ANSOFT HFSS FIELD CALCULATOR COOKBOOK

A BRIEF PRIMER
AND
COLLECTION OF STEP-BY-STEP CALCULATOR RECIPIES
FOR USE IN
HFSS FIELDS POST-PROCESSING

ANSOFT HFSS FIELD CALCULATOR COOKBOOK

A Brief Primer and Collection of Step-by-Step Calculator Recipes for use in HFSS Fields Post-Processing

ANSOFT CORPORATION
Four Station Square, Suite 200
Pittsburgh, PA 15219
(412) 261-3200

VERSION NOTICE:

The calculator routines described in the following pages are intended for use with Ansoft HFSS Version 7 (7.0.04). Earlier or later versions of the software may include differences in the Field Calculator which will require alteration of the routines contained herein.

INTRODUCTION

The following pages contain calculator routines, or “recipes”, for use within the Field Calculator feature of Ansoft’s HFSS Version 7. The field calculator is a very powerful but frequently misunderstood and underutilized tool within the 3D “Fields” Post-Processor.

These routines represent only a small set of the complete capabilities of the calculator. Starting from field data obtained by performing an HFSS solution, the calculator could be used to generate thermal information, voltages and currents, or any other quantity that can be viewed in a 3D environment upon the modeled geometry. This document is intended to give the user a ‘head start’ in using the calculator by codifying some frequently used calculations into easy-to-follow steps. In many cases the steps identified in this document are not the only sequence of operations which can obtain the same results. However, an attempt has been made to identify the routines that require the least number of ‘button clicks’ and stack manipulations to obtain the desired answer.

TABLE OF CONTENTS

Cautionary Notes

The following text provides some brief cautionary notes regarding use of the Post-Processor Field Calculator in Ansoft HFSS. Most of the statements below are fairly generalized, and may not apply to all HFSS projects. When in doubt about the applicability of a particular warning for a particular project, please feel free to contact your local HF Applications Engineer for further assistance.

Field Convergence:

Ansoft HFSS is a finite element method (FEM) field solver, which arrives upon its solution via adaptive meshing convergence. There are different algorithms available for determining *where* in each given model mesh adaptation is performed, but convergence is always evaluated by comparison of S-parameters (for driven solutions), changes in overall scattering energy (for incident wave problems) or resonant frequencies (for eigenmode solutions) from pass to pass. Since these quantities represent the results of the model as a whole, they tend to converge more rapidly than the 'field values' at each point in the modeled space can be said to have 'converged' to some value. As a result, specific field quantities at each mesh point are likely to be less accurate than the overall S-parameter or Eigenfrequency result of a project solution.

In order to obtain high accuracy results from calculations on field data, it is advised that the user take extra precautions to assure that the model's field data is dependable. These extra precautions might include:

-- Running the project to a tighter than usual convergence value
-- Seeding or manually refining the mesh in the areas to be used for calculations
-- Running parametric variations to isolate sensitivity to modeling parameters such as adaptation frequency or circular cross-section facetization.

As long as the accuracy of specific field data points to be used has been assured, the results of the HFSS Field Calculator operations should provide valuable information for the user's electromagnetic design tasks.

Fast Sweep and Dispersive Models:

If an HFSS solution has been performed to include an ALPS Fast Frequency Sweep, the Fields Post-Processor can be 'tuned' to display field data at any point in the frequency band swept. The specific frequency selected for viewing (in the Data→Edit Sources menu of the Fields Post-Processor) need not even be a precise data point at which the S-parameters were calculated. While field calculator operations may be performed at any frequency to which the Fields Post-Processor is set, fast sweep solution field data (away from the 'center frequency' of the sweep) may not be as accurate for lossy and dispersive media, within the interior of solid-meshed finite conductors, etc. For higher accuracy under these conditions, field calculator operations should be performed on a full matrix solution completed at the desired frequency.

Since materials assigned as part of a "Perfectly Matched Layer" (PML) model termination are anisotropic and highly lossy, performing field calculations on the surface of or interior to objects designated as PMLs is not recommended.

Inputs/Excitations:

The user should remember to set the field excitation using DATA→EDIT SOURCES appropriate to the calculation to be performed. In some cases (e.g. FSS calculations) picking the right field solution set (incident, scattered, or total) is also paramount to obtaining the intended result.

Any field calculation which has not yet been completed (such that the calculator stack still shows some form of "text" string rather than a simple numerical value) is merely a placeholder. Altering the field data loaded in the Post-Processor (by altering port excitations, changing frequency, or picking a different solution set using Data→Edit Sources) will result in subsequent evaluation of the placeholder to the newly loaded data. To preserve a placeholder's association to an existing data set before altering the excitation to a different data set, the register stack should be exported using the "Write" button. The correctly associated quantity can be brought back into the stack using "Read" after the field data set selection has been altered.

Units:

All units in Driven HFSS field solutions are expressed in the MKS system, regardless of drawing units. Therefore E-mag is always in V/m, H-mag in A/m, etc. The exception is that when plotting along a geometry (e.g. along a line) the dimension along the X axis of the graph shows the position along the line in the drawing units, while the vertical (field quantity) axis will be in the MKS system.

Eigensolutions:

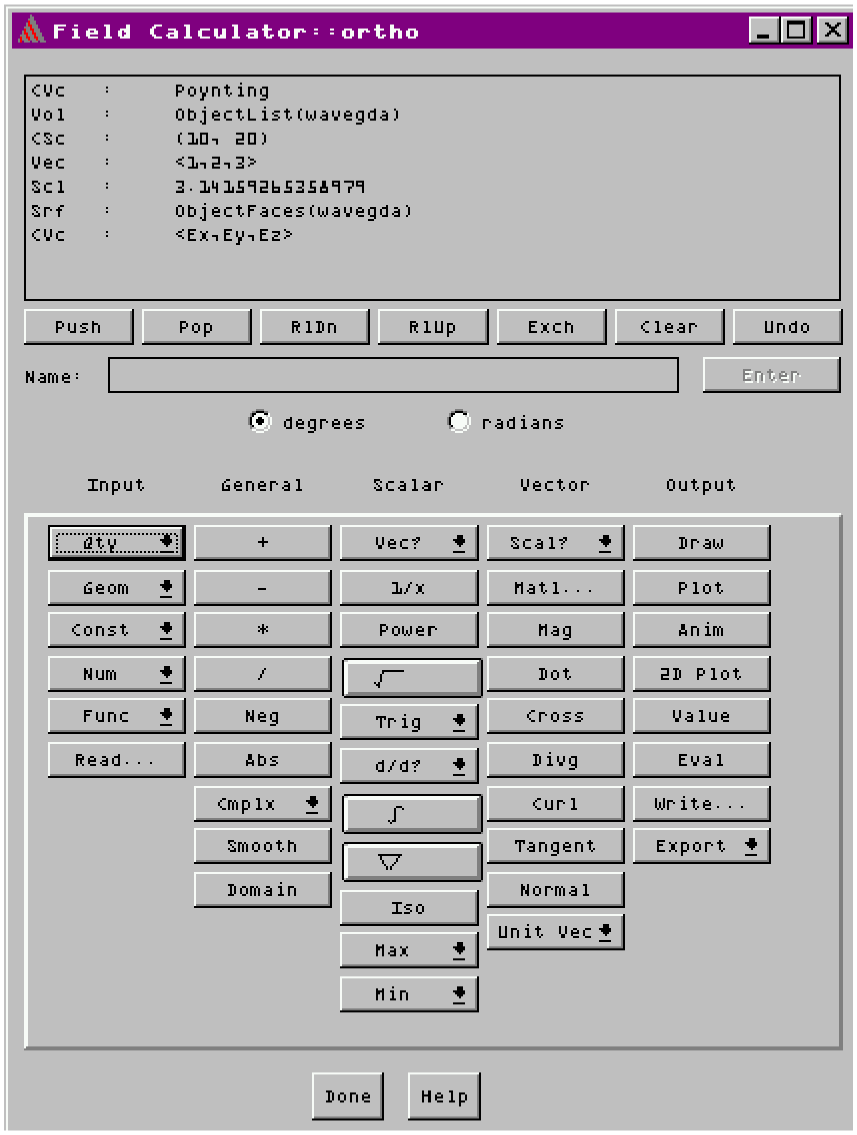
Field values in eigensolutions are normalized to a peak value of 1.0, since there is no real 'excitation' to which to scale the internal field results. If desired, the peak value can be scaled to a user-selected number using the Data→Edit Sources menu.

Macro Implementation:

All calculator operations have direct equivalents in the HFSS Macro Language. There are, however, additional commands in the macro language to permit use of calculator operation results (e.g. to extract the top entry from the calculator stack and save its value to a macro variable for later storage in a database) which do not emulate specific calculator operations. Details on the use and syntax of these commands can be found in the macro manual, "*Introduction to the Ansoft Macro Language*."

Calculator Interface Basics

Most engineers who use HFSS find that the standard post processing features are sufficient for their work. The scattering parameters, Y or Z matrix, animated field plots and far field patterns cover most of what one needs from such a simulation tool. For those few cases where these are not sufficient the post processor within HFSS includes a Field Calculator. Using this calculator one can perform mathematical operations on any of the field quantities within the solution space to derive specialized quantities. The calculator can also be used to plot these calculated quantities, allowing one to compute the desired term and then to display it over some geometric feature to visualize its characteristics.


Field Calculator::ortho

```
CVc   :      Poynting
Vol   :      ObjectList(wavegda)
CSc   :      (10, 20)
Vec   :      <1,2,3>
Scl   :      3.14159265358979
Srf   :      ObjectFaces(wavegda)
CVc   :      <Ex,Ey,Ez>
```

Name:

[✓ degrees] radians

Figure 1: Field Calculator Interface

The Field Calculator (hereafter referred to as the "calculator") interface is shown in Figure 1, above. The top of the calculator contains the *stack*, in which calculator entries are held in *stack registers*. Immediately beneath the stack is the row of *stack command* buttons, as well as a *Name* field for assigning a name to the top entry in the stack. The middle of the calculator contains selection buttons to opt between *degree* and *radian* assumptions for all calculator operations. The default setting is *degrees.* The bottom half of the calculator holds the columns containing the actual calculator buttons, organized in columns by functional groups. These columns are headed *Input, General, Scalar, Vector,* and *Output*. Each will be discussed in further detail below. At the very bottom of the calculator are buttons to exit (*Done*) and to access the online *Help*.

As shown in Figure 2, below, some calculator "buttons" are actually dropdown *menus*, containing multiple "button" options. Buttons which contain multiple options are indicated with an arrow symbol on their face. The extreme bottom of the calculator will also act as a *status bar* providing prompts describing the various button options within these dropdown menus.

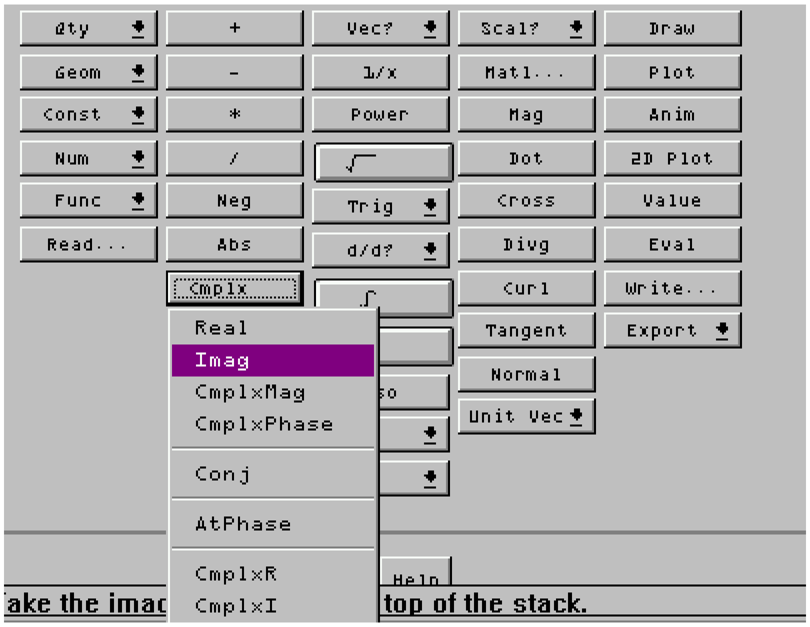


Figure 2: Dropdown Menu "Button" Example

Stack Registers:

Calculator stack registers add to the stack display *above* preceding entries. Therefore, the entry at the *top* of the stack represents the *last* register filled. This convention is opposite to that which many users may be familiar with from the use of hand-held multi-line calculators, which often build their stacks from the bottom up.

Stack Commands:

Stack Commands are those commands which influence the entries in the calculator stack and their position. Many are self-explanatory, and/or match standard stack manipulation conventions. For example. *RlDn* and *RlUp* represent "roll down" and "roll up", which will rotate the stack entries in the direction indicated. *Clear* will empty the stack of all contents, while *Pop* deletes only the last entry from the stack. A full description of all the calculator stack commands can be found in the online help.

Input Column:

The *Input Column* contains all the calculator functions which place new values into the stack. These values include field data, geometry data, and numerical quantities. Field data (e.g. E-field, H-field, and Poynting vector) for the current project solution is input from the *Qty* (quantity) dropdown menu button. Other *Input* dropdown buttons should be self-explanatory. A complete description of each option is available in the online help.

The *Quantities* specifically available from the calculator are the E-field, H-field, J-vol (volume current) and Poynting vector. All quantities are *Peak Phasors*, and **not** RMS quantities, with the phase information captured in the real and imaginary components and the field orientation captured in the vector components. Although the Poynting vector is automatically calculated by the interface as 0.5(E $\times$ H$*$), it will appear in the calculator stack as a *Complex Vector* quantity. The imaginary portion should however be zero. [See ***CVc*** in the section of this document regarding ***Calculator Stack Quantities***, below.]

General Column:

The *General Column* contains calculator operations which act on many different kinds of data (e.g. vector, scalar, complex, etc.). With the exception of the *Cmplx* (complex) menu, all are distinct functions. Most are self-explanatory, with the exception of *Smooth* which performs some data “smoothing” or statistical medianization on the top stack entry. A complete description of each button is available in the online help.

Scalar Column:

The *Scalar Column* contains calculator operations which can only be performed on *scalar* stack entries. Dropdown menus in this column include *Vec?* (convert scalar to vector), *Trig* (trigonometric, containing sin, cos, etc. functions), *d/d?* (derivative with respect to...), *Max* and *Min* (self-explanatory).

Note that the calculator’s $\int$(Integrate) function is located in the *Scalar* column. The implication is clear that integration can only be performed on scalar quantities. To perform integration upon complex quantities, the integration will need to be performed separately on real and imaginary subcomponents.

Vector Column:

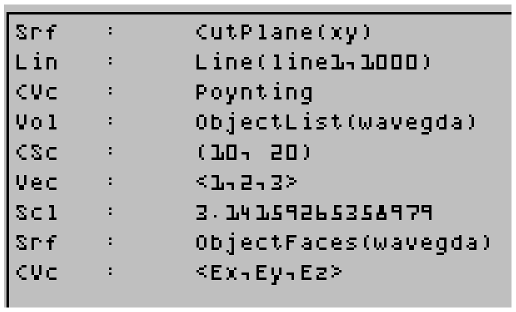
The *Vector Column* contains calculator operations which can only be performed on *vector* stack entries. Dropdown menus in this column are *Scal?* (convert vector to scalar) and *Unit Vec* (create unit vector). Standard vector algebra operations (*Dot, Cross,* etc.) are also present. A complete description of each button is available in the online help.

Output Column:

The *Output Column* contains those calculator operations that result in final data outputs from calculations, including those which result in plotting or displaying calculation results in the Fields Post-Processor’s graphical windows. The only dropdown menu in this column is *Export* (self-explanatory). *Draw* is used to display geometric quantities, *Plot* can be used to display scalar or vector data on and in geometries, and *2D Plot* is used to generate a line graph of a calculated quantity along some positional entity. The *Eval* button obtains final numerical results from stack placeholders (such as integrations). A complete description of each button is available in the online help.

Calculator Stack Quantities

The calculator is capable of performing operations on a number of different data types. In many instances, a calculation requires certain type(s) of data to be present in the correct order in the stack register. In order to show the user the type of data contained in each stack entry, the entry will be preceded by an indicator as shown in Figure 3, below. The following list will describe the definition of each indicator, and provide guidance regarding operations which can convert data from one type to another.



```
Srf   :     CutPlane(xy)
Lin   :     Line(line1,1000)
CVc   :     Poynting
Vol   :     ObjectList(wavegda)
CSc   :     (10, 20)
Vec   :     <1,2,3>
Scl   :     3.14159265358979
Srf   :     ObjectFaces(wavegda)
CVc   :     <Ex,Ey,Ez>
```

Figure 3: Stack Contents showing Data Type Indicators (at left)

Scl:

Scl denotes a *Scalar* quantity. This is a simple numerical value. To convert a scalar to a vector quantity, use the *Vec?* dropdown menu in the Scalar column. The choices *VecX, VecY,* and *VecZ* convert the scalar data to vector data aligned with the X, Y, or Z unit vectors, respectively. The user can also multiply the scalar quantity by a desired vector direction entered manually (*Num* dropdown in the Input column) or obtained using the *Unit Vec* button from the Vector column. To convert a scalar to a complex quantity, use either *CmplxR* (assign the scalar value as the real component of a complex quantity) or *CmplxI* (assign the scalar value as the imaginary component of a complex quantity), both found under the *Cmplx* dropdown in the General Column.

CSc:

CSc denotes a *Complex Scalar* quantity. This is a numerical value with real and imaginary components. Convert to a vector quantity using the same techniques described for ***Scl***, above. Convert to a scalar using *Real* (take the real component)*, Imag* (take the imaginary component), *CmplxMag,* (take the magnitude of the complex number) or *CmplxPhase* (take the phase of the complex number), all within the *Cmplx* dropdown in the General column.

Vec:

Vec denotes a *Vector* (non-complex) quantity. Vectors are always evaluated in the coordinate system of the model. To convert a vector quantity to a scalar, use the *Scal?* dropdown menu from the Vector column. Suboptions *ScalarX, ScalarY,* and *ScalarZ* will take the appropriate scalar component of the vector data. Optionally, you can also *Dot* the vector with another vector to obtain the appropriate scalar result, or use the *Tangent* (return the tangential scalar component of) or *Normal* (return the normal scalar component of) operations to relate the vector quantity to a geometric data (*Lin, Srf*) stack entry. Convert to a Complex quantity using the *CmplxR* and *CmplxI* operations described in ***Scl***, above.

CVc:

CVc denotes a *Complex Vector* quantity. This is a quantity with real and imaginary components for each vector component. In normal calculator usage, the complex nature of the vector components represent the magnitude and phase data of a field quantity, while the vector components themselves represent the orientation of the field quantity in space. Convert to a non-complex Vector as described in ***CSc***, above. Convert the vector to a scalar quantity as described in ***Vec***, above.

Geometric Data:

Geometric data is indicated in the calculator stack by the headers *Lin* (line), *Srf* (surface), and *Vol* (volume). Lines may be straight, curved, or "polylines" in three dimensional space. Lines may also be open (have two endpoints) or closed (ending vertex same as starting vertex). Surfaces need not be planar, and may actually comprise a list of object faces (*faces list*) as well as planar slices through the entire model space (*cutplanes*). Volumes may include sets of discontinuous object volumes created as an *Object List*.

These indicators may exist alone, representing geometric data only, or in combination with one of the categories above, indicating a type of data applied to the geometric entity in question. For example, the notation *SclSrf* identifies a stack entry containing Scalar data on a Surface geometry set. To select only the portion of a given data entry which exists along, on, or within a given geometry quantity, use the *Value* button in the Output column of the calculator. Other operations (e.g. integration, or the *Normal* button) operate when a data quantity is in the second stack register and a geometric quantity is in the top stack register. Full descriptions of the register requirements for each individual command is available in the on-line help.

CALCULATOR RECIPES

The following pages contain calculator recipes for deriving a number of commonly used output parameters from solved HFSS projects. Each calculator recipe will be provided in the format shown below:

EXAMPLE: **Title of Current Calculation**

Description:

The first paragraph will give a brief description of the calculation's intent.

Usage Example(s):

The second paragraph will give an example of a project type on which the calculation might be useful. It may also comment upon the reasons such a calculation might be of interest.

Prerequisites:

The third (optional) paragraph will indicate what must be present before doing the calculator operations, e.g. if certain geometry (lines, faces lists, etc.) need to be generated to use in calculations.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
Each button click shown as a step	resulting stack entry/type
Dropdown Menu→Submenu pick also shown as a single step	Scl : {placeholders for numerical results shown in brackets}
Button steps requiring data entry will have entry quantity shown in {brackets}	resulting stack entry/type *(notes follow in italics)*

Calculating the Current along a Wire or Trace

Description:

Obtains the full complex current in a wire or trace conductor (e.g. microstrip, stripline) at a specific location by integrating the magnetic field along a closed path encircling the conductor.

Usage Example(s):

To find the current distribution along a wire (dipole, monopole, etc.) antenna, this calculation could be repeated at periodic positions along the length of the antenna.

Prerequisites:

A *closed* line for the integration path must be created using Geometry→Create→Line before beginning calculator operations. The line must be orthogonal to the direction of current flow, should not intersect the wire/trace, and should not be too much bigger than the wire/trace.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
Qty→H	CVc : <Hx,Hy,Hz>
Push	*(above entry duplicated)*
Cmplx→Real	Vec : Real(<Hx,Hy,Hz>)
Geom→Line...→{select line}	Lin : Line (line1,1000) *(user line name and point count may differ from example)*
Tangent	SclLin: Value(Line(...),Dot(Real <Hx,Hy,Hz>),Tangent(...
∫	Scl : Integrate(Line(....
Eval	Scl : {numerical output} *(real part of current in **Amps**)*
Cmplx→CmplxR	CSc : {numerical output from above shown as real part of cmplx num}
Exch	*(swaps top two stack entries)*
Pop	*(drops top stack entry)*
Exch	*(swaps top two stack entries;* CVc : <Hx,Hy,Hz> *back on top)*
Cmplx→Imag	Vec : Imag(<Hx,Hy,Hz>)
Geom→Line...→{select line}	Lin : Line (line1,1000) (user line name and point count may differ from example)
Tangent	ScLin: Value(Line(...),Dot(Imag <Hx,Hy,Hz>),Tangent(...
∫	Scl : Integrate(Line(....
Eval	Scl : {numerical output} *(imaginary part of current, **Amps**)*
Cmplx→CmplxI	CSc : {numerical value from above shown as imag part of cmplx num}
Exch	*(swaps top two stack entries)*
Pop	*(drops top stack entry)*
+	CSc : {complex numerical value} *(Final complex current result)*

Calculating the Voltage Drop along a Line

Description:

Provides the complex voltage drop, in volts, between two points by integrating the E-field along a line between the two points.

Usage Example(s):

To find the voltage excited across the width of a slot antenna element; to test whether a voltage exceeds breakdown in a particular dielectric media.

Prerequisites:

The line along which the E-field is to be integrated must be created using Geometry→ Create→Line before the calculator routine can be completed. The line entered should be aligned with the E-field vector orientation.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
`Qty→E`	`CVc : <Ex,Ey,Ez>`
`Push`	*(above entry duplicated)*
`Cmplx→Real`	`Vec : Real(<Ex,Ey,Ez>)`
`Geom→Line...→{select line}`	`Lin : Line (line1,1000)` *(user line name and point count may differ from example)*
`Tangent`	`SclLin: Value(Line(...),Dot(Real <Ex,Ey,Ez>),Tangent(...`
`∫`	`Scl : Integrate(Line(....`
`Eval`	`Scl : {numerical output}` *(real part of voltage in Volts)*
`Cmplx→CmplxR`	`CSc : {numerical output from above shown as real part of cmplx num}`
`Exch`	*(swaps top two stack entries)*
`Pop`	*(drops top stack entry)*
`Exch`	*(swaps top two stack entries;* `CVc : <Ex,Ey,Ez>` *back on top)*
`Cmplx→Imag`	`Vec : Imag(<Ex,Ey,Ez>)`
`Geom→Line...→{select line}`	`Lin : Line (line1,1000)` *(user line name and point count may differ from example)*
`Tangent`	`ScLin: Value(Line(...),Dot(Imag <Ex,Ey,Ez>),Tangent(...`
`∫`	`Scl : Integrate(Line(....`
`Eval`	`Scl : {numerical output}` *(imag. part of voltage, Volts)*
`Cmplx→CmplxI`	`CSc : {numerical value from above shown as imag part of cmplx num}`
`Exch`	*(swaps top two stack entries)*
`Pop`	*(drops top stack entry)*

+	CSc : {complex numerical value} *(Final complex voltage result)*

Calculating Net Power Flow through a Surface

Description:

This recipe allows calculation of power flow through an open or closed surface by integrating the Poynting vector normal to that surface.

Usage Example(s):

This calculation could be used on scattered field data resulting from an incident wave excited HFSS project to evaluate reflection from a radome filter or FSS (frequency selective surface). It might also be used on the closed exterior surface of a solid volume to determine power dissipation within the volume (due to conservation of energy, what goes in a closed surface must come out, unless there is a loss or storage [e.g. standing wave or resonance] mechanism).

Prerequisites:

The surface on which the integration is to be performed must exist prior to completing the calculation. If the surface is the exterior of a solid object, no customer geometry creation is necessary. If the surface is only a subset of an object's faces, or a slice through the entire plane of the model not already defined by a separate 2D entity, then a Faces List and/or Cutplane must be generated to represent the integration location.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
Qty→Poynting	CVc : Poynting
Cmplx→Real	Vec : Real(Poynting) *(discards the unneeded imaginary components)*
Geom→Surface...→{select surface}	Srf : FacesList(eval_srf) *(above is example; user surface shown may vary)*
Normal	SclSrf : Value(FacesList(eval_srf), Dot(Real(Poynting), Normal... *(takes the dot product of the vector data with the normal to the surface(s) selected)*
∫	Scl : Integrate(FacesList...)
Eval	Scl : {numerical value} *(final answer in **watts**)*

Calculating the Average of a Field Quantity on a Surface

Description:

This recipe permits the user to calculate the average of a field quantity on a Surface geometry, by dividing the Integration of the field value on the surface by the surface area.

Usage Example(s):

This calculation could be used to determine the average phase of the E-field at a given cutplane through a project, to find the average current on a trace surface, or to calculate the average H-field tangential to a 2D object used as an aperture. The specific example steps below will be for the first usage example mentioned (average phase of an E-field on a surface), but the format for integration on a surface and for finding the area of the surface is identical for the other applications as well.

Prerequisites:

The surface on which the integration is to be performed must exist before completing the calculation. If the surface is the exterior of a solid object, no customer geometry creation is necessary. If the surface is only a subset of an object's faces, or a slice through the entire plane of the model not already defined by a separate 2D entity, then a Faces List and/or Cutplane must be generated to represent the integration location.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
Qty→{select field quantity}	CVc : <Ex, Ey, Ez> *(E-field used as example)*
{derive desired scalar field data desired. For example, used following Steps: Scal?→ScalarX Cmplx→CmplxPhase}	CSc : ScalarX(<Ex, Ey, Ez>) *(first operation result)* Scl : Phase(ScalarX(<Ex, Ey, Ez>)) *(second operation result)*
Geom→Surface...→{select surface}	Srf : CutPlane(plane1) *(above is example; user surface shown may vary)*
Value	SclSrf : Value(CutPlane(plane1), Phase(ScalarX(<Ex, Ey, Ez>)))
∫	Scl : Integrate(...)
Geom→Surface...→{select surface}	Srf : CutPlane(plane1)
Unit Vec→Normal	Vec : Normal(CutPlane(plane1))
Geom→Surface...→{select surface}	Srf : CutPlane(plane1)
Normal	SclSrf: Value(CutPlane(plane1)... *(takes the dot product of the surface with its own normal)*
∫	Scl : Integrate(CutPlane(... *(top entry is placeholder for the surface area of the plane)*
/	Scl : /(Integrate(Value(...
Eval	Scl : {numerical value} *(final answer in units of scalar data; for this example units are in **degrees** or **radians**)*

Calculating the Peak Electrical Energy in a Volume

Description:

This recipe permits the user to calculate the peak electrical energy in a volume object. The solution is achieved by integrating **E**·**E**∗ within the volume.

Usage Example(s):

This calculation could be used to determine the average total energy with respect to time in a terminating resonant cavity. (In a sealed, one-port structure at resonance, energy is converted back and forth between the electrical and magnetic fields, but maintains the same total quantity; therefore the peak electrical energy is equal to the average total energy.)

Prerequisites:

The volume object within which the integration is to be performed must already exist before the computation can be completed. If the volume for integration consists of the volume of several drawing objects, a single list entry representing their combined volumes must first be created using Geometry→Create→Object List.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
Qty→E	CVc : <Ex, Ey, Ez>
Push	*(duplicates top entry)*
Cmplx→Conj	CVc : Conj(<Ex, Ey, Ez>)
Dot	CSc : Dot(<Ex, Ey, Ez>, Conj(<Ex...
Cmplx→Real	Scl : Real(Dot(<Ex, Ey, Ez>, ... *(note: the dot product of the E field with its conjugate should result in a real quantity, but the calculator does not assume the imaginary part has been cancelled, and will continue to treat the result as complex until this operation.)*
Geom→Volume...→{select volume}	Vol : ObjectList(resonator) *(above is example, user entry may differ)*
∫	Scl : Integrate(ObjectList(...
Eval	Scl : {numerical quantity}
Const→Epsi0	Scl : 8.854187827E-012
Num→Scalar→{enter ε_r for volume}	Scl : {numerical quantity}
*	Scl : {numerical quantity} *(stack entry is volume ε)*
Num→Scalar→0.5	Scl : 0.5
*	Scl : {numerical quantity}
*	Scl : {numerical quantity} *(above is electrical energy in evaluated volume, in **joules**)*

Calculating the Q of a Resonant Cavity

Description:

This recipe permits the user to calculate the Q in a homogeneous dielectric-filled cavity with uniform wall losses, using the equation:

$$Q_u = \frac{\int_\Omega |\mathbf{H}|^2 \, d\Omega}{\frac{s}{2}\oint_\Gamma |\mathbf{n} \times \mathbf{H}|^2 \, d\Gamma + tg\delta \int_\Omega |\mathbf{H}|^2 \, d\Omega}$$

where *s* is skin depth, *tg*δ is dielectric loss tangent, *n* is the surface normal for the cavity wall faces, and Γ and Ω represent wall surface area and cavity volume, respectively.[1]

Usage Example(s):

To calculate the Q of an air- or solid-dielectric filled cavity, fed with a below-cutoff port aperture, or obtained via an eigensolution.

Prerequisites:

The Object (or Object List) representing the cavity total volume must already exist, as must the Faces List corresponding to the total wall surface area of the cavity. Both can be created via the Geometry menu if necessary. The solution should be tuned to the desired resonant frequency for evaluation.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
`Qty→H`	`CVc :  <Hx, Hy, Hz>`
`Push`	*`(above entry duplicated)`*
`Cmplx→Conj`	`CVc :  Conj(<Hx, Hy, Hz>)`
`Dot`	`CSc :  Dot(<Hx, Hy, Hz>, Conj(...`
`Cmplx→Real`	`Scl :  Real(Dot(<Hx, Hy, ...`
`Geom→Volume→{select cavity volume}`	`Vol :  ObjectList(cav_total)` *`(above is example; user entry may differ)`*
`∫`	`Scl :  Integrate(ObjectList(cav...` *`(above represents energy stored in cavity volume)`*
`Push`	*`(above entry duplicated)`*
`Num→Scalar→{enter loss tan for volume}`	`Scl :  {numerical value}` *`(loss tangent for dielectric fill within cavity volume)`*

[1] The above equation is ***only*** valid for cavities filled with one dielectric material across the entire volume. For cavities with different dielectric fills (e.g. a dielectric resonator within a larger metal cavity), dielectric loss must be evaluated using integration by parts for each dielectric material volume. The equation also assumes the same conductivity for all walls, and no nonreciprocal (e.g. ferrite) property to either walls or fill.

*	Scl : *(Integrate(ObjectList(... *(above represents energy lost in dielectric material losses)*
Qty→H	CVc : <Hx, Hy, Hz>
Geom→Surface→{select cavity surfaces}	Srf : ObjectFaces(cav_tot_faces) *(above is example; user entry may differ)*
Unit Vec→Normal	Vec : Normal(ObjectFaces(cav_...
Cross	CVc : Cross(<Hx, Hy, Hz>, Norm...
Push	*(above entry duplicated)*
Cmplx→Conj	CVc : Conj(Cross(<Hx, Hy, Hz>, ...
Dot	CSc : Dot(Cross(<Hx, Hy, Hz>, ...
Cmplx→Real	Scl : Real(Dot(Cross(<Hx, ...
Geom→Surface→{select cavity surfaces}	Srf : ObjectFaces(cav_tot_faces)
∫	Scl : Integrate(ObjectFaces(...
Num→Scalar→2	Scl : 2
Const→Pi	Scl : 3.14159265358979
Const→Frequency	Scl : {current freq, in Hz}
*	Scl : {numerical result, pi*f}
Num→Scalar→{enter μ_r for walls}	Scl : {entered value, unitless}
*	Scl : {numerical result, pi*f*mur}
Const→Mu0	Scl : 1.25663706144E-006
*	Scl : {numerical, pi*f*mur*mu0}
Num→Scalar→{enter wall conductivity}	Scl : {entered value, s/meter}
*	Scl : {numerical, pi*f*mur*mu0*σ}
√	Scl : {numerical, sqrt of above}
*	Scl : {numerical result, 2*above}
1/x	Scl : {numerical result} *(above is skin depth/2)*
*	Scl : *(Integrate(ObjectFaces... *(above is energy lost in walls)*
+	Scl : +(*(Integrate(ObjectFaces...
/	Scl : /(+(*(Integrate(...
Eval	Scl : {numerical result} *(above is Q of homogeneous fill and wall conductivity cavity, unitless)*

Plotting Wave Impedance along a Line

Description:

This recipe generates a 2D plot of wave impedance in ohms vs. length for a line geometry. Wave impedance is obtained directly by taking the ratio of the transverse components of the electric field to the ratio of the transverse components of the magnetic field.

Usage Example(s):

This calculation could be used to display wave impedance vs. position along a length of waveguide with a changing cross-section. It could also be used to display the changes in wave impedance in free space at some boundary (i.e. a frequency selective surface or radome) when performed on an incident wave problem.

Prerequisites:

The line along which the impedance is to be plotted should be defined before performing this calculation. Lines are generated using the Geometry→Create→Line menu pick.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
`Qty→E`	`CVc :  <Ex, Ey, Ez>`
`Smooth`	`CVc :  Smooth(<Ex, Ey, Ez>)` *(provides data medianization for cleaner plot output)*
`Cmplx→CmplxMag`	`Vec :  CmplxMag(Smooth(<Ex, Ey, ...`
`Num→Vector→{enter unit vector in direction of propagation}`	`Vec :  <0, 0, 1>` *(Z-directed unit vector used for example)*
`Cross`	`Vec :  Cross(CmplxMag(Smooth(<...`
`Mag`	`Scl :  Mag(Cross(CmplxMag(Smooth...`
`Qty→H`	`CVc :  <Hx, Hy, Hz>`
`Smooth`	`CVc :  Smooth(<Hx, Hy, Hz>)`
`Cmplx→CmplxMag`	`Vec :  CmplxMag(Smooth(<Hx, Hy,...`
`Num→Vector→{enter unit vector in direction of propagation}`	`Vec :  <0, 0, 1>` *(Z-directed unit vector used for example)*
`Cross`	`Vec :  Cross(CmplxMag(Smooth(<...`
`Mag`	`Scl :  Mag(Cross(CmplxMag(Smooth...`
`/`	`Scl :  /(Mag(Cross(CmplxMag(Sm...`
`Geom→Line→{select line to plot along}`	`Lin :  Line(line1,1000)` *(above line is example; user's may vary)*
`2D Plot`	`{launches 2D Line Plot settings}`
`{select desired settings and click OK}`	`{2D graph displayed}` *(y axis is wave impedance in **ohms** and x axis is position along line in drawing units)*

Plotting the Phase of E Tangential to a Line/Curve

Description:

This recipe generates a 2D plot of the phase of an E-field whose vector component is tangential to a line geometry. The line geometry may also be a curve (faceted polyline).

Usage Example(s):

This calculation could be used to display the change in phase of the E field tangential to a circular path within a cylindrical dielectric resonator, when used on either a driven or eigensolution problem. Identifying the phase change along this curved path is often necessary to determine the mode index (e.g. Mode 10δ) which a particular eigensolution or S-parameter resonance represents.

Prerequisites:

The line along which the phase is to be plotted should be defined before performing this calculation. Lines are generated using the Geometry→Create→Line menu pick.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
`Qty→E`	`CVc : <Ex, Ey, Ez>`
`Geom→Line→{select desired line}`	`Lin : Line(line1,1000)` *(above line is example; user's may vary)*
`Unit Vec→Tangent`	`Vec : Tangent(Line(line1,1000))`
`Cmplx→CmplxR`	`CVc : CmplxR(Tangent(Line(line1...` *(converts unit vector to complex vector)*
`Dot`	`CSc : Dot(<Ex,Ey,Ez>),CmplxR(...`
`Cmplx→CmplxPhase`	`Scl : Phase(Dot(<Ex,Ey,Ex>), ...`
`Geom→Line→{select desired line}`	`Lin : Line(line1,1000)` *(above line is example; user's may vary)*
`2D Plot`	`{launches 2D Line Plot settings}`
`{select desired settings and click OK}`	`{2D graph displayed}` *(y axis is E field phase in **deg** and x axis is position along line in drawing units)*

Plotting the Maximum Magnitude of E Tangential to a Line/Curve

Description:

This recipe generates a 2D plot of the maximum magnitude of an E-field tangential to a line. The line may also be a faceted curve. The maximum magnitude is not necessarily tied to the same input phase value along the length of the line.

Usage Example(s):

This calculation could be used to display the maximum magnitude of an E-field at all points along a line or curve in a transmission line structure, where it is the maximum magnitude and not the magnitude along the line corresponding to a single 'snapshot in time' (single port excitation phase) that is of interest. Such data could be used to determine whether the present design might exceed dielectric breakdown voltage in a particular location.

Prerequisites:

The line along which the field data is to be plotted should be defined before performing this calculation. Lines are generated using the Geometry→Create→Line menu pick.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
`Qty→E`	`CVc : <Ex, Ey, Ez>`
`Geom→Line→{select desired line}`	`Lin : Line(line1,1000)` *(above line is example; user's may vary)*
`Unit Vec→Tangent`	`Vec : Tangent(Line(line1,1000))`
`Cmplx→CmplxR`	`CVc : CmplxR(Tangent(Line(line1...` *(converts unit vector to complex vector)*
`Dot`	`CSc : Dot(<Ex,Ey,Ez>),CmplxR(...`
`Cmplx→CmplxMag`	`Scl : CmplxMag(Dot(<Ex,Ey,Ex ...` *(above quantity is the **maximum** magnitude of the E-field tang. To the line. To obtain the mag. associated with a particular port phase excitation, enter a number into the stack and use the Cmplx→AtPhase operation instead.)*
`Geom→Line→{select desired line}`	`Lin : Line(line1,1000)` *(above line is example; user's may vary)*
`2D Plot`	`{launches 2D Line Plot settings}`
`{select desired settings and click OK}`	`{2D graph displayed}` *(y axis is max E field mag in **V/m** and x axis is position along line in drawing units)*

Plotting the E-Field Vector Along a Line

Description:

This recipe generates a vector plot in the 3D graphical environment of the E-field orientation along a line geometry, relative to a given excitation phase value.

Usage Example(s):

This calculation could be used to display the rotating E-field orientation along the axis of a circularly-polarized horn antenna, or to display the sum E-field orientation resulting from the excitation of two unequal mode amplitudes along a waveguide. (Vector plotting along a line is not directly supported by use of the Plot→Fields menu.)

Prerequisites:

The line along which the field data is to be plotted should be defined before performing this calculation. Lines are generated using the Geometry→Create→Line menu pick. For the graphical vector display, the user needs to identify an appropriate number of sampling points along the line which will display well as individual vectors.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
`Qty→E`	`CVc :  <Ex, Ey, Ez>`
`Num→Scalar→{enter phase reference for which vectors are desired}`	`Scl : 0` *(zero degrees used for example)*
`Cmplx→AtPhase`	`Vec :  AtPhase(<Ex,Ey,Ez>, 0)`
`Geom→Line→{select line}`	`Lin :  Line(line1,75)` *(note reduced point samples along line for better vector display)*
`Value`	`VecLin: Value(Line(line1,75), ...`
`Plot`	`{launches Vector 3D Line Plot settings}`
`{select desired settings and click OK}`	`{Vector Plot along Line displayed}` *(each vector represents E-field value and orientation in V/m)*

Plotting the E-Field Magnitude Normal to a Surface

Description:

This recipe generates a scalar intensity plot of the E-field magnitude normal to a particular surface (or group of object surfaces, list of object faces), relative to a given input phase excitation.

Usage Example(s):

This calculation could be used instead of the automatic Plot→Fields→MagE upon surface, when only the magnitude of the E-field with a particular vector orientation is desired. For example, to evaluate the field available for coupling to a probe structure with a particular orientation.

Prerequisites:

The plane to which the desired field component should be normal must be generated prior to completing the following steps. Planes available for this routine can be generated using the Geometry→Create→Cutplane, Geometry→Create→Faces List, ***or*** Geometry→Create→Surface List menu selections.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
`Qty→E`	`CVc : <Ex, Ey, Ez>`
`Num→Scalar→{enter phase reference for which data is desired}`	`Scl : 0` *(zero degrees used for example)*
`Cmplx→AtPhase`	`Vec : AtPhase(<Ex,Ey,Ez>, 0)`
`Geom→Surface→{select desired cutplane, faces list, or surface list}`	`Srf : FacesList(faces1)` *(faces list used as example)*
`Normal`	`SclSrf: Value(FacesList(faces1), Dot(AtPhase(<Ex,Ey,Ez>,0), ...`
`Plot`	`{launches Scalar Surface Plot settings}`
`{select desired settings and click OK}`	`{Scalar Plot on faces displayed}` *(E-field component magnitude value in V/m)*

Generating an Iso-Surface Contour for a Given Field Value

Description:

This recipe generates a geometry entry called an IsoSurface which represents the surface upon which a selected scalar field quantity has a single value. This surface can be displayed, or used in later operations (to plot other quantities upon, etc.).

Usage Example(s):

This calculation could be used to locate regions of excessive field magnitudes for voltage breakdown or ohmic heating analysis. It could also be used to generate a desired isosurface to be used as an integration surface for another quantity.

Prerequisites:

Prior plotting of the field quantity of interest to determine the isovalue to use may be necessary. Isovalues should be entered in MKS units (e.g. V/m, A/m) unless the problem is an eigensolution, in which case all field values are normalized to a peak of 1.0.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
Qty→E	CVc : <Ex, Ey, Ez> *(IsoSurfaces for other quantities can also be created; E used as example.)*
Smooth	CVc : Smooth(<Ex,Ey,Ez>) *(as this routine generates a surface geometry object, data smoothing is recommended)*
Num→Scalar→{enter phase reference for which vectors are desired}	Scl : 0 *(zero degrees used for example)*
Cmplx→AtPhase	Vec : AtPhase(Smooth(<Ex,Ey,Ez>...
Mag	Scl : Mag(AtPhase(Smooth(<Ex,...
Num→Scalar→{enter desired iso value}	Scl : 0.75 *(eigensolution relative quantity used as example)*
Iso	IsoSurface(Mag(AtPhase(Smooth(...
Draw	{launches Surface Drawing settings}
{pick desired settings}	{IsoSurface contour is displayed}

Generating an Animation on a Plane with respect to Excitation Phase

Description:

This recipe generates animated field output on a single geometric plane, with the animation frames varying as the excitation source phase is stepped through a user-defined range. In this particular case the animation will be of the E field magnitude, although any other derived field quantity could also be animated.

Usage Example(s):

This calculation permits the user to generate their own animated output results in addition to those automatically available from the post-processor. For example, displacement current $D = \varepsilon E$ could be animated vs. input phase.

Prerequisites:

The geometric surface on which plotting is to be performed must already exist.

Calculator Operation	Resulting Stack Display (top entry only unless noted)
`Qty→E`	`CVc : <Ex, Ey, Ez>` *(Animations for other quantities can also be created; E used as example.)*
`Smooth`	`CVc : Smooth(<Ex,Ey,Ez>)`
`Func→Scalar→"PHASE"`	`Scl : PHASE` *(phase is a variable or function)*
`Cmplx→AtPhase`	`Vec : AtPhase(Smooth(<Ex,Ey,Ez>,..`
`Mag`	`Scl : Mag(AtPhase(Smooth(<Ex,...`
`Geom→Surface→{pick surface}`	`Srf : CutPlane(xy)` *(XY plane entity used as example)*
`Anim`	`{launches Animation Plot settings}`
`{pick desired settings}`	`{Animation is displayed}`

Generating an Animation on Multiple Planes with a Positional Variable

Description:

This recipe generates animated field output in which each frame is a snapshot of the fields on a different plane of the modeled volume. Any derived field quantity could be plotted in this manner, but this example will simply use the E-field magnitude at zero degrees input excitation.

Usage Example(s):

This calculation permits the user to generate their own animated output results in addition to those automatically available from the post-processor. For example, peak E field (E dot E conjugate) could be plotted at multiple planes in sequence.

Prerequisites:

This operation will only work in the global coordinate system if using X, Y, or Z positions as the animation 'variable'.

Calculator Operation	**Resulting Stack Display (top entry only unless noted)**
`Qty→E`	`CVc : <Ex, Ey, Ez>` *(Animations for other quantities can also be created; E used as example.)*
`Smooth`	`CVc : Smooth(<Ex,Ey,Ez>)`
`Num→Scalar→"0"`	`Scl : 0`
`Cmplx→AtPhase`	`Vec : AtPhase(Smooth(<Ex,Ey,Ez>,..`
`Mag`	`Scl : Mag(AtPhase(Smooth(<Ex,...`
`Func→Scalar→"X"`	`Scl : X` *(we will animate along YZ planes as our example)*
`Func→Scalar→"XPOS"`	`Scl : XPOS` *(XPOS is a function which invokes the Animation display)*
`Iso`	`Srf : IsoSurface (X, XPOS)` *(An IsoSurface with X = Xpos is a YZ plane that can vary with user-entered limits on XPOS)*
`Anim`	`{launches Animation Plot Settings}`
`{pick desired settings}`	`{displays animation}`

Calculator Data Extraction (Outputs)

Data can be extracted from the calculator stack register via a number of different operations. The data extracted can have many different forms, depending on the nature of the specific stack register of interest (e.g. vector data, scalar data, etc.)

Single-Value Outputs

A single-point scalar, vector, or complex (numerical) result can be simply written down by the user as it appears in the calculator stack. For use by the Ansoft Macro Language (for example, to store a value in a database for later operations), a simple numerical registry entry can also be assigned to a variable, using the command syntax:

```
Assign {variablename} GetTopEntryValue
```

For use as an output variable by the Optimetrics™ parameterization and optimization module, the *Write* button in the calculator *Output* column permits name assignment to the stack entry, followed by a save filename location. Using this extraction technique allows Optimetrics™ to consider the exported stack value directly as an output, or as a variable in a derived output (e.g. optimization goal function) quantity.

Outputs for Calculation in Other Post-Processor Sessions

If the calculator operations performed have obtained a stack entry that is intended for use in still other calculator operations (e.g. after altering the field data set within this post-processing session, or within another problem's post-processing session entirely), the stack entry is saved for this purpose by using the *Write* button in the calculator *Output* column. Note that in this case no variable name will be requested; only a save filename location will be required. This function will *not* work for field values derived upon a specific geometric quantity (those containing either *Lin, Srf,* or *Vol* in the stack data type indicator) as the calculator cannot know that these geometric quantities exist in identical forms in other post-processing sessions.

Outputs for Post-Processing outside of HFSS or Optimetrics

To output a field quantity or calculation result for use by some third-party post-processor, use the *Export* dropdown in the calculator *Output* column. The *Export* dropdown contains options for outputs *To File*, which requires a pre-existing file of three-dimensional (cartesian) location points at which field data is desired, or *On Grid*, which outputs data on a cartesian grid specified by the user by direct entry of X, Y, and Z coordinate ranges and spacings. Further detail regarding the use of the *Export* options can be found in the on-line help.