 4673 of file mwfiltersgui.py.

References mwfiltersgui.ParamEditDlg.transmissionZerosUnitsIndex.

Referenced by mwfiltersgui.ParamEditDlg.on_transmissionZeros_comboBox_currentIndexChanged(), and mwfiltersgui.ParamEditDlg.updateDialogParams().

Here is the caller graph for this function:



10.19.3.21 def mwfiltersgui.ParamEditDlg.updatewFuncTableEntries (*self, list*)

Update predistorsion weights table Widget with list.

Parameters

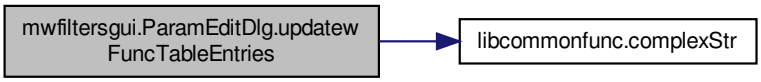
list	= list containing entries to set in tableWidget

Definition at line 4570 of file mwfiltersgui.py.

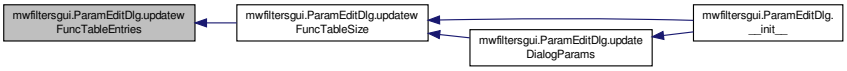
References libcommonfunc.complexStr().

Referenced by mwfiltersgui.ParamEditDlg.updatewFuncTableSize().

Here is the call graph for this function:



Here is the caller graph for this function:



10.19.3.22 def mwfiltersgui.ParamEditDlg.updatewFuncTableSize (*self, list, dims, updateEntries*)

Make the size of the adaptive predistorsion weights table equal to the order of the filter.

Calls self.updatewFuncTableEntries() function and adjusts widget size to table contents. This function is automatically executed by the GUI every time the user changes the filter order in the GUI or activates the adaptive predistiortion checkbox, but only if the synthesis method is 'Predistortion' and adaptive predistortion group box is checked. and is called by the self.updateDialogParams() function.

Parameters

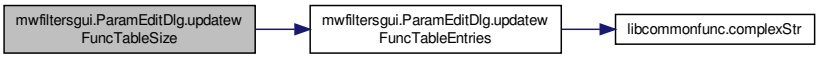
list	= list containing entries to set in tableWidget
dims	= tuple (rows, cols) contaning number of rows and columns
updateEntries	= If True, table entries are updated (calls self.updatewFuncTableEntries(list)). When this function is called by itemChanged signal, this flag must be False to avoid infinite recursion.

Definition at line 4988 of file mwfiltersgui.py.

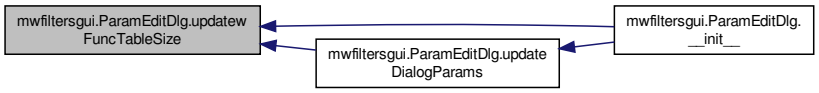
References mwfiltersgui.ParamEditDlg.updatewFuncTableEntries().

Referenced by mwfiltersgui.ParamEditDlg.__init__(), and mwfiltersgui.ParamEditDlg.updateDialogParams().

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.20 libcommonfunc.Parameters Class Reference

Filter, synthesis and sweep parameters.

Public Member Functions

- def readParam

  Parse int, float, complex or string parameters from string stValue.

- def __init__

  Initialize parameters class with list of strings.

- def readChebash

  Translate a list of Chebash parameters into a list of equivalent lossyfilters parameters.

10.20.1 Detailed Description

Filter, synthesis and sweep parameters.

Definition at line 457 of file libcommonfunc.py.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 def libcommonfunc.Parameters.__init__ (*self, stList*)

Initialize parameters class with list of strings.

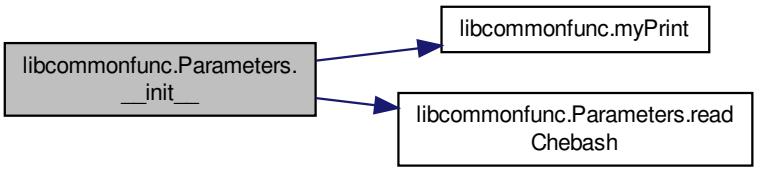
```
    Each string is a line of the parameter file.
    For backwards compatibility of parameter files, missing fields are set to None.
        @param stList = List of strings. Each element of list is a line of input file.
                                    @n If stList is equal to None, an empty instance of Parameters class is cr
                                    (useful to create a new parameter file and for the copy funcion).
        @return Parameter class instance.
```

Flag that indicates if the class instance contains information. empty==False or empty==True Path of output files, including directory and file name

Definition at line 505 of file libcommonfunc.py.

References libcommonfunc.Parameters.empty, libcommonfunc.Parameters.LPalgor, libcommonfunc.Parameters.LPcomplexZeros, libcommonfunc.Parameters.LPmaxIter, libcommonfunc.Parameters.LPNsamples, libcommonfunc.Parameters.LPrealZeros, libcommonfunc.myPrint(), libcommonfunc.Parameters.N, libcommonfunc.Parameters.nuqK11c1, libcommonfunc.Parameters.nuqK21c1, libcommonfunc.Parameters.nuqK21c2, libcommonfunc.Parameters.nuqK21c3, libcommonfunc.Parameters.nuqK22c1, libcommonfunc.Parameters.outDirName, libcommonfunc.Parameters.readChebash(), libcommonfunc.Parameters.transComplexZeros, libcommonfunc.Parameters.transImagZeros, libcommonfunc.Parameters.wFunc, and libcommonfunc.Parameters.zerosArrang.

Here is the call graph for this function:



10.20.3 Member Function Documentation

10.20.3.1 def libcommonfunc.Parameters.readChebash (*self, paramList*)

Translate a list of Chebash parameters into a list of equivalent lossyfilters parameters.

Parameters

paramList	= List of tuples (Chebash_Parameter_Name, value) where value is a string

Returns

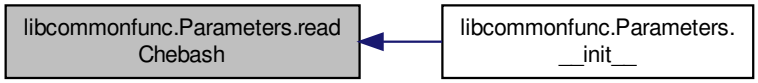
paramList = List of tuples (LossyFilters_Parameter_Name, type, value) where type and value are strings

Definition at line 590 of file libcommonfunc.py.

References mwfiltersgui.SpecMask.BW, libcommonfunc.FrequencyTransformBP.BW, mwfiltersgui.SpecMask.f0, mwfiltersgui.myTableWidget.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, libcommonfunc.Parameters.N, libcommonfunc.Parameters.transComplexZeros, and libcommonfunc.Parameters.transImagZeros.

Referenced by libcommonfunc.Parameters.__init__().

Here is the caller graph for this function:



10.20.3.2 def libcommonfunc.Parameters.readParam (*self, key, type, stValue*)

Parse int, float, complex or string parameters from string stValue.

Parameters

key	= string containing parameter keyword.
type	= string containing parameter type: 'int', 'float', 'complex' or 'string'. 'Float' type allows a value of Inf, inf or INF.
stValue	= string containing parameter or a list of comma-separated parameters. Complex number format is 1+2j.

Returns

parameter value.

Definition at line 468 of file libcommonfunc.py.

The documentation for this class was generated from the following file:

- libcommonfunc.py

10.21 libcommonfunc.parseError Class Reference

Parse error catch.

10.21.1 Detailed Description

Parse error catch.

Definition at line 109 of file libcommonfunc.py.

The documentation for this class was generated from the following file:

- libcommonfunc.py

10.22 dbplot.PlotMarker Class Reference

Class for markers.

Public Member Functions

- def __init__
  Class constructor.
- def updateLabel
  Update marker label according to required precision.
- def updatePosition
  Update marker position after a curve update with DbPlot.update().

10.22.1 Detailed Description

Class for markers.

Definition at line 2313 of file dbplot.py.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 def dbplot.PlotMarker.__init__ (*self, dbplot, axis, nTab, curve, index*)

Class constructor.

Gets the data from dbplot zoomers base in nTab.

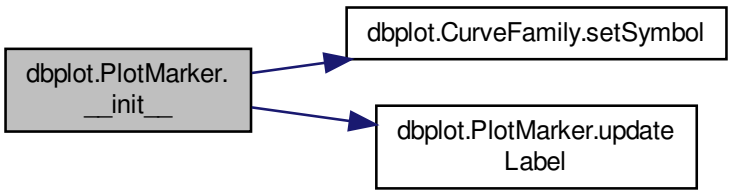
Parameters

dbplot	= DbPlot class instance, to which this marker belongs.
Yaxis	= y-axis to use Qwt.QwtPlot.yLeft or Qwt.QwtPlot.yRight.
nTab	= Tab widget index.
curve	= QwtPlotCurve instance, to which this marker is linked.
index	= Index of curve point. DbPlot class instance, to which this marker belongs. y-axis to use Qwt.-QwtPlot.yLeft or Qwt.QwtPlot.yRight. Tab widget index QwtPlotCurve instance, to which this marker is linked Index of curve x-data closest to x Marker x-coordinate Marker y-coordinate QwtPlot class instance where this marker is attached to. QwtSymbol class instance to use when the marker state is normal QwtSymbol class instance to use when the marker state is selected for move QwtSymbol class instance to use when the marker state is selected for deletion

Definition at line 2324 of file dbplot.py.

References dbplot.CurveFamily.axis, dbplot.PlotMarker.axis, dbplot.PlotMarker.curve, dbplot.MyPicker.dbplot, dbplot.CanvasEventFilter.dbplot, dbplot.PlotMarker.dbplot, dbplot.DbPlot.hPlot, dbplot.CurveFamily.hPlot, dbplot.PlotMarker.hPlot, dbplot.PlotMarker.index, dbplot.MyPicker.nTab, dbplot.PlotMarker.nTab, dbplot.CurveFamily.setSymbol(), dbplot.PlotMarker.symbolDel, dbplot.PlotMarker.symbolMove, dbplot.PlotMarker.symbolNormal, dbplot.PlotMarker.updateLabel(), dbplot.PlotMarker.x, and dbplot.PlotMarker.y.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- dbplot.py

10.23 mwfiltersgui.PredisZeorsDlg Class Reference

Dialog to select predistortion zeros.

Public Member Functions

- def __init__

  Create dialog to select visible curves.

10.23.1 Detailed Description

Dialog to select predistortion zeros.

Definition at line 2666 of file mwfiltersgui.py.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 def mwfiltersgui.PredisZeorsDlg.__init__ (*self, roots_NRP, zerosArrang, parent* = `None`)

Create dialog to select visible curves.

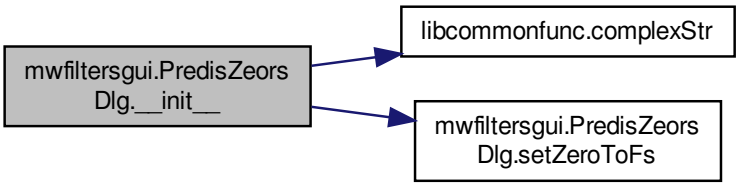
```
@roots_NRP    = Roots of F(s)F(-s) that are in the left-hand s-plane
@zerosArrang  = Predistiortion zeros arrangement (list of bool).
@param parent = Parent window (default None).
```

Handle to the dbplot main window.

Definition at line 2675 of file mwfiltersgui.py.

References libcommonfunc.complexStr(), dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.SetCouplingDlg.mainWindow, mwfiltersgui.MatrixEditDlg.mainWindow, mwfiltersgui.PredisZeorsDlg.mainWindow, mwfiltersgui.PredisZeorsDlg.setZeroToFs(), libcommonfunc.Parameters.zerosArrang, and mwfiltersgui.PredisZeorsDlg.zerosArrang.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.24 dbplot.PrintFilter Class Reference

Filter to transform color of some plot items and fontsize when printing.

Public Member Functions

- def __init__
  Constructor.
- def color
  Modifies a color for printing.
- def font
  Modifies a font for printing.

10.24.1 Detailed Description

Filter to transform color of some plot items and fontsize when printing.

Definition at line 2479 of file dbplot.py.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 def dbplot.PrintFilter.__init__ (*self*)

Constructor.

The class is derived from Qwt5.QwtPlotPrintFilter.

Definition at line 2484 of file dbplot.py.

10.24.3 Member Function Documentation

10.24.3.1 def dbplot.PrintFilter.color (*self, c, item*)

Modifies a color for printing.

Parameters

c	= Color to be modified (QColor).
item	= Type of item where the color belongs.

Returns

Modified color (QColor).

Definition at line 2494 of file dbplot.py.

10.24.3.2 def dbplot.PrintFilter.font (*self, f, item*)

Modifies a font for printing.

Parameters

f	= Font to be modified (QFont).
item	= Type of item where the font belongs.

Returns

All fonts are returned unmodified.

Definition at line 2517 of file dbplot.py.

The documentation for this class was generated from the following file:

- dbplot.py

10.25 mwfiltersgui.SensitivityAnalysis Class Reference

Class for sensitivity analysis.

Public Member Functions

- def __init__
  Constructor: creates SensitivityAnalisys class instance.
- def runAnalysis
  Run Monte Carlo analysis with random variations in coupling matrix elements.
- def randomVariations
  Returns a randomly modified coupling matrix with maximum element variations determined by function arguments.
- def plotSensitivity
  Plot [S] parameters magnitude and phase graph, if they are available.

10.25.1 Detailed Description

Class for sensitivity analysis.

Definition at line 2736 of file mwfiltersgui.py.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 def mwfiltersgui.SensitivityAnalysis.__init__ (*self, parent*)

Constructor: creates SensitivityAnalisys class instance.

```
@param parent = Parent widget, in this case is coupling matrix mainWindow.
```

Copy of the coupling matrix mainWindow class instance. It is necessary to store a copy since it must be accessed by some member functions. QwtPlot that contains the Magnitude and Phase S-parameter plot for all random realizations. Numpy array with columns containing S11 for each random realization. Numpy array with columns containing S21 for each random realization. Numpy array with columns containing S22 for each random realization. Numpy array with columns containing phase of S11 for each random realization. Numpy array with columns containing phase of S21 for each random realization. Numpy array with columns containing phase of S22 for each random realization. Numpy array with columns containing groupDelay for each random realization. [S] parameters class instance, used to compute the [S] parameters after random variations in coupling matrix.

Definition at line 2743 of file mwfiltersgui.py.

References libcommonfunc.Sparameters.groupDelayM, mwfiltersgui.SensitivityAnalysis.groupDelayM, dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.SetCouplingDlg.mainWindow, mwfiltersgui.MatrixEditDlg.mainWindow, mwfiltersgui.PredisZeorsDlg.mainWindow, mwfiltersgui.SensitivityAnalysis.mainWindow, libcommonfunc.Sparameters.phaseS11M, mwfiltersgui.SensitivityAnalysis.phaseS11M, libcommonfunc.Sparameters.phaseS21M, mwfiltersgui.SensitivityAnalysis.phaseS21M, libcommonfunc.Sparameters.phaseS22M, mwfiltersgui.SensitivityAnalysis.phaseS22M, libcommonfunc.Sparameters.S11M, mwfiltersgui.SensitivityAnalysis.S11M, libcommonfunc.Sparameters.S21M, mwfiltersgui.SensitivityAnalysis.S21M, libcommonfunc.Sparameters.S22M, mwfiltersgui.SensitivityAnalysis.S22M, mwfiltersgui.MainWindow.SP, mwfiltersgui.SensitivityAnalysis.SP, libcommonfunc.CouplingMatrices.SP, mwfiltersgui.MainWindow.SPlotMagPhase, and mwfiltersgui.SensitivityAnalysis.SPlotMagPhase.

10.25.3 Member Function Documentation

10.25.3.1 def mwfiltersgui.SensitivityAnalysis.plotSensitivity (*self*)

Plot [S] parameters magnitude and phase graph, if they are available.

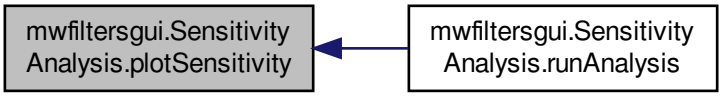
The plotted data is self.S11, self.S21 and self.S22, which contain the [S] parameters for all random realizations, in columns.

Definition at line 2959 of file mwfiltersgui.py.

References libcommonfunc.Sparameters.groupDelayM, mwfiltersgui.SensitivityAnalysis.groupDelayM, dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.SetCouplingDlg.mainWindow, mwfiltersgui.MatrixEditDlg.mainWindow, mwfiltersgui.PredisZeorsDlg.mainWindow, mwfiltersgui.SensitivityAnalysis.mainWindow, libcommonfunc.Sparameters.phaseS11M, mwfiltersgui.SensitivityAnalysis.phaseS11M, libcommonfunc.Sparameters.phaseS21M, mwfiltersgui.SensitivityAnalysis.phaseS21M, libcommonfunc.Sparameters.phaseS22M, mwfiltersgui.SensitivityAnalysis.phaseS22M, libcommonfunc.Sparameters.S11M, mwfiltersgui.SensitivityAnalysis.S11M, libcommonfunc.Sparameters.S21M, mwfiltersgui.SensitivityAnalysis.S21M, libcommonfunc.Sparameters.S22M, and mwfiltersgui.SensitivityAnalysis.S22M.

Referenced by mwfiltersgui.SensitivityAnalysis.runAnalysis().

Here is the caller graph for this function:



10.25.3.2 def mwfiltersgui.SensitivityAnalysis.randomVariations (*self, MatQ, inout, inductive, capacitive, resistive, f0var, Qvar*)

Returns a randomly modified coupling matrix with maximum element variations determined by function arguments.

The random variation has uniform distribution.

Parameters

MatQ	= Coupling matrix to which random variations are applied (MatrixQ class instance). It is not modified, and the modified (deep) copy is returned.
inout	= Maximum random variation in % of nominal value, for input/output (source/load) couplings. It is applied to elements in first and last rows and columns.
inductive	= Maximum random variation in % of nominal value, for inductive couplings. It is applied to positive real parts of out-of-diagonal matrix elements $\Re M_{ij} > 0 \qquad i \neq j$.
capacitive	= Maximum random variation in % of nominal value, for capacitive couplings. It is applied to negative real parts of out-of-diagonal matrix elements $\Re M_{ij} < 0 \qquad i \neq j$.
resistive	= Maximum random variation in % of nominal value, for resistive couplings. It is applied to imaginary parts of out-of-diagonal matrix elements $\Im M_{ij} > 0 \qquad i \neq j$.
f0var	= Maximum random variation in % of nominal value, for resonators resonant frequency.
Qvar	= Maximum random variation in % of nominal value, for resonators quality factor.

Returns

modMatQ = Modified coupling matrix, based on a deep copy of the MatQ argument.

In order to obtain the randomly modified resonant frequency, the real part of diagonal elements corresponding to resonating nodes is unnormalized. The random variation is applied to the unnormalized frequency, and it is normalized again to obtain the modified real part of the matrix element.

Diagonal elements corresponding to non-resonant nodes are not modified.

The modified Q of resonators is obtained applying a random variation to the resonator Q of the original matrix (before applying the random variation to resistive couplings). Then, the resistive couplings are randomly modified and the new imaginary part of the diagonal element is computed accounting for the new values of the resistive couplings.

In general, for each node n, the quality factor is

$$Q_n = \frac{1}{G_n \mathrm{FBW}}$$

where the loss conductance of the unloaded resonator is:

$$G_n = -\sum_{m=1}^{N} \Im M_{mn}$$

The losses corresponding to the new \form#285 for resonator \form#62 are:

$$G'_n = \frac{1}{Q'_n \mathrm{FBW}}$$

But, after the random variation of resistive couplings the losses that we have are

$$G''_n = -\sum_{m=1}^{N} \Im M''_{nm}$$

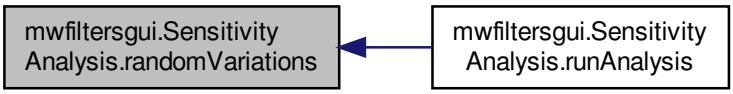
so, in order to achieve a quality factor Q', the modified value of the imaginary part of the diagonal element must be:

$$\Im M'_{nn} = \Im M''_{nn} + G''_n - G'_n$$

Definition at line 2902 of file mwfiltersgui.py.

Referenced by mwfiltersgui.SensitivityAnalysis.runAnalysis().

Here is the caller graph for this function:



10.25.3.3 def mwfiltersgui.SensitivityAnalysis.runAnalysis (*self, N, inout, freq, Q, inductive, capacitive, resistive*)

Run Monte Carlo analysis with random variations in coupling matrix elements.

Calls self.randomVariations() to obtain a randomly modified coupling matrix with maximum element variations determined by function arguments.

Parameters

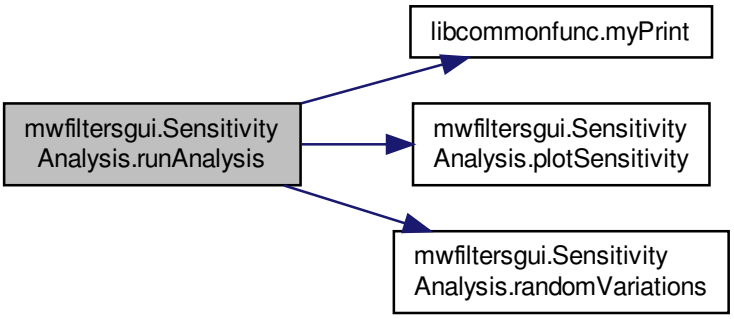
N	= Number of Monte Carlo random realizations.
inout	= Maximum random variation in % of nominal value, for input/output (source/load) couplings. It is applied to elements in first and last rows and columns.
freq	= Maximum random variation in % of nominal value, for resonators resonant frequency.
Q	= Maximum random variation in % of nominal value, for resonators quality factor.
inductive	= Maximum random variation in % of nominal value, for inductive couplings. It is applied to positive real parts of out-of-diagonal matrix elements.
capacitive	= Maximum random variation in % of nominal value, for capacitive couplings. It is applied to negative real parts of out-of-diagonal matrix elements.
resistive	= Maximum random variation in % of nominal value, for resistive couplings. It is applied to imaginary parts of out-of-diagonal matrix elements.

Definition at line 2798 of file mwfiltersgui.py.

References libcommonfunc.Sparameters.groupDelayM, mwfiltersgui.SensitivityAnalysis.groupDelayM, libcommonfunc.myPrint(), libcommonfunc.Sparameters.phaseS11M, mwfiltersgui.SensitivityAnalysis.phaseS11M, libcommonfunc.Sparameters.phaseS21M, mwfiltersgui.SensitivityAnalysis.phaseS21M, libcommonfunc.Sparameters.phaseS22M, mwfiltersgui.SensitivityAnalysis.phaseS22M, mwfiltersgui.SensitivityAnalysis.plotSensitivity(), mwfiltersgui.SensitivityAnalysis.randomVariations(), libcommonfunc.Sparameters.S11M, mwfiltersgui.SensitivityAnalysis.S11M,

libcommonfunc.Sparameters.S21M, mwfiltersgui.SensitivityAnalysis.S21M, libcommonfunc.Sparameters.S22M, and mwfiltersgui.SensitivityAnalysis.S22M.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.26 mwfiltersgui.SetCouplingDlg Class Reference

Dialog to set the type of a coupling.

Public Member Functions

- def __init__

  Create dialog to set the type of a coupling.
- def typeSelected

  Set type selected in self.comboBox to self.item.

10.26.1 Detailed Description

Dialog to set the type of a coupling.

Definition at line 2080 of file mwfiltersgui.py.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 def mwfiltersgui.SetCouplingDlg.__init__ (*self, item, topolText, topolColors, parent* = None)

Create dialog to set the type of a coupling.

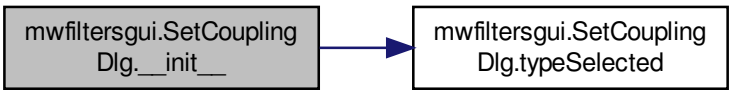
```
@param item = Topology table item to modify
@param topolText = Topology table text, corresponding to ComboBox items
@param topolColors = Topology table colors, corresponding to ComboBox items
@param parent = parent widget (topology table)
```

Topology table widget Topology table item to set ComboBox item text for different kinds of coupling Topology table text, corresponding to ComboBox items Topology table color, corresponding to ComboBox items

Definition at line 2090 of file mwfiltersgui.py.

References mwfiltersgui.SetCouplingDlg.comboBox, mwfiltersgui.SetCouplingDlg.item, dbplot.DbPlot.label, mwfiltersgui.SetCouplingDlg.label, dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, mwfiltersgui.HelpForm.mainWindow, dbplot.EditCurvesDlg.mainWindow, mwfiltersgui.SetCouplingDlg.mainWindow, mwfiltersgui.SetCouplingDlg.topolColors, mwfiltersgui.SetCouplingDlg.topolText, mwfiltersgui.SetCouplingDlg.typeLabels, and mwfiltersgui.SetCouplingDlg.typeSelected().

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.27 libcommonfunc.Sparameters Class Reference

[S] parameters class.

Public Member Functions

- def __init__

  Initialize frequency axis attributes and compute S parameters from Characteristic Polynomials.
- def fromCharPolys

  Compute S-Parameters (S_{11}, S_{12}, S_{21}, and S_{22}) from the characteristic polynomials.
- def groupDelayfunc

  This function returns the group delay as defined in [Cameron eq.
- def fromCouplingMatrix

  Compute S-Parameters (S_{21} and S_{11}) from a coupling matrix.
- def energyPowerFromCM

  Compute stored energy and dissipated power at resonators and lossy couplings, from a coupling matrix.
- def saveSparam

  Save [S] parameters in output file.

10.27.1 Detailed Description

[S] parameters class.

Definition at line 935 of file libcommonfunc.py.

10.27.2 Constructor & Destructor Documentation

10.27.2.1 def libcommonfunc.Sparameters.__init__ (*self, P, CP, FT, freq* = None, *checkPassive* = True)

Initialize frequency axis attributes and compute S parameters from Characteristic Polynomials.

```
@param P  = Parameters class instance, containing filter, synthesis and sweep parameters.
@param CP = CharPolys  class instance, containing characteristic polynomials and constants.
@param FT = Frequency transform class instance, containing methods to normalize frequency and unnormalize
@param freq = Frequency axis (Hz). If equal to None, the frequency sweep based on P.minFreq, P.maxFreq and
@param checkPassive = If checkPassive is True, a warning will be raised when there are [S] parameters larg

[S] parameters computed from characteristic polynomials.
```

$$S_{11} = \frac{1}{\varepsilon_R}\frac{F(s)}{E(s)}$$

$$S_{21} = S_{12} = \frac{1}{\varepsilon}\frac{P(s)}{E(s)}$$

$$S_{22} = \frac{1}{\varepsilon_R}\frac{F_{22}(s)}{E(s)} = (-1)^N \frac{1}{\varepsilon_R}\frac{F(s)^*}{E(s)}$$

except for asymmetrical "Prescribed Insertion Loss technique" case 1 (kS21+kS11), where it is computed as

$$S_{22} = k_{22}\left(1 - \left(\frac{k_{21}}{k_{11}}\right)^2\right) + \left(\frac{k_{21}}{k_{11}}\right)^2 S_{11}$$

In this "Prescribed Insertion Loss technique" case 1, the correct lossy values of S_{11} and S_{21} are obtained because ε_R and ε have been divided by k_{11} and k_{21} , respectively, when processing the polinomials at liblossyfilters.nonUniformQ() function . Fractional bandwidth of the FT used to compute this [S] parameters. Central frequency of the FT used to compute this [S] parameters. Frequency transform used to compute this [S] parameters. Frequency axis (Hz) Normalized ω' axis Dissipation factor. [Cameron eq. (3.118)] [TN 101.1 eq. (14)].

$\sigma = \frac{f_0}{\Delta f}\frac{1}{Q_0}$ Complex frequency variable in normalized s-plane. $s = \sigma + j\omega'$ Parameter S_{11} computed from characteristic polynomials. $S_{11} = \frac{1}{\varepsilon_R}\frac{F(s)}{E(s)}$ Parameter S_{12} computed from characteristic polynomials. $S_{12} = \frac{1}{\varepsilon}\frac{P(s)}{E(s)}$ Parameter S_{21} computed from characteristic polynomials. $S_{21} = \frac{1}{\varepsilon}\frac{P(s)}{E(s)}$ Parameter S_{22} computed from characteristic polynomials. $S_{22} = \frac{1}{\varepsilon_R}\frac{F_{22}(s)}{E(s)} = (-1)^N\frac{1}{\varepsilon_R}\frac{F(s)^*}{E(s)}$ [Cameron eq. (6.15) pp. 212]

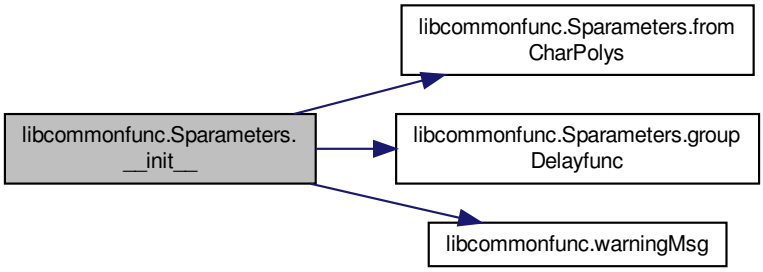
except for asymmetrical "Prescribed Insertion Loss technique" case 1 (kS21+kS11), where it is computed as $S_{22} = 1 - \left(\frac{k_{21}}{k_{11}}\right)^2 + \frac{k_{21}^2}{k_{11}k_{22}}S_{11_{lossless}}$ Group delay for the LowPass prototype Group delay for the unnormalized filter (sec) Phase of transfer function (rad) Phase of return losses (rad) Phase of inverse return losses (rad) Deviation from linear phase. It is the error between the phase of S21 and the straight line that best fits the phase in the center of the band in an interval equal to half the bandwidth. Parameter S_{11} computed from a coupling matrix. Parameter S_{21} computed from a coupling matrix. Parameter S_{22} computed from a coupling matrix. Power dissipated at resonators. 2-D numpy array of shape (Nf, Nr), where Nf is the number of frequencies and Nr the number of resonators. Energy stored at resonators. 2-D numpy array of shape (Nf, Nr), where Nf is the number of frequencies and Nr the number of resonators. Power dissipated at resistive couplings. Tuple ((m, n), Gmn, powerDirect, powerReverse) where power is a 1-D numpy array with one element for each frequency. Only entries n>m are stored. Energy stored at magnetic or electric couplings. Tuple ((m, n), Bmn, energyDirect, energyReverse) where energy is a 1-D numpy array with one element for each frequency. Only entries n>m are stored. Power dissipated at resonators, with reverse excitation. 2-D numpy array of shape (Nf, Nr), where Nf is the number of frequencies and Nr the number of resonators. Energy stored at resonators, with reverse excitation. 2-D numpy array of shape (Nf, Nr), where Nf is the number of frequencies and Nr the number of resonators.

Definition at line 965 of file libcommonfunc.py.

References libcommonfunc.Sparameters.deviationLP, libcommonfunc.Sparameters.energyCoup, libcommonfunc.Sparameters.energyRes, libcommonfunc.Sparameters.energyResRev, libcommonfunc.Sparameters.f0, libcommonfunc.Sparameters.FBW, libcommonfunc.Sparameters.freq, libcommonfunc.Sparameters.fromCharPolys(), libcommonfunc.Sparameters.FT, libcommonfunc.Sparameters.groupDelay, libcommonfunc.Sparameters.groupDelayfunc(),

libcommonfunc.Sparameters.groupDelayLP, libcommonfunc.Sparameters.omega, libcommonfunc.Sparameters.-phaseS11, libcommonfunc.Sparameters.phaseS21, libcommonfunc.Sparameters.phaseS22, libcommonfunc.-Sparameters.powerCoup, libcommonfunc.Sparameters.powerRes, libcommonfunc.Sparameters.powerRes-Rev, libcommonfunc.Sparameters.s, libcommonfunc.Sparameters.S11, libcommonfunc.Sparameters.S11-M, libcommonfunc.Sparameters.S12, libcommonfunc.Sparameters.S21, libcommonfunc.Sparameters.S21M, libcommonfunc.Sparameters.S22, libcommonfunc.Sparameters.S22M, libcommonfunc.Sparameters.sigma, and libcommonfunc.warningMsg().

Here is the call graph for this function:



10.27.3 Member Function Documentation

10.27.3.1 def libcommonfunc.Sparameters.energyPowerFromCM (*self, MatQ*)

Compute stored energy and dissipated power at resonators and lossy couplings, from a coupling matrix.

```
Evaluation points are taken from attribute Sparameters.s .
Output in atributes Sparameters.energyRes, Sparameters.powerRes ,
which are 2D numpy arrays, each column is the energy(f) or power(f) at a resonator.
@param MatQ = MatrixQ class instance, containing the coupling matrix to process.

Dissipated power at \form#38-th resonator can be found as:
```

$$P_i(f) = \frac{G_i}{2}|V_i(f)|^2$$

being V_i the voltage at each resonator and G_i the conductance of unloaded resonators

$$G_i = \frac{Z_0}{Q_i \mathrm{FBW}}$$

where Q_i is the quality factor, $\mathrm{FBW} = \Delta f / f_0$ is the filter fractional bandwidth and Z_0 is the characteristic impedance of the resonators.

The stored energy is:

$$W_i(f) = \frac{Q_i P_i(f)}{2\pi f_0} = \frac{Z_0}{4\pi f_0 \mathrm{FBW}}|V_i(f)|^2$$

```
For resistive couplings between resonators \form#47 and \form#48, with conductance \form#49, the voltage c
the dissipated power becomes:
```

$$P_{ij}(f) = \frac{G_{ij}}{2}\,|V_{ij}(f)|^2$$

```
Following [Pozar eq. (4.14) and (4.16)], the time harmonic complex power is
```

$$P = \frac{1}{2} V I^* = \frac{1}{2} |V|^2 Y^* = Pd + 2j\omega(W_m - W_e)$$

and, accounting that $Y_{ij} = jM_{ij}$, the stored energy at magnetic or electric couplings between resonators i and j, with susceptance $B_{ij} = \Im Y_{ij} = \Re M_{ij}$, is

$$W_{ij}(f) = W_m - W_e = \frac{B_{ij}}{4\omega} |V_{ij}(f)|^2$$

Voltages at each resonator for unitary current excitation at source port are obtained as:

$$V(s) = Z(s) J(s)$$

where $J(s)$ is the input current vector $J(s) = [I_s \ 0 \ldots 0]^T$

If the input is matched, the input current is $I_s = \sqrt{8P_0/Z_0}$. We will assume input power P_0 equal to 1 Watt.

The impedance matrix is computed as:

$$Z(s) = (jM + sI + G)^{-1}$$

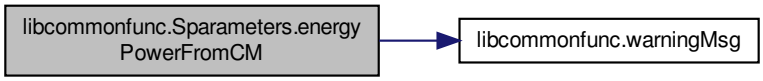
with I a diagonal matrix with the node scaling factor squared in the diagonal except in the non-resonant nodes positions, where it is zero. It means putting zeros in the extraNodesS first positions of the diagonal and in the last extraNodesL positions.

$$G = \begin{bmatrix} G_S & & & & \\ & 0 & & & \\ & & \ddots & & \\ & & & 0 & \\ & & & & G_L \end{bmatrix}$$

Definition at line 1408 of file libcommonfunc.py.

References libcommonfunc.Sparameters.energyCoup, libcommonfunc.Sparameters.energyRes, libcommonfunc.Sparameters.energyResRev, libcommonfunc.Sparameters.f0, libcommonfunc.Sparameters.FBW, libcommonfunc.Sparameters.freq, libcommonfunc.Sparameters.powerCoup, libcommonfunc.Sparameters.powerRes, libcommonfunc.Sparameters.powerResRev, libcommonfunc.Sparameters.s, and libcommonfunc.warningMsg().

Here is the call graph for this function:



10.27.3.2 def libcommonfunc.Sparameters.fromCharPolys (*self, P, evalS, Ps, Es, Fs, eps, epsR*)

Compute S-Parameters (S_{11}, S_{12}, S_{21}, and S_{22}) from the characteristic polynomials.

Evaluation points are taken from attribute Sparameters.s . This method will be called by the BW and TZs optimization routines. Output in atributes Sparameters.S11, Sparameters.S12, Sparameters.S21 and Sparameters.S22 .

Parameters

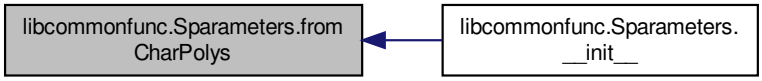
P	= Parameters class instance, containing filter, synthesis and sweep parameters.
evalS	= Vector with the normalized complex frequencies (s) where we want to compute the group delay.
Ps	= Characteristic polynomial of the numerator of S_{21}.
Es	= Characteristic polynomial of the denominator of S_{11} and S_{21}.
Fs	= Characteristic polynomial of the numerator of S_{11}.
eps	= Characteristic constant ε in the denominator of S_{12} and S_{21}
epsR	= Characteristic constant ε_R in the denominator of S_{11}

Definition at line 1154 of file libcommonfunc.py.

References libcommonfunc.Sparameters.S11, libcommonfunc.Sparameters.S12, libcommonfunc.Sparameters.S21, and libcommonfunc.Sparameters.S22.

Referenced by libcommonfunc.Sparameters.__init__().

Here is the caller graph for this function:



10.27.3.3 def libcommonfunc.Sparameters.fromCouplingMatrix (*self, MatQ, evalS* = `None`, *checkPassive* = `True`)

Compute S-Parameters (S_{21} and S_{11}) from a coupling matrix.

```
Evaluation points are taken from attribute Sparameters.s .
Output in atributes Sparameters.S11M and Sparameters.S21M .
@param MatQ = MatrixQ class instance, containing the coupling matrix to process.
@param evalS = Normalized frequency axis (imaginary). If equal to None, it is set to 1j * np.imag(self.s).
@param checkPassive = If checkPassive is True, a warning will be raised when there are [S] parameters larg
@return stErrorCM = String with a message about the error in [S] computed from M coupling matrix versus se
@return CM_error = Relative error in [S] computed from M coupling matrix versus self.S11 and self.S21 (flo

By definition, \form#29 and \form#30 can be found as:
```

$$S_{21}(s) = 2\sqrt{R_S R_L}\, Z'_{N+2,1}(s)$$

$$S_{11}(s) = 1 - 2R_S Z'_{1,1}(s)$$

$$S_{22}(s) = 1 - 2R_L Z'_{N+2,N+2}(s)$$

where:

$$Z'(s) = Y'(s)^{-1}$$

$$Y'(s) = jM + sI + G$$

with I a diagonal matrix with the node scaling factor squared in the diagonal except in the non-resonant nodes positions, where it is zero. It means putting zeros in the extraNodesS first positions of the diagonal and in the last extraNodesL positions.

$$G = \begin{bmatrix} G_S & & & & \\ & 0 & & & \\ & & \ddots & & \\ & & & 0 & \\ & & & & G_L \end{bmatrix}$$

Todo Find out why sign of S11 from CM is opposite than from CP. The sign is changed below to force agreement with S11 from CP.

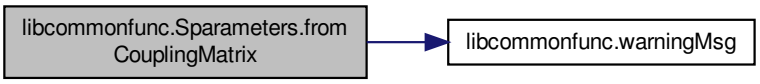
Phase of S11 return losses computed from coupling matrix Phase of S22 return losses computed from coupling matrix Phase of transfer function computed from coupling matrix Group delay computed from coupling matrix Frequency axis with sampling points in the middle of self.freq sampling. It is necessary to plot self.groupDelayM Group

delay computed from Characteristic Polynomials, sampled at self.freq2 It is necessary to compare with self.groupDelayM

Definition at line 1223 of file libcommonfunc.py.

References libcommonfunc.Sparameters.freq, libcommonfunc.Sparameters.freq2, libcommonfunc.Sparameters.groupDelay, libcommonfunc.Sparameters.groupDelay2, libcommonfunc.Sparameters.groupDelayM, libcommonfunc.Sparameters.phaseS11M, libcommonfunc.Sparameters.phaseS21M, libcommonfunc.Sparameters.phaseS22M, libcommonfunc.Sparameters.S11, libcommonfunc.Sparameters.S11M, libcommonfunc.Sparameters.S21, libcommonfunc.Sparameters.S21M, libcommonfunc.Sparameters.S22, libcommonfunc.Sparameters.S22M, and libcommonfunc.warningMsg().

Here is the call graph for this function:



10.27.3.4 def libcommonfunc.Sparameters.groupDelayfunc (*self, Es, Ps, evalS*)

This function returns the group delay as defined in [Cameron eq.

13.7, pp. 475-476] evaluated in the normalized complex frequency vector evalS .

Parameters

Es	= Characteristic polynomial of the denominator of S_{21} and S_{11}.
Ps	= Characteristic polynomial of the numerator of S_{21}.
evalS	= Vector with the normalized complex frequencies (s) where we want to compute the group delay.

Returns

groupD = Vector with the group delay at each frequency. The group delay is computed from the formula:

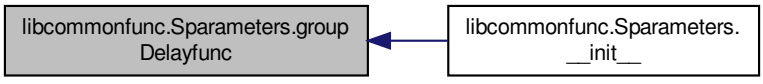
$$\tau_i = Re\left[\frac{E'(s_i)}{E(s_i)} - \frac{P'(s_i)}{P(s_i)}\right]$$

where s_i are the frequency samples.

Definition at line 1188 of file libcommonfunc.py.

Referenced by libcommonfunc.Sparameters.__init__().

Here is the caller graph for this function:



10.27.3.5 def libcommonfunc.Sparameters.saveSparam (*self, P, prec*)

Save [S] parameters in output file.

S11, S12, S21 and S22 are attributes of sParameters class, and of type numpy.array.

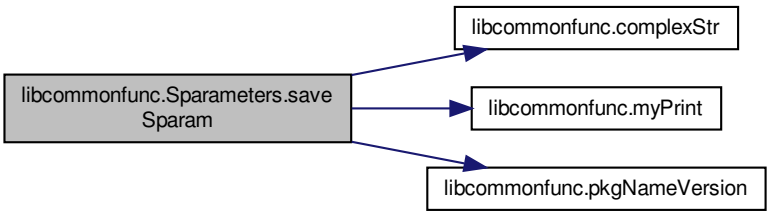
Parameters

P	= Parameter class instance.
prec	= Number of correct significant digits to print.

Definition at line 1519 of file libcommonfunc.py.

References libcommonfunc.complexStr(), libcommonfunc.Sparameters.freq, libcommonfunc.myPrint(), libcommonfunc.pkgNameVersion(), libcommonfunc.Sparameters.S11, libcommonfunc.Sparameters.S12, libcommonfunc.Sparameters.S21, and libcommonfunc.Sparameters.S22.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- libcommonfunc.py

10.28 mwfiltersgui.SpecMask Class Reference

Specification mask class.

Public Member Functions

- def __init__

  Read specification mask from file.
- def specToMask

  Process specification data to obtain the mask curves.
- def maskSamplesOptimization

  Create mask samples to be used as a reference in optimization.
- def dataSymmetrizeDeltaF0

  Sort and symmetrise data with respect to centre frequency self.f0 and add self.f0 to frequency column of array.
- def freqSort

  Sort array by frequency (inplace).

10.28.1 Detailed Description

Specification mask class.

Definition at line 218 of file mwfiltersgui.py.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 def mwfiltersgui.SpecMask.__init__ (*self, fileName, parent =* None)

Read specification mask from file.

```
     @param fileName = File Name
     @param parent = Parent window for the temperature variation dialog


Open a widget to get temperature variation parameters:
Tamb = Ambient Temperature: Temperature at which the filter is built. At ambient temperature, the measured [S]
Tmax = Maximum operating temperature. Due to thermal expansion, the filter at Tmax is actually larger than the
Tmin = Minimum operating temperature. Due to thermal expansion, the filter at Tmin is actually smaller than th
alpha = Thermal expansion coefficient, expressed in parts per million per degree Celsius. Example: 22 ppm/C fo
@return freqShiftRight = Relative frequency shift of the mask to the right = alpha*(Tmax-Tamb)*1e-6.
@return freqShiftLeft  = Relative frequency shift of the mask to the left = alpha*(Tamb-Tmin)*1e-6.


          Helper function that returns next item that does not start with '!' of iterLines iterator.
          Items returned by iterLines iterator must be strings.
          @return line = First item that does not start with '!'


          Helper function that returns the conversion factor corresponding to 'units' argument.
          @param  units = Units. Must be a key in the unitsFactorDict dictionary. String.
          @return factor = Frequency conversion factor


          Helper function that returns the data read form specification mask file.
          The units conversion factors are applied.
          @param line = String to process (line of specification mask file).
          @return data = Numpy array containing the data.
          @return line = String containing next non-comment line of specification mask file after the data, nor
```

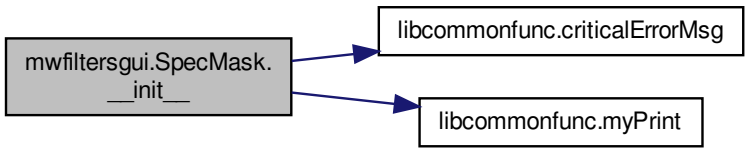
File name Parent window Central frequency Bandwidth Specification for S11 (return losses) in dB. Float. Value of S21 at f0 Value of S21 over bandwidth Specification for S21 variation over bandwidth (insertion losses) in dB pk-pk. Numpy float array Nx2 where N is the number of points, frequency data (Hz) is in the 1st column and Mask data in the 2nd. Specification for S21 out of band rejection in dB. Numpy float array Nx2 where N is the number of points, frequency data (Hz) is in the 1st column and Mask data in the 2nd. Group delay mask (sec). Numpy float array Nx2 where N is the number of points, frequency data (Hz) is in the 1st column and Mask data in the 2nd. Frequency samples in the pass band for optimization Frequency samples in the rejection band for optimization Array with all frequency samples for optimization, in Hz, the first self.NfreqInBandOptim correspond to self.freqInbandOptim and the rest ot self.freqOutBandOptim Array with all frequency samples for optimization, in normalized s-plane, the first self.NfreqInBandOptim correspond to self.freqInbandOptim and the rest ot self.freqOutBandOptim Number of samples in self.evalSoptim corresponding to the pass band = len(self.freqInbandOptim) S11 mask samples in the pass band for optimization S21 mask samples in the pass band for optimization S21 mask samples in the rejection band for optimization Factor to multiply the frequencies that shift to the right Factor to multiply the frequencies that shift to the left

Definition at line 226 of file mwfiltersgui.py.

References mwfiltersgui.SpecMask.BW, libcommonfunc.FrequencyTransformBP.BW, libcommonfunc.criticalErrorMsg(), mwfiltersgui.SpecMask.evalFoptim, mwfiltersgui.SpecMask.evalSoptim, mwfiltersgui.SpecMask.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, mwfiltersgui.SpecMask.fileName, mwfiltersgui.SpecMask.freqFactorLeft, mwfiltersgui.SpecMask.freqFactorRight, mwfiltersgui.SpecMask.freqInbandOptim, mwfiltersgui.SpecMask.freqOutBandOptim, mwfiltersgui.SpecMask.GDspec, dbplot.DbPlot.mainWindow, mwfiltersgui.SpecMask.mainWindow, dbplot.AxisScalingDlg.mainWindow, dbplot.EditCurvesDlg.mainWindow, libcommonfunc.myPrint(), mwfiltersgui.SpecMask.NfreqInBandOptim, mwfiltersgui.SpecMask.S11optim,

mwfiltersgui.SpecMask.S11spec, mwfiltersgui.SpecMask.S21_BW, mwfiltersgui.SpecMask.S21_F0, mwfiltersgui.SpecMask.S21inBandOptim, mwfiltersgui.SpecMask.S21inbandSpecDelta, mwfiltersgui.SpecMask.S21outBandOptim, and mwfiltersgui.SpecMask.S21outbandSpec.

Here is the call graph for this function:



10.28.3 Member Function Documentation

10.28.3.1 def mwfiltersgui.SpecMask.dataSymmetrizeDeltaF0 (*self, inArray, freqCol, hideBW* = `False`)

Sort and symmetrise data with respect to centre frequency self.f0 and add self.f0 to frequency column of array.

Frequencies in the input array are delta with respect to centre frequency self.f0. Frequencies in the result are absolute.

Parameters

inArray	= Input numpy array.
freqCol	= Column index of frequency values.
hideBW	= Flag to hide the centre of the mask. If True, the mask is not displayed at the centre, between the symmetric data points. Default False.

Returns

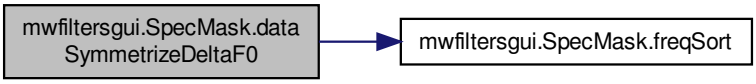
outArray = Output numpy array

Definition at line 607 of file mwfiltersgui.py.

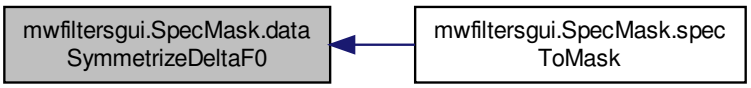
References mwfiltersgui.SpecMask.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, and mwfiltersgui.SpecMask.freqSort().

Referenced by mwfiltersgui.SpecMask.specToMask().

Here is the call graph for this function:



Here is the caller graph for this function:



10.28.3.2 def mwfiltersgui.SpecMask.freqSort (*self, inArray, freqCol*)

Sort array by frequency (inplace).

The input array is modified inplace

Parameters

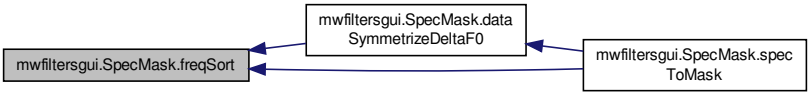
inArray	= Input numpy array.
freqCol	= Column index of frequency values.

Definition at line 631 of file mwfiltersgui.py.

References mwfiltersgui.SpecMask.BW, libcommonfunc.FrequencyTransformBP.BW, mwfiltersgui.SpecMask.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, mwfiltersgui.SpecMask.GDmaskLower, mwfiltersgui.SpecMask.GDmaskUpper, mwfiltersgui.SpecMask.GDspec, mwfiltersgui.SpecMask.S11mask, mwfiltersgui.SpecMask.S11spec, mwfiltersgui.SpecMask.S21_BW, mwfiltersgui.SpecMask.S21_F0, mwfiltersgui.SpecMask.S21inBandMask, mwfiltersgui.SpecMask.S21inbandSpecDelta, mwfiltersgui.SpecMask.S21outBandMask, and mwfiltersgui.SpecMask.S21outbandSpec.

Referenced by mwfiltersgui.SpecMask.dataSymmetrizeDeltaF0(), and mwfiltersgui.SpecMask.specToMask().

Here is the caller graph for this function:



10.28.3.3 def mwfiltersgui.SpecMask.maskSamplesOptimization (*self, FT, Nsamples*)

Create mask samples to be used as a reference in optimization.

Half the samples go to the passband and half to the rejection band (symmetrically). The frequency samples for optimization are obtained by linear interpolation of the mask data. Only the magnitude of [S] is optimized to comply with the S11 and S21 masks. Group delay in the mask file, if any, is not used in optimization,

Parameters

FT	= FrequencyTransform class instance, containing f0 and BW parameters.
Nsamples	= Total number of samples. Half go to the passband and half to the rejection band (symmetrically).

Definition at line 543 of file mwfiltersgui.py.

References mwfiltersgui.SpecMask.evalFoptim, mwfiltersgui.SpecMask.evalSoptim, mwfiltersgui.SpecMask.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, mwfiltersgui.SpecMask.freqInbandOptim, mwfiltersgui.SpecMask.freqOutBandOptim, mwfiltersgui.SpecMask.GDmaskLower, mwfiltersgui.SpecMask.GDmaskUpper, mwfiltersgui.SpecMask.NfreqInBandOptim, mwfiltersgui.SpecMask.S11mask, mwfiltersgui.SpecMask.S11optim, mwfiltersgui.SpecMask.S21_BW, mwfiltersgui.SpecMask.S21_F0, mwfiltersgui.SpecMask.S21inBandMask, mwfiltersgui.SpecMask.S21inBandOptim, mwfiltersgui.SpecMask.S21outBandMask, and mwfiltersgui.SpecMask.S21outBandOptim.

10.28.3.4 def mwfiltersgui.SpecMask.specToMask (*self, SP*)

Process specification data to obtain the mask curves.

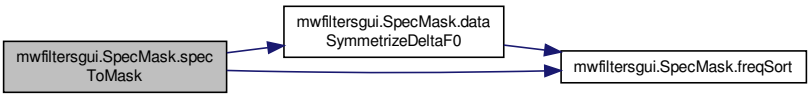
```
    The mask values are computed from specification relative values adding the relevant [S] parameter data.
    @param SP = Sparameters class instance

Apply frequency shift to mask.
@param mask = Mask numpy array
@param type = 'outer' or 'inner'.
'outer' masks expand: the right freq shift is applied to the left portion of the data and the left frequency s
'inner' masks collapse: the left freq shift is applied to the left portion of the data and the right frequency
```

Mask for S11 (return losses) as a function of frequency. Numpy array. Mask for S21 inband (insertion losses) Mask for out-of-band rejection as a function of frequency. Numpy array. Mask for group delay (upper). Mask for group delay (lower).

Definition at line 440 of file mwfiltersgui.py.

References mwfiltersgui.SpecMask.BW, libcommonfunc.FrequencyTransformBP.BW, mwfiltersgui.SpecMask.dataSymmetrizeDeltaF0(), mwfiltersgui.SpecMask.f0, libcommonfunc.Sparameters.f0, libcommonfunc.FrequencyTransformBP.f0, mwfiltersgui.SpecMask.freqFactorLeft, mwfiltersgui.SpecMask.freqFactorRight, mwfiltersgui.SpecMask.freqSort(), mwfiltersgui.SpecMask.GDmaskLower, mwfiltersgui.SpecMask.GDmaskUpper, mwfiltersgui.SpecMask.GDspec, mwfiltersgui.SpecMask.S11mask, mwfiltersgui.SpecMask.S11spec, mwfiltersgui.SpecMask.S21_BW, mwfiltersgui.SpecMask.S21_F0, mwfiltersgui.SpecMask.S21inBandMask, mwfiltersgui.SpecMask.S21inbandSpecDelta, mwfiltersgui.SpecMask.S21outBandMask, and mwfiltersgui.SpecMask.S21outbandSpec.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.29 mwfiltersgui.Stouchstone Class Reference

[S] parameters class, as read from Touchstone file.

Public Member Functions

- def __init__

Set frequency axis and S parameters, including phase and group delay, read from a Touchstone file.

10.29.1 Detailed Description

[S] parameters class, as read from Touchstone file.

It contains the minimum number of attributes to plot the [S] parameters in the results comparison window. Of course, the attributes share the same name as the corresponding ones in libcommonfunc.Sparameters class.

Definition at line 664 of file mwfiltersgui.py.

10.29.2 Constructor & Destructor Documentation

10.29.2.1 def mwfiltersgui.Stouchstone.__init__ (*self, fileName*)

Set frequency axis and S parameters, including phase and group delay, read from a Touchstone file.

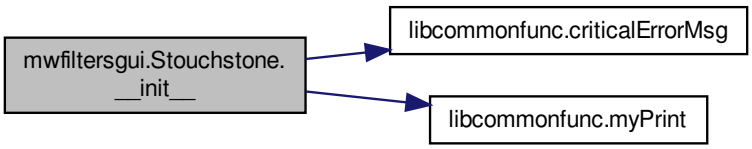
```
Touchstone is a trademark of Agilent Corporation
@param fileName = File Name
```

Frequency axis (Hz) Parameter S_{11} Parameter S_{12} Parameter S_{21} Parameter S_{22} Phase of return losses (rad) Phase of inverse transfer function (rad) Phase of transfer function (rad) Phase of inverse return losses (rad) Frequency axis with sampling points in the middle of self.freq sampling. It is necessary to plot self.groupDelay2 Group delay for the unnormalized filter (sec), sampled at self.freq2

Definition at line 672 of file mwfiltersgui.py.

References libcommonfunc.criticalErrorMsg(), mwfiltersgui.Stouchstone.freq, libcommonfunc.Sparameters.freq, mwfiltersgui.Stouchstone.freq2, libcommonfunc.Sparameters.freq2, mwfiltersgui.Stouchstone.groupDelay2, libcommonfunc.Sparameters.groupDelay2, libcommonfunc.myPrint(), mwfiltersgui.Stouchstone.phaseS11, libcommonfunc.Sparameters.phaseS11, mwfiltersgui.Stouchstone.phaseS12, mwfiltersgui.Stouchstone.phaseS21, libcommonfunc.Sparameters.phaseS21, mwfiltersgui.Stouchstone.phaseS22, libcommonfunc.Sparameters.phaseS22, mwfiltersgui.Stouchstone.S11, libcommonfunc.Sparameters.S11, mwfiltersgui.Stouchstone.S12, libcommonfunc.Sparameters.S12, mwfiltersgui.Stouchstone.S21, libcommonfunc.Sparameters.S21, mwfiltersgui.Stouchstone.S22, and libcommonfunc.Sparameters.S22.

Here is the call graph for this function:



The documentation for this class was generated from the following file:

- mwfiltersgui.py

10.30 libcommonfunc.synthError Class Reference

Synthesis error catch.

10.30.1 Detailed Description

Synthesis error catch.

Definition at line 117 of file libcommonfunc.py.

The documentation for this class was generated from the following file:

- libcommonfunc.py

10.31 mwfiltersgui.TempVarDlg Class Reference

Widget to ask the user for temperature variation parameters.

10.31.1 Detailed Description

Widget to ask the user for temperature variation parameters.

Definition at line 206 of file mwfiltersgui.py.

The documentation for this class was generated from the following file:

- mwfiltersgui.py

Chapter 11

File Documentation

11.1 dbplot.py File Reference

Classes

- class dbplot.DbPlot

  GUI dialog to contain plot.
- class dbplot.AxisScalingDlg

  GUI dialog to contain axis scaling controls.
- class dbplot.CurveFamily

  Data structure containing info about properties of a family of curves and the curve list.
- class dbplot.EditCurvesDlg

  Dialog to select visible curves.
- class dbplot.MyPicker

  Class derived from Qwt.QwtPlotPicker to allow reimplementation of Qwt.QwtPlotPicker.trackerText()
- class dbplot.CanvasEventFilter

  Event filter for the canvas.
- class dbplot.PlotMarker

  Class for markers.
- class dbplot.AutoScaleZoomerBase

  Class that stores zoomer base.
- class dbplot.PrintFilter

  Filter to transform color of some plot items and fontsize when printing.

Packages

- namespace dbplot

  Plot data curves in multiple tabs.

Variables

- string dbplot.MAC = "qt_mac_set_native_menubar"

  Global variable useful only for MacOSX.
- string dbplot.applicationName = 'dBplot'

  Application name, to be displayed in 'About' dialog.

11.1.1 Detailed Description

Definition in file dbplot.py.

11.2 documentation.py File Reference

11.2.1 Detailed Description

Definition in file documentation.py.

11.3 libcommonfunc.py File Reference

Classes

- class libcommonfunc.parseError

  Parse error catch.
- class libcommonfunc.synthError

  Synthesis error catch.
- class libcommonfunc.licenseError

  License error catch.
- class libcommonfunc.Parameters

  Filter, synthesis and sweep parameters.
- class libcommonfunc.Sparameters

  [S] parameters class.
- class libcommonfunc.MatrixQ

  Class that contains a coupling matrix, its name, the number of non-resonant nodes, the Q of resonators and some operations.
- class libcommonfunc.CouplingMatrices

  Coupling matrices class.
- class libcommonfunc.FrequencyTransformBP

  Frequency transformation fro band pass to low pass prototype.

Packages

- namespace libcommonfunc

  Library of common functions for microwave filters synthesis.

Functions

- def libcommonfunc.setGlobalVariablesLibCommonFunc

  Function to allow the GUI set values to the global variables of libcomonfunc module.
- def libcommonfunc.readVerbose

  Returns the value of the global verbose variable.
- def libcommonfunc.pkgNameVersion

  Retrieves software package name and version from string.
- def libcommonfunc.myPrint

  Prints on standard output if console application.
- def libcommonfunc.printHeader

  Prints application header info.

- def libcommonfunc.readParamFile

  Read parameters file and return a Parameter class instance.
- def libcommonfunc.criticalErrorMsg

  Show critical error message.
- def libcommonfunc.warningMsg

  Show warning message.
- def libcommonfunc.informationMsg

  Show information message.
- def libcommonfunc.askQuestion

  Show dialog to ask question.
- def libcommonfunc.listStr

  Nicely convert list of floats to string, rounding to 'prec' significant digits.
- def libcommonfunc.complexStr

  Nicely convert complex number to string, rounding real and imaginary parts to 'prec' significant digits.
- def libcommonfunc.validate

List of Chebash parameters to ignore

chebashIgnoreList = ['n_ceros', 'n_frecpropias_eq', 'real', 'imag', 'tipo_transformacion', 'graf[1]', 'graf[2]', 'graf[3]', 'graf[5]', 'graf[6]', 'graf[7]', 'graf[8]', 'graf[9]', 'graf[10]', 'graf[11]', 'graf[12]', 'graf[13]', 'graf[14]', 'graf[15]', 'graf[16]', 'graf[17]', 'graf[18]']

- def libcommonfunc.copy

  Function to copy instances of the Parameters class (copies the self instance).
- def libcommonfunc.__str__

  Special method to print parameter class (prints the self instance).

Variables

- dictionary libcommonfunc.parameterDict

  Dictionary of parameter keywords and variable names.
- int libcommonfunc.saveSignificantDigits = 14

  Number of significant digits to save in files.
- int libcommonfunc.saveSignificantDigitsEnergy = 4

  Number of significant digits to save in energy and power results.
- libcommonfunc.mainWindow = None

  Module global variable equal either to None or to the GUI QMainWindow class instance, containing the application main window.
- libcommonfunc.verbose = False

  Module global variable equal to True if the GUI has been called with the -v flag command line option.
- libcommonfunc.nogui = False

  Module global variable equal to True if the GUI has been called with the –nogui flag command line option.
- string libcommonfunc.preGuiOutput = ''

  String containing console output generated before the GUI main window exists.
- libcommonfunc.CP = CMSPNone
- list libcommonfunc.nonSymmZeros = [ ]

  def normRealFreq(f): """ Auxiliary function """ return (f/self.f0 - self.f0/f)∗self.f0/self.BW

11.3.1 Detailed Description

Definition in file libcommonfunc.py.

11.4 libfreefilters.py File Reference

Classes

- class libfreefilters.CharPolys

  Characteristic polynomials class.

Packages

- namespace libfreefilters

  Free Filters Library to compute Butterworth, Chebyshev and Quasieliptic characteristic polynomials with minimum insertion loss.

Functions

- def libfreefilters.setGlobalVariablesLibFreeFilters

  Function to allow the GUI set values to the global variables of libcomonfunc module.
- def libfreefilters.polyOmega

  Returns $P_s(s)$ for the input polynomial $P_w(\omega)$, with $s = j\omega$ or $\omega = -js$.
- def libfreefilters.polyCheby

  Compute Chebyshev polynomials from order 0 to N.
- def libfreefilters.polyConj

  Returns the 'Paraconjugate' $P_c(s)$ of the input polynomial $P(s)$, with complex variable restricted to the imaginary axis, $s = j\omega$, and therefore $s^* = -s$.
- def libfreefilters.findEps

  This function finds the maximum of the function $g(s) = \left|\frac{NUM(s)}{\varepsilon DEN(s)}\right|$ in the imaginary axis.
- def libfreefilters.rootErrors

  Report errors in the roots position for a polynomial of the form SQF(s) = F(s)F(-s) with real coefficients.
- def libfreefilters.factorization

  Factorization of polynomial SQF(s) of order 2N as a product of two polynomials, each one of order N.
- def libfreefilters.vecSort

  Sort a vector of complex numbers so that their imaginary parts is in decreasing order.
- def libfreefilters.butterworthFilter

  Compute characteristic polynomials and characteristic constants of a Butterwoth filter.
- def libfreefilters.chebyshevFilter

  Compute characteristic polynomials and characteristic constants of a Chebyshev filter.
- def libfreefilters.quasiElipticFilter

  Compute characteristic polynomials and characteristic constants of a Quasieliptic filter.

Variables

- libfreefilters.mainWindow = None

  Module global variable equal either to None or to the GUI QMainWindow class instance, containing the application main window.
- libfreefilters.importedExtraFilters = False

  Variable set to True if the non-free libextrafilters is available.

11.4.1 Detailed Description

Definition in file libfreefilters.py.

11.5 mwfiltersgui.py File Reference

Classes

- class mwfiltersgui.EnergyDbPlot
  Class derived from DbPlot to allow adding some widgets for plotting energy and power.
- class mwfiltersgui.TempVarDlg
  Widget to ask the user for temperature variation parameters.
- class mwfiltersgui.SpecMask
  Specification mask class.
- class mwfiltersgui.Stouchstone
  [S] parameters class, as read from Touchstone file.
- class mwfiltersgui.myTableWidget
  Subclass for:
- class mwfiltersgui.MainWindow
  GUI main window class.
- class mwfiltersgui.HelpForm
  GUI dialog to display help.
- class mwfiltersgui.SetCouplingDlg
  Dialog to set the type of a coupling.
- class mwfiltersgui.MatrixEditDlg
  GUI dialog to edit coupling matrix.
- class mwfiltersgui.PredisZeorsDlg
  Dialog to select predistortion zeros.
- class mwfiltersgui.SensitivityAnalysis
  Class for sensitivity analysis.
- class mwfiltersgui.CouplingMatrixDlg
  GUI dialog to show coupling matrices.
- class mwfiltersgui.ParamEditDlg
  GUI dialog to edit synthesis parameters.

Packages

- namespace mwfiltersgui
  Graphical User Interface.

Functions

- def mwfiltersgui.__init__
  Derived class constructor.
- def mwfiltersgui.updateInputPower
  Update energy and power plots after inputPower value has been changed by the user.
- def mwfiltersgui.updateExcitation
  Update energy and power plots after excitation value has been changed by the user.
- def mwfiltersgui.appendReverseData
  Set YData for reverse excitation.
- def mwfiltersgui.replaceReverseData
  Replace YData for reverse excitation.
- def mwfiltersgui.sizeHint
- def mwfiltersgui.runSynthesis
  Runs parameter validation, filters synthesis and results plots.
- def mwfiltersgui.usage
  Print error message and command line help.

Variables

- string mwfiltersgui.MAC = "qt_mac_set_native_menubar"

  Global variable useful only for Mac OS X.
- mwfiltersgui.importedExtraFilters = False

  Variable set to True if the non-free libextrafilters is available.
- mwfiltersgui.importedLossyFilters = False

  Variable set to True if the non-free liblossyfilters is available.
- string mwfiltersgui.applicationName = "Microwave filters GUI"

  Program name as run by the user.
- list mwfiltersgui.fileName = args[0]

  Name of the parameters file to process.
- tuple mwfiltersgui.P = readParamFile(fileName)

  Parameters class instance containing filter and synthesis parameters read from the parameters file.
- tuple mwfiltersgui.FT = FrequencyTransformBP(P)

  Create frequency transform instance.
- string mwfiltersgui.stFloatPoint = r'((\d+\.?\d*|\d*\.?\d+)([Ee][-+]?\d+)?)'

  Create validator strings and regular expressions.
- tuple mwfiltersgui.app = QApplication(QtArgs)

  QApplication class instance, containing our application.

11.5.1 Detailed Description

Todo Linux executables created with cx_freeze report sys.stdout.encoding equal to None.

Check end-of-line in a Mac computer. It should work if platform.system() returns something different than 'Windows', 'Microsoft', 'Vista' or 'Linux'.

To avoid program crash when trying to write non-ascii characters, these characters are replaced by '?'. It would be better to enforce writting uft-8 to stdout regardless of sys.stdout.encoding .

Todo Possibility to change styles with a combobox. Not obvious in pyQt.

Definition in file mwfiltersgui.py.

Index