Agilent Technologies

Advanced Design System 2002

Expressions, Measurements, and Simulation Data Processing

February 2002

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms that apply to this software product is available upon request from your Agilent Technologies representative.

Restricted Rights Legend

Use, duplication or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DoD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Agilent Technologies
395 Page Mill Road
Palo Alto, CA 94304 U.S.A.

Contents

Chapter 1: Introduction to Expressions and Functions

This document describes the expressions and functions that are available within Advanced Design System. These functions or equations are divided into two distinct categories based on their roles in ADS. Although there is an overlap among many of the more commonly used functions, they are derived from separate sources, evaluated by ADS at different times, and can have subtle differences in their usages. Thus, they need to be considered separately. Refer to Figure 1-1 in this chapter for an overview of how ADS evaluates and uses Simulator Expressions and Measurement Equations.

Simulator Expressions

The first category of equations or functions are the ones used *internally* during *simulation runtime*, known as **Simulator Expressions** or sometimes as **VarEqn functions**. These expressions or functions can be entered into the program by means of the VarEqn component or used in place of a parameter for any component: for example in a resister, R=sin5. These functions are evaluated at the start of simulation. If a term is undefined at the start of simulation, such as $R=S_{11}$, where the results of S_{11} will not be known until the simulation is complete, an error will be returned.

For information on the general use of VarEqn components, place a VarEqn component on a Schematic, double click it, and then click the *Help* button in the Component dialog box. Or from any ADS Window choose Help > Topics and Index > Components > Circuit Components > Introduction and Simulation Components > Chapter 1, Introduction > VarEqn.

For a list and description of the ADS Simulator Expressions, refer to Chapter 4, Simulator Expression Reference.

Measurement Equations

The second category of equations are the ones used during simulation post processing, known as *Measurement Equations* or *MeasEqn* for short. These are entered into the program by means of the *MeasEqn* component, available on the Simulation palettes in an Analog/RF Systems Schematic window (such as

Simulation-AC or Simulation-Envelope) or from the Controllers palette in a Signal Processing Schematic window.

Many of the more commonly used measurement items are built in, and are found in the palettes of the appropriate simulator components. Many common expressions are included as measurements, which makes it easy for beginning users to use the system. To make simulation and analyses convenient, all the measurement items, including the built-in items, can be edited to meet specific requirements. Underlying each measurement is a *function*; the functions themselves are available for modification. Moreover, it is also possible for you to write entirely new measurements and functions.

The measurement items and their underlying expressions are based on AEL, ADS's Application Extension Language. Consequently, they can serve a dual purpose:

- They can be used on the schematic page, in conjunction with simulations, to process the results of a simulation (this is useful, for example, in defining and reaching optimization goals). The MeasEqn items are processed after the simulation engine has finishing its task and just before the dataset is written.
- They can be used in the Data Display window to process the results of a dataset that can be displayed graphically. Here the MeasEqn items are used to post-process the data written after simulation is complete.

In either of the above cases, the same syntax is used. However, some measurements can be used on the schematic page and not the Data Display window, and vice versa. These distinctions will be noted where they occur.

For information on how to interpret the function descriptions found in the MeasEqn Function Reference, see Chapter 2, Using the MeasEqn Function Reference.

For a complete list of ADS MeasEqn functions, refer to Chapter 3, MeasEqn Function Reference.

AEL Math Functions

As stated earlier, the measurement components and their underlying expressions are based on ADS's Application Extension Language (AEL). AEL includes many math functions that can be used as part of writing AEL code for various purposes. While many of these functions overlap with the Measurement Equations, they are documented in the *AEL* manual. Refer to "Math Functions" on page 9-1 in the *AEL* manual for more information.

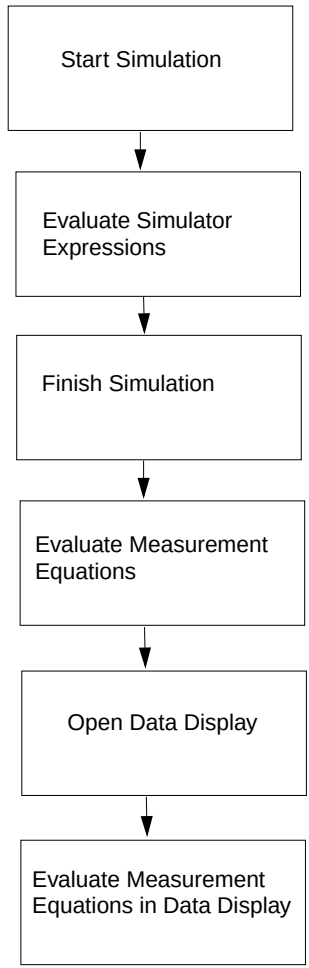

Figure 1-1. How ADS Evaluates and Uses Simulator Expressions and Measurement Equations

Chapter 2: Using the MeasEqn Function Reference

This chapter explains how to interpret the function descriptions found in Chapter 3, MeasEqn Function Reference.

The following figure illustrates how the measurement functions and mathematical functions are described. In the case of AEL measurements, the entries for “Used in” and Available as measurement component?” reads “Not applicable.”

<function name>

Purpose

States what the function does.

Synopsis

Presents the syntax of the function.

Examples

Presents typical uses of the function.

Used in

Lists applicable simulation types, if any.

Available as measurement component?

Indicates whether the measurement function is available as a component within simulation palettes (where applicable).

Defined in

Indicates whether the measurement function is defined in a script or is built in. All AEL functions are built in.

See also

Lists related functions, if any.

In addition, where applicable, a *Description* section gives detailed information about a measurement function’s behavior, including parameter defaults and exceptions.

Where examples are available in the *examples* directory, they will be listed in this section.

Manipulating Simulation Data with Equations

ADS equations are designed to manipulate data produced by the simulator. Equations may reference any simulation output and may be placed (a) in a Data Display window, or (b) in a Schematic window, by means of a MeasEqn (measurement equation) component. Ready-made measurements, found in the various simulator palettes, are simply preconfigured equations.

This description of ADS equations is accompanied by a set of example designs and data display pages. These designs can be found in the project *express_meas_prj*, in the *examples/Tutorial* directory.

Simulation Data

The expressions package has inherent support for two main simulation data features. First, simulation data are normally multidimensional. Each sweep introduces a dimension. All operators and relevant functions are designed to apply themselves automatically over a multidimensional simulation output. Second, the independent (swept) variable is associated with the data (for example, S-parameter data). This independent is propagated through expressions, so that the results of equations are automatically plotted or listed against the relevant swept variable.

Case Sensitivity

All variable names, functions names, and equation names are case sensitive in ADS expressions.

Measurements and Expressions

Refer also to *simple_meas_1.dsn* in */examples/Tutorial/express_meas_prj/networks*

Expressions are available on the schematic page by means of the MeasEqn component. Also available in various simulation palettes are preconfigured measurements. These are designed to help the user by presenting an initial equation, which can be modified to suit the particular instance.

Measurements are evaluated after a simulation is run and the results are stored in the dataset. The tag “meqn_xxx” (where xxx is a number) is placed at the beginning of all measurement results, to distinguish those results from data produced directly by the simulator.

Complex measurement equations are available for both circuit and signal processing simulations. Underlying a measurement is the same generic equations handler that is available in the Data Display window. Consequently, simulation results can be referenced directly, and the expression syntax is identical. All operators and almost all functions are available.

The expression used in an optimization goal or a yield specification is a measurement expression. It may reference any other measurement on the schematic.

It is not possible to reference a VarEqn variable in a MeasEqn equation. However, a MeasEqn equation can reference other MeasEqns, any simulation output, and any swept variable.

Variable Names

Refer also to *names_1.dsn* and *names_1.dds* in */examples/Tutorial/express_meas_prj.*

Variables produced by the simulator can be referenced in equations with various degrees of rigidity. In general a variable is defined as follows:

DatasetName.AnalysisName.AnalysisType.CircuitPath.VariableName

By default, in the Data Display window a variable is commonly referenced as follows:

DatasetName..VariableName

where the double dot “..” indicates that the variable is unique in this dataset. If a variable is referenced without a dataset name, then it is assumed to be in the current default dataset.

When the results of several analyses are in a dataset, it becomes necessary to specify the analysis name with the variable name. The double dot can always be used to pad a variable name instead of specifying the complete name.

Refer also to *names_2.dsn,* and *names_2.dds* in */examples/Tutorial/express_meas_prj.*

In most cases a dataset contains results from a single analysis only, and so the variable name alone is sufficient. The most common use of the double dot is when it is desired to tie a variable to a dataset other than the default dataset.

Refer also to *names_3.dds* in */examples/Tutorial/express_meas_prj*.

Simple Sweeps and Using “[ ]”

Refer also to *sweep_1.dsn, sweep_1.dsn* and *sweep_2.dds* in */examples/Tutorial/express_meas_prj*.

Parameter sweeps are commonly used in simulations to generate, for example, a frequency response or a set of DC IV characteristics. The simulator always attaches the swept variable to the actual data (the data often being called the “attached independent” in equations).

Often after performing a swept analysis we want to look at a single sweep point or a group of points. The sweep indexer “[ ]” can be used to do this. The sweep indexer is zero offset, meaning that the first sweep point is accessed as index 0. A sweep of n points can be accessed by means of an index that runs from 0 to n–1. Also, the what() function can be useful in indexing sweeps. Use what() to find out how many sweep points there are, and then use an appropriate index. Indexing out of range yields an invalid result.

The sequence operator can also be used to index into a subsection of a sweep. Given a parameter X, a subsection of X may be indexed as

```
a=X[start::increment::stop]
```

Because increment defaults to one,

```
a=X[start::stop]
```

is equivalent to

```
a=X[start::1::stop]
```

The “::” operator alone is the wildcard operator, so that X and X[::] are equivalent. Indexing can similarly be applied to multidimensional data. As will be shown later, an index may be applied in each dimension.

S-Parameters and Matrices

Refer also to *sparam_1.dsn* and *sparam_1.dds* in */examples/Tutorial/express_meas_prj.*

As described above, the sweep indexer "[ ]" is used to index into a sweep. However, the simulator can produce a swept matrix, as when an S-parameter analysis is performed over some frequency range. Matrix entries can be referenced as S11 through S*nm*. While this is sufficient for most simple applications, it is also possible to index matrices by using the matrix indexer "()". For example, S(1,1) is equivalent to S11. The matrix indexer is offset by one meaning the first matrix entry is X(1,1). When it is used with swept data its operation is transparent with respect to the sweep. Both indexers can be combined. For example, it is possible to access S(1,1) at the first sweep point as S(1,1)[0]. As with the sweep indexer "[ ]", the matrix indexer can be used with wildcards and sequences to extract a submatrix from an original matrix.

Matrices

Refer also to *matrix_1.dds* in /examples/Tutorial/express_meas_prj.

S-parameters above are an example of a matrix produced by the simulator. Matrices are more frequently found in signal processing applications. Mathematical operators implement matrix operations. Element-by-element operations can be performed by using the dot modified operators (.* and ./).

The matrix indexer conveniently operates over the complete sweep, just as the sweep indexer operates on all matrices in a sweep. As with scalars, the mathematical operators allow swept and nonswept quantities to be combined. For example, the first matrix in a sweep may be subtracted from all matrices in that sweep as

```
Y = X-X[0]
```

Refer also to *matrix_2.dsn* and *matrix_2.dds* in */examples/Tutorial/express_meas_prj*.

Multidimensional Sweeps and Indexing

Refer also to *multi_dim_1.dsn* and *multi_dim_1.dds* in */examples/Tutorial/express_meas_prj*.

In the previous examples we looked at single-dimensional sweeps. Multidimensional sweeps can be generated by the simulator by using multiple parameter sweeps.

Expressions are designed to operate on the multidimensional data. Functions and operators behave in a meaningful way when a parameter sweep is added or taken away. A common example is DC IV characteristics.

The sweep indexer accepts a list of indices. Up to *N* indices are used to index N-dimensional data. If fewer than *N* lookup indices are used with the sweep indexer, then wildcards are inserted automatically to the left. This is best explained by referring to the above example files.

Working with Harmonic Balance (HB) Data

Refer also to *hb_1.dds*
in */examples/Tutorial/express_meas_prj*.

Harmonic balance analysis produces complex voltages and currents as a function of frequency or harmonic number. A single analysis produces 1-dimensional data. Individual harmonic components can be indexed by means of "[ ]." Multitone HB also produces 1-dimensional data. Individual harmonic components can be indexed as usual by means of "[ ]." However. the "mix" function provides as convenient way to select a particular mixing component (for a list of functions, refer to *List of Functions*).

Working with Transient Data

Refer also to *tran_1.dsn* and *tran_1.dds*
in */examples/Tutorial/express_meas_prj*.

Transient analysis produces real voltages and currents as a function of time. A single analysis produces 1-dimensional data. Sections of time-domain waveforms can be indexed by using a sequence within "[ ]."

Working with Envelope Data

Refer also to *env_1.dds*
in */examples/Tutorial/express_meas_prj*.

Envelope analysis produces complex frequency spectra as a function of time. A single envelope analysis can produce 2-dimensional data where the outermost independent variable is time and the innermost is frequency or harmonic number. Indexing can be used to look at a harmonic against time, or a spectrum at a particular time index.

if-then-else Construct

Refer also to *if_then_else_1.dds* in */examples/Tutorial/express_meas_prj.*

The if-then-else construct provides an easy way to apply a condition on a per-element basis over a complete multidimensional variable. It has the following syntax:

A = **if** (*condition*) **then** *true_expression* **else** *false_expression*

Condition, true_expression, and false_expression are any valid expressions. The dimensionality and number of points in these expressions follow the same matching conditions required for the basic operators.

Multiple nested if-then-else constructs can also be used:

A = **if** (*condition*) **then** *true_expression* **else*****if*** *(condition2)* **then** *true_expression* **else** *false_expression*

The type of the result depends on the type of the true and false expressions. The size of the result depends on the size of the condition, the true expression, and the false expression.

The if-then-else construct can be used in a MeasEqn component on a schematic. It has the following syntax:

A = **if** (*condition*) **then** *true_expression* **else** *false_expression*

Generating Data

Refer also to *gen_1.dds*
in */examples/Tutorial/express_meas_prj*.

The simulator produces scalars and matrices. When a sweep is being performed it can produce scalars and matrices as a function of a set of swept variables. It is also possible to generate data by using expressions. Two operators can be used to do this. The first is the sweep generator "[ ]," and the second is the matrix generator "{ }." These operators can be combined in various ways to produce swept scalars and matrices. The data can then be used in the normal way in other expressions. The operators can also be used to concatenate existing data, which can be very useful when combined with the indexing operators.

Operator Precedence

Expressions are evaluated from left to right, unless there are parentheses. Operators are listed from higher to lower precedence. Operators on the same line have the same precedence. For example, a+b*c means a+(b*c), because * has a higher precedence than +. Similarly, a+b-c means (a+b)–c, because + and – have the same precedence (and because + is left-associative).

The operators !, &&, and || work with the logical values. The operands are tested for the values TRUE and FALSE, and the result of the operation is either TRUE or FALSE. In AEL a logical test of a value is TRUE for non-zero numbers or strings with non-zero length, and FALSE for 0.0 (real), 0 (integer), NULL or empty strings. Note that the right hand operand of && is only evaluated if the left hand operand tests TRUE, and the right hand operand of | | is only evaluated if the left hand operand tests FALSE.

The operators >=, <=, >, <, ==, != , AND, OR, EQUALS, and NOT EQUALS also produce logical results, producing a logical TRUE or FALSE upon comparing the values of two expressions. These operators are most often used to compare two real numbers or integers. These operators operate differently in AEL than C with string expressions in that they actually perform the equivalent of *strcmp()* between the first and second operands, and test the return value against 0 using the specified operator.

Table 2-1. Operator Precedence

Operator	Name	Example
( )	function call, matrix indexer	foo(expr_list) X(expr,expr)
[ ]	sweep indexer, sweep generator	X[expr_list] [expr_list]
{ }	matrix generator	{expr_list}
**	exponentiation	expr**expr
!	not	!expr
* / .* ./	multiply divide element-wise multiply element-wise divide	expr * expr expr / expr expr .* expr expr ./ expr
+ -	add subtract	expr + expr expr - expr

Table 2-1. Operator Precedence (continued)

Operator	Name	Example
::	sequence operator wildcard	exp::expr::expr start::inc::stop ::
< <= > >=	less than less than or equal to greater than greater than or equal to	expr < expr expr <= expr expr > expr expr >= expr
== !=	equal not equal	expr == expr expr != expr
&&	logical and	expr && expr
||	logical or	expr || expr

Built-in Constants

The following constants can be used in expressions.

Table 2-2. Built-in Constants

Constant	Description	Value
PI (also pi)	p	3.1415926535898
e	Euler's constant	2.718281822
ln10	natural log of 10	2.302585093
boltzmann	Boltzmann's constant	1.380658e–23 J /degree K
qelectron	electron charge	1.60217733e–19 C
planck	Planck's constant	6.6260755e-34 J–sec
c0	Speed of light in free space	2.99792e+08 m/sec
e0	Permittivity of free space	8.85419e–12 F/m
u0	Permeability of free space	12.5664e–07 H/m
i, j	sqrt(–1)	1i

Budget Measurement Analysis

Budget analysis determines the signal and noise performance for elements in the top-level design. Therefore, it is a key element of system analysis. Budget measurements show performance at the input and output pins of the top-level system elements. This enables the designer to adjust, for example, the gains at various components, to reduce nonlinearities. These measurements can also indicate the degree to which a given component can degrade overall system performance.

Budget measurements are performed upon data generated during a special mode of circuit simulation. AC and HB simulations are used in budget mode depending upon if linear or nonlinear analysis is needed for a system design. The controllers for these simulations have a flag called, "*OutputBudgetIV*" which must be set to "*yes*" for the generation of budget data. Alternatively, the flag can be set by editing the AC or HB simulation component and selecting the "*Perform Budget simulation*" button on the Parameters tab.

Budget data contains signal voltages and currents, and noise voltages at every node in the top level design. Budget measurements are functions that operate upon this data to characterize system performance parameters including gain, power, and noise figure. These functions use a constant reference impedance for all nodes for calculations. By default this impedance is 50 Ohms. The available source power at the input network port is assumed to equal the incident power at that port.

Budget measurements are available in the schematic and the data display windows. The budget functions can be evaluated by placing the budget components from Simulation-AC or Simulation-HB palettes on the schematic. The results of the budget measurements at the terminal(s) are sorted in ascending order of the component names. The component names are attached to the budget data as additional dependent variables. To use one of these measurements in the data display window, first reference the appropriate data in the default dataset, and then use the equation component to write the budget function.

Note The budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

Frequency Plan

A frequency plan of the network is determined for budget mode AC and HB simulations. This plan tracks the reference carrier frequency at each node in a

network. When performing HB budget, there may be more than one frequency plan in a given network. This is the case when double side band mixers are used. Using this plan information, budget measurements are performed upon selected reference frequencies, which can differ at each node. When mixers are used in an AC simulation, be sure to set the “Enable AC frequency conversion” option on the controller, to generate the correct plan.

The budget measurements can be performed on arbitrary networks with multiple signal paths between the input and output ports. As a result, the measurements can be affected by reflection and noise generated by components placed between the terminal of interest and the output port on the same signal path or by components on different signal paths.

Reflection and Backward-Traveling Wave Effects

The effects of reflections and backward-traveling signal and noise waves generated by components along the signal path can be avoided by inserting a forward-traveling wave sampler between the components. A forward-traveling wave sampler is an ideal, frequency-independent directional coupler that allows sampling of forward-traveling voltage and current waves

This sampler can be constructed using the equation-based linear three-port S-parameter component. To do this, set the elements of the scattering matrix as follows: $S12 = S21 = S31 = 1$, and all other $Sij = 0$. The temperature parameter is set to -273.16 deg C to make the component noiseless. A noiseless shunt resistor is attached to port 3 to sample the forward-traveling waves.

MeasEqn

By placing a MeasEqn (simulation measurement equation) component on the schematic, you can write an equation that can be evaluated, following a simulation, and displayed in a Data Display window.

Instance Name

Displays and edits the name of the MeasEqn component. You can place more than one MeasEqn component on the schematic.

Select Parameter

Selects an equation for editing.

Add Adds an equation to the Select Parameter field.

Cut Deletes an equation from the Select Parameter field.

Paste Copies an equation that has been cut and places it in the Select Parameter field.

Meas

Enter your equation in this field.

Display parameter on schematic

Displays or hides a selected equation on the schematic.

Component Options

Refer to Component Options.

User-Defined Functions

By writing some AEL code, you can define your own custom functions. A file called *user_defined_fun.ael* has been set aside for this purpose in the directory *expressions/ael/*. By looking at the other *_fun.ael* files, you can see how to write your code. You can have as many functions as you like in this one file, and they will all be compiled upon program start-up. If you have a large number of functions to define, you may want to organize them into more than one file. In this case, in order to have your functions all compile, you will need to include a line such as

```
load("more_user_defined_fun.ael");
```

in the *expressions_init.ael* file in the same directory.

Chapter 3: MeasEqn Function Reference

This chapter lists and describes the MeasEqn functions that are available within the Advanced Design System. These functions include mathematical functions such as those for matrix conversion, trigonometry, absolute value, and the like. They also include functions specific to simulation, such as S-parameter functions and budget measurement components.

The tables in this chapter indicate whether or not a function is available as a built-in measurement from a palette in the design window. Although they have been designed to make simulation convenient, the built-in measurement items can also be edited by the user to meet specific requirements.

For more information on how to interpret this material, see Chapter 2, Using the MeasEqn Function Reference.

abcdtoh

Purpose

Performs ABCD-to-H conversion

Synopsis

abcdtoh(A)

where A is the chain (ABCD) matrix of a 2-port network.

Examples

h=abcdtoh(a)

Used in

Small-signal and large-signal S-parameter simulations.

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

htoabcd, abcdtoh, abcdtoz

Description

This measurement transforms the chain (ABCD) matrix of a 2-port network to a hybrid matrix.

abcdtos

Purpose

Performs ABCD-to-S conversion

Synopsis

abcdtos(A, zRef)

where A is the chain (ABCD) matrix of a 2-port network and zRef is a reference impedance.

Examples

s=abcdtos(a, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

stoabcd, abcdtoy, abcdtoz

Description

This measurement transforms the chain (ABCD) matrix of a 2-port network to a scattering matrix.

abcdtoy

Purpose

Performs ABCD-to-Y conversion

Synopsis

abcdtoy(A)

where A is the chain (ABCD) matrix of a 2-port network.

Examples

y = abcdtoy(a)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param. Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

ytoabcd, abcdtoz, abcdtoh

Description

This measurement transforms the chain (ABCD) matrix of a 2-port network to an admittance matrix.

abcdtoz

Purpose

Performs ABCD-to-Z conversion

Synopsis

abcdtoz(A)

where A is the chain (ABCD) matrix of a 2-port network.

Examples

z = abcdtoz(a)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

ztoabcd, abcdtoy, abcdtoh

Description

This measurement transforms the chain (ABCD) matrix of a 2-port network to impedance matrix.

abs

Returns the absolute value of a real number or an integer. In the case of a complex number, the abs function:

- accepts one complex argument.
- returns a positive real number.
- returns the magnitude of its complex argument.

Synopsis

y = abs(x)

where x is an integer or real number.

Examples

a = abs(–45)
returns 45

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cint, exp, float, int, log, log10, pow, sgn, sqrt

acos

Purpose

Returns the inverse cosine, or arc cosine, in radians, of a real number or integer

Synopsis

y = acos(x)

where x is an integer or real number, and y ranges from 0 to pi.

Examples

a = acos(–1)
returns 3.142

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

asin, atan, atan2

acpr_vi

Purpose

Computes the adjacent-channel power ratio following a Circuit Envelope simulation

Synopsis

ACPRvals=acpr_vi(voltage, current, mainCh, lowerAdjCh, upperAdjCh, winType, winConst)

where

voltage is the single complex voltage spectral component (for example, the fundamental) across a load versus time;

current is the single complex current spectral component into the same load versus time;

mainCh is the two-dimensional vector defining the main channel frequency limits (as an offset from the single voltage and current spectral component);

lowerAdjCh is the two-dimensional vector defining the lower adjacent-channel frequency limits (as an offset from the single voltage and current spectral component);

upperAdjCh is the two-dimensional vector defining the upper adjacent channel frequency limits (as an offset from the single voltage and current spectral component);

winType is an optional window type and must be one of the following: Kaiser, Hamming, Gaussian, 8510, or NoWindow (leaving this field blank is the equivalent of NoWindow); and

winConst is an optional parameter that affects the shape of the applied window. The default window constants are:

Kaiser: 7.865

Hamming: 0.54

Gaussian: 0.75

8510: 6 (The 8510 window is the same as a Kaiser window with a window constant of 6.)

Examples

Example equations

VloadFund = vload[1]
IloadFund = iload.i[1]
mainlimits = {–16.4 kHz, 16.4 kHz}
UpChlimits = {mainlimits + 30 kHz}
LoChlimits = {mainlimits – 30 kHz}
TransACPR = acpr_vi(VloadFund, IloadFund, mainlimits, LoChlimits, UpChlimits, "Kaiser")

where vload is the named connection at a load, and iload.i is the name of the current probe that samples the current into the node. The {} braces are used to define vectors, and the upper channel limit and lower channel limit frequencies do not need to be defined by means of the vector that defines the main channel limits.

Example file

examples/RF_Board/NADC_PA_prj/NADC_PA_ACPRtransmitted.dds

Used in

Adjacent-channel power computations

Available as measurement component?

Equations listed under Description can be entered by means of a MeasEqn component in the Schematic window. There is no explicit ACPR measurement function.

Defined in

hpeesof/expressions/ael/digital_wireless_fun.ael

See also

acpr_vr, channel_power_vi, channel_power_vr, relative_noise_bw

Description

The user must supply a single complex voltage spectral component (for example, the fundamental) across a load versus time and a single complex current spectral component into the same load. The user must also supply the upper and lower adjacent-channel and main-channel frequency limits, as offsets from the spectral component frequency of the voltage and current. These frequency limits must be

entered as two-dimensional vectors. An optional window and window constant may also be supplied, for use in processing nonperiodic data.

acpr_vr

Purpose

Computes the adjacent-channel power ratio following a Circuit Envelope simulation

Synopsis

ACPRvals=acpr_vr(voltage, resistance, mainCh, lowerAdjCh, upperAdjCh, winType, winConst)

where

voltage is the single complex voltage spectral component (for example, the fundamental) across a resistive load versus time;

resistance is the load resistance in ohms (default is 50 ohms);

mainCh is the two-dimensional vector defining the main-channel frequency limits (as an offset from the single voltage and current spectral component);

lowerAdjCh is the two-dimensional vector defining the lower adjacent-channel frequency limits (as an offset from the single voltage spectral component);

upperAdjCh is the two-dimensional vector defining the upper adjacent-channel frequency limits (as an offset from the single voltage spectral component);

winType is an optional window type and must be one of the following: Kaiser, Hamming, Gaussian, 8510, or NoWindow (leaving this field blank is the equivalent of NoWindow); and

winConst is an optional parameter that affects the shape of the applied window. The default window constants are:

Kaiser: 7.865

Hamming: 0.54

Gaussian: 0.75

8510: 6 (The 8510 window is the same as a Kaiser window with a window constant of 6.)

Examples

Example equations

Vfund = vOut[1]
mainlimits = {–(1.2288 MHz/2), (1.2288 MHz/2)}

UpChlimits = {885 kHz, 915 kHz}
LoChlimits = {–915 kHz, –885 kHz}
TransACPR = acpr_vr(VloadFund, 50, mainlimits, LoChlimits, UpChlimits, "Kaiser")

where vOut is the named connection at a resistive load. The {} braces are used to define vectors.

Note vOut is a named connection on the schematic. Assuming that a Circuit Envelope simulation was run, vOut is output to the dataset as a two-dimensional matrix. The first dimension is time, and there is a value for each time point in the simulation. The second dimension is frequency, and there is a value for each fundamental frequency, each harmonic, and each mixing term in the analysis, as well as the baseband term.

vOut[1] is the equivalent of vOut[::, 1], and specifies all time points at the lowest non-baseband frequency (the fundamental analysis frequency, unless a multitone analysis has been run and there are mixing products). For former MDS users, the notation "vOut[*, 2]" in MDS corresponds to the ADS notation of "vOut[1]".

Example file

examples/Tutorial/ModSources_prj/IS95RevLinkSrc.dds

Used in

Adjacent-channel power computations

Available as measurement component?

Equations listed under Description can be entered by means of a MeasEqn component in the Schematic window. There is no explicit ACPR measurement function.

Defined in

hpeesof/expressions/ael/digital_wireless_fun.ael

See also

acpr_vi, channel_power_vi, channel_power_vr, relative_noise_bw

Description

The user must supply a single complex voltage spectral component (for example, the fundamental) across a resistive load versus time and the load resistance. The user must also supply the upper and lower adjacent-channel and main-channel frequency limits, as offsets from the spectral component frequency of the voltage. These frequency limits must be entered as two-dimensional vectors. An optional window and window constant may also be supplied, for use in processing nonperiodic data.

add_rf

Purpose

Returns the sum of two Timed Complex Envelope signals defined by the triplet in-phase (real or I(t)) and quadrature-phase (imaginary or Q(t)) part of a modulated carrier frequency(Fc)

Synopsis

y = add_rf(T1, T2)

where T1 and T2 are two Timed Complex Envelope signals at two distinct carrier frequencies Fc1 and Fc2.

Examples

y= add_rf(T1, T2)

Used in

Signal processing designs that output Timed Signals using Timed Sinks

Available as measurement component?

Not applicable

Defined in

AEL, signal_proc_fun.ael

See also

Not applicable

Description

This equation determines the sum of two Timed Complex Envelope at a new carrier frequency Fc3. Given Fc1 and Fc2 as the carrier frequencies of the two input waveforms, the output carrier frequency Fc3 will be the greater of the two.

asin

Purpose

Returns the inverse sine, or arc sine, in radians, of a real number or integer

Synopsis

y = asin(x)

where x is an integer or real number and y ranges from –pi/2 to pi/2.

Examples

a = asin(–1)
returns –1.571

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

acos, atan, atan2

atan

Purpose

Returns the inverse tangent, or arc tangent, in radians, of a real number or integer

Synopsis

y = atan(x)

where x is a real number or integer and y ranges from -pi/2 to pi/2.

Examples

a = atan(–1)
returns –0.785

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

acos, asin, atan2

atan2

Purpose

Returns the inverse tangent, or arc tangent, of the rectangular coordinates y and x

Synopsis

w= atan2(y, x)

where y and x are integer or real number coordinates, and w ranges from –pi to pi. atan2(0, 0) defaults to –pi/2.

Examples

a = atan2(1, 0)
returns 1.571

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

acos, asin, atan

ber_pi4dqpsk

Purpose

Returns the symbol probability of error versus signal-to-noise ratio per bit for pi/4 DQPSK modulation

Synopsis

data = ber_pi4dqpsk(vIn, vOut, symRate, noise{, samplingDelay, rotation, tranDelay, pathDelay})

where

vIn and *vOut* are the complex envelope voltage signals at the input and output nodes, respectively, *symRate* is the symbol rate (real) of the modulation signal, and *noise* is the RMS noise vector.

The remaining arguments are optional and will be calculated if not specified by the user: *pathDelay* is the delay from input to output in seconds, rotation is the carrier phase in radians, and *samplingDelay* is the clock phase in seconds.

tranDelay is an optional time in seconds that causes this time duration of symbols to be eliminated from the bit error rate calculation. Usually the filters in the simulation have transient responses, and the bit error rate calculation should not start until these transient responses have finished.

Note that ber_pi4dqpsk returns a list of data:

data[0]= symbol probability of error versus Eb / N0

data[1]= path delay in seconds

data[2]= carrier phase in radians

data[3]= clock phase in seconds

data[4]= complex(Isample, Qsample)

Examples

y= ber_pi4dqpsk(videal[1], vout[1], 0.5e6, {0.1::-0.01::0.02})

Used in

Circuit Envelope simulation, Data Flow simulation

Available as measurement component?

Not applicable

Defined in

AEL, digital_wireless_fun.ael

See also

ber_qpsk, constellation

Description

The arguments *vIn* and *vOut* usually come from a circuit envelope simulation, while *noise* usually comes from a harmonic balance simulation, and is assumed to be additive white Gaussian. It can take a scalar or vector value. The function uses the quasi-analytic approach for estimating BER: for each symbol, Eb / N0 and BER are calculated analytically; then the overall BER is the average of the BER values for the symbols.

ber_qpsk

Purpose

Returns the symbol probability of error versus signal-to-noise ratio per bit for QPSK modulation

Synopsis

data = ber_qpsk(vIn, vOut, symRate, noise{, samplingDelay, rotation, tranDelay, pathDelay})

where

vIn and *vOut* are the complex envelope voltage signals at the input and output nodes, respectively, *symRate* is the symbol rate (real) of the modulation signal, and *noise* is the RMS noise vector.

The remaining arguments are optional and will be calculated if not specified by the user: *pathDelay* is the delay from input to output in seconds, rotation is the carrier phase in radians, and *samplingDelay* is the clock phase in seconds.

tranDelay is an optional time in seconds that causes this time duration of symbols to be eliminated from the bit error rate calculation. Usually the filters in the simulation have transient responses, and the bit error rate calculation should not start until these transient responses have finished.

Note that ber_qpsk returns a list of data:

data[0]= symbol probability of error versus Eb / N0

data[1]= path delay in seconds

data[2]= carrier phase in radians

data[3]= clock phase in seconds

data[4]= complex(Isample, Qsample)

Examples

y= ber_qpsk(videal[1], vout[1], 1e6, {0.15::-0.01::0.04})

Used in

Circuit Envelope simulation, Data Flow simulation

Available as measurement component?

Not applicable

Defined in

AEL, digital_wireless_fun.ael

See also

ber_pi4dqpsk, constellation

Description

The arguments *vIn* and *vOut* usually come from a circuit envelope simulation, while *noise* usually comes from a harmonic balance simulation, and is assumed to be additive white Gaussian. It can take a scalar or vector value. The function uses the quasi-analytic approach for estimating BER: for each symbol, Eb / N0 and BER are calculated analytically; then the overall BER is the average of the BER values for the symbols.

bud_freq

Purpose

Returns the frequency plan of a network

Synopsis

bud_freq({freqIn, pinNumber, “simName”})

or

bud_freq(planNumber{, pinNumber})

This function is used in AC and HB simulations with the budget parameter turned on. For AC, the options are to pass no parameters, or the input source frequency freqIn, for the first parameter if a frequency sweep is performed. freqIn can still be passed if no sweep is performed, table data is just formatted differently. The first argument must be a real number for AC data and the second argument is an integer, used optionally to choose pin references.

Note To use bud_freq() in AC simulation, the AC controller FreqConversion flag must be set to “yes”.

When using this function with HB data, the planNumber is required. The planNumber is an integer which represents the chosen frequency plan.

For both analyses, the second parameter is the pinNumber, which is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the frequency plan displayed references pin 1 of each element; otherwise, the frequency plan is displayed for all pins of each element. (Note that this means it is not possible to select only pin 2 of each element, for example.) By default, the frequency plan is displayed for pin 1 of each element.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Examples

x = bud_freq()
Returns frequency plan for AC analysis.

x = bud_freq(1MHz)
Returns frequency plan for frequency swept AC analysis. By passing the value of

1MHz the plan is returned for the subset of the sweep, when the source value is 1MHz

x = bud_freq(2)
For HB, returns a selected frequency plan, 2, with respect to pin 1 of every network element.

Used in

AC and harmonic balance simulations

Available as measurement component?

BudFreq

Defined in

AEL, budget_fun.ael

See also

Not applicable

Description

When a frequency sweep is performed in conjunction with AC, the frequency plan of a particular sweep point can be chosen.

For HB, this function determines the fundamental frequencies at the terminal(s) of each component, thereby given the entire frequency plan for a network. Sometimes more than one frequency plan exists in a network. For example when double sideband mixers are used. This function gives the user the option of choosing the frequency plan of interest.

Note that a negative frequency at a terminal means that a spectral inversion has occurred at the terminal. For example, in frequency-converting AC analysis, where vIn and vOut are the voltages at the input and output ports, respectively, the relation may be either vOut=alpha*vIn if no spectral inversion has occurred, or vOut=alpha*conj(vIn) if there was an inversion. Inversions may or may not occur depending on which mixer sidebands one is looking at.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_gain

Purpose

Returns budget transducer-power gain

Synopsis

bud_gain(vIn, iIn{, Zs, Plan, pinNumber, “simName”})

or

bud_gain(“SourceName”, {SrcIndx, Zs, Plan})

where vIn and iIn are the input voltage and the input current (flowing into the input port), respectively. “SourceName” is the component name at the input port, and SrcIndx is the frequency index that corresponds to the source frequency to determine which frequency to use from a multitone source as the reference signal. The input source port impedance Zs is an optional parameter. If not specified, Zs is set to 50.0 ohms. Plan is the number of the selected frequency plan, which is only needed for HB.

Note that for AC simulation, both the SrcIndx and Plan arguments must not be specified; these are for HB only.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Examples

x = bud_gain(PORT1.t1.v, PORT1.t1.i)

or

x = bud_gain(“PORT1”)

y= bud_gain(PORT1.t1.v, PORT1.t1.i, 75)

or

y= bud_gain(“PORT1”, , 75., 1)

z = bud_gain(PORT1.t1.v[3], PORT1.t1.i[3], , 1)

or

z= bud_gain(“PORT1”, 3, , 1)

Used in

AC and harmonic balance simulations

Available as measurement component?

BudGain

Defined in

AEL, budget_fun.ael

See also

bud_gain_comp

Description

This is the power gain (in dB) from the input port to the terminal(s) of each component, looking into that component. Power gain is defined as power delivered to the resistive load minus the power available from the source. Note that the fundamental frequency at different pins can be different. If vIn and iIn are passed directly, one may want to use the index of the frequency sweep explicitly to reference the input source frequency.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_gain_comp

Purpose

Returns budget gain compression at fundamental frequencies as a function of power

Synopsis

bud_gain_comp(vIn, iIn{, Zs, Plan, freqIndex, pinNumber, “simName”})

or

bud_gain_comp(“SourceName”, SrcIndx{, Zs, Plan, freqIndex, pinNumber, “simName”})

where vIn and iIn are the input voltage and the input current (flowing into the input port), respectively. SrcIndx is the frequency index that corresponds to the source frequency to determine which frequency to use from a multitone source as the reference signal. Zs is an optional input port impedance whose default value is 50.0 ohms. Plan is the number of the selected frequency plan, which is only needed for HB.

If Plan is not selected, the gain compression is calculated at the harmonic frequency selected by freqIndex

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_gain_comp(PORT1.t1.v[3], PORT1.t1.i[3], , 1)
x = bud_gain_comp(“PORT1”, 3, , 1)
returns the gain compression at the fundamental frequencies as a function of power.

y= bud_gain_comp(PORT1.t1.v[3], PORT1.t1.i[3], , , 1)
y= bud_gain_comp(“PORT1”, 3, , , 1)
returns the gain compression at the second harmonic frequency as a function of power.

Used in

Harmonic balance simulation with sweep

Available as measurement component?

BudGainComp

Defined in

AEL, budget_fun.ael

See also

bud_gain

Description

This is the gain compression (in dB) at the given input frequency from the input port to the terminal(s) of each component, looking into that component. Gain compression is defined as the small signal linear gain minus the large signal gain. Note that the fundamental frequency at each element pin can be different by referencing the frequency plan. A power sweep of the input source must be used in conjunction with HB. The first power sweep point is assumed to be in the linear region of operation.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_gamma

Purpose

Returns the budget reflection coefficient

Synopsis

bud_gamma({Zref, Plan, pinNumber, “simName”})

where Zref is the reference impedance, set to 50.0 ohms by default. Plan is the number of the selected frequency plan, which is only needed for HB.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Examples

x = bud_gamma()
returns reflection coefficient at all frequencies.

y = bud_gamma(75, 1)
returns reflection coefficient at reference frequencies in plan 1

Used in

AC and harmonic balance simulations

Available as measurement component?

BudGamma

Defined in

AEL, budget_fun.ael

See also

bud_vswr

Description

This is the complex reflection coefficient looking into the terminal(s) of each component. Note that the fundamental frequency at different pins can in general be

different, and therefore values are given for all frequencies unless a Plan is referenced.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_ip3_deg

Purpose

Returns the budget third-order intercept point degradation

Synopsis

bud_ip3_deg(vOut, LinearizedElement, fundFreq, imFreq{, zRef})

where vOut is the signal voltage at the output, LinearizedElement is the variable containing the names of the linearized components, fundFreq and imFreq are the harmonic frequency indices for the fundamental and intermodulation frequencies, respectively, and zRef is the reference impedance, set to 50.0 ohms by default.

Example

y=bud_ip3_deg(vOut, LinearizedElement, {1, 0}, {2, –1})
returns the budget third-order intercept point degradation

Used in

Harmonic balance simulation with the BudLinearization Controller

Available as measurement component?

BudIP3Deg

Defined in

AEL, budget_fun.ael

See also

ip3_out, ipn

Description

This measurement returns the budget third-order intercept point degradation from the input port to any given output port. It does this by setting to linear each component in the top-level design, one at a time.

For the components that are linear to begin with, this measurement will not yield any useful information. For the nonlinear components, however, this measurement will indicate how the nonlinearity of a certain component degrades the overall system IP3. To perform this measurement, the BudLinearization Controller needs to be placed in the schematic window. If no component is specified in this controller, all components on the top level of the design are linearized one at a time, and the budget IP3 degradation is computed.

bud_nf

Purpose

Returns the budget noise figure

Synopsis

bud_nf(vIn, iIn, noisevIn{, Zs, BW, pinNumber, “simName”})

or

bud_nf(“SourceName”)

where vIn, iIn, and noisevIn are the signal voltage and current (flowing into the input port) and the noise voltage at the input port, respectively. The input port impedance and the bandwidth are optional parameters. If not specified, *Zs* and *BW* are set to 50.0 ohms and 1 Hz, respectively. BW must be set as the value of Bandwidth used on the noise page of the AC controller.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_nf(PORT1.t1.v, PORT1.t1.i, PORT1.t1.v.noise)

x = bud_nf(“PORT1”)

Used in

AC simulation

Available as measurement component?

BudNF.

Defined in

AEL, budget_fun.ael

See also

bud_nf_deg, bud_tn

Description

This is the noise figure (in dB) from the input port to the terminal(s) of each component, looking into that component. The noise analysis control parameters in the AC Simulation component must be selected: “Calculate Noise” and “Include port noise”. For the source, the parameter “Noise” should be set to yes. The noise figure is always calculated per IEEE standard definition with the input termination at 290 K.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_nf_deg

Purpose

Returns budget noise figure degradation

Synopsis

bud_nf_deg(vIn, iIn, vOut, iOut, vOut.NC.vnc, vOut.NC.name{, Zs, BW})

or

bud_nf_deg(“PORT1”, “Term1”, “vOut”)

where vIn and iIn are the voltage and current at the input port, and vOut and iOut are the voltage and current at the output port. vOut.NC.vnc and vOut.NC.name are the noise contributions and the corresponding component names at the output port, respectively. The input port impedance, bandwidth, and temperature are optional parameters.

If not specified, *Zs* and *BW* are set to 50.0 ohms and 1 Hz, respectively. BW must be set as the value of Bandwidth used on the noise page of the AC controller.

Example

x = bud_nf_deg(PORT1.t1.v, PORT1.t1.i, Term1.t1.v, Term1.t1.i, vOut.NC.vnc, vOut.NC.name)

x = bud_nf_deg(“PORT1”, ”Term1”, “vOut”)

Used in

AC simulation

Available as measurement component?

BudNFDeg

Defined in

AEL, budget_fun.ael

See also

bud_nf, bud_tn

Description

The improvement of system noise figure is given when each element is made noiseless. This is the noise figure (in dB) from the source port to a specified output

port, obtained while setting each component noiseless, one at a time. The noise analysis and noise contribution control parameters in the AC Simulation component must be selected. For noise contribution, the output network node must be labeled and referenced on the noise page in the AC Controller. Noise contributors mode should be set to “Sort by Name.” The option “Include port noise “on the AC Controller should be selected. For the source, the parameter “Noise” should be set to yes. For this particular budget measurement the AC controller parameter “OutputBudgetIV” can be set to no. The noise figure is always calculated per IEEE standard definition with the input termination at 290 K.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_noise_pwr

Purpose

Returns the budget noise power

Synopsis

bud_noise_pwr({Zref, Plan, pinNumber, “simName”})

where Zref is the reference impedance and Plan is the number of the selected frequency plan, which is only needed for HB.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_noise_pwr()
returns the noise power at all frequencies

y = bud_noise_pwr(75, 1)
returns the noise power at reference frequencies in plan 1

Used in

AC and harmonic balance simulations

Available as measurement component?

BudNoisePwr

Defined in

AEL, budget_fun.ael

See also

bud_pwr

Description

This is the noise power (in dBm) at the terminal(s) of each component, looking into the component. If Zref is not specified, the impedance that relates the signal volatage and current is used to calculate the noise power. Note that the fundamental

frequency at different pins can be different, and therefore values are given for all frequencies unless a Plan is referenced.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_pwr

Purpose

Returns the budget signal power in dBm

Synopsis

bud_pwr({Plan, pinNumber, “simName”})

where Plan is the number of the selected frequency plan, which is only needed for HB.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_pwr()
returns the signal power at all frequencies when used in AC or HB simulations

y = bud_pwr(50, 1)
returns the signal power at reference frequencies in plan 1 when used for HB simulations

Used in

AC and harmonic balance simulations

Available as measurement component?

Not applicable

Defined in

AEL, budget_fun.ael

See also

bud_noise_pwr

Description

This is the signal power (in dBm) at the terminal(s) of each component, looking into the component. Note that the fundamental frequency at different pins can be

different, and therefore values are given for all frequencies unless a Plan is referenced.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_pwr_inc

Purpose

Returns the budget incident power

Synopsis

bud_pwr_inc({Zref, Plan, pinNumber, “simName”})

where Zref is the reference impedance, set to 50.0 ohms by default. Plan is the number of the selected frequency plan, which is only needed for HB.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_pwr_inc()
returns incident power at all frequencies

y = bud_pwr_inc(75, 1)
returns incident power at reference frequencies in plan 1

Used in

AC and harmonic balance simulations

Available as measurement component?

BudPwrInc

Defined in

AEL, budget_fun.ael

See also

bud_pwr_refl

Description

This is the incident power (in dBm) at the terminal(s) of each component, looking into the component. Note that the fundamental frequency at different pins can be

different, and therefore values are given for all frequencies unless a Plan is referenced.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_pwr_refl

Purpose

Returns the budget reflected power

Synopsis

bud_pwr_refl({Zref, Plan, pinNumber, “simName”})

where Zref is the reference impedance, set to 50.0 ohms by default. Plan is the number of the selected frequency plan, which is only needed for HB.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_pwr_refl()
returns reflected power at all frequencies

y = bud_pwr_refl(75, 1)
returns reflected power at reference frequencies in plan 1

Used in

AC and Harmonic balance simulations

Available as measurement component?

BudPwrRefl

Defined in

AEL, budget_fun.ael

See also

bud_pwr_inc

Description

This is the reflected power (in dBm) at the terminal(s) of each component, looking into the component. Note that the fundamental frequency at different pins can be

different, and therefore values are given for all frequencies unless a Plan is referenced.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_snr

Purpose

Returns the budget signal-to-noise-power ratio

Synopsis

bud_snr({Plan, pinNumber, “simName”})

where Plan is the number of the selected frequency plan, which is only needed for HB.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_snr()

returns the SNR at all frequencies

or

y = bud_snr(1)
returns the SNR at reference frequencies in plan 1

Used in

AC and harmonic balance simulations

Available as measurement component?

BudSNR

Defined in

AEL, budget_fun.ael

See also

Not applicable

Description

This is the SNR (in dB) at the terminal(s) of each component, looking into that component. Note that the fundamental frequency at different pins can in general be different, and therefore values are given for all frequencies unless a Plan is referenced. The noise analysis control parameter in the AC and Harmonic Balance Simulation components must be selected. For the AC Simulation component select: “Calculate Noise” and “Include port noise.” For the source, the parameter “Noise” should be set to yes. In Harmonic Balance select the “Nonlinear noise” option.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_tn

Purpose

Returns the budget equivalent output-noise temperature

Synopsis

bud_tn(vIn, iIn, noisevIn{, Zs, BW, pinNumber, “simName”})

or

bud_tn(“SourceName”)

where vIn, iIn, and noisevIn are the signal voltage and current (flowing into the input port) and the noise voltage at the input port, respectively. The input port impedance, the bandwidth, and the temperature are optional parameters.

If not specified, *Zs* and *BW* are set to 50.0 ohms and 1 Hz, respectively. If the values of *BW* or *Temp* used in the simulation are different from their default values, be sure to use their correct values in the budget function. BW must be set as the value of Bandwidth used on the noise page of the AC controller.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_tn(PORT1.t1.v, PORT1.t1.i, PORT1.t1.v.noise)
x = bud_tn(“PORT1”)

Used in

AC simulation

Available as measurement component?

BudTN

Defined in

AEL, budget_fun.ael

See also

bud_nf, bud_nf_deg

Description

This is an equivalent output-noise temperature (in degrees Kelvin) from the input port to the terminal(s) of each component, looking into that component. The noise analysis and noise contribution control parameters in the AC Simulation component must be selected: “Calculate Noise” and “Include port noise.” For the source, the parameter “Noise” should be set to yes. The output-noise temperature is always calculated per IEEE standard definition with the input termination at 290 K.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

---

bud_vswr

Purpose

Returns the budget voltage-standing-wave ratio

Synopsis

bud_vswr({Zref, Plan, pinNumber, “simName”})

where Zref is the reference impedance, set to 50.0 ohms by default. Plan is the number of the selected frequency plan, which is only needed for HB.

pinNumber is used to choose which pins of each network element are referenced. If 1 is passed as the pinNumber, the results at pin 1 of each element are returned; otherwise, the results for all pins of each element are returned. By default, the pinNumber is set to 1.

“simName” is the simulation instance name, such as “AC1” or “HB1”, used to qualify the data when multiple simulations are performed.

Example

x = bud_vswr()
returns the vswr at all frequencies

y = bud_vswr(75, 1)
returns the vswr at reference frequencies in plan 1

Used in

AC and harmonic balance simulations

Available as measurement component?

BudVSWR

Defined in

AEL, budget_fun.ael

See also

bud_gamma

Description

This is the VSWR looking into the terminal(s) of each component. Note that the fundamental frequency at different pins can be different, and therefore values are given for all frequencies unless a Plan is referenced.

Note Remember that the budget function can refer only to the default dataset, that is, the dataset selected in the data display window.

carr_to_im

Purpose

Returns the ratio of carrier signal power to IMD power

Synopsis

carr_to_im(vOut, fundFreq, imFreq)

where vOut is the signal voltage at the output port, and fundFreq and imFreq are the harmonic frequency indices for the fundamental frequency and IMD product of interest, respectively.

Example

a = carr_to_im(out, {1, 0}, {2, –1})

Used in

Harmonic balance simulation

Available as measurement component?

CarrToIM

Defined in

AEL, rf_system_fun.ael

See also

ip3_out

Description

This measurement gives the suppression (in dB) of a specified IMD product below the fundamental power at the output port.

cdf

Purpose

Returns the cumulative distribution function

Synopsis

cdf(data, numBins, minBin, maxBin)

where x is the signal, numBins is the number of subintervals or bins used to measure CDF, and minBin and maxBin are the beginning and end, respectively, of the evaluation of the CDF.

Example

cdf(data)
cdf(data, 20)

Used in

Not applicable

Available as measurement component?

This function can only be entered by means of a Eqn component in the Data Display window. There is no measurement component in schematic window.

Defined in

AEL, statistical_fun.ael

See also

histogram, pdf, yield_sens

Description

This function measures the cumulative distribution function of a signal. The default values for minBin and maxBin are the minimum and the maximum values of the data, and numBins is set to log(numOfPts)/log(2.0) by default.

cdrange

Purpose

Returns compression dynamic range

Synopsis

cdrange(nf, inpwr_lin, outpwr_lin, outpwr)

where nf is noise figure at the output port, inpwr_lin and outpwr_lin are input and the output power, respectively, in the linear region, and outpwr is output power at 1 dB compression.

Example

a = cdrange(nf2, inpwr_lin, outpwr_lin, outpwr)

Used in

XDB simulation

Available as measurement component?

CDRange

Defined in

AEL, rf_system_fun.ael

See also

sfdr

Description

The compressive dynamic range ratio identifies the dynamic range from the noise floor to the 1-dB gain-compression point. The noise floor is the noise power with respect to the reference bandwidth.

channel_power_vi

Purpose

Computes the power (in watts) in an arbitrary frequency channel following a Circuit Envelope simulation

Synopsis

Channel_power=channel_power_vi(voltage, current, mainCh, winType, winConst)

where

voltage is the single complex voltage spectral component (for example, the fundamental) across a load versus time;

current is the single complex current spectral component into the same load versus time;

mainCh is the two-dimensional vector defining channel frequency limits (as an offset from the single voltage and current spectral component (note that these frequency limits do not have to be centered on the voltage and current spectral component frequency);

winType is an optional window type and must be one of the following: Kaiser, Hamming, Gaussian, 8510, or NoWindow (leaving this field blank is the equivalent of NoWindow); and

winConst is an optional parameter that affects the shape of the applied window. The default window constants are:

Kaiser: 7.865

Hamming: 0.54

Gaussian: 0.75

8510: 6 (The 8510 window is the same as a Kaiser window with a window constant of 6.)

Examples

Example equations

VloadFund = vload[1]
IloadFund = iload.i[1]
mainlimits = {–16.4 kHz, 16.4 kHz}

Main_Channel_Power = channel_power_vi(VloadFund, IloadFund, mainlimits, Kaiser)

where vload is the named connection at a load, and iload.i is the name of the current probe that samples the current into the node. The {} braces are used to define a vector. Note that the computed power is in watts.

Use the equation

Main_Channel_Power_dBm = 10 * log(Main_Channel_Power) + 30

to convert the power to dBm. Do not use the dBm function, which operates on voltages.

Example file

examples/RF_Board/NADC_PA_prj/NADC_PA_ACPRtransmitted.dds

Used in

Channel power computations

Available as measurement component?

Equations listed under Description can be entered by means of a MeasEqn component in the Schematic window. There is no explicit channel-power measurement function for use with Circuit Envelope data.

Defined in

hpeesof/expressions/ael/digital_wireless_fun.ael

See also

acpr_vi, acpr_vr, channel_power_vr

Description

The user must supply a single complex voltage spectral component (for example, the fundamental) across a load versus time, and a single complex current spectral component into the same load. The user must also supply the channel frequency limits, as offsets from the spectral component frequency of the voltage and current. These frequency limits must be entered as a two-dimensional vector. An optional window and window constant may also be supplied, for use in processing nonperiodic data.

channel_power_vr

Purpose

Computes the power (in watts) in an arbitrary frequency channel following a Circuit Envelope simulation

Synopsis

Channel_power=channel_power_vr(voltage, resistance, mainCh, winType, winConst)

where

voltage is the single complex voltage spectral component (for example, the fundamental) across a resistive load versus time;

resistance is the load resistance in ohms (default is 50 ohms);

mainCh is the two-dimensional vector defining channel frequency limits (as an offset from the single voltage and current spectral component (note that these frequency limits do not have to be centered on the voltage and current spectral component frequency);

winType is an optional window type and must be one of the following: Kaiser, Hamming, Gaussian, 8510, or NoWindow (leaving this field blank is the equivalent of NoWindow); and

winConst is an optional parameter that affects the shape of the applied window. The default window constants are:

Kaiser: 7.865

Hamming: 0.54

Gaussian: 0.75

8510: 6 (The 8510 window is the same as a Kaiser window with a window constant of 6.)

Example

Example equations

Vmain_fund = Vmain[1]
mainlimits = {–16.4 kHz, 16.4 kHz}
Main_Channel_Power = channel_power_vr(Vmain_fund, 50, mainlimits, Kaiser)

where Vmain is the named connection at a resistive load (50 ohms in this case.) The {} braces are used to define a vector. Note that the computed power is in watts. Use the equation

Main_Channel_Power_dBm = 10 * log(Main_Channel_Power) + 30

to convert the power to dBm. Do not use the dBm function, which operates on voltages.

Example file

examples/RF_Board/NADC_PA_prj/NADC_PA_ACPRreceived.dds

Used in

Channel power computations

Available as measurement component?

Equations listed under Description can be entered by means of a MeasEqn component in the Schematic window. There is no explicit channel-power measurement function for use with Circuit Envelope data.

Defined in

hpeesof/expressions/ael/digital_wireless_fun.ael

See also

acpr_vi, acpr_vr, channel_power_vi

Description

The user must supply a single complex voltage spectral component (for example, the fundamental) across a load versus time and the resistance of the load. The user must also supply the channel frequency limits, as offsets from the spectral component frequency of the voltage. These frequency limits must be entered as a two-dimensional vector. An optional window and window constant may also be supplied, for use in processing nonperiodic data.

chop

Purpose

Replace numbers in *x* with magnitude less than *dx* with 0

Synopsis

y=chop(x{, dx})

then y= x if mag(x)>=mag(dx)

and y=0 if mag(x)<mag(dx)

dx is optional, default is 1e–10.

Actually this function is more complicated; it acts independently on the real and complex components of x, comparing each to mag(dx)

Example

chop(1)
1

chop(1e–12)
0

chop(1+1e–12i)
1+0i

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

AEL, elementary_fun.ael

See also

None

chr

Purpose

Returns the character representation of an integer

Synopsis

y = chr(x)

where x is a valid ASCII string representing a character.

Examples

a = chr(64)
“@”

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

Not applicable

circle

Purpose

Used to draw a circle on a Data Display page. Accepts the arguments center, radius, and number of points. Can only be used on polar plots and Smith charts.

Synopsis

a = circle(x, y, z)

where x is the center coordinate (can be a complex number), y is the radius, and z is the number of points

Examples

x = circle(1,1,500)

y = circle(1+j*1,1,500)

Used in

Data Display

Available as measurement component?

Not applicable

Defined in

AEL

See also

Not applicable

cint

Purpose

Given a noninteger real number, returns a rounded integer

Synopsis

y = cint(x)

where x is a real number to be rounded to an integer.

Note 0.5 rounds up, –0.5 rounds down (up in magnitude).

Examples

a = cint(45.6)
46

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, exp, float, int, log, log10, pow, sgn, sqrt

cmplx

Purpose

Given two real numbers representing the real and imaginary components of a complex number, returns a complex number

---

Note Use the real and imag functions to retrieve the real and imaginary components, respectively. The basic math functions operate on complex numbers.

Synopsis

y = cmplx(x, y)

where x is the real component and y is the imaginary component.

Examples

a = cmplx(2, –1)
2 – 1j

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

imag, real

conj

Purpose

Returns the conjugate of a complex number

Synopsis

y = conj(x)

where x is a complex number.

Examples

a = conj(3–4*j)
3.000 + j4.000
or 5.000 / 53.130 in magnitude / degrees

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

mag

const_evm

Purpose

Takes the results of a Circuit Envelope simulation and generates the ideal and distorted constellation and trajectory diagrams, as well as the error vector magnitude, in percent, and a plot of the error vector magnitude versus time

Synopsis

data = const_evm(vfund_ideal, vfund_dist, symbol_rate, sampling_delay, rotation, transient_duration, path_delay)

where

vfund_ideal is a single complex voltage spectral component (for example the fundamental) that is ideal (undistorted). This could be constructed from two baseband signals instead, by using the function cmplx()].

vfund_dist is a single complex voltage spectral component (for example, the fundamental) that has been distorted by the network being simulated. This could be constructed from two baseband signals instead, by using the function cmplx().

symbol_rate is the symbol rate of the modulation signal.

sampling_delay (if nonzero) throws away the first delay = *N* seconds of data from the constellation and trajectory plots. It is also used to interpolate between simulation time points, which is necessary if the optimal symbol-sampling instant is not exactly at a simulation time point. Usually this parameter must be nonzero to generate a constellation diagram with the smallest grouping of sample points.

rotation is a user-selectable parameter that rotates the constellations by that many radians. It does not need to be entered, and it will not affect the error-vector-magnitude calculation, because both the ideal and distorted constellations will be rotated by the same amount.

transient_duration is an optional time in seconds that causes this time duration of symbols to be eliminated from the error-vector-magnitude calculation. Usually the filters in the simulation have transient responses, and the error-vector-magnitude calculation should not start until these transient responses have finished.

path_delay is an optional time in seconds of the sum of all delays in the signal path. If the delay is 0, this parameter may be omitted. If it is non-zero, enter the delay value. This can be calculated by using the function delay_path().

Note that const_evm returns a list of data. So in the above example,

data[0]= ideal constellation
data[1]= ideal trajectory
data[2]= distorted constellation
data[3]= distorted trajectory
data[4]= error vector magnitude versus time
data[5]= percent error vector magnitude

Please refer to the example file listed below to see how these data are plotted.

Example

Example equations

rotation = –0.21
sampling_delay = 1/sym_rate[0, 0] – 0.5 ∗ tstep[0, 0]
vfund_ideal = vOut_ideal[1]
vfund_dist = vOut_dist[1]
symbol_rate = sym_rate[0, 0]
data = const_evm(vfund_ideal, vfund_dist, symbol_rate, sampling_delay, rotation, 1.5ms, path_delay)

where the parameter sampling_delay can be a numeric value, or in this case an equation using sym_rate, the symbol rate of the modulated signal, and tstep, the time step of the simulation. If these equations are to be used in a Data Display window, sym_rate and tstep must be defined by means of a variable (VAR) component, and they must be passed into the dataset as follows: Make the parameter Other visible on the Envelope simulation component, and edit it so that

Other = OutVar = sym_rate OutVar = tstep

In some cases, it may be necessary to experiment with the delay value to get the constellation diagrams with the tightest points.

Example files

examples/RF_Board/NADC_PA_prj/NADC_PA_Test.dsn and ConstEVM.dds and examples/Tutorial/Env_BER_prj/timing_doc.dds.

Used in

Constellation and trajectory diagram generation and error-vector-magnitude calculation

Available as measurement component?

Equations listed under Description can be entered by means of a MeasEqn component in the Schematic window. There is no explicit constellation or error-vector-magnitude measurement function.

Defined in

hpeesof/expressions/ael/digital_wireless_fun.ael

See also

constellation, delay_path, sample_delay_pi4dqpsk, sample_delay_qpsk

Description

The user must supply a single complex voltage spectral component (for example, the fundamental) that is ideal (undistorted), as well as a single complex voltage spectral component (for example, the fundamental) that has been distorted by the network being simulated. These ideal and distorted complex voltage waveforms could be generated from baseband I and Q data. The user must also supply the symbol rate, a delay parameter, a rotation factor, and a parameter to eliminate any turn-on transient from the error-vector-magnitude calculation are optional parameters.

The error vector magnitude is computed after correcting for the average phase difference and RMS amplitude difference between the ideal and distorted constellations.

constellation

Purpose

Generates the constellation diagram from Circuit Envelope, Transient, or Ptolemy simulation I and Q data.

Synopsis

Const = constellation(i_data, q_data, symbol_rate, delay)

where

i_data is the in-phase component of data versus time of a single complex voltage spectral component (for example, the fundamental) (this could be a baseband signal instead, but in either case it must be real-valued versus time);

q_data is the quadrature-phase component of data versus time of a single complex voltage spectral component (for example, the fundamental) (this could be a baseband signal instead, but in either case it must be real valued versus time);

symbol_rate is the symbol rate of the modulation signal; and delay (if nonzero) throws away the first delay = *N* seconds of data from the constellation plot. It is also used to interpolate between simulation time points, which is necessary if the optimal symbol-sampling instant is not exactly at a simulation time point. Usually this parameter must be nonzero to generate a constellation diagram with the smallest grouping of sample points.

Example

Example equations

Rotation = –0.21
Vfund =vOut[1] ∗ exp(j ∗ Rotation)
delay =1/sym_rate[0, 0] – 0.5 ∗ tstep[0, 0]
Vimag = imag(Vfund)
Vreal = real(Vfund)
Const = constellation(Vreal, Vimag, sym_rate[0, 0], delay)

where Rotation is a user-selectable parameter that rotates the constellation by that many radians, and vOut is the named connection at a node. The parameter delay can be a numeric value, or in this case an equation using sym_rate, the symbol rate of the modulated signal, and tstep, the time step of the simulation. If these equations are to be used in a Data Display window, sym_rate and tstep must be defined by means of a variable (VAR) component, and they must be passed into

the dataset as follows: Make the parameter Other visible on the Envelope simulation component, and edit it so that

Other = OutVar = sym_rate OutVar = tstep

In some cases, it may be necessary to experiment with the value of delay to get the constellation diagram with the tightest points.

Note vOut is a named connection on the schematic. Assuming that a Circuit Envelope simulation was run, vOut is output to the dataset as a two-dimensional matrix. The first dimension is time, and there is a value for each time point in the simulation. The second dimension is frequency, and there is a value for each fundamental frequency, each harmonic, and each mixing term in the analysis, as well as the baseband term.

vOut[1] is the equivalent of vOut[::, 1], and specifies all time points at the lowest non-baseband frequency (the fundamental analysis frequency, unless a multitone analysis has been run and there are mixing products). For former MDS users, the notation "vOut[*, 2]" in MDS corresponds to the ADS notation of "vOut[1]".

Example files

examples/RF_Board/NADC_PA_prj/ConstEVMslow.dds

examples/Tutorial/ModSources_prj/QAM_16_ConstTraj.dds

Used in

Constellation diagram generation

Available as measurement component?

Equations listed under Description can be entered by means of a MeasEqn component in the Schematic window. There is no explicit constellation measurement function.

Defined in

hpeesof/expressions/ael/digital_wireless_fun.ael

See also

const_evm

Description

The I and Q data do not need to be baseband waveforms. For example, they could be the in-phase (real or I) and quadrature-phase (imaginary or Q) part of a modulated carrier. The user must supply the I and Q waveforms versus time, as well as the symbol rate. A delay parameter is optional.

contour

Purpose

Generates contour levels on surface data

Synopsis

y = contour(data {, contour_levels})

where data is the data to be contoured, which must be at least two-dimensional real number or integer or implicit, and contour_levels is an optional one-dimensional quantity specifying the levels of the contours, which is normally specified by the sweep generator “[ ],” but can also be specified as a vector. If not provided, contour_levels defaults to six levels equally spaced between the maximum and the minimum of the data.

Examples

a = contour(dB(S11), [1::3::10])

or

a = contour(dB(S11), {1, 4, 7, 10})

produces a set of four equally spaced contours on a surface generated as a function of, say, frequency and strip width.

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

contour_polar

Description

This function introduces three extra inner independents into the data. The first two are "level", the contour level, and "number", the contour number. For each contour level there may be n contours. The contour is an integer running from 1 to n. The contour is represented as an (x, y) pair with x as the inner independent.

contour_polar

Purpose

Generates contour levels on polar or Smith chart surface data

Synopsis

y = contour_polar(data {, contour_levels})

where data is the polar or Smith chart data to be contoured, (and therefore is surface data), and contour_levels is an optional one-dimensional quantity specifying the levels of the contours, which is normally specified by the sweep generator "[ ]," but can also be specified as a vector. If not provided, contour_levels defaults to six levels equally spaced between the maximum and the minimum of the data.

Examples

a = contour_polar(data_polar, [1::4])

or

a = contour_polar(data_polar, {1, 2, 3, 4})

produces a set of four equally spaced contours on a polar or Smith chart surface.

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

AEL, display_fun.ael

See also

contour

cos

Purpose

Returns the cosine of a real number or integer

Synopsis

y = cos(x)

where x is the real number or integer, in radians.

Examples

a = cos(pi/3)
0.500

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

sin, tan

cosh

Purpose

hyperbolic cosine

Synopsis

cosh()

Example

cosh(0)
1

cosh(1)
1.543

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

sinh, tanh

cross

Purpose

Computes the zero crossings of a signal and the interval between successive zero crossings. The independent axis returns the time when the crossing occurred. The dependent axis returns the time interval since the last crossing.

Synopsis

cross(signal, direction)

If direction = +1, compute positive going zero crossings.

If direction = -1, compute negative going zero crossings.

If direction = 0, compute all zero crossings (default).

Example

period=cross(vosc-2.0, 1)

This computes the period of each cycle of the vosc signal. The period is measured from each positive-going transition through 2.0V.

Used In

Not applicable

Available in measurement component?

Not applicable

Defined In

Built in

See Also

None

cross_corr

Purpose

Returns the cross-correlation

Synopsis

cross_corr(v1, v2)

where v1 and v2 are 1-dimensional data

Example

v1 = qpsk..videal[1]
v2 = qpsk..vout[1]
x_corr_v1v2 = cross_corr(v1, v2)
auto_corr_v1 = cross_corr(v1, v1)

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

AEL, digital_wireless_fun.ael

See also

None

cum_prod

Purpose

Returns the cumulative product

Synopsis

cum_prod(x)

The function takes a single argument, so enclose a sequence of numbers in brackets "[x, y, ...]"

Example

```
cum_prod(1)
1

cum_prod([1, 2, 3])
6

cum_prod([i, i])
–1
```

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cum_sum, max, mean, min, prod, sum

cum_sum

Purpose

Returns the cumulative sum

Synopsis

cum_sum(x)

The function takes a single argument, so enclose a sequence of numbers in brackets "[x, y, ...]"

Example

cum_sum([1, 2, 3])
7

cum_sum([i, i]
2i

Used in

Not Applicable

Available as measurement component?

Not Applicable

Defined in

Built in

See also

cum_prod, max, mean, min, prod, sum

dB

Purpose

Returns the decibel measure of a voltage ratio

Synopsis

dB(r{, z1, z2})= 20 log(mag(r) - 10 log(zOutfactor/zInfactor)

where r is a voltage ratio (vOut/vIn), z1 is the source impedance (default is 50), zOutfactor = mag(z2)**2 / real (z2), z2 is the load impedance (default is 50), and zInfactor = mag(z1)**2 / real (z1).

Examples

dB(100)
40

dB(8-6*j)
20

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

dBm, pae

dBm

Purpose

Returns the decibel measure of a voltage referenced to a 1 milliwatt signal

Synopsis

dBm(v{, z})= 20 log(mag(v) - 10 log(real(zOutfactor/50)) + 10

where v is a voltage (the peak voltage), z is an impedance (default is 50), and zOutfactor = mag(z)**2 / real (z).

Examples

dBm(100)
50

dBm(8-6*j)
30

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

dB, pae

dc_to_rf

Purpose

Returns DC-to-RF efficiency

Synopsis

dc_to_rf(vPlusRF, vMinusRF, vPlusDC, vMinusDC, currentRF, currentDC, freq)

where vPlusRF and vMinusRF are RF voltages at the negative terminals, vPlusDC and vMinusDC are DC voltages at the negative terminals, currentRF and currentDC are the RF and DC currents for power calculation, and freq is harmonic index of the RF frequency at the output port.

Example

a = dc_to_rf(vrf, 0, vDC, 0, I_Probe1.i, SRC1.i, 1MHz

Used in

Harmonic balance simulation

Available as measurement component?

DCtoRF

Defined in

AEL, circuit_fun.ael

See also

None

Description

This measurement computes the DC-to-RF efficiency of any part of the network.

deg

Purpose

Converts radians to degrees

Synopsis

deg(x)

Example

deg(1.5708)
90

deg(pi)
180

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

rad

delay_path

Purpose

This function is used to determine the time delay and the constellation rotation angle between two nodal points along a signal path.

Synopsis

delay_path(vin, vout)

where vin is the complex envelope (I + j * Q) at the input node and vout is I + j * Q at the output node.

Example

x = delay_path(vin[1], vout[1])

where vin[1] and vout[1] are complex envelopes around the first carrier frequency in envelope simulation. In return, x[0] is the time delay (in seconds) between vin and vout. x[1] is the rotation angle (in radians) between vin and vout constellations.

or

x = delay_path(T1, T2)

where T1 and T2 are instance names of two TimedSink components.

Used in

Circuit envelope simulation, Ptolemy simulation.

Available as measurement component?

Not applicable

Defined in

Built in

See also

ber_pi4dqpsk, ber_qpsk, const_evm, cross_corr

Description

This function outputs an array of two values. The first value, data[0], is the time delay between vin and vout. The second value, data[1], is the rotation angle between vin-constellation and vout-constellation.

dev_lin_phase

Purpose

Returns deviation (in degrees) from linear phase.

Synopsis

dev_lin_phase(voltGain)

where voltGain is a function of frequency.

Example

a = dev_lin_phase(S21)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

DevLinPhase

Defined in

AEL, rf_system_fun.ael

See also

diff, phasedeg, phaserad, pwr_gain, ripple, unwrap, volt_gain

Description

Given a variable sweep over a frequency range, a linear least-squares fit is performed on the phase of the variable, and the deviation from this linear fit is calculated at each frequency point.

diff

Purpose

Returns the numerical difference

Synopsis

y = diff(data)

returns the numerical difference against the inner independent variable associated with the data.

Examples

group_delay = –diff(unwrap(phaserad(S21),pi))/ (2*pi)

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

AEL, elementary_fun.ael

See also

dev_lin_phase, integrate, phasedeg, phaserad, ripple, unwrap

Description

Calculates a simple numerical difference against the inner independent variable associated with the data. Can be used to calculate group delay.

erf

Purpose

Returns the error function

Synopsis

y = erf(x)

where x is real.

Examples

a = –erf(0.1)
0.112

a = –erf(0.2)
0.223

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

erfc

Description

Calculates the error function, the area under the Gaussian curve exp(–x**2).

erfc

Purpose

Returns the complementary error function

Synopsis

y = erfc(x)

where x is real.

Examples

a = –erfc(0.1)
0.888

a = –erfc(0.2)
0.777

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

erf

Description

Calculates the complementary error function, or 1 minus the error function. For large x, this can be calculated more accurately than the plain error function.

exp

Purpose

Given an integer or real number as an exponent, returns *e* (~2.7183) raised to that exponent

Synopsis

y = exp(x)

where x is the exponent of *e*.

Examples

a = exp(1)
2.71828

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, float, int, log, log10, pow, sgn, sqrt

eye

Purpose

Creates data for an eye diagram plot

Synopsis

eye(data, symbolRate{, Cycles{, Delay}})

data is either numeric data or a time domain waveform typically from the I or Q data channel. symbolRate is the symbol rate of the channel. For numeric data, the symbol rate is the reciprocal of the number of points in one cycle; for a waveform, it is the frequency. Cycles is optional and is the number of cycles to repeat, default is 1. Delay is an optional sampling delay, default is 0.

Example

eye(I_data, symbol_rate)

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

constellation

Description

Refer also to *analysis.dds* in /examples/Tutorial/express_meas_prj.

The cycle parameter is used to display more than one cycle of the eye. The delay parameter is used to adjust the position of the eye opening. Note that delay is not used to remove a transient in the eye diagram. To remove an initial transient, you must use explicit indexing on the original data. Following this, you may want to use a delay to realign the eye opening.

fft

Purpose

Performs the discrete Fourier transform

Synopsis

y = fft(x, length)

Example

fft([1, 1, 1, 1])
[4+0i, 0+0i]

fft([1, 0, 0, 0]
[1+0i, 1+0i]

fft(1, 4)
[1+0i, 1+0i]

Used in

Not applicable

Available as measurement component?

Not Applicable

Defined in

Built in

See also

fs, ts

Description

fft(x) is the discrete Fourier transform of x computed with the fast Fourier transform algorithm. fft() uses a high-speed radix-2 fast Fourier transform when the length of x is a power of two. fft(x, n) performs an n-point discrete Fourier transform, truncating x if length(x) > n and padding x with zeros if length(x) < n.

fft() uses a real transform if x is real and a complex transform if x is complex. If the length of x is not a power of two, then a mixed radix algorithm based on the prime factors of the length of x is used.

find_index

Purpose

Finds the closest index for a given search value

Synopsis

index = find_index(data_sweep, search_value)

To facilitate searching, the find_index function finds the index value in a sweep that is closest to the search value. Data of type int or real must be monotonic. find_index also performs an exhaustive search of complex and string data types.

Examples

Given S-parameter data swept as a function of frequency, find the value of S_{11} at 1 GHz:

```
index=find_index(freq, 1GHz)
a=S11[index]
```

Used in

Use with all simulation data

Available as measurement component?

Not applicable

Defined in

Built in

See also

mix

float

Purpose

Converts an integer to a real (floating-point) number

Note: To convert a real to an integer, use int.

Synopsis

y = float(x)

where x is the integer to convert.

Examples

a = float(10)

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, float, int, log10, pow, sgn, sqrt

fs

Purpose

Performs a time-to-frequency transform

Synopsis

fs(x, fstart, fsto*p, numfreqs, dim, windowType, windowConst, tstart, tstop, interpOrder, transformMethod*)

See detailed Description below.

Examples

The following example equations assume that a transient simulation was performed from 0 to 5 ns with 176 timesteps, on a 1-GHz-plus-harmonics signal called vOut:

y=fs(vOut)
returns the spectrum (0, 0.2GHz, ... , 25.6GHz), evaluated from 0 to 5 ns.

y=fs(vOut, 0, 10GHz)
returns the spectrum (0, 0.2GHz, ... , 10.0GHz), evaluated from 0 to 5 ns.

y=fs(vOut, 0, 10GHz, 11)
returns the spectrum (0, 1.0GHz, ... , 10.0GHz), evaluated from 0 to 5 ns.

y=fs(vOut, , , , , , , 3ns, 5ns)
returns the spectrum (0, 0.5GHz, ... , 32.0GHz), evaluated from 3 to 5 ns.

y=fs(vOut, 0, 10GHz, 21, , , , 3ns, 5ns)
returns the spectrum (0, 0.5GHz, ... , 10.0GHz), evaluated from 3 to 5 ns.

y=fs(vOut, 0, 10GHz, 11, , "Blackman")
returns the spectrum (0, 1.0GHz, ... , 10.0GHz), evaluated from 0 to 5 ns after a Blackman window is applied.

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

fft, fspot

Description

fs(x) returns the frequency spectrum of the vector x by using a chirp-z transform. The values returned are peak, complex values.

x will typically be data from a transient, signal processing, or envelope analysis.

Transient simulation uses a variable timestep and variable order algorithm. The user sets an upper limit on the allowed timestep, but the simulator will control the timestep so that the local truncation error of the integration is also controlled. The nonuniformly sampled data are uniformly resampled for fs.

If the Gear integration algorithm is used, the order can also change during simulation. fs can use this information when resampling the data. This variable order integration depends on the presence of a special dependent variable, tranorder, which is output by the transient simulator. When the order varies, the Fourier integration will adjust the order of the polynomial it uses to interpolate the data between timepoints.

If the tranorder variable is not present, or if the user wishes to override the interpolation scheme, then interpOrder may be set to a nonzero value:

1 = use only linear interpolation

2 = use quadratic interpolation

3 = use cubic polynomial interpolation

Only polynomials of degree one to three are supported. The polynomial is fit from the timepoint in question backwards over the last n points. This is because time-domain data are obtained by integrating forward from zero; previous data are used to determine future data, but future data can never be used to modify past data.

The data are uniformly resampled, with the number of points being determined by increasing the original number of points to the next highest power of two.

The data to be transformed default to all of the data. The user may specify tstart and tstop to transform a subset of the data.

The starting frequency defaults to 0 and the stopping frequency defaults to 1/(2*newdeltat), where newdeltat is the new uniform timestep of the resampled data. The number of frequencies defaults to (fstop–fstart)*(tstop–tstart)+1. The user may change these by using fstart, fstop, and numfreqs. Note that numfreqs specifies the

number of frequencies, not the number of increments. Thus, to get frequencies at (0, 1, 2, 3, 4, 5), numfreqs should be set to 6, not 5.

The data to be transformed may be windowed. The window is specified by windowType, with an optional window constant windowConst. The window types and their default constants are:

0 = None

1 = Hamming 0.54

2 = Hanning 0.50

3 = Gaussian 0.75

4 = Kaiser 7.865

5 = 8510 6.0 (This is equivalent to the time-to-frequency transformation with normal gate shape setting in the 8510 series network analyzer.)

6 = Blackman

7 = Blackman–Harris

The default time-to-frequency transform is done by means of a chirp-z transform. This may be changed by using transformMethod:

1 = Chirp-z transform

2 = Discrete Fourier integral evaluated at each frequency

3 = Fast Fourier transform

When the data to be operated on is of the baseband type, such as VO[0] from a Circuit Envelope analysis, where VO is an output node voltage and [0] is index for DC, then in order to obtain a single sided spectrum, only the real part of VO[0] should be used as the argument. ie, x=fs(real(VO[0],...).

This is necessary because the fs() function has no way of knowing the data VO[0] is baseband. Even though VO[0] contains an imaginary part of all zeroes, it is still represented by a complex data type. When the first argument of fs() is complex, the result will be a double-sided spectrum by default.

An alternative method of obtaining a single-sided spectrum from the above baseband data is to specify the frequencies ranges in the spectrum, using the fstart, fstop, and numfreqs parameters of the fs() function.

For example, y=fs(VO[0], 0, 25e3, 251). This will yield a spectrum from 0 to 25 kHz with 26 frequencies and 1 kHz spacing.

This does not apply to data from Transient analysis nor Ptolemy simulation because voltage data from Transient and baseband data from Ptolemy are real.

fspot

Purpose

Performs a single-frequency time-to-frequency transform

Synopsis

fspot(x, fund, harm, windowType, windowConst, interpOrder, tstart)

See detailed Description below.

Examples

The following example equations assume that a transient simulation was performed from 0 to 5 ns on a 1-GHz-plus-harmonics signal called vOut:

fspot(vOut)
returns the 200-MHz component, integrated from 0 to 5 ns.

fspot(vOut, , 5)
returns the 1-GHz component, integrated from 0 to 5 ns.

fspot(vOut, 1GHz, 1)
returns the 1-GHz component, integrated from 4 to 5 ns.

fspot(vOut, 0.5GHz, 2, , , , 2.5ns)
returns the 1-GHz component, integrated from 2.5 to 4.5 ns.

fspot(vOut, 0.25GHz, 4, "Kaiser")
returns the 1-GHz component, integrated from 1 to 5 ns, after applying the default Kaiser window to this range of data.

fspot(vOut, 0.25GHz, 4, 3, 2.0)
returns the 1-GHz component, integrated from 1 to 5 ns, after applying a Gaussian window with a constant of 2.0 to this range of data.

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

fft, fs

Description

fspot(x) returns the discrete Fourier transform of the vector x evaluated at one specific frequency. The value returned is the peak component, and it is complex. The harmth harmonic of the fundamental frequency fund is obtained from the vector x. The Fourier transform is applied from time tstop–1/fund to tstop, where tstop is the last timepoint in x.

When x is a multidimensional vector, the transform is evaluated for each vector in the specified dimension. For example, if x is a matrix, then fspot(x) applies the transform to every row of the matrix. If x is three dimensional, then fspot(x) is applied in the lowest dimension over the remaining two dimensions. The dimension over which to apply the transform may be specified by dim; the default is the lowest dimension (dim=1). x must be numeric. It will typically be data from a transient, signal processing, or envelope analysis.

fund must be greater than zero. It is used to specify the period 1/fund for the Fourier transform. fund defaults to a period that matches the length of the independent axis of x.

harm may be any positive number. harm defaults to 1. Specifying harm=0 will compute the dc component of x.

The data to be transformed may be windowed. The window is specified by windowType, with an optional window constant windowConst. The window types and their default constants are:

0 = None

1 = Hamming 0.54

2 = Hanning 0.50

3 = Gaussian 0.75

4 = Kaiser 7.865

5 = 8510 6.0

6 = Blackman

7 = Blackman-Harris

windowType can be specified either by the number or by the name.

By default, the transform is performed at the end of the data from tstop-1/fund to tstop. By using tstart, the transform can be started at some other point in the data. The transform will then be performed from tstart to tstart+1/fund.

Unlike with fft or fs, the data to be transformed are not zero padded or resampled. fspot works directly on the data as specified, including nonuniformly sampled data from a transient simulation.

Transient simulation uses a variable timestep and variable order algorithm. The user sets an upper limit on the allowed timestep, but the simulator will control the timestep so the local truncation error of the integration is controlled. If the Gear integration algorithm is used, the order can also be changed during simulation. fspot can use all of this information when performing the Fourier transform. The time data are not resampled; the Fourier integration is performed from timestep to timestep of the original data.

When the order varies, the Fourier integration will adjust the order of the polynomial it uses to compute the shape of the data between timepoints.

This variable order integration depends on the presence of a special dependent variable, tranorder, which is output by the transient simulator. If this variable is not present, or if the user wishes to override the interpolation scheme, then interpOrder may be set to a nonzero value:

1 = use only linear interpolation

2 = use quadratic interpolation

3 = use cubic polynomial interpolation

Only polynomials of degree one to three are supported. The polynomial is fit because time domain data are obtained by integrating forward from zero; previous data are used to determine future data, but future data can never be used to modify past data.

fun_2d_outer

Purpose

Applies a function to the outer dimension of two-dimensional data.

Synopsis

fun_2d_outer(data, fun)

where data must be two-dimensional data, and fun is some function (usually mean, max, or min) that will be applied to the outer dimension of the data.

Example

fun_2d_outer(data, min)

Used in

max_outer, mean_outer, min_outer functions.

Available as measurement component?

No, but the function can be used on a schematic page, in a measurement equation.

Defined in

AEL, statistcal_fun.ael

See also

max_outer, mean_outer, min_outer

Description

Functions such as mean, max, and min operate on the inner dimension of two-dimensional data. The function fun_2d_outer enables these functions to be applied to the outer dimension. As an example, assume that a Monte Carlo simulation of an amplifier was run, with 151 random sets of parameter values, and that for each set the S-parameters were simulated over 26 different frequency points. S21 becomes a [151 Monte Carlo iteration X 26 frequency] matrix, with the inner dimension being frequency, and the outer dimension being Monte Carlo index. Now, assume that it is desired to know the mean value of the S-parameters at each frequency. Inserting an equation mean(S21) computes the mean value of S21 at each Monte Carlo iteration. If S21 is simulated from 1 to 26 GHz, it computes the mean value over this frequency range, which usually is not very useful. The function fun_2d_outer allows the mean to be computed over each element in the outer dimension.

ga_circle

Purpose

Generates an available-gain circle

Synopsis

ga_circle(S{, gain, numOfPts})

where S is the scattering matrix of a 2-port network, gain is the specified gain in dB, and numOfPts is the desired number of points per circle. The default value for gain is min(max_gain(S)) – {1, 2, 3}, and the default value for numOfPts is 51.

Example

circleData = ga_circle(S, 2, 51)
circleData = ga_circle(S, {2, 3, 4}, 51)
return the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

GaCircle

Defined in

AEL, circle_fun.ael

See also

gl_circle, gp_circle, gs_circle

Description

This expression generates the constant available-gain circle resulting from a source mismatch. The circle is defined by the loci of the source-reflection coefficients resulting in the specified gain.

A gain circle is created for each value of the swept variable(s). Multiple gain values can be specified for a scattering parameter that has dimension less than four. This measurement is supported for 2-port networks only.

gain_comp

Purpose

Returns gain compression

Synopsis

gain_comp(Sji)

where Sji is a power-dependent complex transmission coefficient obtained from large-signal S-parameter simulation.

Example

gc = gain_comp(S21[::, 0])

Used in

Large-signal S-parameter simulations

Available as measurement component?

GainComp

Defined in

AEL, rf_system_fun.ael

See also

phase_comp

Description

This measurement calculates the small-signal minus the large-signal power gain, in dB. The first power point (assumed to be small) is used to calculate the small-signal power gain.

generate

Purpose

Generates a sequence of real numbers

Synopsis

generate(start, stop, npts)

where start is the first number, stop is the last number, and npts is the number of numbers in the sequence.

Example

a = generate(9, 4, 6)
return the sequence 9., 8., 7., 6., 5., 4.

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

Not applicable

Description

This function generates a sequence of real numbers. The modern way to do this is to use the sweep generator “[ ].”

get_attr

Purpose

Gets a data attribute

Synopsis

a = get_attr(data, "attr_name"{, eval})

where data is a frequency swept variable, attr_name is the name of an attribute, and eval is true or false as to whether to evaluate the attribute.

Example

get_attr(data, "fc", true)
10GHz

get_attr(data, "dataType")
"TimedData"

get_attr(data, "TraceType", false)
"Spectral"

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

set_attr

Description

This function only works with frequency swept variables.

gl_circle

Purpose

Generates a load-mismatch gain circle

Synopsis

gl_circle(S{, gain, numOfPts})

where S is the scattering matrix of a 2-port network, gain is the specified gain in dB, and numOfPts is the desired number of points per circle. The default value for gain is 10*log(1 / (1 – mag(S22)**2)) – {1, 2, 3}, and the default value for numOfPts is 51.

Example

circleData = gl_circle(S, 2, 51)
circleData = gl_circle(S, {2, 3, 4}, 51)
return the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

GlCircle

Defined in

AEL, circle_fun.ael

See also

ga_circle, gp_circle, gs_circle

Description

This expression generates the unilateral gain circle resulting from a load mismatch. The circle is defined by the loci of the load-reflection coefficients that result in the specified gain.

A gain circle is created for each value of the swept variable(s). Multiple gain values can be specified for a scattering parameter that has dimension less than four. This measurement is supported for 2-port networks only.

gp_circle

Purpose

Generates a power gain circle

Synopsis

gp_circle(S{, gain, numOfPts})

where S is the scattering matrix of a 2-port network, gain is the specified gain in dB, and numOfPts is the desired number of points per circle. The default value for gain is min(max_gain(S)) – {1, 2, 3}, and the default value for numOfPts is 51

Example

circleData = gp_circle(S, 2, 51)
circleData = gp_circle(S, {2, 3, 4}, 51)
return the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

GpCircle

Defined in

AEL, circle_fun.ael

See also

ga_circle, gl_circle, gs_circle

Description

This expression generates a constant-power-gain circle resulting from a load mismatch. The circle is defined by the loci of the output-reflection coefficients that result in the specified gain.

A gain circle is created for each value of the swept variable(s). Multiple gain values can be specified for a scattering parameter that has dimension less than four. This measurement is supported for 2-port networks only.

gs_circle

Purpose

Returns a source-mismatch gain circle

Synopsis

gs_circle(S{, gain, numOfPts})

where S is the scattering matrix of a 2-port network, gain is the specified gain in dB, and numOfPts is the desired number of points per circle. The default value for gain is 10*log(1 / (1 – mag(S11)**2)) – {1, 2, 3}, and the default value for numOfPts is 51.

Example

circleData = gs_circle(S, 2, 51)
circleData = gs_circle(S, {2, 3, 4}, 51)
return the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

GsCircle

Defined in

AEL, circle_fun.ael

See also

ga_circle, gl_circle, gp_circle

Description

This expression generates the unilateral gain circle resulting from a source mismatch. The circle is defined by the loci of the source-reflection coefficients that result in the specified gain.

A gain circle is created for each value of the swept variable(s). Multiple gain values can be specified for a scattering parameter that has dimension less than four. This measurement is supported for 2-port networks only.

histogram

Purpose

Generates a histogram representation

Synopsis

histogram(x, numBins, minBin, maxBin)

where x is the signal, numBins is number of subintervals or bins used to measure the histogram, and minBin and maxBin are the beginning and end, respectively, of the evaluation of the histogram.

Example

y = histogram(data)

y = histogram(data, 20)

Used in

Not applicable

Available as measurement component?

This function can only be entered by means of a Eqn component in the Data Display window. There is no measurement component in schematic window.

Defined in

Built in

See also

cdf, pdf, yield_sens

Description

This function creates a histogram that represents data. The default values for minBin and maxBin are the minimum and the maximum values, respectively, of the data, and numBins is set to log(numOfPts)/log(2.0) by default.

htoabcd

Purpose

Performs H-to-ABCD conversion

Synopsis

htoabcd(H)

where H is the hybrid matrix of a 2-port network.

Example

a = htoabcd(h)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

abcdtoh, htoz, ytoh

Description

This measurement transforms the hybrid matrix of a 2-port network to a chain (ABCD) matrix.

htos

Purpose

Performs H-to-S conversion

Synopsis

htos(H, Z)

where H is the hybrid matrix of a 2-port network., and Z is the reference impedance.

Example

s = htos(h, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

htoy, htoz, stoh

Description

This measurement transforms the hybrid matrix of a 2-port network to a scattering matrix.

htoy

Purpose

Performs H-to-Y conversion

Synopsis

htoy(H)

where H is the hybrid matrix of a 2-port network

Example

Y = htoy(H)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

htos, htoz, ytoh

Description

This measurement transforms the hybrid matrix of a 2-port network to an admittance matrix.

htoz

Purpose

Performs H-to-Z conversion

Synopsis

htoz(H)

where H is the hybrid matrix of a 2-port network

Example

z = htoz(h)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

htos, htoy, ytoh

Description

This measurement transforms the hybrid matrix of a 2-port network to an impedance matrix.

identity

Purpose

Returns the identity matrix

Synopsis

Y = identity(2)

Y = identity(2, 3)

The identity matrix is defined as follows. If one argument is supplied, then a square matrix is returned with ones on the diagonal and zeros elsewhere. If two arguments are supplied, then a matrix with size rows × cols is returned, again with ones on the diagonal.

Example

a = identity(2)

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

ones, zeros

ifc

Purpose

Returns frequency-selective current in Harmonic Balance analysis

Synopsis

ifc(iOut, harm_freq_index)

where iOut is the current through a branch, and harm_freq_index is the harmonic index of the desired frequency. Note that the harm_freq_index argument's entry should reflect the number of tones in the harmonic balance controller. For example, if one tone is used in the controller, there should be one number inside the braces; two tones would require two numbers separated by a comma.

Example

The following example is for two tones in the harmonic balance controller:

ifc(I_Probe1.i, {1, 0})

Used in

Harmonic Balance simulation

Available as measurement component?

Ifc

Defined in

AEL, circuit_fun.ael

See also

pfc, vfc

Description

This measurement gives the RMS current value of one frequency-component of a harmonic balance waveform.

ifc_tran

Purpose

Returns frequency-selective current in Transient analysis

Synopsis

ifc_tran(iOut, fundFreq, harmNum)

where iOut is the current through a branch, fundFreq is the fundamental frequency and harmNum is the harmonic number of the fundamental frequency (positive integer value only).

Example

ifc_tran(I_Probe1.i, 1GHz, 1)

Used in

Transient simulation

Available as measurement component?

IfcTran

Defined in

AEL, circuit_fun.ael

See also

pfc_tran, vfc_tran

Description

This measurement gives RMS current, in current units, for a specified branch at a particular frequency of interest. fundFreq determines the portion of the time-domain waveform to be converted to the frequency domain. This is typically one full period corresponding to the lowest frequency in the waveform. harmNum is the harmonic number of the fundamental frequency at which the current is requested.

imag

Purpose

Returns the imaginary component of a complex number

Synopsis

y = imag(x)

where x is a complex number.

Examples

```
a = imag(1–1*j)
-1.000
```

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cmplx, real

indep

Purpose

Returns the independent attached to the data

Synopsis

Y = indep(x)

Y = indep(x, dimension)

Y = indep(x, "indep_name")

indep() returns the independent (normally the swept variable) attached to simulation data. When there is more than one independent, then the independent of interest may be specified by number or by name. If no independent specifications are passed, then indep() returns the innermost independent.

Example

Given S-parameters versus frequency and power: Frequency is the innermost independent, so its index is 1. Power has index 2.

freq = indep(S, 1)
freq = indep(S, "freq")

power = indep(S, 2)
power = indep(S, "power")

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

find_index

int

Purpose

Returns the largest integer not greater than a given real value

Synopsis

y = int(x)

where x is the real value.

Examples

a = int(4.3);
4

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, exp, float, log10, pow, sgn, sqrt

integrate

Purpose

Returns the intergral of data

Synopsis

y = integrate(data{, start, stop{, incr}})

returns the intergral of data from start to stop with increment incr.

Examples

x =[0::0.01::1.0]

y = vs(2*exp(-x*x) / sqrt(pi), x)

z= integrate(y, 0.1, 0.6, 0.001)

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

AEL, circuit_fun.ael

See also

diff

Description

Returns the intergral of data from start to stop with increment incr using the composite trapezoidal rule on uniform subintervals. The default values for start and stop are the first and last points of the data, respectively. The default value for incr is “(stop - start) / (nPts - 1)” where nPts is the number of original data points between start and stop, inclusively.

interp

Purpose

Returns linearly interpolated data

Synopsis

y = interp(data{, start, stop{, incr}})

returns linearly interpolated data between start and stop with increment incr.

Examples

y = interp(data{, start, stop{, incr}})

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

None

Description

Returns linearly interpolated data between start and stop with increment incr. The default values for start and stop are the first and last points of the data, respectively. The default value for incr is “(stop - start) / (nPts - 1)” where nPts is the number of original data points between start and stop, inclusively.

inverse

Purpose

Performs a matrix inverse

Synopsis

y = inverse(x)

inversion of real and complex general matrices.

Example

inverse({{1, 2}, {3, 4}});
{{–2, 1}, {1.5, –0.5}}

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

None

ip3_in

Purpose

Returns the input third-order intercept (TOI) point

Synopsis

ip3_in(vOut, ssGain, fundFreq, imFreq, zRef)

where vOut is the signal voltage at the output, ssGain is the small signal gain in dB, fundFreq and imFreq are the harmonic frequency indices for the fundamental and intermodulation frequencies, respectively, and zRef is the reference impedance.

Example

y=ip3_in(vOut, 22, {1, 0}, {2, –1}, 50)

Used in

Harmonic balance simulation

Available as measurement component?

IP3in

Defined in

AEL, rf_system_fun.ael

See also

ip3_out, ipn

Description

This measurement determines the input third-order intercept point (in dBm) at the input port with reference to a system output port.

ip3_out

Purpose

Returns the output third-order intercept (TOI) point

Synopsis

ip3_out(vOut, fundFreq, imFreq, zRef)

where vOut is the signal voltage at the output, fundFreq and imFreq are the harmonic frequency indices for the fundamental and intermodulation frequencies, respectively, and zRef is the reference impedance.

Example

y=ip3_out(vOut, {1, 0}, {2, –1}, 50)

Used in

Harmonic balance simulation

Available as measurement component?

IP3out

Defined in

AEL, rf_system_fun.ael

See also

ip3_in, ipn

Description

This measurement determines the output third-order intercept point (in dBm) at the system output port.

ipn

Purpose

Returns the output *n*th-order intercept (TOI) point

Synopsis

ipn(vPlus, vMinus, iOut, fundFreq, imFreq, n)

where vPlus and vMinus are the voltages at the positive and negative output terminals, respectively; iOut is the current through a branch; fundFreq and imFreq are the harmonic indices of the fundamental and intermodulation frequencies, respectively; and n is the order of the intercept.

Example

y=ipn(vOut, 0, I_Probe1.i, {1, 0}, {2, -1}, 3)

Used in

Harmonic balance simulation

Available as measurement component?

IPn

Defined in

AEL, circuit_fun.ael

See also

ip3_in, ip3_out

Description

This measurement determines the output nth-order intercept point (in dBm) at the system output port.

ispec_tran

Purpose

Returns current spectrum

Synopsis

ispec_tran(iOut, fundFreq, numHarm)

where iOut is the current through a branch, fundFreq is the fundamental frequency value and numHarm is the number of harmonics of fundamental frequency (positive integer value only).

Example

y=ispec_tran(I_Probe1.i, 1GHz, 8)

Used in

Transient simulation

Available as measurement component?

IspecTran

Defined in

AEL, circuit_fun.ael

See also

pspec_tran, vspec_tran

Description

This measurement gives a current spectrum for a specified branch. The measurement gives a set of RMS current values at each frequency. fundFreq determines the portion of the time-domain waveform to be converted to frequency domain. This is typically one full period corresponding to the lowest frequency in the waveform. numHarm is the number of harmonics of fundamental frequency to be included in the currents spectrum.

it

Purpose

Returns time-domain current waveform

Synopsis

it(iOut, tmin, tmax, numOfPnts)

where iOut is the current through a branch, tmin and tmax are start time are stop time, respectively, and numOfPts is the number of points (integer values only).

ExamplezRef

y=it(I_Probe1.i, 0, 10nsec, 201)

Used in

Harmonic balance simulation

Available as measurement component?

It

Defined in

AEL, circuit_fun.ael

See also

vt

Description

This measurement converts a harmonic-balance current frequency spectrum to a time-domain current waveform.

l_stab_circle

Purpose

Returns a load (output) stability circle

Synopsis

l_stab_circle(S{, numOfPts})

where S is the scattering matrix of a 2-port network and numOfPts is the desired number of points per circle and is set to 51 by default.

Example

circleData=l_stab_circle(S, 51)
returns the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

L_StabCircle

Defined in

AEL, circle_fun.ael

See also

l_stab_region, s_stab_circle, s_stab_region

Description

The expression generates a load stability circle. The circle is defined by the loci of load-reflection coefficients where the magnitude of the source-reflection coefficient is 1.

A circle is created for each value of the swept variable(s). This measurement is supported for 2-port networks only.

l_stab_region

Purpose

Indicates the region of stability of the load (output) stability circle

Synopsis

l_stab_region(S)

where S is the scattering matrix of a 2-port network.

Example

region = l_stab_region(S)
returns “Outside” or “Inside”.

Used in

Small-signal S-parameter simulations

Available as measurement component?

Not applicable

Defined in

AEL, circle_fun.ael

See also

l_stab_circle, s_stab_circle, s_stab_region

Description

This expression returns a string identifying the region of stability of the corresponding load stability circle.

ln

Purpose

Returns the natural logarithm (ln) of an integer or real number

Synopsis

y = ln(x)

where x is the integer or real number.

Examples

a = ln(e);
returns 1

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, exp, float, int, pow, sgn, sqrt

log

Purpose

Returns the base 10 logarithm of an integer or real number

Note: log10(x) perform the same operation.

Synopsis

y = log(x)

where x is the integer or real number.

Examples

a = log(10)
1

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, exp, log10, float, int, pow, sgn, sqrt

log10

Purpose

Returns the base 10 logarithm of an integer or real number

Note: log(x) perform the same operation.

Synopsis

y = log10(x)

where x is the integer or real number.

Examples

a = log10(10)
1

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, exp, log, float, int, pow, sgn, sqrt

mag

Purpose

Returns the magnitude of a complex number

Synopsis

y = mag(x)

where x is a complex number.

Examples

```
a = mag(3–4*j)
5.000
```

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

conj

map1_circle

Purpose

Returns source-mapping circles from port 1 to port 2

Synopsis

circleData=map1_circle(S{, numOfPts})

where S is the scattering matrix of a 2-port network and numOfPts is the desired number of points per circle and is set to 51 by default.

Example

circleData=map1_circle(S, 51)
returns the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

Map1Circle

Defined in

AEL, circles_fun.ael

See also

map2_circle

Description

The expression maps the set of terminations with unity magnitude at port 1 to port 2. The circles are defined by the loci of terminations on one port as seen at the other port.

A source-mapping circle is created for each value of the swept variable(s). This measurement is supported for 2-port networks only.

map2_circle

Purpose

Returns source-mapping circles, from port 2 to port 1

Synopsis

circleData=map2_circle(S{, numOfPts})

where S is the scattering matrix of a 2-port network and numOfPts is the desired number of points per circle and is set to 51 by default.

Example

circleData=map2_circle(S, 51)
returns the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

Map2Circle

Defined in

AEL, circle_fun.ael

See also

map1_circle

Description

The expression maps the set of terminations with unity magnitude at port 2 to port 1. The circles are defined by the loci of terminations on one port as seen at the other port.

A source-mapping circle is created for each value of the swept variable(s). This measurement is supported for 2-port networks only.

max

Purpose

Returns the maximum value

Synopsis

Y = max(x)

The function takes a single argument, so enclose a sequence of numbers in brackets "[x, y, ...]"

Example

a = max([1, 2, 3])
3

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cum_prod, cum_sum, mean, min, prod, sum

max_gain

Purpose

Returns the maximum available and stable gain

Synopsis

max_gain(S)

where S is a scattering matrix of 2-port network.

Example

y=max_gain(S)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

MaxGain

Defined in

AEL, rf_system_fun.ael

See also

sm_gamma1, sm_gamma2, stab_fact, stab_meas

Description

Given a 2 x 2 scattering matrix, this measurement returns the maximum available and stable gain between the input and the measurement ports.

max_index

Purpose

Returns the index of the maximum

Synopsis

max_index(x)

The function takes a single argument, so enclose a sequence of numbers in brackets "[x, y, ...]"

Example

```
max_index([1, 2, 3])
2

max_index([3, 2, 1])
0
```

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built-in

See also

min_index

max_outer

Purpose

Computes the maximum across the outer dimension of two-dimensional data.

Synopsis

max_outer(data)

where data must be two-dimensional data.

Example

max_outer(data)

Used in

Not applicable

Available as measurement component?

No, but the function can be used on a schematic page, in a measurement equation.

Defined in

AEL, statistcal_fun.ael

See also

fun_2d_outer, mean_outer, min_outer

This function can be applied to the data in the example: .../examples/Tutorial/DataAccess_prj/Truth_MonteCarlo.dds.

Description

The max function operates on the inner dimension of two-dimensional data. The max_outer function just calls the fun_2d_outer function, with max being the applied operation. As an example, assume that a Monte Carlo simulation of an amplifier was run, with 151 random sets of parameter values, and that for each set the S-parameters were simulated over 26 different frequency points. S21 becomes a [151 Monte Carlo iteration X 26 frequency] matrix, with the inner dimension being frequency, and the outer dimension being Monte Carlo index. Now, assume that it is desired to know the maximum value of the S-parameters at each frequency. Inserting an equation max(S21) computes the maximum value of S21 at each Monte Carlo iteration. If S21 is simulated from 1 to 26 GHz, it computes the maximum value over this frequency range, which usually is not very useful. Inserting an equation max_outer(S21) computes the maximum value of S21 at each Monte Carlo iteration.

mean

Purpose

Returns the mean

Synopsis

mean(x)

The function takes a single argument, so enclose a sequence of numbers in brackets "[x, y, ...]"

Example

mean([1, 2, 3])
2

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cum_prod, cum_sum, max, min, prod, sum

mean_outer

Purpose

Computes the mean across the outer dimension of two-dimensional data.

Synopsis

mean_outer(data)

where data must be two-dimensional data.

Example

mean_outer(data)

Used in

Not applicable

Available as measurement component?

No, but the function can be used on a schematic page, in a measurement equation.

Defined in

AEL, statistcal_fun.ael

See also

fun_2d_outer, max_outer, min_outer
This function can be applied to the data in the example:
.../examples/Tutorial/DataAccess_prj/Truth_MonteCarlo.dds.

Description

The mean function operates on the inner dimension of two-dimensional data. The mean_outer function just calls the fun_2d_outer function, with mean being the applied operation. As an example, assume that a Monte Carlo simulation of an amplifier was run, with 151 random sets of parameter values, and that for each set the S-parameters were simulated over 26 different frequency points. S21 becomes a [151 Monte Carlo iteration X 26 frequency] matrix, with the inner dimension being frequency, and the outer dimension being Monte Carlo index. Now, assume that it is desired to know the mean value of the S-parameters at each frequency. Inserting an equation mean(S21) computes the mean value of S21 at each Monte Carlo iteration. If S21 is simulated from 1 to 26 GHz, it computes the mean value over this frequency range, which usually is not very useful. Inserting an equation mean_outer(S21) computes the mean value of S21 at each Monte Carlo frequency.

median

Purpose

Returns the median

Synopsis

median(x)

The function takes a single argument, so enclose a sequence of numbers in brackets "[x, y, ...]"

Example

median([1, 2, 3, 4])
2.5

Used in

Not applicable

Available as measurement component?

This function can only be entered by means of a Eqn component in the Data Display window. There is no explicit measurement component.

Defined in

AEL, statistcal_fun.ael

See also

mean, sort

min

Purpose

Returns the minimum value of a swept parameter

Synopsis

y = min(x)

The function takes a single argument, so enclose a sequence of numbers in brackets "[x, y, ...]"

Examples

a = min([1, 2, 3]);
1

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cum_prod, cum_sum, max, mean, prod, sum

min_index

Purpose

Returns the index of the minimum

Synopsis

y = min_index(x)

The function takes a single argument, so enclose a sequence of numbers in brackets "[x, y, ...]"

Example

min_index([3, 2, 1])
2

min_index([1, 2, 3])
0

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

max_index

min_outer

Purpose

Computes the minimum across the outer dimension of two-dimensional data.

Synopsis

min_outer(data)

where data must be two-dimensional data.

Example

min_outer(data)

Used in

Not applicable

Available as measurement component?

No, but the function can be used on a schematic page, in a measurement equation.

Defined in

AEL, statistcal_fun.ael

See also

fun_2d_outer, max_outer, mean_outer
This function can be applied to the data in the example:
.../examples/Tutorial/DataAccess_prj/Truth_MonteCarlo.dds.

Description

The min function operates on the inner dimension of two-dimensional data. The min_outer function just calls the fun_2d_outer function, with min being the applied operation. As an example, assume that a Monte Carlo simulation of an amplifier was run, with 151 random sets of parameter values, and that for each set the S-parameters were simulated over 26 different frequency points. S21 becomes a [151 Monte Carlo iteration X 26 frequency] matrix, with the inner dimension being frequency, and the outer dimension being Monte Carlo index. Now, assume that it is desired to know the minimum value of the S-parameters at each frequency. Inserting an equation min(S21) computes the minimum value of S21 at each Monte Carlo iteration. If S21 is simulated from 1 to 26 GHz, it computes the minimum value over this frequency range, which usually is not very useful. Inserting an equation min_outer(S21) computes the minimum value of S21 at each Monte Carlo iteration.

mix

Purpose

Returns a component of a spectrum based on a vector of mixing indices

Synopsis

mix(xOut, harmIndex{, Mix})

where xOut is a voltage or a current spectrum and harmIndex is the desired vector of harmonic frequency indices (mixing terms). Mix is a variable consisting of all possible vectors of harmonic frequency indices (mixing terms) in the analysis.

Example

y = mix(vOut, {2, -1})

z = mix(vOut*vOut/50, {2, -1}, Mix)

Used in

Harmonic balance simulation

Available as measurement component?

This equation can be entered by means of a MeasEqn component in Harmonic balance simulation palette in the Schematic window. There is no explicit Mix measurement component.

Defined in

Built in

See also

find_index

Description

This function returns the mixing component of a voltage or a current spectrum corresponding to particular harmonic-frequency indices or mixing terms. Note that the third argument, Mix, is required whenever the first argument is a spectrum obtained from an expression that operates on the voltage and/or current spectrums.

moving_average

Purpose

Returns the moving_average of a sequence of data

Synopsis

moving_average(data, numPoints)

where data is a one-dimensional sequence of numbers in brackets “[x, y, ...]”, and numPoints is the number of points to be averaged together.

Example

moving_average([1, 2, 3, 7, 5, 6, 10], 3)

[1, 2, 4, 5, 6, 7, 10]

Used in

Not applicable

Available as measurement component?

There is no explicit measurement component, but the function can be used on a schematic page.

Defined in

AEL, statistcal_fun.ael

See also

Not applicable

Description

The first value of the smoothed sequence is the same as the original data. The second value is the average of the first three. The third value is the average of data elements 2, 3, and 4, etc. If numPoints were set to 7, for example, then the first value of the smoothed sequence would be the same as the original data. The second value would be the average of the first three original data points. The third value would be the average of the first five data points, and the fourth value would be the average of the first seven data points. Subsequent values in the smoothed array would be the average of the seven closest neighbors. The last points in the smoothed sequence are computed in a way similar to the first few points. The last point is just the last point in the original sequence. The second from last point is the average of the last three

points in the original sequence. The third from the last point is the average of the last five points in the original sequence, etc.

mu

Purpose

Returns the geometrically derived stability factor for the load

Synopsis

mu(S)

where S is a scattering matrix of a 2-port network.

Examples

x=mu(S)

Used in

Small-signal and large-signal S-parameter simulations.

Available as measurement component?

Mu

Defined in

AEL, circuit_fun.ael

See also

mu_prime

Description

This measurement gives the distance from the center of the Smith chart to the nearest output (load) stability circle.

This stability factor is given by

mu = {1-|S11|**2} / {|S22 - conj(S11)*Delta | + |S12*S21| }

where Delta is the determinant of the S-parameter matrix. Having mu > 1 is the single necessary and sufficient condition for unconditional stability of the 2-port network.

Reference

[1] M. L. Edwards and J. H. Sinsky, “A new criterion for linear 2-port stability using geometrically derived parameters", IEEE Transactions on Microwave Theory and Techniques, Vol. 40, No. 12, pp. 2303-2311, Dec. 1992.

mu_prime

Purpose

Returns the geometrically derived stability factor for the source

Synopsis

mu_prime(S)

where S is a scattering matrix of 2-port network.

Examples

y=mu_prime(S)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

MuPrime

Defined in

AEL, circuit_fun.ael

See also

mu

Description

This measurement gives the distance from the center of the Smith chart to the nearest unstable-input (source) stability circle.

This stability factor is given by

mu_prime = {1-|S22|**2} / {|S11 - conj(S22)*Delta | + |S21*S12| }

where Delta is the determinant of the S-parameter matrix. Having mu_prime > 1 is the single necessary and sufficient condition for unconditional stability of the 2-port network.

Reference

[1] M. L. Edwards and J. H. Sinsky, “A new criterion for linear 2-port stability using geometrically derived parameters”, IEEE Transactions on Microwave Theory and Techniques, Vol. 40, No. 12, pp. 2303-2311, Dec. 1992.

ns_circle

Purpose

Returns noise-figure circles

Synopsis

ns_circle(nf, NFmin, Sopt, rn{, numOfPts})

where nf is the specified noise figure and is set by default to max(NFmin) + {0, 1, 2, 3}. NFmin is the minimum noise figure, Sopt is the optimum mismatch, rn is the equivalent normalized noise resistance of a 2-port network (rn = *Rn*/*zRef* where *Rn* is the equivalent noise resistance and *zRef* is the reference impedance), and numOfPts is the desired number of points per circle and is set to 51 by default.

Example

circleData=ns_circle(0+NFmin, NFmin, Sopt, Rn/50, 51)
circleData=ns_circle({0, 1}+NFmin, NFmin, Sopt, Rn/50, 51)
returns the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

NsCircle

Defined in

AEL, circle_fun.ael

See also

Not applicable

Description

The expression generates constant noise-figure circles. The circles are defined by the loci of the source-reflection coefficients that result in the specified noise figure. NFmin, Sopt, and Rn are generated from noise analysis.

A circle is created for each value of the swept variable(s).

ns_pwr_int

Purpose

Returns the integrated noise power

Synopsis

ns_pwr_int(Sji, nf, resBW)

where Sji is the complex transmission coefficient, nf is noise figure at the output port (in dB), and resBW is the user-defined resolution bandwidth.

Example

Y=ns_pwr_int(S21, nf2, 1MHz)

Used in

Small-signal S-parameter simulation

Available as measurement component?

NsPwrInt

Defined in

AEL, rf_system_fun.ael

See also

ns_pwr_ref_bw, snr

Description

This is the integrated noise power (in dBm) calculated by integrating the noise power over the entire frequency sweep. The noise power at each frequency point is calculated by multiplying the noise spectral density by a user-defined resolution bandwidth.

ns_pwr_ref_bw

Purpose

Returns noise power in a reference bandwidth

Synopsis

Y =ns_pwr_ref_bw(Sji, nf, resBW)

where Sji is the complex transmission coefficient, nf is noise figure at the output port (in dB), and resBW is the user-defined resolution bandwidth.

Example

Y=ns_pwr_ref_bw(S21, nf2, 1MHz)
returns the noise power with respect to the reference bandwidth.

Used in

Small-signal S-parameter simulation

Available as measurement component?

NsPwrRefBW

Defined in

AEL, rf_system_fun.ael

See also

ns_pwr_int, snr

Description

This is the noise power calculated by multiplying the noise spectral density at a frequency point by a user-defined resolution bandwidth. Unlike NsPwrInt, this gives the noise power (in dB) at each frequency sweep.

ones

Purpose

Returns a matrix of ones

Synopsis

Y=ones(2)

This is the ones matrix. If only one argument is supplied, then a square matrix is returned. If two are supplied, then a matrix of ones with size rows × cols is returned.

Example

a = ones(2)
{{1, 1}, {1, 1}}

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

identity, zeros

pae

Purpose

Returns power-added efficiency

Synopsis

pae(vPlusOut, vMinusOut, vPlusIn, vMinusIn, vPlusDC, vMinusDC, iOut, iIn, iDC, outFreq, inFreq)

where vPlusOut and vMinusOut are output voltages at the positive and negative terminals; vPlusIn and vMinusIn are input voltages at the positive and negative terminals; vPlusDC and vMinusDC are DC voltages at the positive and negative terminals; iOut, iIn, and iDc are the output, input, and DC currents, respectively; and outFreq and inFreq are harmonic indices of the fundamental frequency at the output and input port, respectively.

Example

y=pae(vOut, 0, vIn, 0, v1, 0, I_Probe1.i, I_Probe2.i, I_Probe3.i, 1, 1)

Used in

Harmonic balance simulation

Available as measurement component?

PAE

Defined in

AEL, circuit_fun.ael

See also

dB, dBm

Description

This measurement computes the power-added efficiency (in percent) of any part of the circuit.

pdf

Purpose

Returns a probability density function

Synopsis

pdf(x, numBins, minBin, maxBin)

where x is the signal, numBins is number of subintervals or bins used to measure PDF, and minBin and maxBin are the beginning and end, respectively, of the evaluation of the PDF.

Example

y = pdf(data)

y = pdf(data, 20)

Used in

Not applicable

Available as measurement component?

This function can be entered by means of a Eqn component in the Data Display window. There is no measurement component in schematic window

Defined in

AEL statistical_fun.ael

See also

cdf, histogram, yield_sens

Description

This function measures the probability density function of a signal. The default values for minBin and maxBin are the minimum and the maximum values of the data and numBins is set to log(numOfPts)/log(2.0) by default.

permute

Purpose

Permutes data based on the attached independents

Synopsis

y = permute(data, permute_vector)

where data is any N-dimensional square data (all inner independents must have the same value N) and permute_vector is any permutation vector of the numbers 1 through N. The permute_vector defaults to {N::1}, representing a complete reversal of the data with respect to its independent variables. If permute_vector has fewer than N entries, the remainder of the vector, representing the outer independent variables, is filled in. In this way, expressions remain robust when outer sweeps are added.

Examples

a = permute(data)
a = permute(data, {3, 2, 1})
reverses the (three inner independents of) the data.

a = permute(data, {1, 2, 3})
preserves the data.

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

None

pfc

Purpose

Returns frequency-selective power

Synopsis

pfc(vPlus, vMinus, iOut, harm_freq_index)

where vPlus is the voltage at the positive and negative terminals, iOut is the current through a branch, and harm_freq_index is the harmonic index of the desired frequency. Note that the harm_freq_index argument's entry should reflect the number of tones in the harmonic balance controller. For example, if one tone is used in the controller, there should be one number inside the braces; two tones would require two numbers separated by a comma.

Example

The following example is for two tones in the harmonic balance controller:

y=pfc(vOut, 0, I_Probe1.i, {1, 0})

Used in

Harmonic balance simulation

Available as measurement component?

Pfc

Defined in

AEL, circuit_fun.ael

See also

ifc, vfc

Description

This measurement gives the RMS power value of one frequency component of a harmonic balance waveform.

pfc_tran

Purpose

Returns frequency-selective power

Synopsis

pfc_tran(vPlus, vMinus, iOut, fundFreq, harmNum)

where vPlus and vMinus are the voltages at the positive terminals, iOut is the current through a branch measured for power calculation, and fundFreq is fundamental frequency and harmNum is the harmonic number of the fundamental frequency (positive integer value only).

Example

y=pfc_tran(v1, v2, I_Probe1.i, 1GHz, 1)

Used in

Transient simulation

Available as measurement component?

PfcTran

Defined in

AEL, circuit_fun.ael

See also

ifc_tran, vfc_tran

Description

This measurement gives RMS power, delivered to any part of the circuit at a particular frequency of interest. fundFreq determines the portion of the time-domain waveform to be converted to frequency domain. This is typically one full period corresponding to the lowest frequency in the waveform. harmNum is the harmonic number of the fundamental frequency at which the power is requested.

phase

Purpose

Phase in degrees

Synopsis

y = phase(x)

Example

phase(1i)
90

phase(1+1i)
45

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built-in

See also

phaserad

phase_comp

Purpose

Returns the phase compression (phase change)

Synopsis

Y = phase_comp(Sji)

where Sji is a power-dependent parameter obtained from large-signal S-parameters simulation.

Example

y=phase_comp(S21[::, 0])

Used in

Large-signal S-parameter simulations

Available as measurement component?

PhaseComp

Defined in

AEL, rf_systems_fun.ael

See also

gain_comp

Description

This measurement calculates the small-signal minus the large-signal phase, in degrees. The first power point (assumed to be small) is used to calculate the small-signal phase.

phasedeg

Purpose

Phase in degrees

Synopsis

y = phasedeg(x)

Example

phase(1i)
90

phase(1+1i)
45

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built-in

See also

dev_lin_phase, diff, phase, phasedeg, phaserad, ripple, unwrap

phaserad

Purpose

Phase in Radians

Synopsis

y = phaserad(x)

Example

phaserad(1i)
1.5708

phaserad(1+1i)
0.785398

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

dev_lin_phase, diff, phase, phasedeg, phaserad, ripple, unwrap

plot_vs

Purpose

Attaches an independent to data for plotting

Synopsis

plot_vs(dependent, independent)

where dependent is any N-dimensional square data (all inner independents must have the same value N) and permute_vector is any permutation vector of the numbers 1 through N. The permute_vector defaults to {N::1}, representing a complete reversal of the data with respect to its independent variables. If permute_vector has fewer than N entries, the remainder of the vector, representing the outer independent variables, is filled in. In this way, expressions remain robust when outer sweeps are added.

Example

```
a=[1, 2, 3]
b=[4, 5, 6]
c=plot_vs(a, b)
```

Builds c with independent b, and dependent a.

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

AEL, display_fun.ael

See also

indep, vs

polar

Purpose

Builds a complex number from magnitude and angle (in degrees)

Synopsis

polar(mag, angle)

Example

```
polar(1, 90)
0+1i

polar(1, 45)
0.707107+0.707107i
```

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

None

pow

Purpose

Raises an integer or real number to a given power

Synopsis

z = pow(x, y)

where x is the integer or real number and y is the exponent of that number.

Examples

a = pow(4, 2);

returns 16

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, exp, float, int, log10, sgn, sqrt

pspec

Purpose

Returns power frequency spectrum

Synopsis

pspec(vPlus, vMinus, iOut)

where vPlus and vMinus are voltages at the positive terminals, and iOut is the current through a branch measured for power calculation.

Example

y=pspec(vOut, 0, I_Probe1.i)

Used in

Harmonic balance simulation

Available as measurement component?

Pspec

Defined in

AEL, circuit_fun.ael

See also

Not applicable

Description

This measurement gives a power frequency spectrum in harmonic balance analyses.

pspec_tran

Purpose

Returns transient power spectrum

Synopsis

pspec_tran(vPlus, vMinus, iOut, fundFreq, numHarm)

where vPlus and vMinus are the voltages at the positive and negative terminals, iOut is the current through a branch measured for power calculation, fundFreq is the fundamental frequency, and numHarm is the number of harmonics of the fundamental frequency (positive integer value only).

Example

y=pspec_tran(v1, v2, I_Probe1.i, 1GHz, 8)

Used in

Transient simulation

Available as measurement component?

PspecTran

Defined in

AEL, circuit_fun.ael

See also

ispec_tran, vspec_tran

Description

This measurement gives a power spectrum, delivered to any part of the circuit. The measurement gives a set of RMS power values at each frequency. fundFreq is the fundamental frequency determines the portion of the time-domain waveform to be converted to frequency domain (typically one full period corresponding to the lowest frequency in the waveform). numHarm is the number of harmonics of the fundamental frequency to be included in the power spectrum.

prod

Purpose

Returns the product

Synopsis

prod(x)

Example

prod([1, 2, 3]
6

prod([4, 4, 4])
64

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built-in

See also

sum

pt

Purpose

Returns total power

Synopsis

pt(vPlus, vMinus, iOut)

where vPlus and vMinus are the voltages at the positive and negative terminals, respectively, and iOut is the current through a branch.

Example

y=pt(vOut, 0, I_Probe1.i)

Used in

Harmonic balance simulation

Available as measurement component?

Pt

Defined in

AEL, circuit_fun.ael

See also

pspec

Description

This measurement calculates the total power of a harmonic balance frequency spectrum.

pwr_gain

Purpose

Returns power gain

Synopsis

y= pwr_gain(S, Zs, Zl{, Zref})

where S is the 2×2 scattering matrix, and Zs and Zl are the input and output impedances, respectively. Zref is the reference impedance, set by default to the port impedance.

Example

y=pwr_gain(S, 50, 75)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

PwrGain

Defined in

AEL, rf_system_fun.ael

See also

stos, volt_gain, volt_gain_max

Description

This measurement is used to determine the power gain (in dB), i.e. the power delivered to the load minus the power available from the source (where power is in dBm).

rad

Purpose

Degrees to radians

Synopsis

rad(x)

Example

rad(90)
1.5708

rad(45)
0.785398

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

deg

real

Purpose

Returns the real part of a complex number

Synopsis

y = real(x)

where x is a complex number.

Examples

a = real(1–1j);

returns 1

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cmplx, imag

relative_noise_bw

Purpose

Computes the relative noise bandwidth of the smoothing windows used by the fs() function

Synopsis

RelNoiseBW=relative_noise_bw(winType, winConst)

where

winType is a window type and must be one of the following: Kaiser, Hamming, Gaussian, 8510, or NoWindow (leaving this field blank is the equivalent of NoWindow); and

winConst is an optional parameter that affects the shape of the applied window. The default window constants are as follows:

Kaiser: 7.865

Hamming: 0.54

Gaussian: 0.75

8510: 6 (The 8510 window is the same as a Kaiser window with a window constant of 6.)

Example

Example equations

winType = Kaiser
winConst = 8
relNoiseBW = relative_noise_bw(winType, winConst)
Vfund=vOut[1]
VoltageSpectralDensity = 0.5 * fs(Vfund, , , , , winType, winConst)
PowerSpectralDensity = 0.5 * mag(VoltageSpectralDensity**2)/50/relNoiseBW

where vOut is the named connection at a 50-ohm load, and it is an output from a Circuit Envelope simulation.

Note vOut is a named connection on the schematic. Assuming that a Circuit Envelope simulation was run, vOut is output to the dataset as a two-dimensional matrix. The first dimension is time, and there is a value for each time point in the simulation. The second dimension is frequency, and there is a value for each fundamental frequency, each harmonic, and each mixing term in the analysis, as well as the baseband term.

vOut[1] is the equivalent of vOut[::, 1], and specifies all time points at the lowest non-baseband frequency (the fundamental analysis frequency, unless a multitone analysis has been run and there are mixing products). For former MDS users, the notation "vOut[*, 2]" in MDS corresponds to the ADS notation of "vOut[1]".

Used in

The following functions: acpr_vi, acpr_vr, channel_power_vi, channel_power_vr

Available as measurement component?

Equations listed under Description can be entered by means of a MeasEqn component in the Schematic window. There is no explicit relative noise bandwidth measurement function.

Defined in

hpeesof/expressions/ael/digital_wireless_fun.ael

See also

acpr_vi, acpr_vr, channel_power_vi, channel_power_vr, fs

Description

The relative noise bandwidth function is used to account for the fact that as windows are applied, the effective noise bandwidth increases with respect to the normal resolution bandwidth. The resolution bandwidth is determined by the time span and not by the displayed frequency resolution.

ripple

Purpose

Returns deviation from the average

Synopsis

ripple(x)

where x can be a gain or group delay data over a given frequency range.

Example

y=ripple(pwr_gain(S21))

Used in

Not applicable

Available as measurement component?

GainRipple

Defined in

AEL, elementary_fun.ael

See also

dev_lin_phase, diff, mean, phasedeg, phaserad, unwrap

Description

This function measures the deviation of x from the average of x.

round

Purpose

Rounds to the nearest integer

Synopsis

round(x)

Example

round(0.1)
0

round(0.5)
1

round(0.9)
1

round(–0.1)
0

round(–0.5)
–1

round(–0.9)
–1

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

int

s_stab_circle

Purpose

Returns source (input) stability circles

Synopsis

s_stab_circle(S{, numOfPts})

where S is the scattering matrix of a 2-port network and numOfPts is the desired number of points per circle and is set to 51 by default.

Example

circleData = s_stab_circle(S, 51)
returns the points on the circle(s).

Used in

Small-signal S-parameter simulations

Available as measurement component?

S_StabCircle

Defined in

AEL, circle_fun.ael

See also

l_stab_circle, l_stab_region, s_stab_region

Description

This expression generates source stability circles. The circles are defined by the loci of source-reflection coefficients where the magnitude of the load-reflection coefficient is 1.

A circle is created for each value of the swept variable(s). This measurement is supported for 2-port networks only.

s_stab_region

Purpose

Indicates the region of stability of the source (input) stability circle

Synopsis

s_stab_region(S)

where S is the scattering matrix of a 2-port network.

Example

region = s_stab_region(S)
returns “Outside” or “Inside”.

Used in

Small-signal S-parameter simulations

Available as measurement component?

Not applicable

Defined in

AEL, circle_fun.ael

See also

l_stab_circle, l_stab_region, s_stab_circle

Description

This expression returns a string identifying the region of stability of the corresponding source stability circle.

sample_delay_pi4dqpsk

Purpose

This function calculates the optimal sampling point within a symbol for a given pi4dqpsk waveform.

Synopsis

sample_delay_pi4dqpsk(vlQ, symbolRate, delay, timeResolution)

where

vlQ is the complex envelope (I + j * Q) of a pi/4 DQPSK signal.

symbolRate is the symbol rate of the pi/4 DQPSK signal.

path is the time delay on the waveform before the sampling starts. If the delay is 0, this parameter may be omitted. If it is non-zero, enter the delay value. This can be calculated using the function delay_path().

timeResolution is the time step (typically one-tenth of a symbol time or less) used to search for the best sampling point in a given symbol period.

Example

a = sample_delay_pi4dqpsk(vout[1], 25e3, 1.5e-6, 0.15e-6)

Used in

Envelope simulation

Available as measurement component?

Not applicable

Defined in

Built in

See also

ber_pi4dqpsk, ber_qpsk, const_evm

Description

Calculates the optimal sampling point for a given waveform. "Optimal" is defined as the sampling point that provides the lowest bit error rate.

sample_delay_qpsk

Purpose

This function calculates the optimal sampling point within a symbol for a given QPSK waveform.

Synopsis

sample_delay_qpsk(vlQ, symbolRate, delay, timeResolution)

where

vlQ is the complex envelope (I + j * Q) of a QPSK signal.

symbolRate is the symbol rate of the QPSK signal.

path is the time delay on the waveform before the sampling starts. If the delay is 0, this parameter may be omitted. If it is non-zero, enter the delay value. This can be calculated using the function delay_path().

timeResolution is the time step (typically one-tenth of a symbol time or less) used to search for the best sampling point in a given symbol period.

Example

a = sample_delay_qpsk(vout[1], 25e3, 1.5e-6, 0.15e-6)

Used in

Envelope simulation

Available as measurement component?

Not applicable

Defined in

Built in

See also

ber_pi4dqpsk, ber_qpsk, const_evm

Description

Calculates the optimal sampling point for a given waveform. "Optimal" is defined as the sampling point that provides the lowest bit error rate.

set_attr

Purpose

Sets a data attribute

Synopsis

a = set_attr(data, "attr_name", attribute_value)

Example

set_attr(data, "TraceType", "Spectral")

set_attr(data, "TraceType", 10GHz)

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

get_attr

sfdr

Purpose

Returns the spurious-free dynamic range

Synopsis

sfdr(vOut, ssgain, nf, noiseBW, fundFreq, imFreq, zRef)

where vOut is the output voltage, ssgain is the small-signal gain (in dB), nf is the noise figure at the output port, noiseBW is the noise bandwidth, fundFreq and imFreq are the harmonic frequency indices for the fundamental and intermodulation frequencies, respectively, and *zRef* is the reference impedance.

Example

y=sfdr(vIn, 12, nf2, , {1, 0}, {2, -1}, 50)

Used in

Small-signal S-parameter simulations

Available as measurement component?

SFDR

Defined in

AEL, rf_system_fun.ael

See also

ip3_out

Description

This measurement determines the spurious-free dynamic-range ratio for noise power with respect to the reference bandwidth. *zRef* is an optional parameter that, if not specified, is set to 50.0 ohms.

sgn

Purpose

Returns the integer sign of an integer or real number, as either 1 or –1

Synopsis

y = sgn(x)

where x is an integer or real number.

Examples

a = sgn(–1)
returns –1

a=sgn(1)
returns 1

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, exp, float, int, log10, pow, sqrt

sin

Purpose

Returns the sine of an integer or real number

Synopsis

y = sin(x)

where x is an integer or real number, in radians.

Examples

a = sin(pi/2)
returns 1

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cos, tan

sinc

Purpose

Returns the sinc of an integer or real number

Synopsis

y = sinc(x)

where x is an integer or real number, in radians.

Examples

a = sinc(0.5)
0.637

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

sin

Description

The sinc function is defined as sinc(x) = sin(pi*x)/ (pi*x) and sinc(0)=1.

sinh

Purpose

hyperbolic sin

Synopsis

sinh()

Example

sinh(0)
0

sinh(1)
1.1752

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cosh, tanh

size

Purpose

Returns the row and column size of a vector or matrix

Synopsis

Y = size(X)

Example

Given 2-port S-parameters versus frequency, and given 10 frequency points. Then for ten 2×2 matrices, size() returns the dimensions of the S-parameter matrix, and its companion function sweep_size() returns the size of the sweep:

size(S)
returns {2, 2}

sweep_size(S)
returns 10

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

sweep_size

sm_gamma1

Purpose

Returns the simultaneous-match input-reflection coefficient

Synopsis

sm_gamma1(S)

where S is a scattering matrix of 2-port network.

Example

y=sm_gamma1(S)

Used in

Small-signal and large-signal S-parameter simulations.

Available as measurement component?

SmGamma1

Defined in

AEL, circuit_fun.ael

See also

max_gain, sm_gamma2, stab_fact, stab_meas

Description

This complex measurement determines the reflection coefficient that must be presented to the input (port 1) of the network to achieve simultaneous input and output reflections. If the Rollett stability factor stab_fact(S) is less than unity for the analyzed circuit, then sm_gamma1(S) returns zero. It is, in effect, undefined when stab_fact(S) < 1.

sm_gamma2

Purpose

Returns the simultaneous-match output-reflection coefficient

Synopsis

sm_gamma2(S)

where S is a scattering matrix of 2-port network.

Example

y=sm_gamma2(S)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

SmGamma2

Defined in

AEL, rf_system_fun.ael

See also

max_gain, sm_gamma1, stab_fact, stab_meas

Description

This complex measurement determines the reflection coefficient that must be presented to the output (port 2) of the network to achieve simultaneous input and output reflections. If the Rollett stability factor stab_fact(S) is less than unity for the analyzed circuit, then sm_gamma2(S) returns zero. It is, in effect, undefined when stab_fact(S) < 1.

sm_y1

Purpose

Returns the simultaneous-match input admittance

Synopsis

sm_y1(S, Z)

where S is a scattering matrix of a 2-port network, and Z is a port impedance.

Example

y=sm_y1(S, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

SmY1

Defined in

AEL, rf_system_fun.ael

See also

sm_y2

Description

This complex measurement determines the admittance that must be presented to the input (port 1) of the network to achieve simultaneous input and output reflections.

sm_y2

Purpose

Returns the simultaneous-match output admittance

Synopsis

sm_y2(S, Z)

where S is a scattering matrix of 2-port network.and
Z is a port impedance.

Example

y=sm_y2(S, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

SmY2

Defined in

AEL, circuit_fun.ael

See also

sm_y1

Description

This complex measurement determines the admittance that must be presented to the input (port 2) of the network to achieve simultaneous input and output reflections.

sm_z1

Purpose

Returns the simultaneous-match input impedance

Synopsis

sm_z1(S, Z)

where S is a scattering matrix of a 2-port network, and Z is a port impedance.

Example

y=sm_z1(S, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

SmZ1

Defined in

AEL, circuit_fun.ael

See also

sm_z2

Description

This complex measurement determines the impedance that must be presented to the input (port 1) of the network to achieve simultaneous input and output reflections.

sm_z2

Purpose

Returns the simultaneous-match output impedance

Synopsis

Y = sm_z2(S, Z)

where S is a scattering matrix of 2-port network, and Z is a port impedance.

Example

y=sm_z2(S, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

SmZ2

Defined in

AEL, circuit_fun.ael

See also

sm_z1

Description

This complex measurement determines the impedance that must be presented to the output (port 2) of the network to achieve simultaneous input and output reflections.

snr

Purpose

Returns the signal-to-power noise ratio

Synopsis

snr(vOut, vOut.noise{, fundFreq})

where vOut and vOut.noise are the signal and noise voltages at the output port, and fundFreq is the harmonic frequency index for the fundamental frequency. Note that fundFreq is not optional; it is required for harmonic balance simulations, but it is not applicable in AC simulations.

Example

y=snr(vOut, vOut.noise, {1, 0})
returns the signal-to-power noise ratio for a harmonic balance simulation.

y=snr(vOut, vOut.noise)
returns the signal-to-power noise ratio for an AC simulation.

Used in

Harmonic balance simulations

Available as measurement component?

SNR

Defined in

AEL, rf_system_fun.ael

See also

ns_pwr_int, ns_pwr_ref_bw

Description

This measurement gives the ratio of the output signal power (at the fundamental frequency for a harmonic balance simulation) to the total noise power (in dB).

sort

Purpose

Returns a sorted variable

Synopsis

sort(data, sortOrder, indepName)

where data is a multidimensional scalar variable, *sortOrder* is the sorting order, {“ascending”, ”decending”}. (If not specified, it is set to “ascending.”) *indepName* is used to specify the name of the independent variable for sorting. (If not specified, the sorting is done on the dependent.)

Example

y = sort(data)

y = sort(data, ”decending”, ”freq”)

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

None

Description

This measurement returns a sorted variable in ascending or descending order. The sorting can be done on the independent or dependent variables. String values are sorted by folding them to lower case.

spec_power

Purpose

Returns the integrated signal power (dBm) of a spectrum

Synopsis

spec_power(sinkInstanceName{, lowerFrequencyLimit, upperFrequencyLimit})

where sinkInstanceName is the instance of the FFTAnalyzer or SpecAnalyzer sink in the DSP schematic window (values in dBm). lowerFrequencyLimit and upperFrequencyLimit are optional and define the lower and upper frequency limits to be used in calculating the integrated power. The frequency unit used must match that used in SpecAnalyzer. The default unit is MHz. The entire spectral frequency range will be used if lowerFrequencyLimit and upperFrequencyLimit are not specified.

Example

total_power=spec_power(Mod_Spectrum, 60, 71)
returns the integrated power between 60 and 71 MHz

total_power=spec_power(Mod_Spectrum, indep(m1), indep(m2))
returns the integrated power between markers 1 and 2

where Mod_Spectrum is the instance ID of an FFTAnalyzer or SpecAnalyzer sink

Used in

Agilent Ptolemy simulations

Available as measurement component?

This function is available for use in a MeasEqn component.

Defined in

AEL, signal_proc_fun.ael

See also

None

Description

This function will return the total integrated power (dBm) of a spectrum. The frequency window over which the integrated power will be calculated can be specified, otherwise the entire spectral frequency range will be used.

The FFTAnalyzer and SpecAnalyzer sinks are valid for this measurement. These sinks should have a termination resistor shunted to ground at their input for a measurement referenced to an impedance such as 50 Ohms. This termination resistor value should be set to the Ref_Resistance value specified in the FFTAnalyzer or SpecAnalyzer sink. The display parameter of the sinks must be set to dBm.

spur_track

Purpose

Returns the maximum power of all signals appearing in a user-specifiable IF band, as a single RF input signal is stepped. If there is no IF signal appearing in the specified band, for a particular RF input frequency, then the function returns an IF signal power of -500 dBm.

Synopsis

IFspur=spur_track(vs(vout, freq), if_low, if_high, rout)

where vout is the IF output node name, if_low is the lowest frequency in the IF band, if_high is the highest frequency in the IF band, rout is the load resistance connected to the IF port, necessary for computing power delivered to the load. IFspur computed above will be the power in dBm of the maximum signal appearing in the IF band, versus RF input frequency. Note that it would be easy to modify the function to compute dBV instead of dBm.

Example

IFspur=spur_track(vs(HB.VIF1, freq), Fiflow[0, 0], Fifhigh[0, 0], 50)

where VIF1 is the named node at the IF output, Fiflow is the lowest frequency in the IF band, Fifhigh is the highest frequency in the IF band, and 50 is the IF load resistance. Fiflow and Fifhigh are passed parameters from the schematic page (although they can be defined on the data display page instead.) These parameters, although single-valued on the schematic, become matrices when passed to the dataset, where each element of the matrix has the same value. The [0, 0] syntax just selects one element from the matrix.

Used in

Receiver spurious response simulations

Available as measurement component?

No, but the function can be used on a schematic page, in a measurement equation.

Defined in

AEL, digital_wireless_fun.ael

See also

spur_track_with_if

This function can be applied to the data in the example: .../examples/Com_Sys/Spur_Track_prj/MixerSpurs2MHz.dds.

Description.

This function is meant to aid in testing the response of a receiver to RF signals at various frequencies. This function shows the maximum power of all signals appearing in a user-specifiable IF band, as a single RF input signal is stepped. There could be fixed, interfering tones present at the RF input also, if desired. The maximum IF signal power may be plotted or listed versus the stepped RF input signal frequency. If there is no IF signal appearing in the specified band, for a particular RF input frequency, then the function returns an IF signal power of -500 dBm.

spur_track_with_if

Purpose

Returns the maximum power of all signals appearing in a user-specifiable IF band, as a single RF input signal is stepped. In addition, it shows the IF frequencies and power levels of each signal that appears in the IF band, as well as the corresponding RF signal frequency.

Synopsis

IFspur=spur_track_with_if(vs(vout, freq), if_low, if_high, rout)

where vout is the IF output node name, if_low is the lowest frequency in the IF band, if_high is the highest frequency in the IF band, rout is the load resistance connected to the IF port, necessary for computing power delivered to the load. IFspur computed above will be the power in dBm of the maximum signal appearing in the IF band, versus RF input frequency. Note that it would be easy to modify the function to compute dBV instead of dBm.

Example

IFspur=spur_track_with_if(vs(HB.VIF1, freq), Fiflow[0, 0], Fifhigh[0, 0], 50)

where VIF1 is the named node at the IF output, Fiflow is the lowest frequency in the IF band, Fifhigh is the highest frequency in the IF band, and 50 is the IF load resistance. Fiflow and Fifhigh are passed parameters from the schematic page (although they can be defined on the data display page instead.) These parameters, although single-valued on the schematic, become matrices when passed to the dataset, where each element of the matrix has the same value. The [0, 0] syntax just selects one element from the matrix.

Used in

Receiver spurious response simulations

Available as measurement component?

No, but the function can be used on a schematic page, in a measurement equation.

Defined in

AEL, digital_wireless_fun.ael

See also

spur_track

This function can be applied to the data in the example: .../examples/Com_Sys/Spur_Track_prj/MixerSpurs2MHz.dds.

Description

This function is meant to aid in testing the response of a receiver to RF signals at various frequencies. This function, similar to the spur_track function, shows the maximum power of all signals appearing in a user-specifiable IF band, as a single RF input signal is stepped. In addition, it shows the IF frequencies and power levels of each signal that appears in the IF band, as well as the corresponding RF signal frequency. There could be fixed, interfering tones present at the RF input also, if desired. The maximum IF signal power may be plotted or listed versus the stepped RF input signal frequency.

sqrt

Purpose

Returns the square root of a positive integer or real number

Synopsis

y = sqrt(x)

where x is a positive integer or real number.

Examples

a = sqrt(4)
returns 2

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

abs, cint, exp, float, int, log10, pow, sgn

stab_fact

Purpose

Returns the Rollett stability factor

Synopsis

stab_fact(S)

where S is the scattering matrix of a 2-port network.

Example

k = stab_fact(S)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

StabFact

Defined in

AEL, rf_system_fun.ael

See also

max_gain, sm_gamma1, sm_gamma2, stab_meas

Description

Given a 2 x 2 scattering matrix between the input and measurement ports, this function calculates the stability factor.

The Rollett stability factor is given by

k = {1- |S11|**2 - |S22|**2 + |S11*S22 - S12*S21| **2} / {2*|S12*S21|}

The necessary and sufficient conditions for unconditional stability are that the stability factor is greater than unity and the stability measure is positive.

Reference

[1] Guillermo Gonzales, Microwave Transistor Amplifiers, second edition, Prentice-Hall, 1997.

stab_meas

Purpose

Returns the stability measure

Synopsis

stab_meas(S)

where S is the scattering matrix of a 2-port network.

Example

b = stab_meas(S)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

StabMeas

Defined in

AEL, rf_system_fun.ael

See also

max_gain, sm_gamma1, sm_gamma2, stab_fact

Description

Given a 2 x 2 scattering matrix between the input and measurement ports, this function calculates the stability measure.

The stability measure is given by

b = 1+ |S11|**2 - |S22|**2 - |S11*S22 - S12*S21| **2

The necessary and sufficient conditions for unconditional stability are that the stability factor is greater than unity and the stability measure is positive.

Reference

[1] Guillermo Gonzales, Microwave Transistor Amplifiers, second edition, Prentice-Hall, 1997.

stddev

Purpose

Returns the standard deviation

Synopsis

stddev(x{, flag})

where x is the data and flag is used to indicate how stddev normalizes. By default, flag is set to 0, which means that stddev normalizes by N-1, where N is the length of the data sequence. Otherwise, stddev normalizes by N.

Example

y = stddev(data)

y = stddev(data, 1)

Used in

Not applicable

Available as measurement component?

This function can only be entered by means of a Eqn component in the Data Display window.

Defined in

AEL, statistical_fun.ael

See also

mean

Description

This function calculates the standard deviation of the data.

stoabcd

Purpose

Performs S-to-ABCD conversion

Synopsis

stoabcd(S, zRef)

where S is a scattering matrix of a 2-port network and zRef is a reference impedance.

Example

a = stoabcd(S, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

abcdtoh, stoh, stoy

Description

This measurement transforms the scattering matrix of a 2-port network to a chain (ABCD) matrix.

stoh

Purpose

Performs S-to-H conversion

Synopsis

stoh(S, zRef)

where S is a scattering matrix of a 2-port network and zRef is a reference impedance.

Example

h = stoh(S, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

htos, stoabcd, stoy

Description

This measurement transforms the scattering matrix of a 2-port network to a hybrid matrix.

stos

Purpose

Performs S-to-S conversion

Synopsis

stos(S, zRef, zNew)

where S is a scattering matrix, zRef is a normalizing impedance, and zNew is a new normalizing impedance.

Example

y = stos(S, 50, 75)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

stoy, stoz

Description

This function changes the normalizing impedance of a scattering matrix.

stoy

Purpose

Performs S-to-Y conversion

Synopsis

stoy(S, zRef)

where S is a scattering matrix of a 2-port network and zRef is a reference impedance.

Example

y = stoy (S, 50.0)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

stoh, stoz, ytos

Description

This measurement transforms a scattering matrix to an admittance matrix.

stoz

Purpose

Performs S-to-Z conversion

Synopsis

stoz(S, Z0)

where S is a scattering matrix of a 2-port network and z0 is a reference impedance.

Example

z = stoz(S, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

stoh, stoy, ztos

Description

This measurement transforms a scattering matrix to an impedance matrix.

sum

Purpose

Returns the sum

Synopsis

Y = sum(X)

Example

a = sum([1, 2, 3])
returns 6

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

max, mean, min

sweep_dim

Purpose

Returns the dimensionality of the data

Synopsis

sweep_dim(x)

Example

```
sweep_dim(1)
0

sweep_dim([1, 2, 3])
1
```

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

sweep_size

sweep_size

Purpose

Returns the sweep size of a data object

Synopsis

Y = sweep_size(X)

This function returns a vector with an entry corresponding to the length of each sweep.

Example

Given 2-port S-parameters versus frequency, and given 10 frequency points, there are then ten 2×2 matrices. sweep_size() is used to return the sweep size of the S-parameter matrix, and its companion function size() returns the dimensions of the S-parameter matrix itself:

a=sweep_size(S)
returns 10

size(S)
returns {2, 2}

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

size, sweep_dim

tan

Purpose

Returns the tangent of an integer or real number

Synopsis

y = tan(x)

where x is an integer or real number, in radians.

Examples

a = tan(pi/4)
returns 1

a = tan(+/-pi/2)
returns +/- 1.633E16

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

cos, sin

tanh

Purpose

hyperbolic tangent

Synopsis

tanh(x)

Example

tanh(0)
0

tanh(1)
0.761594

tanh(–1)
–0.761594

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

sinh, cosh

trajectory

Purpose

Generates the trajectory diagram from I and Q data, which are usually produced by a Circuit Envelope simulation

Synopsis

Traj = trajectory(i_data, q_data)

where

i_data is the in-phase component of data versus time of a single complex voltage spectral component (for example, the fundamental). This could be a baseband signal instead, but in either case it must be real valued versus time.

q_data is the quadrature-phase component of data versus time of a single complex voltage spectral component (for example, the fundamental). This could be a baseband signal instead, but in either case it must be real valued versus time

Examples

Rotation = –0.21

Vfund=vOut[1] *exp(j * Rotation)

Vimag = imag(Vfund)

Vreal = real(Vfund)

Traj = trajectory(Vreal, Vimag)

where Rotation is a user-selectable parameter that rotates the trajectory diagram by that many radians, and vOut is the named connection at a node.

Note vOut is a named connection on the schematic. Assuming that a Circuit Envelope simulation was run, vOut is output to the dataset as a two-dimensional matrix. The first dimension is time, and there is a value for each time point in the simulation. The second dimension is frequency, and there is a value for each fundamental frequency, each harmonic, and each mixing term in the analysis, as well as the baseband term.

vOut[1] is the equivalent of vOut[::, 1], and specifies all time points at the lowest non-baseband frequency (the fundamental analysis frequency, unless a multitone analysis has been run and there are mixing products). For former MDS users, the notation "vOut[*, 2]" in MDS corresponds to the ADS notation of "vOut[1]".

Used in

Trajectory diagram generation

Available as measurement component?

Equations listed under Description can be entered by means of a MeasEqn component in the Schematic window. There is no explicit trajectory measurement function.

Defined in

hpeesof/expressions/ael/digital_wireless_fun.ael

See also

constellation, const_evm

Description

The I and Q data do not need to be baseband waveforms. For example, they could be the in-phase (real or I) and quadrature-phase (imaginary or Q) part of a modulated carrier. The user must supply the I and Q waveforms versus time.

transpose

Purpose

Transposes a matrix

Synopsis

Y = transpose(y)

This function transposes a matrix, but does not perform a conjugate transpose for complex matrices.

Example

a={{1, 2}, {3, 4}}
b=transpose(a)
returns {{1, 3}, {2, 4}}

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

None

ts

Purpose

Performs a frequency-to-time transform

Synopsis

ts(x, *tstart, tstop, numtpts, dim, windowType, windowConst, nptsspec*)

See detailed Description below.

Example

The following examples of ts assume that a harmonic balance simulation was performed with a fundamental frequency of 1 GHz and order = 8:

Y=ts(vOut) returns the time series (0, 20ps, ... , 2ns), 0 to 5 ns.

Y=ts(vOut, 0, 1ns) returns the time series (0, 10ps, ..., 1ns).

Y=ts(vOut, 0, 10ns, 201) returns the time series (0, 50ps, ... , 10ns).

Y=ts(vOut, , , , , , , 3) returns the time series (0, 20ps, ... , 2ns), but only uses harmonics from 1 to 3 GHz.

Used in

Harmonic balance simulations

Available as measurement component?

Not applicable

Defined in

Built in

See also

fft, fs, fspot

Description

ts(x) returns the time domain waveform from a frequency spectrum. When x is a multidimensional vector, the transform is evaluated for each vector in the specified dimension. For example, if x is a matrix, then ts(x) applies the transform to every row of the matrix. If x is three dimensional, then ts(x) is applied in the lowest dimension over the remaining two dimensions. The dimension over which to apply the transform may be specified by dimension; the default is the lowest dimension (dimension=1).

x must be numeric. It will typically be data from a harmonic balance analysis.

By default, two cycles of the waveform are produced with 101 points, starting at time zero, based on the lowest frequency in the input spectrum. These may be changed by setting tstart, tstop, or numtpts.

All of the harmonics in the spectrum will be used to generate the time domain waveform. When the higher-order harmonics are known not to contribute significantly to the time domain waveform, only the first n harmonics may be requested for the transform, by setting nptsspec = n.

The data to be transformed may be windowed by a window specified by windowType, with an optional window constant windowConst. The window types allowed and their default constants are:

0 = None

1 = Hamming 0.54

2 = Hanning 0.50

3 = Gaussian 0.75

4 = Kaiser 7.865

5 = 8510 6.0 (This is equivalent to the frequency-to-time transformation with normal gate window setting in the 8510 series network analyzer.)

6 = Blackman

7 = Blackman-Harris

windowType can be specified either by the number or by the name.

type

Purpose

Returns the type of the data

Synopsis

type(x)

Returns a string, which is one of “Integer”, “Real”, “Complex” or “String”

Example

type(1)
“Integer”

type(1i)
“Complex”

type(“type”)
“String”

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

what

unwrap

Purpose

Unwraps phase

Synopsis

y = unwrap(phase{, jump})

where phase is a swept real variable and jump is the absolute jump. By default, jump is set to 180.

Example

unwrap(phase(S21))

unwrap(phaserad(S21, pi))

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built-in

See also

dev_lin_phase, diff, phase, phasedeg, phaserad, ripple, unwrap

Description

This measurement unwraps a phase by changing an absolute jump greater than jump to its 2*jump complement.

vfc

Purpose

Returns frequency-selective voltage

Synopsis

vfc(vPlus, vMinus, harm_freq_index)

where vPlus and vMinus are the voltages at the positive and negative terminals, and harm_freq_index is the harmonic index of the desired frequency. Note that the harm_freq_index argument's entry should reflect the number of tones in the harmonic balance controller. For example, if one tone is used in the controller, there should be one number inside the braces; two tones would require two numbers separated by a comma.

Example

The following example is for two tones in the harmonic balance controller:

y=vfc(vOut, 0, {1, 0})

Used in

Harmonic balance simulation

Available as measurement component?

Vfc

Defined in

AEL, circuit_fun.ael

See also

ifc, pfc

Description

This measurement gives the RMS voltage value of one frequency-component of a harmonic balance waveform.

vfc_tran

Purpose

Returns the transient frequency-selective voltage

Synopsis

vfc_tran(vPlus, vMinus, fundFreq, harmNum)

where vPlus and vMinus are the voltages at the positive and negative terminals, fundFreq is the fundamental frequency, and harmNum is the harmonic number of the fundamental.

Example

y=vfc_tran(vOut, 0, 1GHz, 1)

Used in

Transient simulations

Available as measurement component?

VfcTran

Defined in

AEL, circuit_fun.ael

See also

ifc_tran, pfc_tran

Description

This measurement gives the RMS voltage across any two nodes at a particular frequency of interest. The fundamental frequency determines the portion of the time-domain waveform to be converted to frequency domain. This is typically one full period corresponding to the lowest frequency in the waveform. The harmonic number is the fundamental frequency at which the voltage is requested (positive integer value only).

volt_gain

Purpose

Returns the voltage gain

Synopsis

y= volt_gain(S, Zs, Zl{, Zref})

where S is the 2×2 scattering matrix, and Zs and Zl are the input and output impedances, respectively. Zref is the reference impedance, set by default to the port impedance.

Example

y=volt_gain(S, 50, 75)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

VoltGain

Defined in

circuit_fun.ael

See also

pwr_gain, volt_gain_max

Description

This measurement determines the ratio of the voltage across the load to the voltage available from the source. The network-parameter transformation function “stos” can be used to change the normalizing impedance of the scattering matrix.

volt_gain_max

Purpose

Returns the voltage gain at maximum power transfer

Synopsis

Y = volt_gain_max(S, Zs, Zl{, Zref})

where S is the 2×2 scattering matrix, and Zs and Zl are the input and output impedances, respectively. Zref is the reference impedance, set by default to the port impedances.

Example

y=volt_gain_max(S, 50, 75)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

Not available

Defined in

AEL, rf_system_fun.ael

See also

pwr_gain, volt_gain

Description

This measurement determines the ratio of the voltage across the load to the voltage available from the source at maximum power transfer. The network-parameter transformation function “stos” can be used to change the normalizing impedance of the scattering matrix.

vs

Purpose

Attaches an independent to data

Synopsis

vs(dependent, independent)

Example

```
a=[1, 2, 3]
b=[4, 5, 6]
c=vs(a, b)
```

Builds c with independent b, and dependent a.

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

indep

vspec_tran

Purpose

Returns the transient voltage spectrum

Synopsis

vspec_tran(vPlus, vMinus, fundFreq, numHarm)

where vPlus and vMinus are the voltages at the positive and negative terminals, fundFreq is the fundamental frequency, and numHarm is the number of harmonics of the fundamental frequency (positive integer value only).

Example

y=vspec_tran(v1, v2, 1GHz, 8)

Used in

Transient simulation

Available as measurement component?

VspecTran

Defined in

AEL, circuit_fun.ael

See also

ispec_tran, pspec_tran

Description

This measurement gives a voltage spectrum across any two nodes. The measurement gives a set of RMS voltages at each frequency. The fundamental frequency determines the portion of the time-domain waveform to be converted to the frequency domain. This is typically one full period corresponding to the lowest frequency in the waveform. numHarm is the number of harmonics of the fundamental frequency to be included in the voltage spectrum.

vswr

Purpose

Returns the voltage standing-wave ratio (VSWR)

Synopsis

vswr(Sii)

where Sii is the complex reflection coefficient.

Example

y=vswr(S11)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

VSWR

Defined in

AEL, rf_system_fun.ael

See also

yin, zin

Description

Given a complex reflection coefficient, this measurement returns the voltage standing wave ratio.

vt

Purpose

Returns time-domain voltage waveform

Synopsis

vt(vPlus, vMinus, tmin, tmax, numOfPnts)

where vPlus and vMinus are the voltages at the positive and negative nodes, respectively, tmin and tmin are the start time and stop time, respectively, and numOfPts is the number of points (integer values only).

Example

y=vt(vOut, 0, 0, 10nsec, 201)

Used in

Harmonic balance simulation

Available as measurement component?

Vt

Defined in

AEL, circuit_fun.ael

See also

it

Description

This measurement converts a harmonic-balance voltage frequency spectrum to a time-domain voltage waveform.

vt_tran

Purpose

Returns the transient time-domain voltage waveform

Synopsis

Y = vt_tran(vPlus, vMinus)

where vPlus and vMinus are the terminals across which the voltage is measured.

Example

y=vt_tran(v1, v2)

Used in

Transient simulations

Available as measurement component?

VtTran

Defined in

AEL, circuit_fun.ael

See also

vt

Description

This measurement produces a transient time-domain voltage waveform for specified nodes. vPlus and vMinus are the nodes across which the voltage is measured.

what

Purpose

Returns size and type of data

Synopsis

what(x)

Example

None

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

type

Description

what() is used to find out the dimensions of a piece of data, the attached independents, the type, and (in the case of a matrix) the number of rows and columns. Use what() by entering a listing column and using the trace expression what(x).

yield_sens

Purpose

Returns the yield as a function of a design variable

Synopsis

yield_sens(pf_data{, numBins})

where pf_data is a binary-valued scalar data set indicating the pass/fail status of each value of a companion independent variable, and numBins is the number of subintervals or bins used to measure yield_sens.

Example

yield_sens(pf_data)

yield_sens(pf_data, 20)

Used in

Monte Carlo simulation

Available as measurement component?

This function can only be entered by means of a Eqn component in the Data Display window (or by choosing *Insert* > *Equation,* or clicking the Eqn button on the left side of the window). There is no measurement component in schematic window.

Defined in

AEL, statistical_fun.ael

See also

cdf, histogram, pdf

Description

This function measures the yield as a function of a design variable. The default value for numBins is set to log(numOfPts)/log(2.0) by default. For more information and an example refer to "Creating a Sensitivity Histogram" on page 3-18 in the *Tuning, Optimization and Statistical Design* manual.

yin

Purpose

Returns the port input admittance

Synopsis

yin(Sii, Z)

where Sii is a complex reflection coefficient and Z is a reference impedance.

Example

y=yin(S11, 50)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

Yin

Defined in

AEL, network_fun.ael

See also

vswr, zin

Description

Given a reflection coefficient and the reference impedance, this measurement returns the input admittance looking into the measurement ports.

yopt

Purpose

Returns optimum admittance for noise match

Synopsis

yopt(gammaOpt, zRef)

where gammaOpt is a optimum reflection coefficient and zRef is a reference impedance.

Example

y = yopt(Sopt, 50)

Used in

Small-signal S-parameter simulations

Available as measurement component?

Yopt

Defined in

Built in

See also

zopt

Description

This complex measurement produces the optimum source admittance for noise matching. gammaOpt is the optimum reflection coefficient that must be presented at the input of the network to realize the minimum noise figure (NFmin).

ytoabcd

Purpose

Performs Y-to-ABCD conversion

Synopsis

ytoabcd(Y)

where Y is an admittance matrix.

Example

a = ytoabcd(Y)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

abcdtoh, htoabcd

Description

This measurement transforms an admittance matrix of a 2-port network into a hybrid matrix.

ytoh

Purpose

Performs Y-to-H conversion

Synopsis

ytoh(Y)

where Y is an admittance matrix.

Example

h = ytoh(Y)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

htoy, ytoabcd

Description

This measurement transforms an admittance matrix of a 2-port network into a hybrid matrix.

ytos

Purpose

Performs Y-to-S conversion

Synopsis

S = ytos(Y, zRef)

where Y is an admittance matrix and zRef is a reference impedance.

Example

s = ytos(Y, 50.0)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

stoy, ytoz

Description

This measurement transforms an admittance matrix into a scattering matrix.

ytoz

Purpose

Performs Y-to-Z conversion

Synopsis

Z = ytoz(Y)

where Y is an admittance matrix

Example

z = ytoz(Y)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

ytos, ztoy

Description

This measurement transforms an admittance matrix to an impedance matrix.

zeros

Purpose

Returns a matrix of zeros

Synopsis

Y = zeros(2)

Y = zeros(2, 3)

This is the zeros matrix. If only one argument is supplied, then a square matrix is returned. If two are supplied, then a matrix of zeros with size rows × cols is returned.

Example

a=zeros(2);
returns {{0, 0}, {0, 0}}

b=(2, 3)
returns {{0, 0, 0}, {0, 0, 0}}

Used in

Not applicable

Available as measurement component?

Not applicable

Defined in

Built in

See also

identity, ones

zin

Purpose

Returns the port input impedance

Synopsis

zin(Sii, Z)

where Sii is a complex reflection coefficient and Z is a reference impedance.

Example

y=zin(S11, 50.0)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

Zin

Defined in

AEL, network_fun.ael

See also

vswr, yin

Description

Given a reflection coefficient and the reference impedance, this measurement returns the input impedance looking into the measurement ports.

zopt

Purpose

Returns the optimum impedance for noise matching

Synopsis

zopt(gammaOpt, zRef)

where gammaOpt is an optimum reflection coefficient and zRef is a reference impedance.

Example

y = zopt(Sopt, 50)

Used in

Small-signal S-parameter simulations

Available as measurement component?

Zopt

Defined in

AEL, circuit_fun.ael

See also

yopt

Description

This complex measurement produces the optimum source impedance for noise matching. gammaOpt is the optimum reflection coefficient that must be presented at the input of a network to realize the minimum noise figure (NFmin).

ztoabcd

Purpose

Performs Z-to-ABCD conversion

Synopsis

ztoabcd(Z)

where Z is an impedance matrix.

Example

a = ztoabcd(Z)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

abcdtoz, ytoabcd, ztoh

Description

This measurement transforms an impedance matrix of a 2-port network into a chain (ABCD) matrix.

ztoh

Purpose

Performs Z-to-H conversion

Synopsis

ztoh(Z)

where Z is an impedance matrix.

Example

h = ztoh(Z)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

htoz, ytoh, ztoabcd

Description

This measurement transforms an impedance matrix of a 2-port network into a hybrid matrix.

ztos

Purpose

Performs Z-to-S conversion

Synopsis

ztos(Z, zRef)

where Z is an impedance matrix and zRef is a reference impedance.

Example

s = ztos(Z, 50.0)

Used in

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

stoz, ytos, ztoy

Description

This measurement transforms an impedance matrix to a scattering matrix.

ztoy

Purpose

Performs Z-to-Y conversion

Synopsis

ztoy(Z)

where Z is an impedance matrix.

Example

y = ztoy(Z)

Used in...

Small-signal and large-signal S-parameter simulations

Available as measurement component?

This equation can be entered by means of a MeasEqn component in S_Param Simulation and LSSP Simulation palettes in the Schematic window. There is no explicit measurement component.

Defined in

AEL, network_fun.ael

See also

stoz, ytos, ztoy

Description

This measurement transforms an impedance matrix to an admittance matrix.

Chapter 4: Simulator Expression Reference

This chapter lists and describes the expressions (functions) that are available within the Advanced Design System at simulation runtime. These expressions include mathematical functions such as those for matrix conversion, trigonometry, absolute value, and the like.

Unlike the measurement equations (MeasEqn) described in Chapter 3, these equations are used *internally* during *simulation time*, and are known as Simulator Expressions or sometimes as VarEqn expressions. These expressions can be entered into the program by means of the VarEqn component or used in place of a parameter for any component: for example in a resister, R=sin5. These expressions are evaluated at the start of simulation. If a term is undefined at the start of simulation, such as $R=S_{11}$, where the results of S_{11} will not be known until the simulation is complete, an error will be returned.

The VarEqn component is available in the Data Items palette in an Analog/RF Systems Schematic window or from the Controllers palette in a Signal Processing Schematic window.

For information on the general use of VAR components, place a VAR component on a Schematic, double click it, and then click the *Help* button in the Component dialog box. Or from any ADS Window choose Help > Topics and Index > Components > Circuit Components > Introduction and Simulation Components > Chapter 1, Introduction > VAR.

In this chapter you will find:

- A list of the Simulator Expressions.
- A list of simulator variables and constants.
- A list of mathematical operators and hierarchy that can be used with these expressions.

How to Enter Simulator Expressions

The basic ways to enter the equations or expressions that are available at simulation runtime are as follows:

- Place and setup a VarEqn component.
- Place any component and enter a function in place of a component parameter.

- Place and set up a FDD (frequency-defined device) or SDD (symbolically-defined device).

Simulator Expressions

The following table lists the Simulator Expressions available in ADS and a brief description of each.

Table 4-1.

Function	Description
d_atan2(a, b, c, d)	derivative of atan2()
sin (x)	sine function
cos (x)	cosine function
tan (x)	tangent function
cot(x)	cotangent function
sinh(x)	hyperbolic sine function
cosh(x)	hyperbolic cosine function
tanh(x)	hyperbolic tangent function
coth(x)	hyperbolic cotangent function
exp(x)	exponential function
ln(x)	natural log function
log(x)	log base 10 function
sqrt(x)	square root function
conj(x)	complex-conjugate function
mag(x)	magnitude function
phaserad(x)	phase (in radians) function
imag(x)	imaginary-part function
real(x)	real-part function
phasedeg(x)	phase (in degrees) function
phase(x)	phase (in degrees) function
deg(x)	radian-to-degree conversion function
rad(x)	degree-to-radian conversion function
int(x)	convert-to-integer function

Function	Description
atan2(y, x)	arctangent function (two real arguments)
complex(x, y)	real-to-complex conversion function
polar(x, y)	polar-to-rectangular conversion function
dbpolar(x, y)	(dB,angle)-to-rectangular conversion function
vswrpolar(x, y)	(VSWR,angle)-to-rectangular conversion function
ripple(x, y, z, v)	ripple(amplitude, intercept, period, variable) sinusoidal ripple function
db(x)	decibel function
max(x, y)	maximum function
min(x, y)	minimum function
spectrum(...)	spectrum(function(0), Npts, Period, Delay, Window) returns spectral array
impulse(...)	impulse(CpxFunction(0), Npts, TimeStep, CenterFreq, Window) returns impulse array
abs(A0)	absolute value function
sgn(A0)	signum function
arcsinh(A0)	arcsinh function
arctan(A0)	arctan function
acos(A0)	Arc-cosine(x) for x real and -1 <= x <= 1
asin(A0)	Arc-sine(x) for x real and -1 <= x <= 1
acosh(A0)	Arc-hyperbolic-cosine(x) for x real and x >= 1
atanh(A0)	Arc-hyperbolic-tangent(x) for x real and
jn(A0, A1)	Bessel function
i(...)	current function
deembed(A0)	de-embedding function
delay(A0)	group delay function
deriv(A0, A1)	derivative function
fread(A0)	raw-file reading function
interp1(A0, A1)	interpolation function--one independent variable
interp2(A0, A1, A2)	interpolation function--two independent variables

Function	Description
interp3(A0, A1, A2, A3)	interpolation function--three independent variables
interp4(A0, A1, A2, A3, A4)	interpolation function--four independent variables
interp(...)	somewhat generalized scalar interpolation function
lookup(...)	data lookup function
access_data(...)	datafile dependents' lookup/interpolation function
access_all_data(...)	datafile indep+dep lookup/interpolation function
polarcpx(...)	polar to rectangular conversion function
scalearray(A0, A1)	scalar times a vector (array) function
qinterp(A0, A1)	quick and dirty interpolation function
ramp(A0)	ramp function
step(A0)	step function
setDT(A0)	Turns on discrete time transient mode (returns argument)
readraw(A0, A1, A2)	rawfile reading routine
readlib(A0, A1, A2, A3)	rawfile-from-library reading routine
readdata(...)	library or rawfile reading routine
rect(A0, A1, A2)	rectangular pulse function
rem(...)	remainder function. Examples: rem(10,4) returns 2, because 10 divided by 4 is 2 with a remainder of 2. rem(5,5) returns 0. rem(x,y) = x - int(x/y)*y. If either x or y is a complex number, it's replaced by its mag() value.
limit_warn([A0, A1, A2, A3, A4])	limit, default and warn function
awg_dia(A0)	wire gauge to diameter in meters
rpsmooth(A0)	rectangular-to-polar smoothing function
sens(A0, A1)	sensitivity function
sinc(A0)	sin(x)/x function
sprintf(A0, A1)	formatted print utility
strcat(...)	string concatenation utility

Function	Description
system(A0)	UNIX system call function
rawtoarray(...)	rawfile pointer to sym array conversion function
makearray(...)	(1:real|2:complex|3:string, A1, A2..) or (array, startIndex, stopIndex)
list(...)	
length(A0)	returns number of elements in array
v(...)	voltage function
inoise(A0)	noise current function
vnoise(A0)	noise voltage function
nf(A0)	noise figure function
vss(...)	small-signal voltage function
vlsb(...)	small-signal lower-sideband voltage function
vusb(...)	small-signal upper-sideband voltage function
iss(...)	small-signal current function
ilsb(...)	small-signal lower-sideband current function
iusb(...)	small-signal upper-sideband current function
dphase(A0, A1)	Continuous phase difference (radians) between A0 and A1
dbm(A0, A1)	convert voltage and impedance into dBm
dbmtoa(A0, A1)	convert dBm and impedance into short circuit current
dbmtov(A0, A1)	convert dBm and impedance into open circuit voltage
dbmtow(A0)	convert dBm to watts; takes one argument, e.g., dbmtow (-10) returns 1E-4 watts; can be combined with other functions, such as polar, where the polar argument is in degrees
dbwtow(A0)	convert dBW to watts; see dbmtow for more information
innerprod(...)	inner-product function
thd(A0)	total-harmonic-distortion function
norm(A0)	norm function
rms(...)	root-mean-square function

Function	Description
toi(A0)	third-order-intercept function
freq_mult_coef(A0)	frequency multiplier polynomial generator function
freq_mult_poly(A0, A1)	frequency multiplier polynomial evaluation function
window(A0, A1, A2, A3, A4)	spectral windowing function
internal_window(A0, A1, A2, A3, A4, A5)	internal spectral windowing function
generate_pulse_train_spectra(A0, A1, A2, A3, A4, A5)	generate a pulse train spectra
internal_generate_pulse_train_spectra(A0, A1, A2, A3, A4, A5, A6)	internal generate a pulse train spectra
sym_set(A0, A1)	set sym variable to a given value
log_amp(A0, A1, A2, A3)	successive detection logarithmic amplifier
log_amp_cas(A0, A1, A2, A3)	true logarithmic amplifier
generate_qpsk_pulse_spectra(A0, A1, A2, A3, A4, A5, A6)	generate a QPSK pulse train spectra
internal_generate_qpsk_pulse_spectra(A0, A1, A2, A3, A4, A5, A6, A7)	internal generate a QPSK pulse train spectra
generate_piqpsk_spectra(A0, A1, A2, A3, A4, A5, A6)	generate a pi/4 QPSK pulse train spectra
internal_generate_piqpsk_spectra(A0, A1, A2, A3, A4, A5, A6, A7)	internal generate a pi/4 QPSK pulse train spectra
bin(A0)	function convert a binary to integer
itob(A0, [A1])	convert integer to binary
generate_gmsk_pulse_spectra(A0, A1, A2, A3, A4, A5, A6, A7)	generate a gmsk pulse train spectra
internal_generate_gmsk_pulse_spectra(A0, A1, A2, A3, A4, A5, A6, A7, A8)	internal generate a gmsk pulse train spectra
generate_qam16_spectra(A0, A1, A2, A3, A4, A5, A6)	generate a 16-QAM pulse train spectra
internal_generate_qam16_spectra(A0, A1, A2, A3, A4, A5, A6, A7)	internal generate a 16-QAM pulse train spectra
generate_gmsk_iq_spectra(A0, A1, A2, A3, A4, A5, A6)	generate the gmsk `i' or `q'

Function	Description
internal_generate_gmsk_iq_spectra(A0, A1, A2, A3, A4, A5, A6, A7)	internal generate the gmsk `i' or `q'
get_fund_freq(A0)	Get the frequency associated with a specified fundamental index
internal_get_fund_freq(A0, A1)	internal function to get frequency for a specified fundamental index
ktoc(A0)	convert Kelvin to Celsius
ctok(A0)	convert Celsius to Kelvin
ftoc(A0)	convert Fahrenheit to Celsius
ctof(A0)	convert Celsius to Fahrenheit
ftok(A0)	convert Fahrenheit to Kelvin
ktof(A0)	convert Kelvin to Fahrenheit
get_array_size(A0)	Get the size of the array
eval_poly(A0, A1, A2)	polynomial evaluation function
pwl(...)	piecewise-linear function
phase_noise_pwl(...)	piecewise-linear function for computing phase noise
pwlr(...)	piecewise-linear-repeated function
pulse(time, [low, high, delay, rise, fall, width, period])	periodic pulse function
cos_pulse(time, [low, high, delay, rise, fall, width, period])	periodic cosine shaped pulse function
erf_pulse(time, [low, high, delay, rise, fall, width, period])	periodic error function shaped pulse function
damped_sin(time, [offset, amplitude, freq, delay, damping, phase])	damped sin function
multi_freq(time, amplitude, freq1, freq2, n, [seed])	multifrequency function
exp_pulse(time, [low, high, delay1, tau1, delay2, tau2])	exponential pulse function
sffm(time, [offset, amplitude, carrier_freq, mod_index, signal_freq])	signal frequency FM
bitseq(time, [clockfreq, trise, tfall, vlow, vhigh, bitseq])	bit sequence function

Function	Description
lfsr(A0, A1)	lfsr(taps, seed) returns a string containing complete sequence
read_data(...)	read_data("file|dataset", "locName", "fileType")
read_lib(...)	read_lib("libName", "item", "fileType")
get_block(A0, A1)	HPvar tree from block name function
get_max_points(A0, A1)	maximum points of independent variable
dsexpr(A0, A1)	Evaluate a dataset expression to an hpvar
dstoarray(A0, [A1])	Convert an hpvar to an array
get_attribute(...)	value of attribute of a set of data
dep_data(A0, A1, [A2])	dependent variable value
names(A0, A1)	array of names of indepVars and/or depVars in dataset
index(A0, A1, [A2, A3])	get index of name in array
cxform(A0, A1, A2)	transform complex data
transform(A0, A1, A2, [A3, A4])	transform complex depVars in dataset
stypexform(A0, A1, A2, A3)	Y, Z, H, G, to S matrix transform
miximt_coef(A0, A1, A2, A3, A4)	Mixer IMT polynomial generator function
amp_harm_coef(A0, A1, A2)	Amplifier polynomial generator function
miximt_poly(A0, A1, A2, A3, A4)	Mixer IMT polynomial evaluation function
gcdata_to_poly(A0, A1)	Fit gain compression data to a polynomial
compute_poly_coef(A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11, A12)	Compute polynomial coefficients for user specified nonlinearities such TOI, Psat etc
cpx_gain_poly(A0, A1, A2)	Compute complex gain for given polynomial coefficients
coef_count(A0)	Count the number of coefficients
imt_hpvar_to_array(A0, A1, A2, A3)	Convert IMT hpvar data to an array
imt_hbdata_to_array(A0, A1, A2, A3, A4, A5)	Convert 2-tone HB data to an IMT array
echo(A0)	echo-arguments function
value(A0)	print-value function

Simulator Variables and Constants

When you are using Simulator Expressions, keep in mind that certain variables and constants are reserved words in ADS. You can use these variables and constants, but you cannot redefine them to something else. The following table lists the simulator variables/constants available in ADS and a brief description of each.

Table 4-2.

Variable/Constant Name	Description
time = 0 s	the analysis time
timestep = 1 s	the analysis time step
tranorder = 1	the transient analysis integration order
freq = 1e+006 Hz	the analysis frequency
Nsample = 0	signal processing analysis sample number
ScheduleCycle = 0	signal processing schedule cycle number
DefaultValue = -1	signal processing default parameter value
noisefreq = 1e+006 Hz	the spectral noise analysis frequency
ssfreq = 1e+006 Hz	the small-signal mixer analysis frequency
temp = 25 C	the analysis temperature
e = 2.71828	2.71838...
ln10 = 2.30259	ln(10)
c0 = 2.99792e+008 m/s	the speed of light
e0 = 8.85419e-012	vacuum permittivity
u0 = 1.25664e-006	vacuum permeability
boltzmann = 1.38066e-023	Boltzmann's constant
qelectron = 1.60218e-019	the charge of an electron
planck = 6.62608e-034	Planck's constant
hugereal = 1.79769e+308	largest real number
tinyreal = 2.22507e-308	smallest real number
sourceLevel = 1	used for source-level sweeping
dcSourceLevel = 1	used for DC source-level sweeping
logRshunt = 0	used for DC Rshunt sweeping
logNodesetScale = 0	used for DC nodeset simulation

Table 4-2.

Variable/Constant Name	Description
logRforce = 0	used for HB Rforce sweeping
mcindex = 0	index for Monte Carlo sweeps
doeindex = 0	index for Design of Experiment sweeps
CostIndex = 0	index for optimization cost plots
mcTrial = 0	trial counter for Monte Carlo based simulations
optIter = 0	optimization job iteration counter
doeIter = 0	doe experiment iteration counter
DeviceIndex = 0	device Index used for noise contribution or DC OP output
LinearizedElementIndex = 0	index for BudLinearization sweep
DF_Value = -1e+009	reference to corresponding value defined in Data Flow controller
DF_ZERO_OHMS = 1e-013	symbol for use as zero ohms
DF_DefaultInt = -1e+009	reference to default int value defined in Data Flow controller

Mathematical Operators and Hierarchy

Expressions are evaluated from left to right, unless there are parentheses. Operators are listed from higher to lower precedence. Operators on the same line have the same precedence. For example, a+b*c means a+(b*c), because * has a higher precedence than +. Similarly, a+b-c means (a+b)–c, because + and – have the same precedence (and because + is left-associative).

The operators !, &&, and || work with the logical values. The operands are tested for the values TRUE and FALSE, and the result of the operation is either TRUE or FALSE. In AEL a logical test of a value is TRUE for non-zero numbers or strings with non-zero length, and FALSE for 0.0 (real), 0 (integer), NULL or empty strings. Note that the right hand operand of && is only evaluated if the left hand operand tests TRUE, and the right hand operand of | | is only evaluated if the left hand operand tests FALSE.

The operators >=, <=, >, <, ==, != , AND, OR, EQUALS, and NOT EQUALS also produce logical results, producing a logical TRUE or FALSE upon comparing the values of two expressions. These operators are most often used to compare two real

numbers or integers. These operators operate differently in AEL than C with string expressions in that they actually perform the equivalent of *strcmp()* between the first and second operands, and test the return value against 0 using the specified operator.

Table 4-3. Operator Precedence

Operator	Name	Example
( )	function call, matrix indexer	foo(expr_list) X(expr,expr)
[ ]	sweep indexer, sweep generator	X[expr_list] [expr_list]
{ }	matrix generator	{expr_list}
**	exponentiation	expr**expr
!	not	!expr
* / .* ./	multiply divide element-wise multiply element-wise divide	expr * expr expr / expr expr .* expr expr ./ expr
+ -	add subtract	expr + expr expr - expr
::	sequence operator wildcard	exp::expr::expr start::inc::stop ::
< <= > >=	less than less than or equal to greater than greater than or equal to	expr < expr expr <= expr expr > expr expr >= expr
== !=	equal not equal	expr == expr expr != expr
&&	logical and	expr && expr
||	logical or	expr || expr

Index