



Aether 4.0 Reference Manual

Three-dimensional electromagnetic fields

PO Box 13595, Albuquerque, NM 87192 U.S.A.
Telephone: +1-505-220-3975
Fax: +1-617-752-9077
E mail: techinfo@fieldp.com
Internet: <http://www.fieldp.com>

Contents

1	Introduction to Aether	4
1.1	Program functions	4
1.2	Solution procedure and package components	5
1.3	Computational modes	7
1.4	Learning Aether	7
2	Time-domain solutions of the Maxwell equations	9
2.1	Basic equations and limiting conditions	9
2.2	Element divisions on a regular mesh	10
2.3	Advancing the electric field	12
2.4	Advancing the magnetic field	14
2.5	Courant stability condition	15
2.6	Absorbing layers	16
3	Time variations	18
3.1	Tabular functions	18
3.2	Algebraic functions	20
3.3	Standard parametric pulses	20
3.4	Fourier transforms	25
3.5	Pulses in the frequency domain	27
4	Frequency-domain techniques	29
4.1	Complex-number values in the RF mode	29
4.2	Converting harmonic functions to phasors	29
4.3	Electromagnetic energy and power expressions	30
5	Pulse mode commands	32
5.1	Using MetaMesh with Aether	32
5.2	Pulse mode setup dialog	32
5.3	Syntax and structure of the Aether control script	36
5.4	Run control commands	37
5.5	Current source commands	40
5.6	Commands to set material properties	42
5.7	Diagnostic commands	44
6	Res mode commands	46
6.1	Res mode setup dialog	46
6.2	Res mode script commands	48
6.3	Calculation methods and data records	50
7	RF mode commands	52
7.1	RF mode setup dialog	52
7.2	Script commands	54
7.3	Monitoring solution convergence	57

8	Aerial – file and analysis operations	59
8.1	File operations	59
8.2	Global solution analysis	62
9	Aerial – plane plots	66
10	Aerial – slice plots	71
10.1	Setting the slice view	71
10.2	Setting slice plot properties	72
10.3	Analyses in a slice	74
10.4	Vector tools	78
11	Aerial – surface plots	80
12	Aerial – script operation	86
13	Probe – history file plot utility	89
13.1	Introduction	89
13.2	Loading data files	90
13.3	Plot settings	90
13.4	Plot functions	92
13.5	Information	94
13.6	Using Probe with Aether	94
14	MagWinder – defining applied current	96
14.1	Program function	96
14.2	Starting an assembly	97
14.3	Adding a coil	98
14.4	Adding a part	99
14.5	Position and orientation of parts and coils	100
14.6	Part models	101
14.7	Editing coils and parts	106
14.8	Displaying an assembly	108
14.9	Additional MagWinder features	110
14.10	Structure of the coil definition file	111
14.11	Coil commands	113
14.12	Part commands	114
14.13	Structure of the current element file	116
15	Output file formats	117

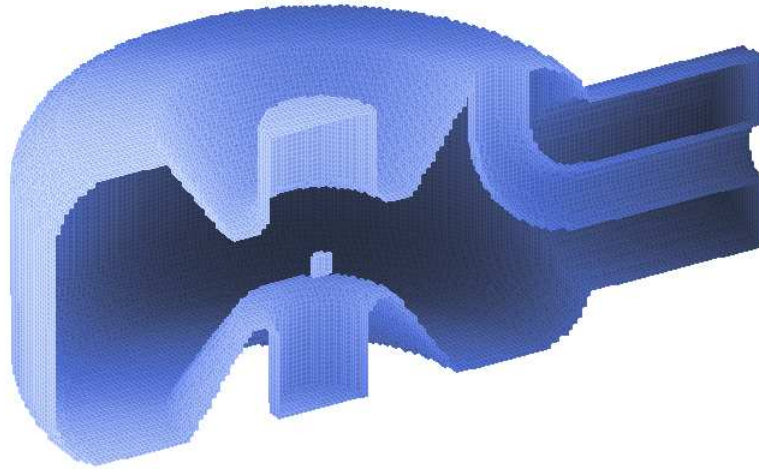


Figure 1: Mesh representation of a three-dimensional resonant cavity.

1 Introduction to Aether

Aether¹ is a unified, finite-element package for calculating electromagnetic fields in three-dimensional structures. The program employs the FETD (finite-element time-domain) method for fast solutions. **Aether** covers the full range of RF and microwave applications as well as pulsed or harmonic magnetic fields with eddy currents. The codes are optimized for personal computers with efficient memory utilization and high speed. A typical modern machine can represent meshes with millions of elements and accomplish solutions in minutes.

1.1 Program functions

Aether handles the two main classes of electromagnetic calculations:

- Time-domain (initial-value) solutions: evolution of pulsed fields driven by specified currents or an electrostatic charge.
- Frequency-domain (boundary-value) solutions: steady-state excitation of electromagnetic waves at a single frequency.

Time-domain calculations are appropriate for pulsed-power devices, electromagnetic interference from transient events and the diffusion of magnetic fields into conductive structures. In the frequency domain, **Aether** handles both closed and open structures. Closed system tasks include searches for resonant frequencies, generation of mode field distributions and calculation of Q factors. Open-system solutions include antenna simulation and the determination of S parameters for microwave networks.

¹Aether was the Greek god of the pure air that the gods breathed as opposed to the ordinary air breathed by mortals. Aether was one of the fundamental deities of the cosmos, the soul from which all life emanated. The name has come down to us as the ether, the mysterious substance postulated to carry electromagnetic waves.

1.2 Solution procedure and package components

Electromagnetic fields are described by the Maxwell equations. Digital computers cannot solve partial differential equations directly. Instead, the equations must be converted to an equivalent set of coupled linear equations. The *computational mesh* is the key to this conversion. The strategy is to divide the solution space into small pieces, or *elements*. The division should satisfy the following conditions:

- The elements must be small compared to the scale length for variations of field quantities (*e.g.*, the electromagnetic wavelength).
- Each element should have a unique material identity. Therefore, the boundaries between elements (*facets*) should lie approximately on the boundaries between materials (*e.g.*, dielectrics, ferrites, conductors,...).

Users of older programs may consider mesh generation as a task to be avoided. Our modern graphical, interactive programs make the procedure relatively easy. Furthermore, volume meshes provide the most effective path to modeling systems with complex distributions of three-dimensional material objects.

A complete **Aether** solution consists of several steps. We have designed the package so that you can proceed in an organized way, checking results at each stage. The following list shows the main tasks and the associated programs.

- **Visualize and define the geometry**
Geometer provides an interactive environment where you can build a three-dimensional assembly from *parts*. Parts may range from simple geometric shapes (boxes, spheres and cones) to complex objects created with 3D CAD programs. When the assembly is complete, **Geometer** writes an archiving text script. You may also create or modify the geometry script with a text editor.
- **Generate the mesh**
MetaMesh reads the geometry script, divides the solution space into elements and associates them with material objects of the solution volume. The program creates a binary record of the element boundaries and material identities. **MetaMesh** has two- and three-dimensional plotting capabilities to check the resulting mesh.
- **Set up run controls**
Information on run times, drive current sources, material properties and diagnostics is required for an **Aether** run. The program features interactive dialogs where you can enter information and then create a text script to document the run. Again, you can work directly with an editor.
- **Generate linear equations and advance them in time**
Aether, the main program of the package, analyzes the mesh and run parameters and generates the appropriate finite-element equations. The program then advances electric and magnetic fields in time until a specified condition is reached. For pulsed fields, **Aether** runs to a given time, recording diagnostic files. In a resonance search, the program excites the structure and performs Fourier transforms of probe signals to identify peaks. In a

Table 1: Programs and associated files of the **Aether** package

Program	File name	I/O	Function
Geometer	AnyName.OTL	I	Extrusion and turning outlines
	AnyName.PTH	I	Path of a neon model
	AnyName.STL	I	Shape import from CAD program
	RunName.MIN	I/O	MetaMesh script
MetaMesh	Runname.MIN	I	Geometry definition script
	RunName.MDF	O	Mesh definition file
MagWinder	AnyName.CDF	I	Definitions of drive coils
	AnyName.WND	O	Current element representation
Aether	RunName.MDF	I	Geometry definition
	RunName.AIN	I	Run control script
	AnyName.WND	I	Drive current elements
	RunName.P01	O	History monitor output
	RunName.001	O	Space files, <i>Pulse</i> mode
	RunName.AOU	O	Space file, <i>RF</i> mode
Probe	RunName.P01	I	History monitor record
Aerial	RunName.001	I	Space file, <i>Pulse</i> mode run
	RunName.AOU	I	Space file, <i>RF</i> mode
	AnyName.SCR	I	Automatic analysis script

solution involving RF or microwave fields, **Aether** runs a time-domain solution to an equilibrium and then converts the field values to phasor form for analysis.

- **Analyze the results**

A pulsed-field solution creates two types of output: 1) history records of field quantities as a function of time at monitor points and 2) space files that record field quantities at all spatial positions at a given time. History files are analyzed with **Probe**, a plotting and analysis utility compatible with all Field Precision initial-value codes. The **Aerial** post-processor is used for space files. This program creates a variety of plots and includes numerical analysis routines for electromagnetic fields. Boundary-value solutions for RF fields result in a single output file with information on the amplitude and phase of field quantities over space. This information may also be plotted and analyzed with **Aerial**.

The package includes the **MagWinder** utility. This program provides an alternate method for entering drive currents into **Aether**. It builds current-element sets to represent three-dimensional coil assemblies. The elements are converted to drive current densities in **Aether**. **MagWinder** is particularly useful for pulsed and RF magnets.

The programs of the **Aether** package communicate through data files. Table 1 summarizes the **Aether** programs and functions of the files. Additional text files may be used to define temporal modulations of drive currents and conductivity.

1.3 Computational modes

Aether can operate in three computational modes to address different types of electromagnetic solutions:

- **Pulse mode**

The program performs a straightforward time-domain solution to find pulsed electromagnetic fields. In a standard solution, the fields $|\mathbf{E}|$ and $|\mathbf{H}|$ have zero values at all positions at $t = 0.0$ s. Fields are generated by current distributions with prescribed spatial and temporal variations. An alternate pulse-initiation method is to start with $|\mathbf{H}| = 0.0$ and non-zero values of $|\mathbf{E}|$ calculated by the electrostatic program **HiPhi**. In this state (which could represent a charged transmission line), pulses are typically generated by changing the conductivity of a region (*i.e.*, a switch). Both methods may be combined in a solution. *Pulse* mode calculations may generate history files (temporal records of field quantities at specified positions) and spatial files (spatial records at specified times).

- **Res mode**

The activity in this mode is to find the resonant modes of three-dimensional structures. **Aether** excites a time-domain solution with a uniform current density in a small drive region. The Fourier transform of the drive waveform is a rounded step function with a user-specified frequency width and central value. The program samples the field response at one or more probe points. The result of the calculation is a set of Fourier-transformed probe signals and a table of identified peaks in the listing file. The location of the drive region and the positions of the probes may be chosen for preferential detection of mode types. The resonant frequencies may be used in an *RF* mode calculation to find field distributions and Q values for the modes.

- **RF mode**

Aether determines a frequency-domain solution by running a time-domain solution to equilibrium and then converting the field values to phasor form. In contrast to the *Pulse* mode, the waveforms for drive currents are harmonic functions at a single frequency f_0 . Drive regions may have prescribed spatial variations of current density and phase offsets. This versatility makes it relatively easy to excite modes of transmission lines and waveguides. The output is a single space file listing field phasors and material properties.

1.4 Learning Aether

It is important to start with realistic expectations. **Aether** is a tool to leverage your experience with electromagnetic fields. It creates numerical solutions for complex three-dimensional systems. These solutions would be difficult or even impossible to find with analytic techniques such as series expansions. Like any computational aid, **Aether** is most effective when you have a solid understanding of the theory underlying your application.

- Except for highly specialized cases, it is improbable that a computer program could devise the optimal approach to your application through *artificial intelligence*. The quality of solutions from technical software depends largely on your knowledge.

- Good solutions come from careful planning, organization and checking. The button-pushing mode generally accomplishes nothing beyond the consumption of time.

The implication is that it is important to read this instruction manual and to approach your application in steps, starting from simple examples. Two basic computer skills are essential. The first is a knowledge of the file structure of your hard disk and an ability to make and to modify directories. The second is the ability to read and to create text files. We supply a good file manager and text editor if you don't already have such programs.

The **MetaMesh** manual supplied with **Aether** covers the subject of mesh generation. The manual gives detailed instructions on using **Geometer** and **MetaMesh**. The **Aether** package includes a tutorial manual of worked electromagnetic examples and a library of input files. The examples are a quick introduction to **Aether** capabilities and procedures. Furthermore, they can provide good starting points for your calculations.

This manual covers methods used in the programs and gives detailed information on program features. The following chapter reviews the mechanics of time-domain numerical solutions of the Maxwell equations. Although it is not necessary to understand all details, it is helpful to skim the chapter to get a sense of how the programs work. Chapter 3 covers methods to define temporal functions in **Aether**. The chapter also reviews Fourier transforms and the frequency content of standard pulses. Chapter 4 covers **Aether** techniques to create frequency-domain solutions.

The next three chapters review how to set up calculations in the **Aether** computation modes: *Pulse* (Chap. 5), *Res* (Chap. 6) and *RF* (Chap. 7). The first section of each chapter describes use of interactive dialogs for quick generation of control scripts. The following sections give detailed descriptions of advanced script commands that you can set or modify with a text editor.

Capabilities of the **Aerial** post-processor are described in the next five chapters. The program automatically adjusts its capabilities to match the solution type (time- or frequency-domain). Chapter 8 discusses file operations and numerical analysis functions. Chapter 9 describes plane plots, attractive displays of two-dimensional information for publications and presentations. The slice plot menu, described in Chap. 10, is a useful environment for interactive solution analysis. Point and scan calculations of can be controlled with mouse operations. You can also generate plots of electric and magnetic field lines. Chapter 11 reviews surface plots, three-dimensional views of the solution space. Here you can plot field quantities over a plane normal to a Cartesian or over the surfaces of regions. You can also add three-dimensional field lines. Chapter 12 concludes the discussion of **Aerial** with a review of analysis scripts. You can define automatic analysis sequences that can be invoked in an interactive window or in the background under batch file control.

The **Probe** utility, described in Chap. 13, is used to plot and to analyze history records created by *Pulse* mode solutions. Chapter 14 is a complete reference on **MagWinder**. This utility program is an efficient path to define drive currents to represent complex magnet coils. Finally, Chap. 15 describes the **Aether** binary file format for advanced users who want to write custom analysis programs.

2 Time-domain solutions of the Maxwell equations

2.1 Basic equations and limiting conditions

We must solve the full set of Maxwell equations to determine electromagnetic fields. This section reviews the equation set in a form appropriate for most pulse-power and microwave applications. The differential form of the equations in SI units is

$$\nabla \times \mathbf{E} = -\frac{\partial(\mu\mathbf{H})}{\partial t}, \quad (1)$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial(\epsilon\mathbf{E})}{\partial t}, \quad (2)$$

$$\nabla \cdot (\epsilon\mathbf{E}) = \rho, \quad (3)$$

$$\nabla \cdot (\mu\mathbf{H}) = 0. \quad (4)$$

The field quantities are the electric field \mathbf{E} in units of volts/m and the magnetic field intensity \mathbf{H} in units of amperes/m. Materials are characterized by the *dielectric permittivity* ϵ and the *magnetic permeability* μ . The permittivity can be expressed as the product $\epsilon = \epsilon_r\epsilon_0$, where ϵ_r is the *relative permittivity* or *dielectric constant* and $\epsilon_0 = 8.85419 \times 10^{-12}$ (farad/m). Similarly, we can write the permeability as $\mu = \mu_r\mu_0$, where μ_r is the *relative permeability* and $\mu_0 = 1.25664 \times 10^{-6}$ (henry/m).

We shall adopt several limiting assumptions that simplify numerical calculations and significantly reduce the run time for three-dimensional calculations. We assume that materials are isotropic so that the values of ϵ and μ do not depend on direction. This condition means that \mathbf{E} is parallel to the *electric flux density* \mathbf{D} and that \mathbf{H} is parallel to the *magnetic flux density* \mathbf{B} . Another condition inherent in the form of Eqs. 1-4 is that the value of ϵ is independent of $|\mathbf{E}|$ and that μ does not depend on $|\mathbf{H}|$. In other words, fields have moderate levels that do not change the properties of materials. In this limit, field quantities are related by the relationships:

$$\mathbf{D} = \epsilon\mathbf{E}, \quad (5)$$

$$\mathbf{B} = \mu\mathbf{H}. \quad (6)$$

When Eqs. 5 and 6 hold, the Maxwell equations are linear and should always lead to stable solutions.

The driving terms in Eqs. 1-4 are the space-charge density ρ and the current density \mathbf{J} . They are related through the equation of continuity,

$$\frac{\partial\rho}{\partial t} + \nabla \cdot \mathbf{J} = 0, \quad (7)$$

an expression of conservation of charge.

Aether is designed to model pulse-power and microwave devices rather than to simulate the dynamics of charged-particle beams or plasmas. If we assume that there is no free space-charge

density ($\rho = 0$), it is sufficient to solve only Eqs. 1 and 2 to find a field solution. We can verify this assertion by taking the divergence of the equations. Using the condition $\nabla \cdot \mathbf{J} = 0$ and the vector identity

$$\nabla \cdot (\nabla \times \mathbf{A}) = 0, \quad (8)$$

we can show that the conditions of Eqs. 3 and 4 are contained in Eqs. 1 and 2 if \mathbf{E} and \mathbf{H} are initially zero.

For numerical electromagnetic solutions, it is useful to start from the integral form of the Maxwell equations. It is convenient to divide the current density into two components:

$$\mathbf{J} = \mathbf{J}_0 + \sigma \mathbf{E}. \quad (9)$$

In Eq. 9, \mathbf{J}_0 is the applied current density. We specify this quantity to generate electromagnetic pulses and waves. The second component (a representation of Ohm's law) is the resistive current density driven by the electric field. The quantity σ is the electrical conductivity in siemens/m.

Using Stoke's theorem, Eqs. 1 and 2 can be rewritten as

$$\oint \mathbf{E} \cdot d\mathbf{l} = - \int \int dA \frac{\partial}{\partial t} (\mu \mathbf{H} \cdot \mathbf{n}), \quad (10)$$

$$\oint \mathbf{H} \cdot d\mathbf{l} = \int \int dA \left[\frac{\partial}{\partial t} (\epsilon \mathbf{E} \cdot \mathbf{n}) + \mathbf{J}_0 \cdot \mathbf{n} + \sigma \mathbf{E} \cdot \mathbf{n} \right]. \quad (11)$$

The circuit and surface integrals may be applied over any surface in space. Because the conductivity does not appear in a time derivative, a time variation of conductivity can easily be incorporated in **Aether** solutions. This is a useful feature for modeling pulsed-power switches.

2.2 Element divisions on a regular mesh

Digital computers cannot solve partial differential equations. Such equations have a unique solution at every point in space and contain an infinite amount of information. One resolution is to limit attention to a large but finite number of points and to apply the equations in an average sense around the points. The general technique of numerical electromagnetics is to divide space into small volumes or *elements*. If the element size is small compared to scale length for field variations, we can represent \mathbf{E} and \mathbf{H} at discrete points and determine intermediate values from interpolation functions. Field values at different points are related by applying the integrals of Eqs. 10 and 11. The end result is a large set of coupled linear equations that can be easily advanced by a computer.

In contrast to other programs of the **AMaze** series, **Aether** uses a *regular* rather than a *conformal* mesh. The term *regular* implies that elements have the shape of boxes rather than generalized hexahedrons. A conformal mesh is practical when dealing with a single scalar field quantity like the electrostatic potential. With two coupled vector quantities, the linear equations to connect points on a conformal mesh have a huge number of terms, greatly increasing storage requirements and run time. Furthermore, highly distorted elements can introduce spurious modes or numerical instabilities in electromagnetic calculations. With the memory

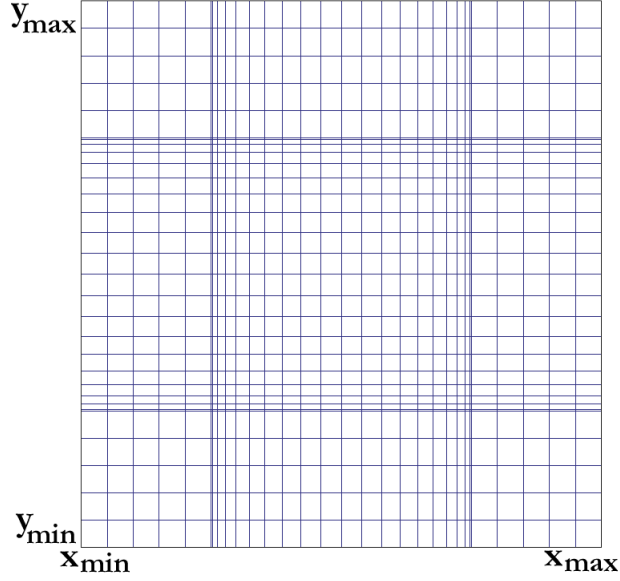


Figure 2: Projection of a regular mesh normal to the z axis.

capacity of modern computers, the best approach to electromagnetic calculations is to use a regular mesh with a large number of elements.²

The **Aether** mesh (Figure 2) covers a solution volume in Cartesian space with dimensions $x_{min} \leq x \leq x_{max}$, $y_{min} \leq y \leq y_{max}$ and $z_{min} \leq z \leq z_{max}$. The facets of elements lie at positions x_i , y_j and z_k , wheremesh!indices

$$0 \leq i \leq I, \quad x_0 = x_{min}, \quad x_I = x_{max}, \quad (12)$$

$$0 \leq j \leq J, \quad y_0 = y_{min}, \quad y_J = y_{max}, \quad (13)$$

$$0 \leq k \leq K, \quad z_0 = z_{min}, \quad z_K = z_{max}. \quad (14)$$

An element has an index (i, j, k) – the range of element indices is $1 \leq i \leq I$, $1 \leq j \leq J$ and $1 \leq k \leq K$. The widths of an element along the Cartesian axes is

$$\Delta x_i = x_i - x_{i-1}, \quad (15)$$

$$\Delta y_j = y_j - y_{j-1}, \quad (16)$$

$$\Delta z_k = z_k - z_{k-1}, \quad (17)$$

Finally, the average position of an element is given by

$$\bar{x}_i = (x_i + x_{i-1})/2, \quad (18)$$

$$\bar{y}_j = (y_j + y_{j-1})/2, \quad (19)$$

$$\bar{z}_k = (z_k + z_{k-1})/2. \quad (20)$$

²Note that **Aether** will automatically convert conformal meshes from **MetaMesh** to a regular form on input.

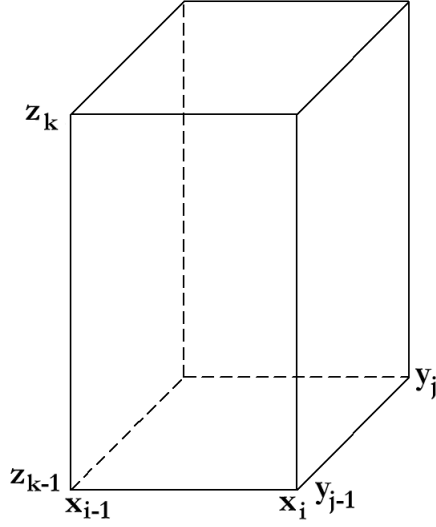


Figure 3: Boundaries of element $[i, j, k]$.

Figure 3 illustrates index conventions for an element. Material properties $\epsilon_{i,j,k}$, $\mu_{i,j,k}$ and $\sigma_{i,j,k}$ are associated with elements. As in the other **AMaze** programs, elements are divided into contiguous groups called *regions* to represent different types of materials. In the derivation of difference equations, we assume that material properties are uniform over an element. Therefore, the element sizes must be small enough to represent continuous or discrete variations of properties. Material properties may be uniform over the elements of a region, or may vary according to specified mathematical relationships.

To achieve second-order accuracy advancing Eqs. 10 and 11, it is necessary to offset the discrete representation of \mathbf{E} and \mathbf{H} in both time and space. Along the time axis, \mathbf{E} is defined at integer multiples of the time step ($t_n = n\Delta t$), while \mathbf{H} is defined at half-integer multiples, ($n + \Delta/2t$). In space, we apply the following convention. Electric field values are defined at the center points of elements (Eqs. 18 - 20). The drive current \mathbf{J}_0 is also defined at element centers and calculated a half-integer times. The magnetic intensity is defined at *nodes*, where a node is the intersection of element facets. The positions of nodes are given by Eq. 12 - 14. The mesh includes $I \times J \times K$ values of \mathbf{E} and $(I+1) \times (J+1) \times (K+1)$ values of \mathbf{H} . As in any numerical method in a finite volume, there is always a question of boundary conditions. There are no values of \mathbf{E} outside the solution volume to calculate curl integrals to advance the outer values of \mathbf{H} . We shall see in the next chapter that it is possible to implement conducting-wall, open-circuit and open-space conditions by setting boundary values to $\mathbf{H} = \mathbf{0}$. In this case, the number of node values of magnetic field to advance is $(I-1) \times (J-1) \times (K-1)$.

2.3 Advancing the electric field

To begin, we shall convert Eq. 11 to a difference equation. A major advantage of a regular mesh is that the vector components of electric field may be advanced independently. As an example, consider the difference form of Eq. 11 to advance E_z . Figure 4 shows element $[i, j, k]$ viewed from the $+z$ direction. The right-hand side of Eq. 11 can be approximated as:

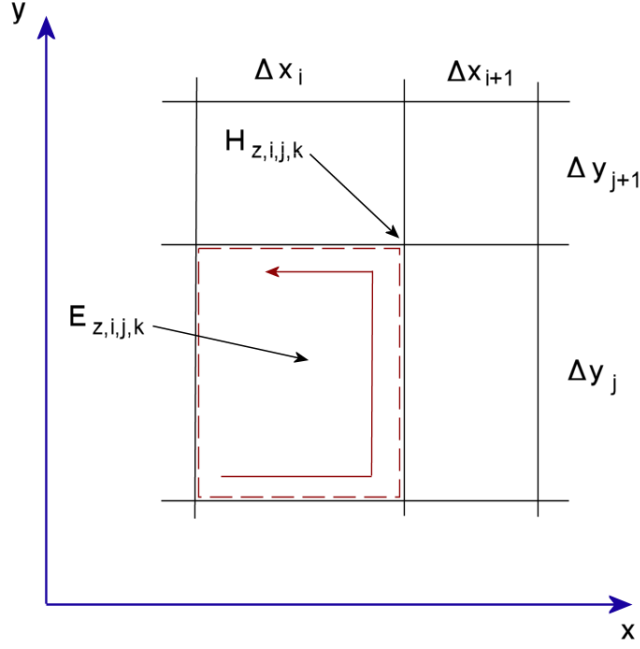


Figure 4: Path of the circuit integral of \mathbf{H} to advance the z component of electric field.

$$\Delta x_i \Delta y_j \left(\epsilon_{i,j,k} \frac{\partial E_{z,i,j,k}}{\partial t} + \sigma_{i,j,k} E_{z,i,j,k} + J_{0z,i,j,k} \right). \quad (21)$$

The dashed line in Fig. 4 is the path for the circuit integral on the left-hand side of the equation. Using averages of \mathbf{H} over the eight points of the element, the integral can be written as:

$$\begin{aligned} \text{Int} H_{z,i,j,k} = & \quad (22) \\ & [H_{x,i-1,j-1,k-1} + H_{x,i-1,j-1,k} + H_{x,i,j-1,k-1} + H_{x,i,j-1,k}] \frac{\Delta x_i}{4} + \\ & [H_{y,i,j-1,k-1} + H_{y,i,j-1,k} + H_{y,i,j,k-1} + H_{y,i,j,k}] \frac{\Delta y_j}{4} - \\ & [H_{x,i,j,k-1} + H_{x,i,j,k} + H_{x,i-1,j,k-1} + H_{x,i-1,j,k}] \frac{\Delta x_i}{4} - \\ & [H_{y,i-1,j,k-1} + H_{y,i-1,j,k} + H_{y,i-1,j-1,k-1} + H_{y,i-1,j-1,k}] \frac{\Delta y_j}{4}. \end{aligned}$$

Final step to complete the difference equation is to introduce centered expressions in time. The superscript n denotes an expression evaluated at time $t = n\Delta t$. We can rewrite Eq. 21 as

$$\Delta x_i \Delta y_j \left[\epsilon_{i,j,k} \frac{E_{z,i,j,k}^{n+1} - E_{z,i,j,k}^n}{\Delta t} + \sigma_{i,j,k} \frac{E_{z,i,j,k}^{n+1} + E_{z,i,j,k}^n}{2} + J_{0z,i,j,k}^{n+1/2} \right]. \quad (23)$$

The combination of Eqs. 22 and 23 gives an equation to advance the z component of electric field in element $[i, j, k]$:

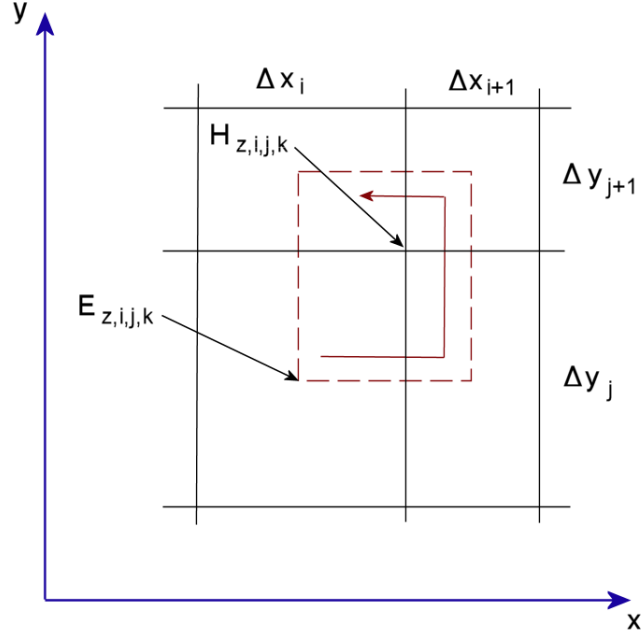


Figure 5: Path of the circuit integral of \mathbf{E} to advance the z component of magnetic field.

$$E_{z,i,j,k}^{n+1} = \left[\frac{1}{\epsilon_{i,j,k}/\Delta t + \sigma_{i,j,k}/2} \right] \times \quad (24)$$

$$\left[\left(\frac{\epsilon_{i,j,k}}{\Delta t} - \frac{\sigma_{i,j,k}}{2} \right) E_{z,i,j,k}^n - J_{z,i,j,k}^{n+1/2} + \frac{Int H_{z,i,j,k}^{n+1/2}}{\Delta x_i \Delta y_j} \right].$$

2.4 Advancing the magnetic field

Conversion of Eq. 10 to difference form gives a complete set of equations to advanced \mathbf{E} and \mathbf{H} . To illustrate, consider an expression to advance $H_{z,i,j,k}$. Figure 5 shows the integration surface. It passes through node $[i, j, k]$, encompassing sections of the facets of eight elements. We can write the right-hand side as

$$\int dx \int dy \mu(x, y) \left(\frac{H_{z,i,j,k}^{n+3/2} - H_{z,i,j,k}^{n+1/2}}{\Delta t} \right). \quad (25)$$

We shall denote the circuit integral of weighted magnetic permeability centered on node $[i, j, k]$ as $MuA_{z,i,j,k}$. Averaging over connected elements gives the approximate expression:

$$MuA_{z,i,j,k} = \frac{1}{4} \times \quad (26)$$

$$[(\mu_{i,j,k} + \mu_{i,j,k+1})\Delta x_i \Delta y_j + (\mu_{i+1,j,k} + \mu_{i+1,j,k+1})\Delta x_{i+1} \Delta y_j +$$

$$(\mu_{i,j+1,k} + \mu_{i,j+1,k+1})\Delta x_i \Delta y_{j+1} + (\mu_{i+1,j+1,k} + \mu_{i+1,j+1,k+1})\Delta x_{i+1} \Delta y_{j+1}].$$

The electric-field integral around boundary of the surface is given by:

$$\text{Int}E_{z,i,j,k} = \tag{27}$$

$$\begin{aligned} & [E_{x,i,j,k} + E_{x,i,j,k+1} - E_{x,i,j+1,k} - E_{x,i,j+1,k+1}] \frac{\Delta x_i}{4} + \\ & [E_{x,i+1,j,k} + E_{x,i+1,j,k+1} - E_{x,i+1,j+1,k} - E_{x,i+1,j+1,k+1}] \frac{\Delta x_{i+1}}{4} + \\ & [E_{y,i+1,j,k} + E_{y,i+1,j,k+1} - E_{y,i,j,k} - E_{y,i,j,k+1}] \frac{\Delta y_j}{4} + \\ & [E_{y,i+1,j+1,k} + E_{y,i+1,j+1,k+1} - E_{y,i,j+1,k} - E_{y,i,j+1,k+1}] \frac{\Delta y_{j+1}}{4}. \end{aligned} \tag{28}$$

Combining terms from Eqs. 26 and 27, we can write the expression to advance node values of H_z as:

$$H_{z,i,j,k}^{n+3/2} = H_{z,i,j,k}^{n+1/2} - \frac{\Delta t}{\text{Mu}A_{z,i,j,k}} \text{Int}E_{z,i,j,k}. \tag{29}$$

2.5 Courant stability condition

The time-domain solution process for internal elements is straightforward. The first operation in a cycle is to advance electric field components over a short time step in all elements (Eq. 24), replacing variables in situ because they do not depend on neighboring values of \mathbf{E} . The second operation is to advance components of \mathbf{H} (Eq. 29). The time step Δt must be shorter than the minimum propagation time for radiation across an element to ensure numerical stability. If this condition were violated, signals could propagate faster than the speed of light. The speed of electromagnetic pulses in element $[i,j,k]$ is

$$v_{ijk} = \frac{c}{\sqrt{\epsilon_{ijk}\mu_{ijk}}}. \tag{30}$$

The Courant condition for stability is written as

$$\Delta t \leq \min \left(\frac{\Delta x_i}{v_{ijk}}, \frac{\Delta y_j}{v_{ijk}}, \frac{\Delta z_k}{v_{ijk}} \right), \tag{31}$$

where Δx_i , Δy_j and Δz_k are the lengths of element $[i,j,k]$ along the three coordinate directions. If you do not specify a value for the time step, **Aether** searches the mesh and finds the largest possible value of Δt consistent with Eq. 31 and adjusts it by a safety factor with a default value of 0.8. The time step is reported in the listing file.

To minimize run times, you should avoid large differences in element resolution when constructing meshes. Remember that the time step depends on the shortest element dimension. Also, try to maintain a uniform light speed over all regions. As an example, consider representing a metal body in a solution where pulses propagate through a dielectric medium with $\epsilon_r = 2.71$. The propagation velocity in the dielectric is $v = 0.607c$. One option is to assign the following properties to the conducting medium: $\epsilon_r = 2.71 \times 10^6$, $\mu_r = 10^{-6}$ and $\sigma = 0.0$. The choice gives a characteristic impedance that is a factor of 10^{-6} smaller than the value in the dielectric but maintains the electromagnetic propagation velocity. Large differences in dielectric constant or magnetic permeability can lengthen run times. For example, suppose a water-filled

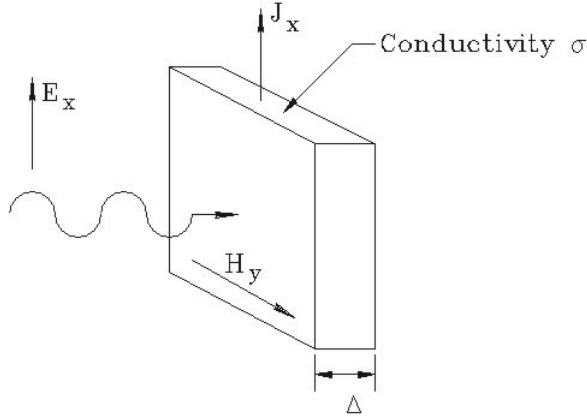


Figure 6: Traveling wave incident on a absorbing layer.

transmission line drives a load in a small vacuum region. The stability condition in the vacuum elements leads to a relatively short time step. In consequence, a large number of cycles may be required to track pulses through the water region.

2.6 Absorbing layers

A perfect absorbing boundary is useful in many electromagnetic field solutions. One application is termination of transmission lines or waveguides. A boundary with zero reflectivity may also be used to represent the free-space condition where all radiation travels outward. In **Aether** we employ a thin conducting layer to approximate an ideal absorber. Figure 6 shows the geometry of a layer on the boundary of the solution volume. A plane wave arrives at the layer at normal incidence. The absorbing layer has conductivity σ and thickness Δ . It has the same values of dielectric permittivity (ϵ) and magnetic permeability (μ) as the adjacent material on the left-hand side. In the limit that $\Delta \ll \lambda$, the wave electric field E_x is approximately uniform over the layer depth. The electric field creates a linear current density

$$J_x = (\sigma\Delta) E_x, \quad \text{A/m} \quad (32)$$

in the layer. The quantity $1/\sigma\Delta$ (with units of ohms) is called the *surface resistance* of the termination layer. The magnetic field on the right hand side of the layer is zero (the natural boundary condition in Aether). The field on the right-hand side is therefore $H_y = J_x$. The electric and magnetic fields of the plane wave are related by:

$$\frac{E_x}{H_y} = \sqrt{\frac{\mu}{\epsilon}} = \eta, \quad (33)$$

where η is the impedance of the adjacent medium. For vacuum, it has the value $\eta_0 = 376.7 \Omega$. Combining the relationships gives the following prescription for the conductivity of a matched layer:

$$\sigma = \frac{1}{\Delta\sqrt{\mu/\epsilon}} = \frac{1}{\Delta Z_0}. \quad (34)$$

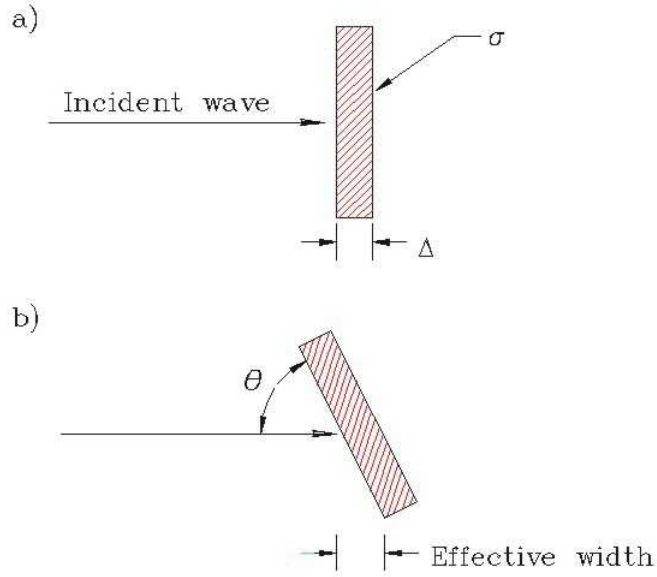


Figure 7: Absorbing boundary for two-dimensional solutions. *a)* Normal incidence, $\theta = 90^\circ$. *b)* Increase in effective layer thickness for $\theta < 90^\circ$.

Equation 34 holds for a wave normally incident on a thin layer (Fig. 7*a*). Here, the propagation vector of the wave makes an angle $\theta = 90^\circ$ with respect to the layer surface. For optimum performance, we must modify the conductivity if the wave arrives at an angle $\theta < 90^\circ$. In this case, the effective layer thickness is $\Delta/\sin\theta$, reducing the surface resistance by a factor of $\sin\theta$. In this case the termination is under-matched. The degradation in performance is not severe. A matched layer absorbs more than 90 per cent of the incident wave energy over the range $30^\circ \leq \theta \leq 90^\circ$.

The absorber performance can be optimized for waves that are not normal to the surface (e.g., an absorbing wedge). Figure 7*b* shows that if a wave is incident at angle θ , then the effective thickness of the layer increases. We can compensate the effect by setting the layer conductivity equal to

$$\sigma = \frac{\sin\theta}{\Delta Z_0}. \quad (35)$$

3 Time variations

Aether supports several options to specify time variations of drive currents or material conductivity. The choice depends on the type of calculation. Section 3.1 discusses the most flexible method, representation of a modulation function by a table of values. The advantage is that you can model any waveform and even incorporate experimental data. Section 3.2 covers the use of mathematical functions to define time variations. This approach is easier to implement than tables and is better suited for representing extended periodic variations. **Aether** includes a flexible algebraic parser that handles functions with multiple levels of parentheses. **Aether** also includes a set of parametric models for common pulse shapes. Section 3.3 reviews standard pulses in the time domain. The frequency spectrum is important for resonance and frequency-domain calculations. For example, for a resonant-mode search the drive pulse should resolve into span of frequency that brackets the modes of interest. Section 3.4 reviews the theory of Fourier analysis, while Sect. 3.5 summarizes the frequency content of standard pulses.

3.1 Tabular functions

Arbitrary time variations $f(t)$ may be defined in **Aether** through flexible *tabular functions*. A tabular function is a text file consisting of a set of data lines that contain two real-number values: t and $f(t)$. The file must be available in the working directory. You can prepare tabular function files with a text editor or spreadsheet. It is also easy to incorporate published data or digitized experimental traces. The following rules apply:

- A file may contain up to 256 data lines and must terminate with the *EndFile* command.
- Values of t and $f(t)$ may be written in any valid real-number format (*e.g.*, 5.016, 7.83E+03, 103, ...).
- A file may also contain blank lines and comment lines. A comment line begins with an asterisk ('*'). You may also include text annotations in any format after the *EndFile* command.
- Although it is best to order data lines in order of increasing time, strict order is not required. **Aether** sorts the list before use and records the final order in the listing file.
- The time interval between data entries need not be uniform. You can concentrate values near times when there are large changes in the modulation function.
- You can record absolute or relative values of t and $f(t)$. **Aether** has the option to renormalize values upon loading. With this feature, you can create a library of standard normalized functions.
- **Aether** supports either cubic spline or linear interpolation between values in the table. Cubic splines require a minimum of five lines of data. The spline method is usually more accurate, but may exhibit large oscillations between data points for noisy data. Always use linear interpolation if the data has discontinuities of value or slope.

- Time in an **Aether** run always starts at $t = 0.0$ s.
- By default, the interpolation routines return a value $f(t) = 0.0$ if t is outside the range of the table. You can set a switch that designates that a table should be interpreted as a periodic function. In this case, if the table has the range $0.0 \leq t \leq t_{max}$ and $t > t_{max}$, then the routine returns the value $f(t) = f[\text{mod}(t, t_{max})]$.

To illustrate, consider a pulse with the following variation in the range $0.0 \leq t \leq 2\Delta t$, where the width at half-height is $\Delta t = 0.01$ s:

$$f(t) = \frac{1 - \cos(\pi t / \Delta t)}{2}, \quad (36)$$

The following table defines a single instance of the pulse:

```

0.000  0.000
0.001  0.024
0.002  0.095          0.012  0.905
0.003  0.206          0.013  0.794
0.004  0.345          0.014  0.655
0.005  0.500          0.015  0.500
0.006  0.655          0.016  0.345
0.007  0.794          0.017  0.206
0.008  0.905          0.018  0.095
0.009  0.976          0.019  0.024
0.010  1.000          0.020  0.000
0.011  0.976          0.030  0.000
                        0.100  0.000
                        1.000  0.000
                        ENDFILE

```

The next example illustrates how the table would be modified to create a series of pulses separated by an interval of 0.05 seconds.

```

0.000  0.000
0.001  0.024
0.002  0.095          0.012  0.905
0.003  0.206          0.013  0.794
0.004  0.345          0.014  0.655
0.005  0.500          0.015  0.500
0.006  0.655          0.016  0.345
0.007  0.794          0.017  0.206
0.008  0.905          0.018  0.095
0.009  0.976          0.019  0.024
0.010  1.000          0.020  0.000
0.011  0.976          0.030  0.000
                        0.050  0.000
                        ENDFILE

```

3.2 Algebraic functions

Aether features a flexible and robust algebraic parser to interpret temporal functions. An algebraic function is a string (up to 230 characters) that may include the following entities:

- The time as a variable: **\$t**. The dollar sign is required so that the parser can differentiate the time from the letter *t* in functions like **tan**.
- Real and/or integer numbers in any valid format (*e.g.*, 3.1415, 476, 1.367E23, 6.25E-02, 8.92E+04,...). Integers are converted to real numbers for evaluation.
- Binary operations: + (addition), - (subtraction), * (multiplication), / (division) and ^ (exponentiation).
- Functions: **abs** (absolute value), **sin** (sine), **cos** (cosine), **tan** (tangent), **ln** (normal logarithm), **log** (base 10 logarithm), **exp** (normal exponent) and **sqrt** (square root).
- Up to 20 sets of parentheses to any depth.
- Any number of spaces as delimiters.

The parser conforms to the standard algebraic rules and includes a comprehensive error checker. Errors may include unbalanced parentheses, unrecognized characters and sequential binary operations. As an example, the expression

```
1 - exp(-1.0*($t^2)/24))
```

corresponds to

$$1 - \exp\left[-\left(\frac{t^2}{24}\right)\right]. \quad (37)$$

Operations to evaluate the expressions are performed with double-precision arithmetic.

3.3 Standard parametric pulses

A third option to define temporal modulations is to use a standard pulse model. The motivation for including some of the models will be apparent when we consider the frequency spectrum in the following sections. The examples show how the pulse specification would appear in the *SMOD* command (Sect. 5.4).

```
SMOD SourceNo GAUSSIAN Tp Tw  
SMOD(3) = GAUSSIAN (2.5E-9 0.5E-9)
```

Figure 8 illustrates the *Gaussian* pulse with mathematical variation

$$f(t) = \exp\left[-\frac{(t - T_p)^2}{(T_w/2)^2}\right]. \quad (38)$$

The parameters are the time at peak amplitude (T_p in s) and the pulse width (T_w in s).

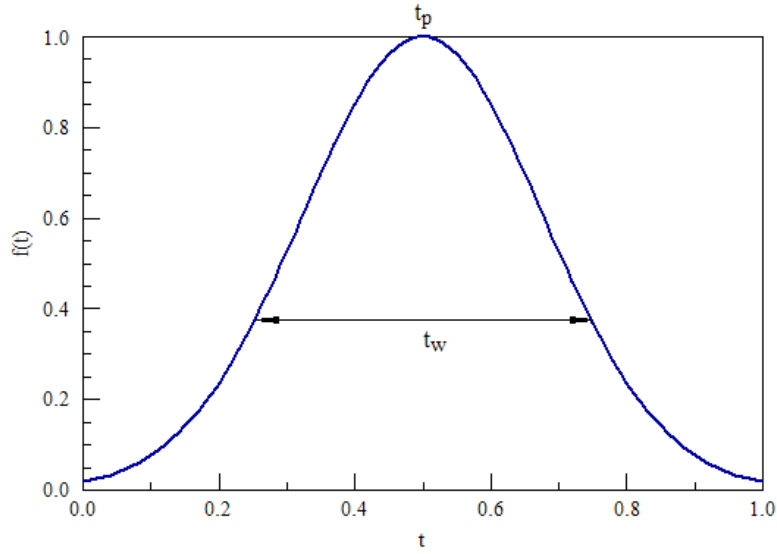


Figure 8: Parameters to define the *Gaussian* pulse.

SMOD SourceNo SIN Fo
SMOD(1) = SIN (3.7E9)

The function is a sine that starts at $t = 0$. Specify the frequency f_0 in Hz.

$$f(t) = \sin(2\pi f_0 t). \quad (t > 0.0) \quad (39)$$

SMOD SourceNo STEP Ts Tr
SMOD(6) = STEP (1.0E-9, 0.1E-9)

A smoothly-rising step function. The parameters are the start time (T_s in s) and the rise time (T_r in s). A value $T_r = 0.0$ gives a discontinuous step.

$$f(t) = 0.0, \quad (t \leq T_s) \quad (40)$$

$$f(t) = \frac{1 - \cos[\pi(t - T_s)]/T_r}{2}, \quad (T_s < t \leq T_s + T_r) \quad (41)$$

$$f(t) = 1.0. \quad (t > T_s + T_r) \quad (42)$$

SMOD SourceNo SQUARE Ts Te Tr
SMOD(6) = SQUARE (1.0E-9, 2.0E-9, 0.1E-9)

A square pulse with a smooth rise and fall (Fig. 9). The parameters are the start time (T_s in s), the end time (T_e in s) and the rise and fall times (T_r in s). A value $T_r = 0.0$ gives a discontinuous square pulse. **Aether** issues an error message if $T_r < (T_e - T_s)/2$.

$$f(t) = 0.0, \quad (t \leq T_s) \quad (43)$$

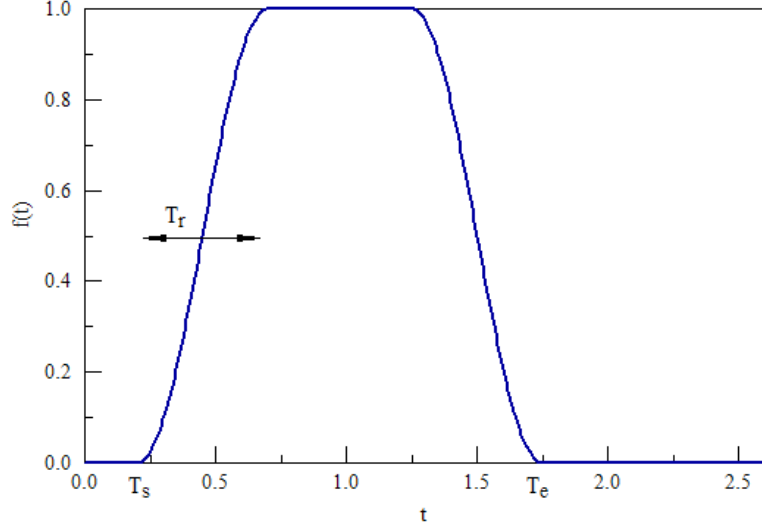


Figure 9: Parameters to define the smoothed *Square* pulse.

$$f(t) = \frac{1 - \cos[\pi(t - T_s)/T_r]}{2}, \quad (T_s < t \leq T_s + T_r) \quad (44)$$

$$f(t) = 1.0. \quad (T_s + T_r > t \leq T_e - T_r) \quad (45)$$

$$f(t) = \frac{1 - \cos[\pi(T_e - t)/T_r]}{2}, \quad (T_e - T_r < t \leq T_e) \quad (46)$$

$$f(t) = 0.0. \quad (t > T_e) \quad (47)$$

SMOD SourceNo SINC Tc Tp
SMOD(6) = SINC (5.0E-9, 3.5E9)

The sinc function (shown in Fig. 10) is of interest because the frequency spectrum is uniform over the range $-1/2T_p \leq f \leq +1/2T_p$. In the command, the parameter T_c is the time of maximum amplitude and T_p is the time interval to the first zero crossing of the function. The mathematical variation is

$$f(t) = \frac{\sin[\pi(t - T_c)/T_p]}{\pi(t - T_c)/T_p}. \quad (48)$$

SMOD SourceNo RAISEDCOS Tc Tp Beta
SMOD(6) = RAISEDCOS (1.0E-9, 2.0E-9 0.5)

The sinc function extends to infinity with an amplitude that decreases linearly with $(t - T_c)$. At best, it can only be approximated in a finite run time. It is usually better to sacrifice some frequency uniformity and to employ a pulse with better localization in time. The *raised cosine filter* is a sinc function with a truncation factor that decreases as $(t - T_c)^3$. In the command, the parameter T_c is the time of maximum amplitude, T_p is the time of the first zero crossing and β is the *roll-off factor* (a number between 0.0 and 1.0). The function has the mathematical variation:

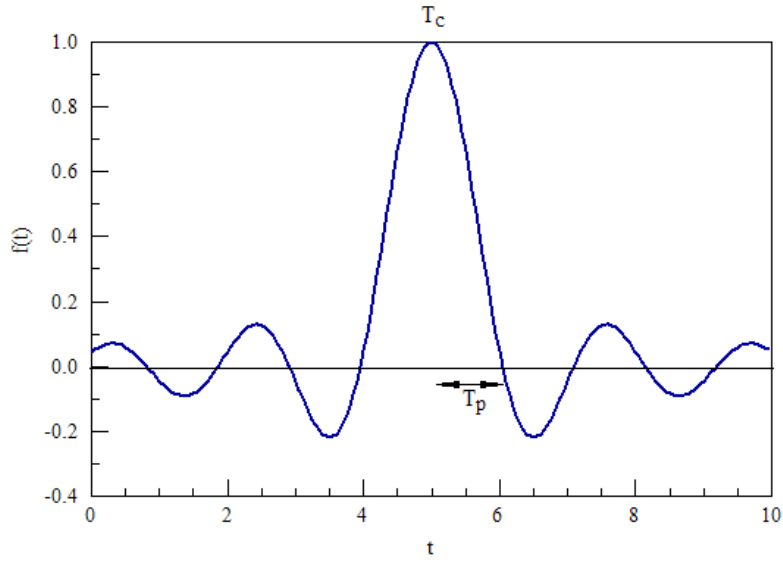


Figure 10: Parameters to define the *Sinc* pulse.

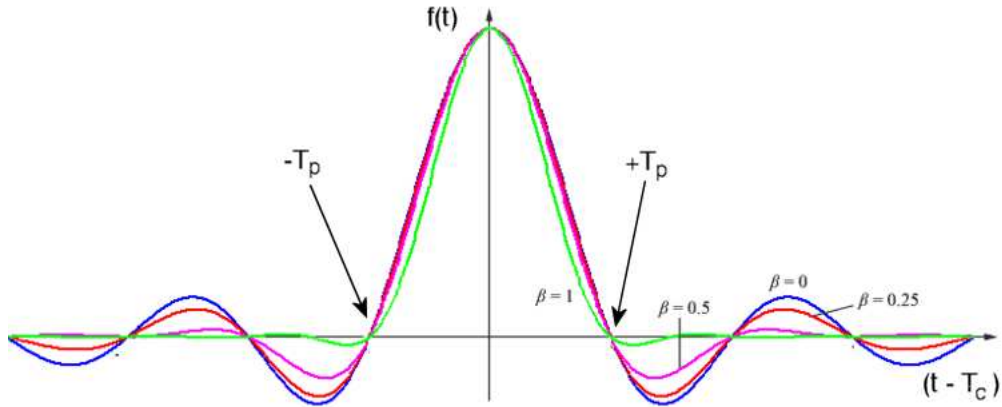


Figure 11: Parameters to define the *RaiseCos* pulse.

$$f(t) = \frac{\sin[\pi(t - T_c)/T_p]}{\pi(t - T_c)/T_p} \frac{\cos[\pi\beta(t - T_c)/T_p]}{1 - 4[\beta(t - T_c)/T_p]^2}, \quad (49)$$

Figure 11 shows the pulse for different choices of β .

The pulses we have discussed so far have finite-width frequency spectra at $f = 0.0$ Hz. When working with resonant modes, we want to center the spectrum at a given frequency f_0 . The next section shows that a harmonic modulation of the temporal function at f_0 shifts the frequency spectrum. Accordingly, the standard pulse set includes following modulated functions:

SMOD SourceNo GAUSSMOD Tp Tw F0
SMOD(3) = GAUSSMOD (2.5E-9 0.5E-9 2.5E9)

The parameters are the time of maximum amplitude (T_p in s), the pulse width (T_w in s) and the modulation frequency (f_0 in Hz). The mathematical variation is

$$f(t) = \exp \left[-\frac{(t - T_p)^2}{(T_w/2)^2} \right] \cos(2\pi f_0 t). \quad (50)$$

SMOD SourceNo SQUAREMOD Ts Te Tr F0

SMOD(6) = SQUAREMOD (1.0E-9, 2.0E-9, 0.1E-9, 5.0E9)

A square pulse modulated at frequency f_0 with a smooth rise and fall. The parameters are the start time (T_s in s), the end time (T_e in s), the rise and fall times (T_r in s) and the modulation frequency (f_0 in Hz).

$$f(t) = 0.0, \quad (t \leq T_s), \quad (51)$$

$$f(t) = \frac{1 - \cos[\pi(t - T_s)/T_r]}{2} \cos(2\pi f_0 t), \quad (T_s < t \leq T_s + T_r) \quad (52)$$

$$f(t) = \cos(2\pi f_0 t) \quad (T_s + T_r > t \leq T_e - T_r), \quad (53)$$

$$f(t) = \frac{1 + \cos[\pi(T_e - t)/T_r]}{2} \cos(2\pi f_0 t), \quad (T_s < t \leq T_s + T_r) \quad (54)$$

$$f(t) = 0.0. \quad (t > T_e). \quad (55)$$

SMOD SourceNo STEPMOD Ts Tr F0

SMOD(6) = STEPMOD (1.0E-9, 0.1E-9 1.23E10)

A smoothly-rising step function modulated at frequency f_0 (Hz). The parameters are the start time (T_s in s), the rise time (T_r in s) and the modulation frequency (f_0 in Hz).

SMOD SourceNo SINCMOD Tc Tp F0

SMOD(6) = SINCMOD (5.0E-9, 3.5E9)

The sinc function modulated by frequency f_0 . The parameters are the time of maximum amplitude (T_c in s), the time interval to the first zero crossing of the envelope function (T_p in s) and the modulation frequency (f_0 in Hz). The mathematical variation is

$$f(t) = \frac{\sin[\pi(t - T_c)/T_p]}{\pi(t - T_c)/T_p} \cos(2\pi f_0 t). \quad (56)$$

SMOD SourceNo RCOSMOD Tc Tp Beta F0

SMOD(6) = RCOSMOD (1.0E-9, 2.0E-9 0.5 6.4E10)

The parameters are the time of maximum amplitude (T_c in s), the time interval to the first zero crossing of the envelope function (T_p in s), the roll-off factor (β a number between 0.0 and 1.0) and the modulation frequency (f_0 in Hz). The function has the mathematical variation:

$$f(t) = \frac{\sin[\pi(t - T_c)/T_p]}{\pi(t - T_c)/T_p} \frac{\cos[\pi\beta(t - T_c)/T_p]}{1 - 4[\beta(t - T_c)/T_p]^2} \cos(2\pi f_0 t), \quad (57)$$

3.4 Fourier transforms

This section reviews basic equations of Fourier transforms in the notation used in signal processing. The results are applied in the following section to determine the frequency content of the standard pulses. The inverse Fourier transform is also employed for *Res* mode calculations. A temporal function $F(t)$ may be written as an integral over harmonic functions of frequency f :

$$F(t) = \int_0^{\infty} df [A(f) \sin(2\pi ft) + B(f) \cos(2\pi ft)]. \quad (58)$$

Here, time has units of seconds and frequency is in units of Hz (1/s). The coefficients of the frequency components are given by

$$A(f) = \int_{-\infty}^{\infty} dt F(t) \cos(2\pi ft), \quad (59)$$

$$B(f) = \int_{-\infty}^{\infty} dt F(t) \sin(2\pi ft). \quad (60)$$

For succinctness and symmetry, the transform equations are usually written in terms of complex exponential functions. They are related to the harmonic functions by:

$$\cos(2\pi ft) = \frac{\exp(2\pi ift) + \exp(-2\pi ift)}{2}, \quad (61)$$

$$\sin(2\pi ft) = \frac{\exp(2\pi ift) - \exp(-2\pi ift)}{2i}. \quad (62)$$

Substituting in Eq. 58, we find

$$F(t) = \int_0^{\infty} df [(a - ib) \exp(2\pi ift) + (a + ib) \exp(-2\pi ift)]. \quad (63)$$

The convention is to extend the integral over the range $-\infty < f < \infty$ and to introduce a complex Fourier amplitude $H(f)$ such that

$$H(f) = a(f) + iB(f), \quad (f < 0.0) \quad (64)$$

$$H(f) = a(f) - iB(f). \quad (f > 0.0) \quad (65)$$

This leads to the standard form of the Fourier transform to move between time and frequency representations of a function:

$$F(t) = \int_{-\infty}^{\infty} df H(f) \exp(-2\pi ift), \quad (66)$$

$$H(f) = \int_{-\infty}^{\infty} dt F(t) \exp(2\pi ift). \quad (67)$$

We can view $F(t)$ and $H(f)$ as two ways to display information on the same function. The notation $F(t) \Leftrightarrow H(f)$ designates that the forms are related by the Fourier transform.

Temporal functions in **Aether** are always real numbers. In this case, Eqs. 64 and 65 imply that

$$H(f) = H(-f)^*, \quad (68)$$

where the asterisk denotes complex conjugation. In this case the negative-frequency information is redundant. Nonetheless, we should be aware of its presence because most packaged FFT (Fast Fourier Transform) routines return both negative and positive frequency values. The quantity of interest in a numerical calculations is the function *power* contained within a frequency interval Δf at f . The convention is to consider the frequency as varying from 0 to ∞ and to define the *one-sided power spectral density* of the function $F(t)$ as

$$P_{sd}(f) = |H(f)|^2 + |H(-f)|^2 = 2|H(f)|^2 \quad (0 \leq f \leq \infty). \quad (69)$$

The latter form applies when $F(t)$ is a real function. The fractional power in a frequency interval given by

$$\frac{P_{sd}(f)\Delta df}{\int_0^\infty df P_{sd}(f)}. \quad (70)$$

To conclude, we shall list some useful properties of the Fourier transform. Assume that we know the transform $H(f)$ of a function $F(t)$. Consider preserving the functional form of $F(t)$ but scaling all time values by a factor a . The revised transform can be written as

$$F(at) \Leftrightarrow \frac{1}{|a|} H\left(\frac{f}{a}\right). \quad (71)$$

Equation 71 implies that short time pulses result in a broad frequency spectrum. Similarly, we can determine the time function that corresponds to a transformed function where all frequency values are scaled by factor b :

$$\frac{1}{|b|} F\left(\frac{t}{b}\right) \Leftrightarrow H(bf). \quad (72)$$

A small frequency range requires a long pulse duration. Next, suppose the $F(t)$ is shifted in time. The modified transform is

$$F(t - t_0) \Leftrightarrow H(f) \exp 2\pi i f t_0. \quad (73)$$

The implication is that a time shift introduces phase differences in the complex Fourier coefficients but does not affect the power spectral density.

Introduction of a shift in frequency is an important task for numerical calculations. Suppose we have a time function that has a power special density with the desired frequency width centered at $f = 0.0$. If we want to shift the range so that the power is centered near a resonance frequency $f = f_0$, we can use the modified time function,

$$F(t) \exp(-2\pi i f_0 t) \Leftrightarrow H(f - f_0). \quad (74)$$

Time-domain numerical simulations are limited to real numbers, so we must use a modulated function of the form $F(t) \cos(2\pi f_0 t)$. Application of Eq. 67 leads to the following relationship:

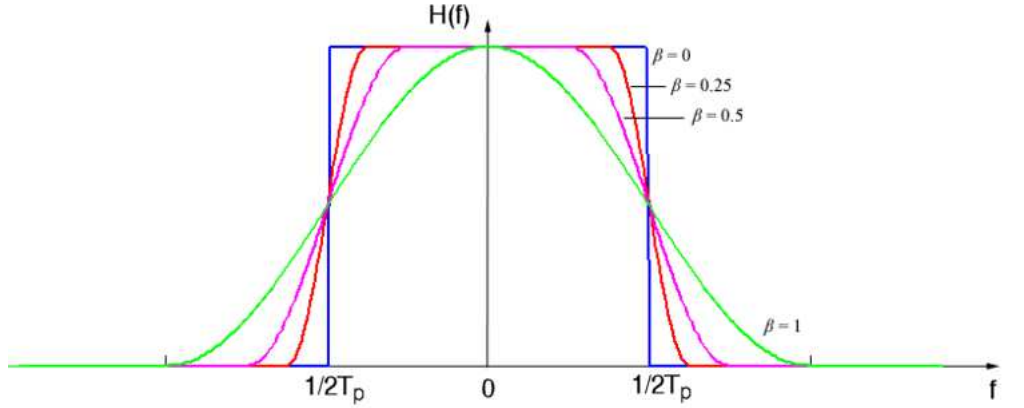


Figure 12: Frequency spectrum of the *RaiseCos* pulse.

$$F(t) \cos(2\pi i f_0 t) \Leftrightarrow \frac{1}{2} [H(f + f_0) + H(f - f_0)]. \quad (75)$$

The transformation introduces a negative frequency component in the complex Fourier spectrum, but does not change the power spectrum density.

3.5 Pulses in the frequency domain

We can apply Eq. 67 to find the frequency content of some members of the standard pulse set. To begin, consider the *Sin* function. If it extends over an infinite duration, the Fourier transform is a delta function at frequency f_0 . In reality, we usually start the function at $t = 0.0$ and run it for the length of the calculation, T_{max} . In this case, the harmonic function acts like a modulated square pulse with duration T_{max} . The amplitude of the Fourier spectrum is given by

$$|H(f)| = T_{max} \frac{\sin[\pi(f - f_0)T_{max}]}{\pi(f - f_0)T_{max}} = T_{max} \text{sinc}[(f - f_0)T_{max}]. \quad (76)$$

The approximate drive frequency range is $f_0 - 1/2T_{max} < f < f_0 + 1/2T_{max}$. For a pulse duration of 10 cycles, the relative frequency spread is $\leq \pm 5\%$. When T_{max} is much larger than the RF period, the sinc function approaches the delta function $\delta(f_0)$.

The frequency spectrum of the modulated Gaussian pulse is

$$|H(f)| = \frac{\sqrt{\pi}T_w}{2} \exp \left[- \left(\frac{\pi T_w (f - f_0)}{2} \right)^2 \right]. \quad (77)$$

The frequency spread is about $\Delta f \cong 2/\pi T_w$. Therefore, we can use a modulated Gaussian pulse where the envelope duration is much larger than $1/f_0$ to excite a limited range of frequencies.

The most useful drive pulse for resonance searches is the modulated raised cosine filter. The frequency spectrum is

$$|H(f)| = T_p, \quad |\Delta f| \leq \frac{1 - \beta}{2T_p} \quad (78)$$

$$|H(f)| = \frac{T_p}{2} \left[1 + \cos \left(\frac{\pi T_p}{\beta} \left[|\Delta f| - \frac{1 - \beta}{2T_p} \right] \right) \right], \quad \frac{1 - \beta}{2T_p} < |\Delta f| \leq \frac{1 + \beta}{2T_p} \quad (79)$$

$$|H(f)| = 0, \quad |\Delta f| \geq \frac{1 + \beta}{2T_p} \quad (80)$$

where $\Delta f = f - f_0$. When $\beta = 0.0$, the time pulse is the unmodified sinc function and the spectrum is uniform over the range $|f - f_0| \leq 1/2T_p$. Figure 12 shows $|H(f)|$ for different values of β . Higher values of β give better localization in time at the expense of increase frequency spread and a non-uniform distribution. For a resonance search with frequency width Δf_w near f_0 , **Aether** uses the *RCosMod* function with modulation frequency f_0 , roll off factor $\beta = 0.5$ and $T_p = 1/\Delta f_w$.

4 Frequency-domain techniques

4.1 Complex-number values in the RF mode

In the RF mode, **Aether** converts the steady-state time domain solution to phasor representation (amplitude and phase) using the procedure described in the following section. Recorded values in the output file `RunName.AOU` are complex-number representations of element field quantities, with **H** values centered in space and time (Chap. 15). In **Aether**, a time-domain harmonic field component at frequency f_0 with amplitude A and phase ϕ has time variation,

$$f(t) = A \cos(2\pi f_0 t + \phi). \quad (81)$$

Therefore, a quantity with $\phi = 0.0$ varies as $\cos(2\pi f_0 t)$ while $\phi = -90^\circ$ implies a variation $\sin(2\pi f_0 t)$. The complex-number form of Eq. 81 is:

$$f(t) = \text{Re}[(A \exp j\phi) \exp(2\pi j f_0 t)]. \quad (82)$$

The temporal factor $\exp(2\pi j f_0 t)$ is common to all element quantities. The first factor on the right-hand side of Eq. 82 (called the *phasor*) contains the unique amplitude and phase information:

$$\mathbf{F} = A \exp j\phi = A(\cos \phi + j \sin \phi). \quad (83)$$

Aether stores the real and imaginary parts of the phasors ($F_r = A \cos \phi$ and $F_i = A \sin \phi$) for all element quantities of the solution volume in the output file. Values of amplitude and phase are determined in **Aerial** from the inverse equations:

$$A = \sqrt{F_r^2 + F_i^2}, \quad (84)$$

$$\phi = \tan^{-1}(F_i/F_r). \quad (85)$$

4.2 Converting harmonic functions to phasors

Quantities in the equilibrium state of an *RF* mode solution vary harmonically with frequency f_0 and period $\tau = 1/f_0$. The amplitude and phase of such functions can be determined by comparing values at two times separated by one quarter RF period. Consider a harmonic function measured at times t_m and $t_m + \tau/4$:

$$f_1 = A \cos[2\pi f_0 t_m + \phi], \quad (86)$$

$$f_2 = A \cos[2\pi f_0 (t_m + \tau/4) + \phi]. \quad (87)$$

Using the trigonometric identity $\cos(x + 90^\circ) = -\sin(x)$, we can rewrite Eq. 87 as

$$f_2 = -A \sin[2\pi f_0 t_m + \phi]. \quad (88)$$

The phase of the oscillation is given by

$$\phi = -\tan^{-1}\left(\frac{f_2}{f_1}\right) - 2\pi f_0 t_m. \quad (89)$$

Knowing the phase, the amplitude can be determined from one of the two equations,

$$A = \frac{f_1}{\cos(2\pi f_0 t_m + \phi)}, \quad (90)$$

$$A = -\frac{f_2}{\sin(2\pi f_0 t_m + \phi)}. \quad (91)$$

The best accuracy is achieved by using the equation with the largest value of the denominator.

In an *RF* mode solution, **Aether** adjusts the time step so that there are an integral number in one-quarter period, $\Delta t = \tau/4N_q$. The program runs a time-domain solution to a steady state and then makes a temporary record of field quantities for all elements. Magnetic-field values are centered in time and computed at the element centers. After advancing N_q steps, **Aether** compares the present values of quantities to the recorded ones to determine phasors to record in the output file.

4.3 Electromagnetic energy and power expressions

Aether and **Aerial** calculate several volume and surface integrals that are useful for applications (*e.g.*, calculation of capacitance and inductance, estimation of resonator Q factors,...). This section documents formulas for the quantities for both the *Pulse* and *RF* modes.

In the *Pulse* mode (where all recorded quantities are real numbers), integrals represent instantaneous values at the time the data file was recorded. Volume integrals over the electric and magnetic field energy densities yield the total field energy:

$$u_e = \frac{\epsilon_r \epsilon_0}{2} (E_x^2 + E_y^2 + E_z^2), \quad (\text{J/m}^3) \quad (92)$$

$$u_m = \frac{\mu_r \mu_0}{2} (H_x^2 + H_y^2 + H_z^2). \quad (\text{J/m}^3). \quad (93)$$

A numerical volume integral is a sum over elements of the product of the average energy density times the element volume. The expressions of Eqs. 92 and 93 are evaluated at the element centers.

The ohmic power density from real current flow is given by:

$$p = \sigma (J_x E_x + J_y E_y + J_z E_z). \quad (\text{W/m}^3) \quad (94)$$

By convention, the current density recorded in the data files and used in the power-density calculation is the sum of drive and resistive currents:

$$\mathbf{J} = \mathbf{J}_0 + \sigma \mathbf{E}. \quad (95)$$

The Poynting vector gives the flux of electromagnetic power:

$$\mathbf{S} = \mathbf{E} \times \mathbf{H}. \quad (\text{W/m}^2) \quad (96)$$

Aerial calculates surface integrals of \mathbf{S} over the boundaries of regions. The procedure for region N is to loop through all elements of the solution volume, checking those with the target region number. For the z component of power flux to and from element $[i, j, k]$, the program calculates the quantity

$$S_{z,i,j,k} = E_{x,i,j,k}H_{y,i,j,k} - H_{x,i,j,k}E_{y,i,j,k}, \quad (97)$$

Here, $S_{z,i,j,k}$ is the Poynting vector component at the center of the element. The program checks the upper element with index $[i, j, k + 1]$. If the element has a different region ($N_u \neq N$), **Aether** supplements the power leaving region N for region N_u by

$$\Delta P = \Delta P + S_{z,i,j,k}\Delta x_i\Delta y_j. \quad (98)$$

If the lower element $[i, j, k - 1]$ has a different region number, then the power leaving the target element is

$$\Delta P = \Delta P - S_{z,i,j,k}\Delta x_i\Delta y_j. \quad (99)$$

A similar calculation is applied to element facets normal to the x and y directions. The listed magnitude of power from region N to N' may not be exactly equal to the power from N' to N because of the offset in positions for evaluating the Poynting vector. Note that the results are not valid for regions that have single-element thickness (such as an absorbing layer).

RF mode calculations have complex-number results which are stored as phasors in the output file. The energy and power densities at a point are interpreted as time-averaged values, where the average is taken over an RF period. Volume integral quantities may be calculated from phasors and their complex conjugates:

$$\bar{u}_e = \frac{\epsilon_r \epsilon_0}{4} (E_x E_x^* + E_y E_y^* + E_z E_z^*), \quad (\text{J/m}^3) \quad (100)$$

$$u_m = \frac{\mu_r \mu_0}{4} (H_x H_x^* + H_y H_y^* + H_z H_z^*), \quad (\text{J/m}^3) \quad (101)$$

$$\bar{p} = \frac{\sigma}{2} (E_x E_x^* + E_y E_y^* + E_z E_z^*). \quad (\text{W/m}^3) \quad (102)$$

$$(103)$$

Finally, the time-averaged Poynting vector may be written in terms of phasors as

$$\mathbf{S} = \frac{1}{2} \text{Re} (\mathbf{E} \times \mathbf{H}^*). \quad (\text{W/m}^2) \quad (104)$$

5 Pulse mode commands

This chapter reviews how to set up *Pulse* mode runs for time-domain solutions. The ultimate aim is to create an input script, a text file with a name of the form `RunName.AIN` (**A**ether **I**Nput). The easiest way to generate a script is to click on the *Setup* command in **Aether** and use the interactive dialog. Section 5.2 describes this option. The remainder of the chapter gives detailed descriptions of the script structure and commands of the *Pulse* mode. There are two reasons why it may be useful to enter information directly into a script with a text editor:

- You can easily make parameter changes in an existing script.
- There are several advanced features that are not available in the *Setup* dialog.

5.1 Using MetaMesh with Aether

A solution in any **Aether** mode requires a computational mesh prepared with **MetaMesh**. The **MetaMesh** manual gives a complete description of all capabilities of the program. The prepared input files discussed in the **Aether** tutorial manual can serve as a quick introduction. **MetaMesh** can create either conformal or non-conformal meshes. The solution programs **HiPhi**, **Magnum**, **HeatWave** and **RFE3** can use meshes in either form. It is important to note that **Aether** accepts only non-conformal meshes. Here are some considerations for using **MetaMesh** with **Aether**:

- Do not include *Surface* commands in *Part* structures.
- Always include the command `Smooth 0` in the *Global* section.
- The *Coat* command (which affects the identity of nodes) is generally not required because **Aether** is an element-based code.
- You can quickly convert a conformal mesh to non-conformal form by enclosing the *Part* sections within the script directives `#NOFIT` and `#ENDNOFIT`.

5.2 Pulse mode setup dialog

Click on the *Setup* command to use the run setup dialog. **Aether** displays a load-file window listing available mesh files (with suffix `MDF`). You can navigate in the dialog to change the working directory. Pick a file and click *OK*. The program next displays the dialog of Fig. 13 for picking the computational mode.

For *Pulse* mode calculations, **Aether** displays the dialog of Fig. 14. The dialog is divided into sections that reflect the parts of the script discussed in the following sections. The *Control* section has the following entries:

DUnit

Set the length units used for the mesh. Common choices are listed in the menu. Pick *Custom* if

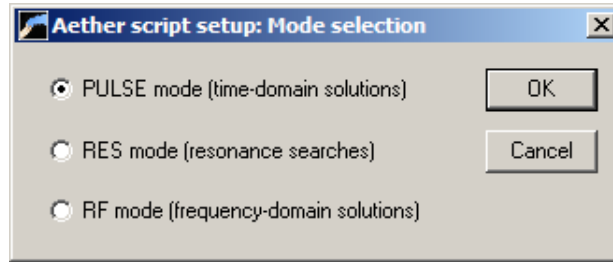


Figure 13: Initial setup dialog to pick the solution mode

your dimensions do not appear in the list. In this case, **Aether** adds a blank *DUnit* command in the script. Use a text editor to enter the conversion factor.

TMax

The maximum simulation time in seconds.

Dt

The time step in seconds. If no value is supplied, **Aether** will calculate one that satisfies the Courant condition (Sect. 2.5).

In the *Source* section, enter the spatial and temporal variations of current density to drive electromagnetic pulses.

Element file prefix

One way to specify the spatial variation of \mathbf{j} is to create a file of current elements using **Magwinder** (Chap. 14). Enter the prefix of a winding file available in the working directory. The full file name is `FPrefix.WND`. The elements are associated with Source Number 1. You can assign spatial variation of current density for up to four additional sources using a text editor and the advanced script commands described in Sect. 5.5.

If no entry appears for an element file, **Aether** writes a set of comment lines in the script to show the syntax of the J_x , J_y , J_z and J_r commands. In this case, you must edit the script, entering commands that set current density in chosen regions. The next three commands define the time variation of current for Source 1. Pick one of the options.

Modulation table

Specify the time variation of current density for Source 1 by interpolation on a table of values. Enter the full name of the file that contains the table. The file must be available in the working directory. Section 3.1 describes the format of the table file. You can specify time-variations for up to four additional sources using a text editor and the advanced script commands described in Sect. 5.5.

Modulation function

As an alternative to a table, you can set the time variation for Source 1 by a mathematical function. Section 3.2 describes the syntax of algebraic expressions for functions.

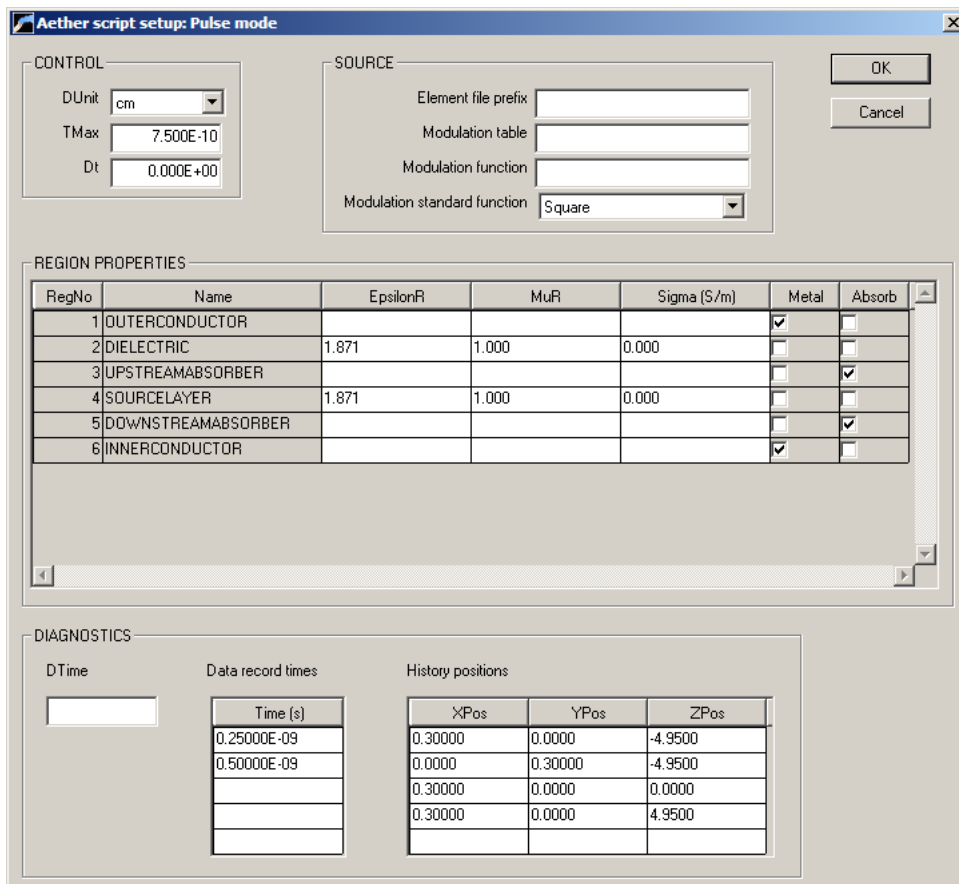


Figure 14: Dialog to create a *Pulse* mode script.

Modulation standard function

Pick one of the standard functions of Sect. 3.3 to define the time variation of Source Number 1. **Aether** writes the associated *SMod* command with the function parameters in symbolic form. Use a text editor to enter numerical values.

Set material properties in the *Region properties* grid. The grid has one row for each region of the solution volume. Region names are displayed if you assigned them in **MetaMesh**. To enter standard properties for a region, supply values of the relative dielectric constant (ϵ_r), the relative magnetic permeability (μ_r) and the conductivity (σ) in S/m. Alternatively, check one of the boxes for special material properties (metal or absorbing layer). Uncheck the boxes if you change your mind and want to enter values manually. For an absorbing layer, **Aether** writes the symbolic command:

```
ABSLAYER(RegNo) = DeltaZ
```

Supply a numerical value for Δz with a text editor (see Sect. 2.6).

The final group of commands controls diagnostics. There are two output types:

- *Space files* are complete spatial records made at a specified time. These files can be loaded into **Aerial** (Chaps. 8 - 12) for analysis and plotting.
- *Time files* are temporal records made at specified positions. Use the **Probe** program (Chap. 13) to generate plots of the recorded quantities.

DTime

Write space files at uniform time intervals. Enter *DTime* in seconds.

Data record times

Write space files at specified times. Enter up to five values in seconds. The entries must appear in chronological order. You can use a text editor to enter up to 95 additional record times in the script.

History positions

Enter the locations of record points for create time files. **Aether** generates individual files of the variation of electromagnetic quantities for each location. Enter up to five positions. You can use a text editor to add up to ten additional record locations.

Click *OK* when you have completed entries in the dialog. Save the file with the default or with a different prefix.

5.3 Syntax and structure of the Aether control script

The **Aether** control script is a text file with data lines containing commands and parameters. The script must end with the *EndFile* command. The program makes no distinction between upper and lower case. Entries on a line may be separated by the following delimiters:

Space, blank
Comma [,]
Tab
Colon [:]
Equal sign [=]
Left parenthesis [(]
Right parenthesis [)]

You may use any number of delimiters in a line. This means that you can add indentations and customize the appearance of the script. For example, the two lines

```
Epsi 2 5.56  
Epsi(2) = 5.56
```

are equivalent.

Aether ignores blank lines and comment lines. Comment lines begin with the symbol [*] (asterisk). Most parameters are real numbers. The following formats are valid:

```
1.000  
5.67E6  
6.8845E+09  
5
```

The final number is interpreted as 5.0.

The commands of the *Pulse* mode divide into the following groups, reflecting in the structure of the setup dialog of the previous section:

- **Run control.** Set parameters that control the calculation, such as the time limit.
- **Current sources.** Set the spatial and temporal variations of currents that drive electromagnetic pulses.
- **Region properties.** Set the material characteristics of the regions that constitute the solution volume.
- **Diagnostics.** Control times for space files and positions for time files.

Aether accepts commands in any order – the program reads and analyzes all commands before initiating the solution. Nonetheless, we recommend grouping commands by the listed categories for clarity. Here is an example of an **Aether** script for a *Pulse* mode solution:

```

* ---- CONTROL ----
Mode = Pulse
Mesh = MagDipole3D
DUnit = 100.0
TMax = 2.51E-09
* ---- CURRENT SOURCES ----
SFile(1) = MagDipole3D
SMod(1) > sin(1.885E10*$t)
* ---- REGION PROPERTIES ----
AbsLayer(1) 0.2
Vacuum(2)
* ---- DIAGNOSTICS ----
SetTime = 2.50E-10
SetTime = 2.50E-9
History = 0.000 0.000 0.000

```

EndFile

Note that you can place any amount of text after the *EndFile* command. With this feature, you can add annotations that may be helpful when you return to a simulation.

5.4 Run control commands

This section, reviews commands that control general program operation.

MODE [Pulse, RF, Res]

MODE = Pulse

Set the calculation mode. The command must appear in the **Aether** script. The commands discussed in this chapter are valid when the choice is *Pulse*.

MESH MPrefix

MESH = PulseLine302

The parameter is the prefix of the **MetaMesh** output file (MPrefix.MDF) that defines the solution geometry. If the command does not appear in a script FPrefix.AIN, **Aether** searches for the default file FPrefix.MDF.

DUNIT Unit

DUNIT = CM

DUNIT = 100.0

Use this command to set the length units of the **MetaMesh** input file. The *Unit* argument may be one of the following strings for common units: *angstrom*, *nanometer*, *micrometer*, *mil*, *mm*, *cm*, *inch*, *foot*, *yard*, *meter*, *kilometer* or *mile*. For custom units, enter a real number equal to the number of mesh units per meter. For example, if you used dimensions of centimeters in the **MetaMesh** script, set $DUnit = 100.0$. The quantity $DUnit$ is recorded in the output file and is used in **Aerial** for the input and output of positions. Default: $DUnit = 1.0$.

TMAX TMax**TMAX = 4.5E-9**

This required command sets the termination time for the simulation, t_{max} . Enter the value in units of seconds. Note that calculations always start at $t = 0.0$ s.

DT Dt**DT = 2.5E-13**

By default, **Aether** automatically picks the maximum time step consistent with the Courant stability condition (Sect. 2.5). Use this command to set Δt manually. The option is useful if you want space files and time-file entries to occur at specific intervals. **Aether** terminates with an error message if Δt is larger than the Courant value.

RUNTIME TRun**RUNTIME = 180.0**

Set a maximum value for the run time of the solution. Enter the value in minutes. **Aether** stops when either the simulation time equals t_{max} or when the run time equals t_{run} . This command may be useful to ensure that solutions controlled by a batch file do not run out of hand. Default: $t_{run} = \infty$.

INTERP [SPLINE, LINEAR]**INTERP = Linear**

This command controls the type of interpolation performed when tables are used to define spatial or temporal variations. Cubic spline interpolations are more accurate for smooth data. Because they preserve continuous values of the first and second derivatives, cubic splines may give anomalous results for noisy or discontinuous data. In this case, use the *Linear* option. Default: *Spline*.

SYMBOUND BndMark**SYMBOUND = XUp**

Sometimes it is possible to apply symmetry conditions to reduce the size of the mesh and the duration of a run. Symmetry boundaries must be parallel to the direction of radiation propagation. The condition on the surface is that the gradient of \mathbf{H} normal to the boundary is zero:

$$\nabla_n \mathbf{H} = 0. \quad (105)$$

In other words, the magnetic field has the same value on each side of the boundary. Any of the six sides of the solution box may be defined as a symmetry boundary using the options *XUp* (boundary at x_{max}), *XDn* (boundary at x_{min}), *YUp* (boundary at y_{max}), *YDn* (boundary at y_{min}), *ZUp* (boundary at z_{max}) or *ZDn* (boundary at z_{min}). The script may contain multiple *SymBound* statements.

EDDYCURRENT [Xi] **EDDYCURRENT = 500.0**

Aether can calculate non-radiative magnetic-field diffusion into resistive materials in the presence of eddy currents. In the eddy-current limit, displacement currents are small compared to real currents. The approximation is satisfied as long as the propagation time for light across the system is small compared to the time-scale for changes of real current. It would be quite inefficient to resolve time scales on the order of the system size divided by the speed of light in vacuum. In response to this command, **Aether** multiplies the relative dielectric constants of all materials by a factor ξ^2 to slow the speed of light by $1/\xi$. If a value of ξ does not appear, **Aether** automatically takes the following steps before starting the calculation:

1. Adjusts values of ϵ_r so that the speed of light, $c/\sqrt{\epsilon_r\mu_r}$, has the same value in all media.
2. Determines a good value of ξ to ensure that the propagation time of radiation across the system is much smaller than t_{max} .
3. Multiplies values of ϵ_r by ξ^2 .

COURANT CSafety **COURANT = 0.75**

A run may become unstable if elements have widely-different dimensions in the x , y and z directions. In this case, use this command to reduce the safety factor for automatic calculation of the time step from the default value of 0.80. Alternately, you may be able to reduce the run time by raising the safety factor for a mesh with approximately cubic elements. The safety factor must be in the range between 0.0 and 1.0.

EINIT FPrefix **EINIT = TLineCharge**

The initial static electric field distribution in devices like charged transmission lines, Blumlein lines and capacitors may have complex three-dimensional variations. If you have the **HiPhi** code, you can set electric fields at $t = 0.0$ s directly from an electrostatic solution. There are two important considerations for importing **HiPhi** solutions into **Aether**:

- The two programs must use the same mesh. You should plan the division into regions so that the mesh serves both the electrostatic and electromagnetic solutions. The mesh must be non-conformal. The **MetaMesh** script should include the command *Smooth 0* and should not include *Surface* commands.
- For a valid physical solution, the initial state of the electromagnetic solution must be consistent with the electrostatic solution. Currents in the electromagnetic solution should be close to zero at $t = 0.0$.

5.5 Current source commands

Currents generate electromagnetic pulses in **Aether**. It is necessary to specify both the spatial and time variation of current density. You can assign a fixed value or spatial variations to the components of current density in regions of the solution volume. Regarding time variations, you can define up to ten independent modulation functions. Region current densities may be associated with one of the modulation functions, giving a complete definition of \mathbf{j} in space and time. The current density in a region is the product of the spatial variation and the modulation functions. Normally, the modulation functions are normalized and the region quantities determine the magnitude of the current density. The first set of commands defines spatial variations.

SFILE(SourceNo) SFilePrefix
SFILE(2) = Solenoid

Define a general current-density variation over regions the solution volume by loading a file of current segments created with **Magwinder** (Chap. 14). **Aether** analyzes the segments to determine values of drive current density (J_{0x}, J_{0y}, J_{0z}) in mesh elements. In this case, the currents may extend over multiple regions of the solution volume. The temporal variation of current density is given by modulation function number *SourceNo*.

JX(RegNo,SourceNo) = Jx
JX(5,2) = 5.0E8

Assign a uniform value j_x for the x component of current over all elements that have region number *RegNo*. The temporal variation of current density is given by modulation function number *SourceNo*.

JY(RegNo,SourceNo) = Jy
JZ(RegNo,SourceNo) = Jz

Assign uniform values for the y and z components of current density

JX(RegNo,SourceNo) > JxFunc
JX(2,1) > 2.3E7*cos(0.31416*\$y)

Assign a spatial variation from an algebraic function to the x component of current density over all elements that have region number *RegNo*. The temporal variation of current density is given by modulation function number *SourceNo*. The syntax of algebraic functions for temporal variations was discussed in Sect. 3.2. The difference for spatial functions is that the allowed variables are $\$x$, $\$y$, $\$z$ and $\$r$, where

$$r = \sqrt{x^2 + y^2}. \quad (106)$$

An expression may include either the variable set $[\$x, \$y, \$z]$ or $[\$r, \$z]$. Enter the spatial quantities in the length units set by *DUnit* (i.e., if *DUnit* = 100.0, the spatial variables return values in centimeters).

JY(RegNo,SourceNo) > JyFunc

JZ(RegNo,SourceNo) > JzFunc

Assign spatial variations for the y and z components of current density from algebraic functions.

The following commands are used to specify time variations. You can define up to ten independent modulation functions that may be associated with any region source. The quickest way to define a modulation function is to use one of the functions discussed in Sect. 3.3. You can create custom functions using the other options.

SMOD(SourceNo) FuncName [Parameters]

SMOD(3) = GAUSSIAN (2.5E-9 0.5E-9)

Assign one of the standard functions described in Sect. 3.3 to the modulation function. The quantity *SourceNo* is an integer from 1 to 10. The string *FuncName* is one of the options listed in Table 2, followed by real-value parameters. The number of parameters depends on the function type.

SMOD(SourceNo) > SFunction

SMOD(1) > 1.0 - exp(-5.6E8*\$t)

Define a custom modulation function as an algebraic function of time. The quantity *SourceNo* is an integer from 1 to 10. The symbol $>$ designates that the remainder of the line contains the function expression. Section 3.2 describes the function syntax. The variable t (which designates the simulation time) may appear multiple times in the function. Spatial variables (*e.g.*, x) are not allowed.

SMOD(SrcNo) STabName [TMult, VMult]

SMOD(1) = SwitchSequence.MOD

SMOD(SrcNo) STabName SINGLE [TMult, VMult]

SMOD(3) = Double.MOD Single 4.0E-5 0.25

Associate a modulation function with a table of values. The quantity *SrcNo* is an integer from 1 to 10, while *STabName* is the name of the file that contains the table. The file must be available in the working directory. It contains a set of up to 256 data lines $[t, f(t)]$ in the format described in Sect. 3.1. In the default *Single* mode, the function is interpreted as a one-time occurrence. If the simulation time exceeds T_{max} (the maximum time of entries in the table), **Aether** takes $f(t) = 0.0$. You can include the optional real-number parameters *TMult* and *VMult*, multiplication factors for t and $f(t)$ values as they are loaded in the program. Use *Spline* interpolation for continuous functions $f(t)$, and *Linear* interpolation for discontinuous or noisy functions.

SMOD(SourceNo) STabName PERIODIC [TMult, VMult]

SMOD(5) = PulseTrain.MOD PERIODIC

Use this variant of the *SMod* command to define a periodic function of time. The function repeats indefinitely. The returned value is $f[\text{mod}(t, T_{max})]$, where T_{max} is the maximum time for entries in the table.

Table 2: Standard functions and parameters

Function	Param01	Param02	Param03	Param04
Gaussian	t_c	t_w		
Sin	f_0			
Step	t_s	t_r		
Square	t_s	t_e	t_r	
Sinc	t_c	t_w		
RaisedCos	t_c	t_w	β	
GaussMod	t_c	t_w	f_0	
StepMod	t_s	t_r	f_0	
SquareMod	t_s	t_e	t_r	f_0
SincMod	t_c	t_w	f_0	
RCosMod	t_c	t_w	β	f_0

5.6 Commands to set material properties

Materials in **Aether** are characterized by three properties:

- The dielectric constant or relative permittivity, ϵ_r .
- The relative permeability, μ_r .
- The electrical conductivity, σ .

The first two quantities are dimensionless, while the conductivity has the SI units of siemens per meter (S/m). The conductivity is related to the volume resistivity by

$$\sigma = \frac{1}{\rho}, \quad (107)$$

where ρ has dimensions Ω -m. The regions of the solution volume created in **MetaMesh** represent different types of materials. Values of ϵ_r , μ_r and σ are therefore associated with regions. The quantities may have uniform values over the region or follow a prescribed spatial variation. The conductivity may also have a time variation, an essential feature for switch simulations.

EPSI(RegNo) Epsi

EPSI(5) = 2.78

Set a uniform value of the dielectric constant (relative permittivity, ϵ/ϵ_0) for elements of region *RegNo*.

EPSI(RegNo) > EpsiFunction**EPSI(2) > 8.0*exp(-(\$x + \$y)/2.5)**

Define a spatial variation of the dielectric constant over region *RegNo*. The string *EpsiFunction* is an algebraic expression (Sect. 3.2) that may contain the variable set ($\$x, \$y, \$z$) or ($\$r, \z). The variables return position values in length units set by *DUnit*. In other words, if the solution volume extends from $-0.005 \text{ m} \leq x \leq 0.005 \text{ m}$ and $DUnit = 1000$, then the variable $\$x$ has the range -5.0 to 5.0.

MU(RegNo) Mu**MU(4) = 500.0**

Set a uniform value of the relative magnetic permeability (μ/μ_0) for elements of region *RegNo*.

MU(RegNo) > MuFunction**MU(3) > sin(3.14159*\$z)**

Define a spatial variation of the relative magnetic permeability over region *RegNo*. The string *MuFunction* is an algebraic expression (Sect. 3.2) that may contain the variable set ($\$x, \$y, \$z$) or ($\$r, \z).

SIGMA(RegNo) Sigma**SIGMA(12) = 56.8**

Set a uniform value of the electrical conductivity σ for elements of region *RegNo*. Enter the number in units of s/m.

SIGMA(RegNo) > SigmaFunction**SIGMA(5) = 100.0*56.8**

Define a spatial variation of the electrical conductivity over region *RegNo*. The string *SigmaFunction* is an algebraic expression (Sect. 3.2) that may contain the variable set ($\$x, \$y, \$z$) or ($\$r, \z).

SIGMOD(RegNo) SigTabName [TMult SigMult]**SIGMOD(3) = cos_square.mod 40.0E-6 25.0**

Define a modulation function for the conductivity of region *RegNo* from a table. The uniform value of conductivity or the spatial variation is multiplied by the modulation function. To avoid confusion, we suggest including the magnitude of conductivity in the spatial part and using a modulation factor in the range 0.0 to 1.0. The string *SigTabName* is the name of a table (Sect. 3.1) in the working directory. The optional real-number parameters t_{mult} and σ_{mult} are multiplication factors for the time and conductivity values. If no *SigMod* command appears, the conductivity retains a fixed value over the duration of the calculation.

SIGMOD(RegNo) > SigFunction**SIGMOD(5) > 1.0 + exp(-\$t/2.0E-6)**

Define a modulation function for the conductivity of region *RegNo* from an algebraic expression of the variable $\$t$ (Sect. 3.2).

VACUUM(RegNo)

VACUUM(6)

Set $\epsilon_r = 1.0$, $\mu_r = 1.0$ and $\sigma = 0.0$ for elements of region *RegNo*.

METAL(RegNo) [EpsiR MuR]

METAL(5) (2.78, 1.00)

Assign a low value of impedance η so that region *RegNo* acts as a metal (*i.e.*, excluding electric and magnetic fields). The default material properties are $\epsilon_r = 10^6$, $\mu_r = 10^{-6}$ and $\sigma = 0.0$. In this case, the impedance is $\eta = 1.0 \mu\Omega$ and the electromagnetic propagation velocity equals the speed of light in vacuum. If the speed of light is less than c in all other regions of the solution volume, you can set optional values of ϵ_r and μ_r to slow electromagnetic pulses in the metal. Choose ϵ_r and μ_r equal to the smallest values in the other regions of the solution volume. This change minimizes run time but does not affect the resulting fields.

ABSLAYER(RegNo) DeltaZ [EpsiR MuR]

ABSLAYER(7) = 0.025

Set material properties in the elements of region *RegNo* to those of an ideal absorbing layer (Sect. 2.6). Generally, an absorbing region is a single layer of elements of thickness Δz . Enter the thickness in units set by *DUnit*. For example, if *DUnit* = 100.0, specify Δz in centimeters. If the layer is adjacent to a vacuum region, **Aether** sets $\epsilon_r = 1.0$, $\mu_r = 1.0$ and $\sigma = 1/(376.7 \times \Delta z)$. Include the optional values *EpsiR* and *MuR* if the layer is adjacent to a non-vacuum region.

5.7 Diagnostic commands

In the *Pulse* mode, **Aether** can record two types of data files:

- Space files are binary files that contain complete spatial information at specified times. These files may be analyzed with **Aerial**.
- Time files are text records of field quantities at specified positions. These files may be analyzed with **Probe**.

The following three commands control the times for recording space files.

NSTEP N_s

NSTEP = 250

Create space files at time intervals of $N_s \Delta t$, where Δt is the time step for the calculation. The default value is $N_s = \infty$. A script may contain only one *NStep* command.

DTIME Dt

DTIME = 5.0E-10

This command sets a uniform time interval for space files. Enter *Dt* in seconds. The program generates files at (or as soon as possible after) the calculated time. The interval may not be

exact unless it is an integer multiple of Δt . The default value is $Dt = \infty$. A script may contain only one *DTime* command.

SETTIME DTime(n)
SETTIME = 1.45E-9

This command sets one or more times for space files. Enter the time in seconds. **Aether** generates a file at (or as soon as possible after) the specified time. The exact time depends on the time step, Δt . A script may contain up to 100 *SetTime* commands. They must appear in order of increasing value of the time.

The *NStep*, *SetTime* and *DTime* commands work in conjunction. Space files have names of the form *RunName*.001, *RunName*.002, The suffix numbers are assigned consecutively as the files are created.

The following commands control the generation of time files.

HISTORY XHist YHist ZHist
HISTORY = (0.00, 0.00, 5.00)

Specify a position for a time file. Enter the coordinate values in units specified by *DUnit*. In other words, if $DUnit = 1000$, supply the position in millimeters. A script may contain up to 15 *History* commands.

NHSTEP Nhs
NHSTEP = 20

By default, entries are made in time files at each time step. Small values of Δt may result in very large time files. Enter a value of N_{hs} greater than one to reduce the file size. In this case, entries are made at intervals $N_{hs}\Delta t$. The default value is $N_{hs} = 1$.

6 Res mode commands

This chapter reviews how to set up *Res* mode solutions to find the frequencies of resonant modes in three-dimensional assemblies. The solution strategy (described in Sect. 6.3) is to excite the structure with a pulse that spans a specified range of frequencies. A chosen field quantity is monitored at one or more locations. **Aether** calculates the Fourier transforms of the probe signals at the end of the run. Resonances correspond to peaks in the frequency spectrum. The program records the raw transforms and also interpolates to find peak frequencies and writes a table in the listing file (`RunName.ALS`). The next section describes how to create an input script using the *Setup* dialog. Subsequent sections describe allowed script commands in the *Res* mode. The final section discusses data created during a run.

6.1 Res mode setup dialog

Click on the *Setup* command to use the run setup dialog. Pick a **MetaMesh** file and then choose the *Res* mode option in the dialog of Fig. 13. **Aether** displays the dialog of Fig. 15. The function of several of the fields has already been discussed in the previous chapter. Here, we will concentrate on differences from the *Pulse* mode setup dialog.

In the *Control* group, note that there are no fields to set the maximum run time (t_{max}) and the time step (Δt). These quantities are determined from the search parameters. As in the *Pulse* mode, the *DUnit* field specifies units of length used in the input mesh.

Central frequency

Enter the central frequency f_0 of the excitation band in Hz. The excitation pulse is a modulated raised cosine function (Eq. 57) with $\beta = 0.5$ and modulation frequency f_0 . As discussed in Sect. 3.5, the frequency spectrum has the shape of a square function centered at f_0 with smoothed edges.

Frequency band

Enter the frequency span of the modulated raised cosine function in units of Hz. The value Δf is the full width (at half maximum) of the excitation spectrum. The maximum value is $\Delta f = 2f_0$. If the value is 0.0 or unspecified, **Aether** uses the default $\Delta f = 0.3f_0$.

The *Source* group is relatively simple compared to that of the *Pulse* mode because there are fewer options. Resonant modes are excited by currents uniformly distributed over one or more designated source regions. The drive waveform is determined by the control commands, so there are no *SMod* commands.

Region number

Specify the number of a source region (an integer). You can specify multiple source regions using the script commands described in the next section.

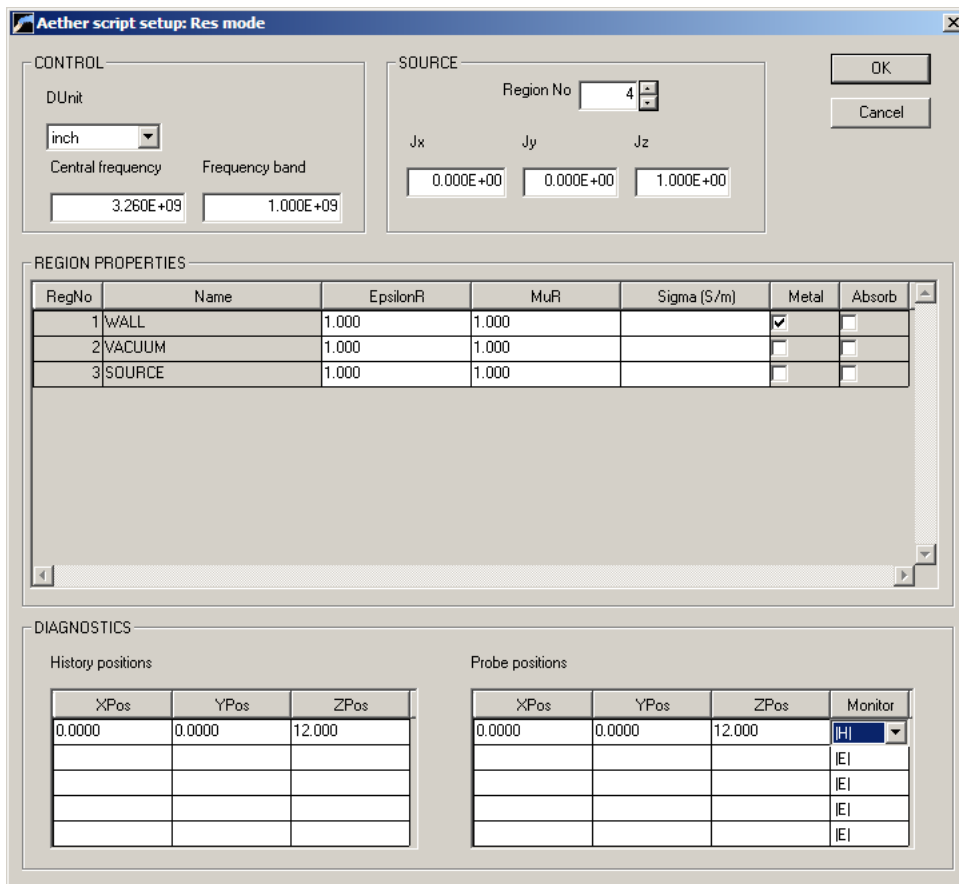


Figure 15: Dialog to create a *Res* mode script.

Jx, Jy, Jz

Set relative values of current density components in the source region. Pick values that preferentially excite the mode of interest. For example, to excite the TE_{111} mode in a circular cavity, one option is to use a source on the axial midplane moved off axis in the x direction with J_y as the only non-zero current component.

The *Region properties* grid is identical to that in the *Pulse* mode dialog. Because **Aether** does not create space files in the *Res* mode, there are no fields in the *Diagnostics* section to set recording times. You can initiate up to five *History* records if you want to view the time variation of fields at specified locations. You can use this feature to check if the fields have reached equilibrium at the end of the run.

Probe positions

Aether records a field component at specified points to provide data for the Fourier transform and peak identification. At least one probe position is required for a *Res* mode solution. The program can record and transform data at up to five probe locations. Enter the position of one or more probes in length units set by *DUnit*. In the *Monitor* column, pick a field component to record. **Aether** will preferentially detect a resonant mode if the probe is near a position where the field component has a maximum. With the proper choice of probe parameters, you can optimize the search for modes with known characteristics.

Click *OK* when you have completed entries in the dialog. Save the file with the default or with a different prefix.

6.2 Res mode script commands

This section reviews *Res* mode script commands, with emphasis on differences from the *Pulse* mode (Chap. 5). Table 6.2 shows an example of an **Aether** script. The following commands have the same function as those in the *Pulse* mode.

```
MESH MPrefix  
DUNIT DUnit  
INTERP [Spline,Linear]  
SYMBOUND [XDn,XUp,YDn,YUp,ZDn,ZUp]  
COURANT CCafety  
EPSI(RegNo) Epsi  
EPSI(RegNo) > EpsiFunction (space)  
MU(RegNo) Mu  
MU(RegNo) > MuFunction (space)  
SIGMA(RegNo) Sigma  
SIGMA(RegNo) > SigmaFunction (space)  
VACUUM(RegNo)  
METAL(RegNo) [EpsiR MuR]  
ABSLAYER(RegNo) DeltaZ [EpsiR MuR]
```

There are three commands that do not appear in the *Pulse* mode.

Table 3: **Aether** script example for a *Res* mode calculation.

```

* Aether 1.0 Script (Field Precision)
* File: KLYSTRON_OUTPUT3D.AIN
* NReg      RegName
* =====
*      1      CAVITYWALL
*      2      VACUUM
*      3      RESISTOR
*      4      DRIVE
* ---- CONTROL ----
Mode = RES
Mesh = KLYSTRON_OUTPUT3D
DUnit = 100.0
Freq = (1.5E9, 3.0E9)
Source(4) = 0.0 0.0 1.0
* ---- REGION PROPERTIES ----
Metal(1)
Vacuum(2)
AbsLayer(3) 0.10
Vacuum(4)
* ---- DIAGNOSTICS ----
History = -5.50 0.00 0.00
Probe = -5.50 0.00 0.00 Hy

EndFile

```

FREQ f_0 [Δf]

FREQ = (32.0E9 16.0E9)

Aether searches for resonances in a frequency band of width Δf with center at f_0 . Enter values in Hz. If Δf is not specified, the default is $\Delta f = 0.3f_0$. The frequency band must enclose the mode of interest. Smaller values of Δf may increase the accuracy for identifying the mode frequency, but significantly increase the run time. The minimum value is $\Delta f = 0.2f_0$ and the is $\Delta f = 2f_0$.

SOURCE RegNo Jx Jy Jz

SOURCE(5) = (0.0, 0.0, 1.0)

The resonant mode is excited by a harmonic current source in one or more regions of the solution volume. This command specifies that region *RegNo* acts as a source and gives the direction of drive current density within the region. The parameters J_x , J_y and J_z are the real-number components of a unit vector pointing in the direction of the current.

PROBE XProbe YProbe ZProbe [|E|,Ex,Ey,Ez,|H|,Hx,Hy,Hz]
PROBE(5.6,0.0,0.2) = |H|

Enter the locations and recorded field quantity for up to five probes. The coordinate values should be in units set by *DUnit*. In the absence of a specification, the default component is $|H|$. The time-variation of the field component at the location is written to a file with a name of the form `RunName.P5n` during the recording phase of the run. The recorded data are replaced with their discrete Fourier transform. The contents of the file may be plotted with the **Probe** utility. **Aether** also makes a listing of peaks of the cavity-response spectrum for each probe in the listing file `RunName.ALS`.

6.3 Calculation methods and data records

The material in this section is included for reference. It is not essential to understand the details because **Aether** takes care of the process automatically. In a *Res* mode calculation, **Aether** determines the Courant time step for numerical stability Δt_c and also defines a maximum time to resolve an the RF oscillation at the central frequency:

$$\Delta t_{rf} = \frac{1}{50f_0}. \quad (108)$$

The minimum value of Δt_c and Δt_{rf} is used as an initial estimate of the time step Δt . **Aether** uses the raised cosine function with modulation frequency f_0 (described in Sect. 3.3) as the waveform for current density in the source region. The pulse has rolloff factor $\beta = 0.5$ and pulse width

$$t_p = \frac{1}{\Delta f}. \quad (109)$$

The run is divided into two phases: initialization and recording. The structure is excited by the pulse in the initialization phase. In the recording phase, field quantities for the ringing structure are stored. For a good representation of the raised cosine function, the initialization time taken as

$$t_{init} = 6t_p. \quad (110)$$

The center of the pulse occurs at $t_c = t_{init}/2$. With regard to the run time, the duration of the initialization phase is inversely proportional to the width of the exciting frequency spectrum.

The maximum frequency to resolve in the discrete Fourier transform is

$$f_{max} = f_0 + \frac{\Delta f}{2}. \quad (111)$$

The time interval between data records in the probe files is set by the Nyquist limit,

$$\Delta t_{rec} = \frac{1}{2f_{max}}. \quad (112)$$

The number of simulation time steps per data record is

$$N_{rec} = \frac{t_{rec}}{\Delta t}. \quad (113)$$

The total number of data points recorded is $R = 256$. The total recording time is

$$t_{rec} = RN_{rec}\Delta t. \quad (114)$$

7 RF mode commands

This chapter describes how to use **Aether** for frequency-domain solutions. Here, the electromagnetic fields result from steady-state excitation of a three-dimensional structure at a single frequency. The following strategy is used to find a frequency-domain solution with a time-domain code (Sect. 4.2):

- Drive a structure with one or more current sources at a specified frequency, allowing time for the fields to settle to an equilibrium.
- Determine the amplitude and phase of field components in all elements and store the results in complex-number (phasor) form.

The procedure takes far less side time than a direct frequency-domain solution involving inversion of a three-dimensional matrix. The precaution is that you must take care to ensure that the fields have settled to a steady state for good accuracy.

The following section explains how to create an input script for the RF mode using the *Setup* dialog. Section 7.2 covers the script commands recognized in the *RF* mode. Section 7.3 describes features of the code to monitor solution convergence.

7.1 RF mode setup dialog

Use the *Setup* command to activate the setup dialog. Pick a **MetaMesh** file and then choose the *RF* mode option in the dialog of Fig. 13. **Aether** displays the dialog of Fig. 16. The function of several of the fields has already been discussed in the previous two chapters. This section emphasizes differences from the *Pulse* and *Res* mode setup dialogs.

The commands of the control group set the drive frequency and the length of the run to reach equilibrium.

Frequency

Enter the drive frequency f_0 in Hz.

NPeriod

The integer quantity *NPeriod* is the duration of the time-domain solution in RF periods. By default, **Aether** uses a modulation waveform with a smooth start over three RF periods to avoid high-frequency numerical noise. Choose a value *NPeriod* > 3. Increase the value if the solution does not reach equilibrium. There are several control options you can exercise by editing the *NPeriod* command in the script.

The *Source* group has two commands.

Element file prefix

One way to specify the spatial variation of current density $\mathbf{j}(x, y, z)$ is to create a file of current

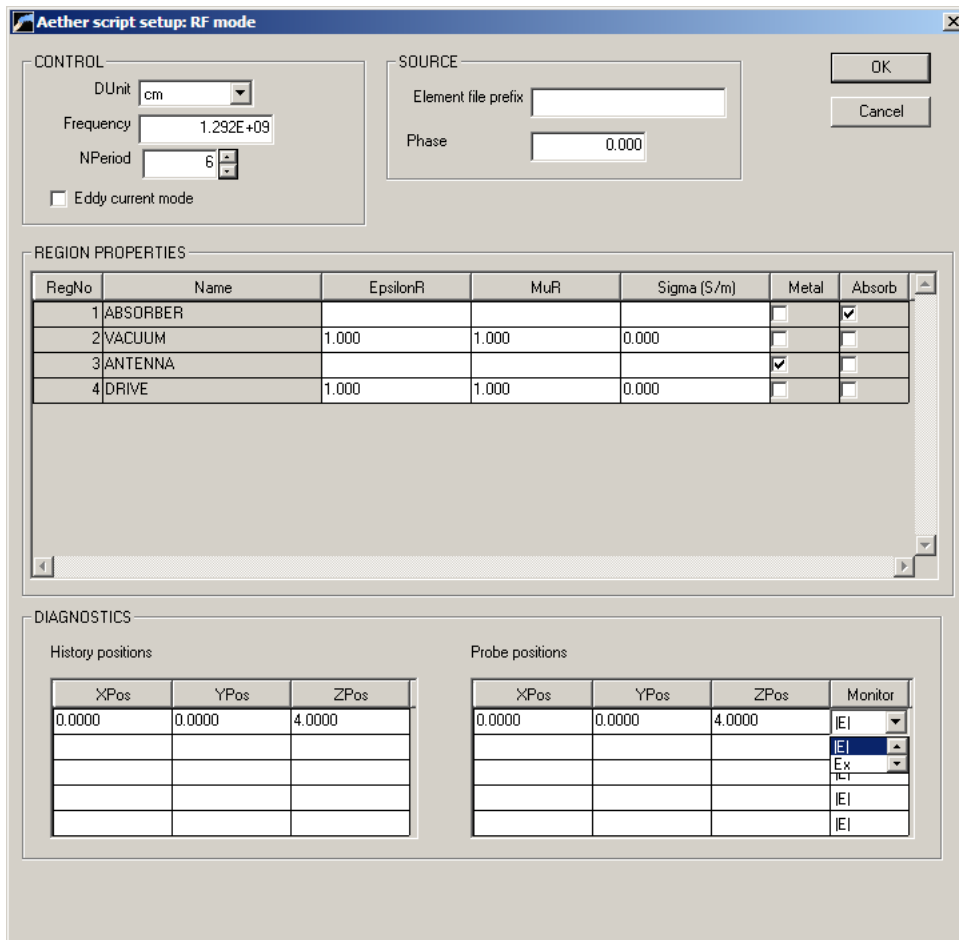


Figure 16: Dialog to create an *RF* mode script.

elements using **Magwinder** (Chap. 14). Enter the prefix of a winding file available in the working directory. The full file name is `FPrefix.WND`. The elements are associated with Source Number 1. You may assign spatial variation of current density for up to four additional sources using a text editor and the script commands described in the next section.

If no entry appears for an element file, **Aether** writes a set of comment lines in the script to show the syntax of the J_x , J_y , J_z and J_r commands. In this case, you must edit the script, entering commands that set current density in chosen regions.

Phase

Specify the phase of the drive currents in degrees. After the smooth start, the current has the variation $\cos(2\pi ft + \phi)$. You may specify up to 10 different drive phases by editing the script.

The *Region properties* grid is identical to those in the *Pulse* and *Res* mode dialogs. As in the *Pulse* mode, you can set up 5 positions for history records. Strategically-placed history monitors are a good way to check convergence of the time-domain solution.

Probe positions

Aether records a field component at specified points to provide a check on the solution convergence. Enter the position of one or more probes in length units set by *DUnit*. In the *Monitor* column, pick a field component to record.

Click *OK* when you have completed entries in the dialog. Save the file with the default or with a different prefix.

7.2 Script commands

This section reviews *RF* mode script commands. Table 7.2 shows an example of a complete script. The following commands have functions identical to those of the *Pulse* mode (Chap. 5):

MESH MPrefix
DUNIT DUnit
INTERP [Spline,Linear]
FORMAT [Text, Binary]
SYMBOUND [XDn,XUp,YDn,YUp,ZDn,ZUp]
RUNTIME Trun SFILE(SourceNo) SFilePrefix
COURANT CSafety
EPSI(RegNo) Epsi
EPSI(RegNo) > EpsiFunction (space)
MU(RegNo) Mu
MU(RegNo) > MuFunction (space)
SIGMA(RegNo) Sigma
SIGMA(RegNo) > SigmaFunction (space)
VACUUM(RegNo)
METAL(RegNo) [EpsiR MuR]

ABSLAYER(RegNo) DeltaZ [EpsiR MuR]

JX(RegNo,SourceNo) = Jx

JY(RegNo,SourceNo) = Jy

JZ(RegNo,SourceNo) = Jz

JX(RegNo,SourceNo) > JxFunc

JY(RegNo,SourceNo) > JyFunc

JZ(RegNo,SourceNo) > JzFunc

JR(RegNo,SourceNo) > JrFunc

The following commands either do not appear in the *Pulse* mode or have different functions:

FREQ f0

FREQ = 1.292E9

In an *RF* mode simulation, all current density sources vary harmonically at frequency f_0 . Enter the value in Hz.

NPERIOD NPeriod [NRise] [NStop]

NPERIOD = (10,3,8)

The RF period is $\tau = 1/f_0$. The run time to generate an equilibrium is $t_{max} = N_{period}\tau$. **Aether** records field values at this time and then runs for an addition quarter cycle. The program determines the amplitude and phase of quantities by comparing values. Increase N_{period} if the solution has not converged. The optional parameter N_{rise} gives the number of cycles for the smooth rise and fall of the drive current, necessary to prevent high-frequency noise associated with discontinuities. The default is $N_{rise} = 3$. The final optional parameter N_{stop} determines whether the drive currents are turned off before the end of the time-domain solution. If $N_{stop} < N_{period}$, the current density smoothly falls to zero over at $t = N_{stop}\tau$. The following rules apply:

- When exciting a resonance in a high- Q structure, the drive currents should be turned off before the end of the solution to allow the fields to equilibrate (i.e., $N_{stop} < N_{period}$).
- When driving a damped structure, the drive currents should reach an equilibrium value and remain active during the entire solution (i.e., $N_{stop} > N_{period}$).

The default value is $N_{stop} = \infty$.

SMOD SourceNo Phase

SMOD(2) = 45.0

Drive currents specified by the commands *SFile*, *Jx*, *Jy* and *Jz* must be associated with a modulation function. You can define up to ten modulation functions with this command. Although all the functions are harmonic variations at frequency f_0 , they may have different phases. Enter the phase in degrees. Modulation function n has the variation $f_n(t) = \cos(2\pi f_0 t + \phi_n)$ for $t > N_{rise}\tau$. A script must contain at least one *SMod* command.

Table 4: **Aether** script example for an *RF* mode calculation. Note the annotations after the *EndFile* command.

```

* File: HALFWAVE.AIN
* NReg      RegName
* =====
*      1      ABSORBER
*      2      VACUUM
*      3      ANTENNA
*      4      DRIVE
* ---- CONTROL ----
Mode = RF
Mesh = HalfWave
DUnit = 1.0
Freq = 3.0E8
NPeriod = 4 2
* ---- CURRENT SOURCES ----
SMod(1) = 0.0
* ---- REGION PROPERTIES ----
AbsLayer(1) 0.025
Vacuum(2)
Metal(3)
Epsi(4) = 1.0
Mu(4) = 1.0
Sigma(4) = 0.0
Jz(4,1) = 133.0
* ---- DIAGNOSTICS ----
History = 0.50 0.00 0.00
Probe = 0.50 0.00 0.00 Ez

EndFile

Total length of antenna: 1.0 m
Lambda: 2.0 m
Frequency = c/Lambda = 1.5E8
Source cross section: 12*(2.5E-2)^2 = 7.5E-3 m2
Total current: 1.0 A -> 133 A/m2

```

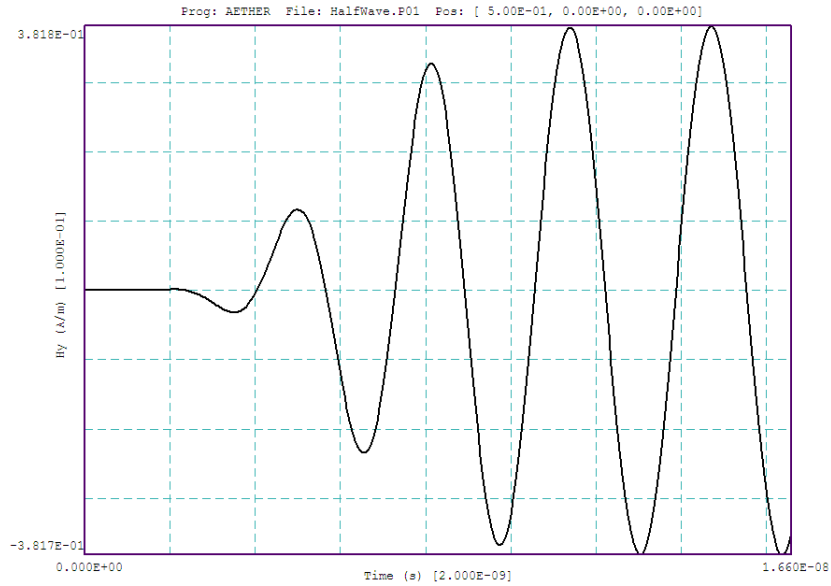



Figure 17: Monitoring solution convergence in the *RF* mode with a history record.

PROBE XProbe YProbe ZProbe [|E|,Ex,Ey,Ez,|H|,Hx,Hy,Hz]
PROBE(5.6,0.0,0.2) = |H|

Enter the locations (in units set by *DUnit*) and a recorded field component for up to five probes. In the absence of a specification, the default field component is **|H|**. **Aether** monitors the field and makes a listing of the relative change at time intervals of $\tau/4$ in the listing file *RunName.ALS*. With this feature, you can check solution convergence at several locations in the solution volume.

7.3 Monitoring solution convergence

For accuracy, the time-domain solution should be close to an equilibrium at the end of the run when values are converted to phasor form. **Aether** provides two options to monitor convergence. The first is to set one or more history monitors and to inspect the signal with the **Probe** utility. Figure 17 shows an example for a strongly-damped antenna solution. The monitor is at a point about halfway between the antenna and the absorbing boundary of the solution volume. The figure shows that the solution quickly settles into a steady-state after the smooth start.

The second option is to place a probe which records quantitative on convergence. After the slow start, **Aether** writes table in the listing file each quarter period. The following is an example for a single probe:

```

----- RF Probe Check -----
Time: 1.333333E-08
Elapsed RF periods: 4.00
Probe Quantity Q dQ
=====
1 Ez 1.285262E+02 0.00329

```

The listing contains a row for each probe. **Aether** compares values at the present time and one quarter period earlier to find the amplitude of the harmonic variation. A data row shows

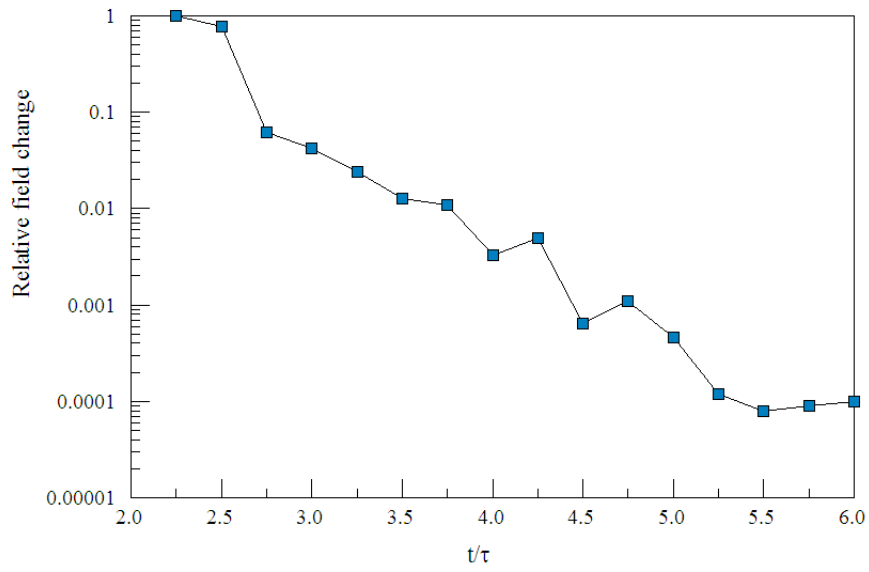


Figure 18: Monitoring solution convergence in the RF mode with a field probe.

the absolute value of the amplitude and the relative change since the last reported point. The logarithmic plot of Fig. 18 illustrates a typical convergence history for a strongly-damped system.

8 Aerial – file and analysis operations

8.1 File operations

The function of the **Aerial** post-processor is create plots and to calculate numerical quantities from **Aether** binary solution files. The program has the following popup menus: *File operations*, *Slice plots*, *Plane plots*, *Surface plots*, *Analysis*, and *Help*. Initially, only the *File operations* and *Help* menus are active. You must load a data file in order to create plots or to perform analyses. This section reviews options in the *File operations* menu.

Aerial adjusts plot and analysis operations, depending on whether you load a *Pulse* mode or an *RF* mode solution. Use the following command to load an *RF* mode solution:

RF MODE: LOAD SOLUTION

An **Aether** run in the *RF* mode produces a single output file of complex-number field values with a name of the form `RunName.AOU`. Changing the directory in the load dialog changes the program working directory. Pick an available file and click *OK*. The program loads the solution and updates the status bar. If data retrieval is successful, the analysis and plot menus become active.

The following two commands are used to load *Pulse* mode solutions:

PULSE MODE: SET SERIES

A *Pulse* mode run may create several data files with the same prefix. Use this command to specify a file prefix for subsequent load operations. Moving to a new directory in the load dialog changes the program working directory. **Aerial** counts the number of files in the series, records the times and displays the dialog of Fig. 19. Pick a solution file to load and click *OK*.

PULSE MODE: LOAD SOLUTION

Load a different solution file in a series. **Aerial** displays the dialog of Fig. 19. Pick a different solution and pick *OK*. This command is active only when a series has been specified.

The remaining commands have the same function for all solution modes.

SOLUTION FILE INFORMATION

The command shows a message box with information on the currently-loaded data file.

RUN SCRIPT

Sometimes you may want to perform complex or repetitive analysis operations on a set of similar solutions. Script operation is a powerful feature of **Aerial**. This command displays a dialog with a list of analysis scripts (suffix *SCR*) that you have created. Pick a file and click *OK*. Changing directories in the dialog changes the working directory of the program.

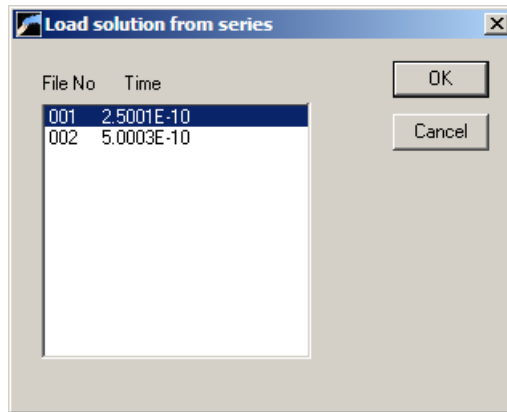


Figure 19: Dialog to pick a solution file from a series

The analysis script can load data files, open and close history files, and perform any of the numerical functions described in this manual. Chapter 12 reviews the analysis script language.

CREATE SCRIPT

Use this command to create scripts using the internal editor. A box requests a file prefix. The resulting script file will be saved as `FPREFIX.SCR`. Next, the program opens the file in the editor and writes the reference list of allowed commands shown in Table 5. Enter commands in the space above *EndFile*. After saving the file, you can run it using the *Run script* command.

EDIT SCRIPT

Use this command to change an existing script file. The dialog lists files in the current directory with the subscript `SCR`. Changing directories does not change the working directory of the program.

OPEN DATA FILE

Several of the analysis commands like *Point calculation* and *Line scan* generate quantitative information. You can record the data generated during an analysis session by opening a data file. Supply a file prefix in the dialog or accept the default. The text data file has a name of the form `FPREFIX.DAT` and will be stored in the working directory. You can use an editor to view the file or to extract information for mathematical analysis programs or spreadsheets.

CLOSE DATA FILE

Use this command if you want to start a new file to record data. The data file is automatically closed when you exit **Aerial**. Otherwise, you must close the file before using the *Edit data file* command or loading the file into another program. Failure to close the file may result in a Windows Resource Sharing Error.

EDIT DATA FILE

View or modify files with names of the form `FPREFIX.DAT`.

Table 5: Create script - default file content

```

* AMaze script file
* Insert commands here...
ENDFILE

    --- Script command summary ----
INPUT FileName
    [Close current solution file and load FileName]
OUTPUT FPrefix
    [Close current data file and open FPrefix.DAT]
NSCAN 100
    [Set the number of points in a line scan]
REFPHASE PhiRef
    [Set the reference phase in degrees, RF mode ]
FORMAT [AmpPhase, Complex]
    [Set the format for writing field values, RF mode]
NORMALIZE NFact
    [Normalize the solution, RF mode]
SAVE FPrefix
    [Save the solution to FPrefix.AOU, RF mode]
POINT xp yp zp
    [Point field calculation at the given coordinates]
LINE xp1 yp1 zp1 xp2 yp2 zp2
    [Scan along a line between the given coordinates]
PATH PathName
    [Point calculations at coodinates in file PathName]
INTEGRALS
    [Write volume and surface integrals to the data file]
LINEINT xs ys zs xe ye ze
    [Line integrals of E and H]
MATRIX FileName XMin XMax NX YMin YMax NY ZMin ZMax NZ
    [Write a matrix of field values to the file FileName]
ENDFILE
    [Terminate the analysis]

```

EDIT FILE

Use the program editor to view or to modify any text file.

The *Help* menu shows program information and contains the following command:

AETHER MANUAL

Displays this document in your default PDF viewer. The file `aether.pdf` must be in the same directory as `aerial.exe`.

8.2 Global solution analysis

SOLUTION INTEGRALS

In response to this command, **Aether** performs a variety of volume and surface integrals and write the results to the data record file. Table 8.2 shows an example. The listing of volume integrals at the top shows electric field energy, magnetic field energy and power dissipation for the full volume and organized by regions. For *Pulse* mode solutions, the quantities apply to the time the solution was recorded. The following expressions are used:

$$U_e = \int \int \int dV \frac{\epsilon_r \epsilon_0 \mathbf{E} \cdot \mathbf{E}}{2}, \quad (115)$$

$$U_h = \int \int \int dV \frac{\mu_r \mu_0 \mathbf{H} \cdot \mathbf{H}}{2}, \quad (116)$$

$$P = \int \int \int dV \sigma \mathbf{E} \cdot \mathbf{E}. \quad (117)$$

In the RF mode, the energy and power quantities are time-averaged values, given in terms of the peak values of the spatial amplitudes of field quantities by:

$$U_e = \int \int \int dV \frac{\epsilon_r \epsilon_0 \mathbf{E} \cdot \mathbf{E}^*}{4}, \quad (118)$$

$$U_h = \int \int \int dV \frac{\mu_r \mu_0 \mathbf{H} \cdot \mathbf{H}^*}{4}, \quad (119)$$

$$P = \int \int \int dV \frac{\sigma \mathbf{E} \cdot \mathbf{E}^*}{2}. \quad (120)$$

The Poynting integrals at the bottom shows the flux of electromagnetic energy over the surfaces of regions in the solution volume. They are organized to show the power leaving the regions of the solution volume. For each region, Aether shows the power flow to other regions that share a boundary. Here, Region 0 represents the space outside the solution volume. The instantaneous Poynting vector for Pulse mode solutions is given by

$$\mathbf{P} = \mathbf{E} \times \mathbf{H}, \quad (121)$$

The time-averaged power flux for RF mode solutions is

Table 6: Listing created by the *Solution integrals* command.

```

Global and regional analysis of RF solution file
Time-averaged field energy and resistive power dissipation
Global electric field energy:  1.37743E-07 (J)
Global magnetic field energy:  1.22177E-07 (J)
Global total field energy:    2.59920E-07 (J)
Global power dissipation:    4.18491E+01 (W)

RegNo      Ue          Uh          Ut          P          Volume
          (J)          (J)          (J)          (W)          (m3)
=====
  1  1.74492E-09  5.31921E-10  2.27684E-09  4.18491E+01  6.15125E-01
  2  1.21823E-07  1.21193E-07  2.43015E-07  0.00000E+00  7.98944E+00
  3  0.00000E+00  7.26828E-16  7.26828E-16  0.00000E+00  8.93750E-03
  4  1.41752E-08  4.52353E-10  1.46276E-08  0.00000E+00  1.62500E-03

Time-averaged Poynting vector flux
Power from region:  1
  RegNo >>  0  Area:  2.52150E+01 (m2)  Power:  2.09900E+01 (W)
  RegNo >>  2  Area:  2.40000E+01 (m2)  Power: -2.08320E+01 (W)
Power from region:  2
  RegNo >>  1  Area:  2.40000E+01 (m2)  Power:  4.23337E+01 (W)
  RegNo >>  3  Area:  4.65000E-01 (m2)  Power: -9.54218E+00 (W)
  RegNo >>  4  Area:  4.00000E-02 (m2)  Power: -3.02340E+01 (W)
Power from region:  3
  RegNo >>  2  Area:  4.65000E-01 (m2)  Power:  0.00000E+00 (W)
  RegNo >>  4  Area:  6.50000E-02 (m2)  Power:  0.00000E+00 (W)
Power from region:  4
  RegNo >>  2  Area:  4.00000E-02 (m2)  Power:  3.87016E+01 (W)
  RegNo >>  3  Area:  6.50000E-02 (m2)  Power:  2.79935E-01 (W)

```

$$\mathbf{P} = \frac{\text{Re}(\mathbf{E} \times \mathbf{H}^*)}{2}. \quad (122)$$

The field quantities \mathbf{E} and \mathbf{H} are taken at the centers of elements of the power source region. Therefore, reported values are always zero for metal regions and may be ambiguous for regions with single-element thickness like absorbing layers.

LINE INTEGRAL

Compute and record values of the quantities $\int \mathbf{E} \cdot d\mathbf{l}$ and $\int \mathbf{H} \cdot d\mathbf{l}$ between any two points in the solution volume. In the dialog, enter coordinates for the start and end points in units set by *DUnit*.

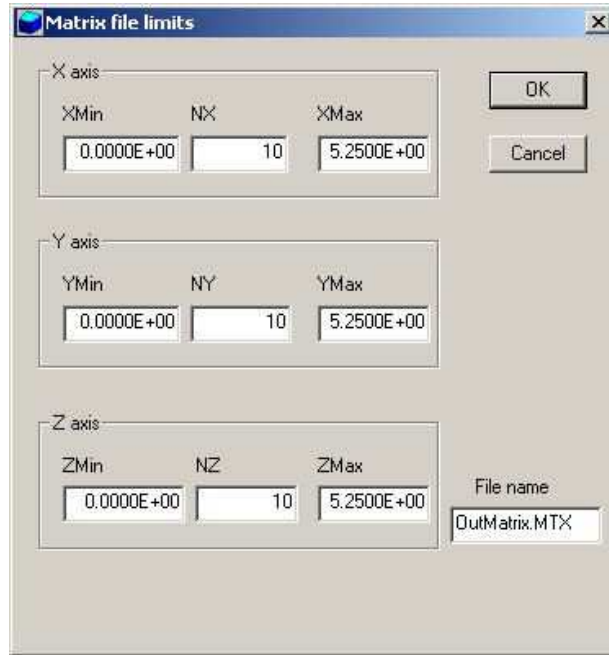


Figure 20: Create matrix file dialog.

CREATE MATRIX FILE

This command controls a feature that is useful if you want to write your own analysis routines or port results to mathematical software. In response to the command, **Aerial** performs interpolations over a specified box region on a regular grid of values. Clicking on *Create matrix file* calls up the dialog of Fig. 20. Specify the dimensions of the box along each axis (in units set by *DUnit*) and the number of calculation intervals. To illustrate, calculations are performed at positions with x coordinates given by

$$x = x_{min} + \frac{n (x_{max} - x_{min})}{n_x}, \quad (123)$$

where $n = 0, 1, 2, \dots, n_x$. For example, if you set $x_{min} = 0.5$, $x_{max} = 1.5$ and $n_x = 10$, the calculations are performed at points with $x = 0.5, 0.6, \dots, 1.4, 1.5$. You can also specify an output file name. For *Pulse* mode solutions, **Aerial** writes components of **E** and **H** at the recording time of the space file. Steady-state complex-number values are recorded for *RF* mode solutions. Numbers are written in amplitude/phase or real/imaginary format, depending on the mode set with the *Number format* command.

The following three commands are active only when an *RF* mode solution is loaded.

NUMBER FORMAT

Complex-number values may be recorded in a matrix file in two formats: 1) amplitude and phase (in degrees) or 2) real and imaginary parts. Use this command to toggle between the formats.

NORMALIZE SOLUTION

Solutions for resonant modes in high-Q structures generally have arbitrary normalization. Use

this command to multiple all stored quantities by a real-number factor A_n . For example, A_n may be equal to the desired level of a field quantity at a point in the solution divided by a value determined with the *Point calculation* command in the *Slice plot* menu.

SAVE SOLUTION

Use this command to save a renormalized solution in the standard **Aether** binary format.

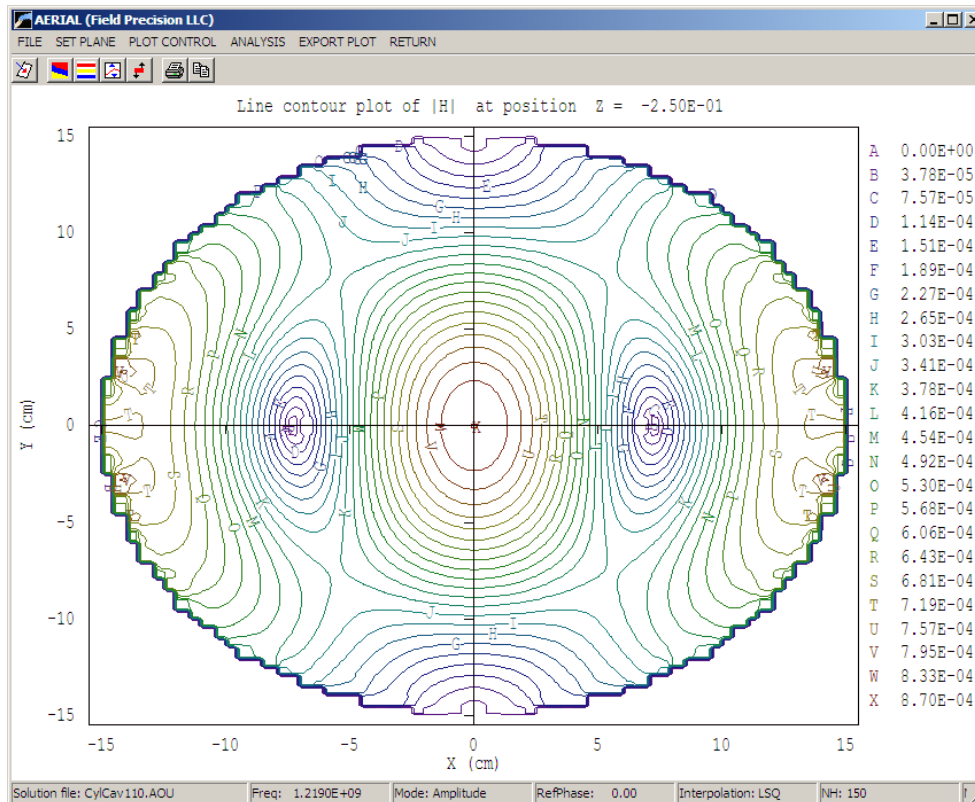


Figure 21: Screen display in the *Plane plot* menu. The plot style is *Contour lines 2D*.

9 Aerial – plane plots

When a data file has been loaded, click *Plane plot* to enter the plotting menu. Plane plots are two-dimensional views that show the variation of quantities over a plane normal to one of the Cartesian axes. Plane plots provide simple and quick views of the solution space. The technique is to generate a rectangular mesh of values over a specified planar region and then to create plots in a variety of styles. No attempt is made to connect the plot mesh with the mesh of the simulation. Slice plots (discussed in the next chapter) are also two-dimensional in a plane normal to an axis. They are constructed using the computational mesh. The two plots types have relative advantages:

- Plane plots offer a wider variety of styles for presentations.
- True-scale slice plots support interactive analysis, using the mouse to specify coordinates. It is also possible to display calculated lines of **E** and **H**.

Figure 21 shows the **Aerial** screen in the *Plane plot* menu for an *RF* mode solution. The *File* menu contains several of the commands discussed in the previous section. In the *Pulse* mode, use the *Load solution* command to work with a different solution in a series. You must return

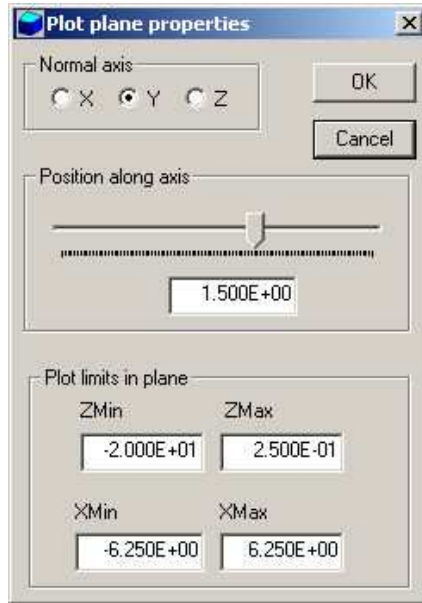


Figure 22: Set plane dialog.

to the main menu to set a different series or to load an RF solution. The status bar displays current settings of program parameters appropriate to the plot style and solution mode.

SET PLANE

This command brings up the dialog of Fig. 22 to set the plane for the plot. Set the normal axis with the radio buttons at the top. For example, for a choice of z , the plot will be created in the x - y plane. You can use the slider bar to set the position along the normal axis or type a value in the box. The range of the slider bar is automatically set to the limits of the solution volume along the normal axis. The boxes at the bottom determine the plot range in the normal plane. The default settings are the limits of the solution volume. Note that plane plots are constructed to fill the maximum area. They do not preserve scaling in the normal plane.

The following commands are in the *Plot control* pop-up menu:

PLOT STYLE

This command brings up the dialog of Fig. 23 to set the plot style. As an example, Fig. 21 shows the *Contour lines 2D* plot style. The numbers at the bottom give the resolution of the mesh used to create the plot. Higher values give more detail but require longer regeneration times. The default is a 100×100 mesh.

PLOT QUANTITY

Set the quantity to be plotted. The choices include the components of the electric field E_x , E_y and E_z or the magnitude of the electric field vector:

$$|E| = \sqrt{E_x^2 + E_y^2 + E_z^2}. \quad (124)$$

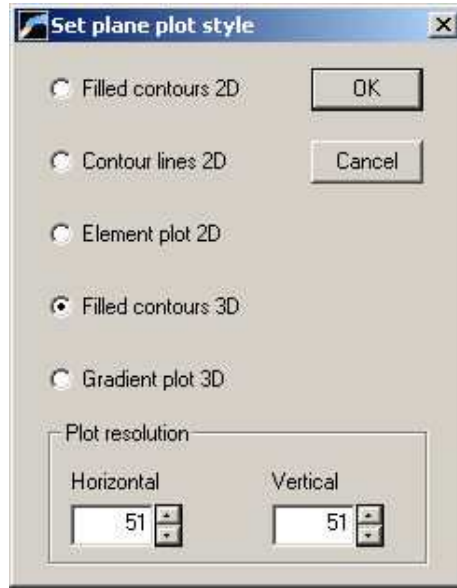


Figure 23: Plot style dialog.

Other quantities include the components and magnitude of the magnetic field \mathbf{H} and the magnetic flux density $\mathbf{B} = \mu\mathbf{H}$. The current density quantities j_x , j_y , j_z and $|\mathbf{j}|$ represent the sum of applied and ohmic currents. They have non-zero values only in drive regions or regions with $\sigma \neq 0.0$. The Poynting vector

$$\mathbf{S} = \mathbf{E} \times \mathbf{H}, \quad (125)$$

gives the flux of electromagnetic energy in units of W/m^2 . The quantity u is the electromagnetic field energy density (in J/m^3). The resistive power density p (W/m^3) has nonzero values in regions with $\sigma \neq 0.0$. Plots of the remaining quantities (ϵ_r , μ_r and σ) are useful mainly if you have defined spatial variations of dielectric constant, relative magnetic permeability or electrical conductivity over a region.

In a *Pulse* mode solution, the field, energy and power quantities are instantaneous values at the time the space file was recorded. In the *RF* mode, the program displays time-averaged values of u and p . The interpretation of the field quantities (\mathbf{E} , \mathbf{H} , \mathbf{B} and \mathbf{S}) is covered in the discussion of the *Toggle reference/amplitude* command.

PLOT LIMITS

Set limits for the plotted quantity. When *Autoscale* is active, **Aerial** automatically sets limits based on the range of values of computed quantities in the displayed plane. Set values manually if you want to compare quantities in different planes.

ROTATE PLOT

This command is active only for the *Filled contours 3D* and *Gradient plot 3D* styles. You can rotate the plot in 90° increments for the best view.

The next commands control the interpretation of complex-number quantities in *RF* mode displays:

TOGGLE REFERENCE/AMPLITUDE

This command determines how the RF field quantities **E**, **H**, **B** and **S** are displayed. In the *Reference* case, **Aerial** shows a snapshot of the oscillating field quantities at a given phase ϕ_{ref} . In this case, the quantities may have positive and negative values. In the *Amplitude* mode, **Aerial** displays the peak value of the oscillating component or spatial magnitude of the field quantity. The status bar shows the current option.

REFERENCE PHASE

Enter the phase ϕ_{ref} for *Reference* mode displays in degrees.

The commands of the *Analysis* popup menu were discussed in Sect. 8.2. The commands of the *Export plot* menu are used to generate printed output or to create plot files.

DEFAULT PRINTER

With this command, you can port the current **Aerial** plot to any installed Windows printer. Note that the current screen plot is sent to the default printer. If necessary, change the default using the *Settings* command of Windows before issuing the command.

COPY TO CLIPBOARD

Copy the current screen plot to the clipboard in Windows Metafile format. You can then paste the image into graphics software.

SAVE PLOT FILE

Use this command to create a graphics file of the current plot in either Windows Bitmap (BMP) or Portable Network Graphics (PNG) formats. In the dialog, specify the format, the size in pixels and the file prefix. The graphics file is created in the current directory.

The creation of plots for presentations may involve some effort. With the following two commands, you can save all the current view parameters and immediately restore the plot.

SAVE NAMED VIEW

Save the view parameters for the current plot. Quantities such as the slice axis, slice position and zoom limits are saved for two dimensional plots. Parameters for three-dimensional plots include the viewpoint position, displayed regions and cut planes. The information is stored in a text file in the current directory with a name of the form `FPREFIX.FPV`.

LOAD NAMED VIEW

Load a view file and refresh the plot. Note that you must be in the appropriate plot menu to retrieve a view. Views of plane plots must be loaded to the Plane Plot Menu.

The file contains the complete set of plot parameters. This excerpt illustrates the format:

```
Program: Aerial
2D/3D: 2D
DisplayBy: Regions
Outline: On
NSlice: 40
SliceAxis: YAxis
PlotType: LogElemUp
XMin: -1.500000E+00
XMax: 4.250000E+00
...
```

If a specific solution file is loaded, the plot will be restored exactly. The saved view feature in **Aerial** has two useful features if a different meshes are loaded:

- Dynamic adaptation to different solutions.
- Option for user control of the view parameter set.

Regarding the first feature, there are situations where you want to create consistent views of a set of solutions with different geometries, maintaining a similar appearance. Some plot properties (like the viewpoint rotation matrix) are applicable to any solution, but others (like region cut planes or slice plot limits) depend on the geometry. **Aerial** checks each plot parameter for validity. If a parameter is outside the allowed range for the currently-loaded solution, the program computes an alternative. The goal is to preserve as many features of the view as possible.

You can modify view files with an editor. The order of entries is not rigid. On input, **Aerial** uses a free-form parser. If a parameter is missing, the program simply makes no change from the value current in the program. The implication is that you can modify a saved view to include only elements essential to your application. For example, you could compare a series of assemblies with different sizes, maintaining an orthographic 3D view from the same point in Cartesian space. In this case, the view file would contain only the entries:

```
DView: 1.000000E+37
R11: 8.660253E-01
R12: -5.000002E-01
R13: 0.000000E+00
R21: 1.669031E-01
R22: 2.890846E-01
R23: 9.426408E-01
R31: -4.713208E-01
R32: -8.163510E-01
R33: 3.338061E-01
```

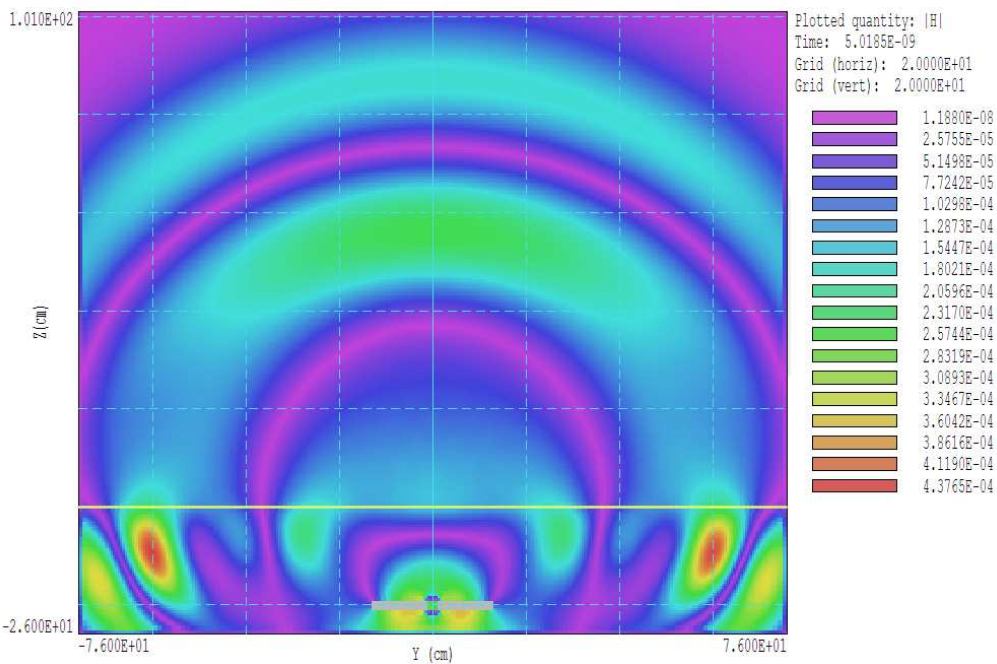


Figure 24: Slice plot example.

10 Aerial – slice plots

10.1 Setting the slice view

Slice plots are two-dimensional views that show the variation of quantities over a plane normal to one of the Cartesian axes. In contrast to plane plots, slice plots are based on the structure of the mesh projected to a slice plane. To facilitate the process, slices are constructed at discrete locations along the normal axis corresponding to the planes of the foundation mesh. The precise rendering of mesh information enables point-and-click analysis operations (point calculation, line scan, ...) in the slice.

As with *Plane* plots, the *File* menu contains several of the commands discussed in Sect. 8.1. In the *Pulse* mode, you can use the *Load solution* command to work with a different solution in a series. You must return to the main menu to set a different series or to load an RF solution. The *Plot view* popup menu contains commands to set the slice plane and to adjust the dimensions of the plot.

SET SLICE PLANE PROPERTIES

This command calls up the same dialog as the *Set plane* command in the plane plot menu (Fig. 22). You can change the normal axis, change the position along the normal axis, and set plot limits in the normal plane.

SLICE NORMAL TO X
SLICE NORMAL TO Y
SLICE NORMAL TO Z

Quick commands to change the normal axis.

JUMP FORWARD
STEP FORWARD
STEP BACKWARD
JUMP BACKWARD

Move along the slice normal axis by small or large steps. The small step is approximately one layer of the foundation mesh and the large step is 5 layers. The term *forward* implies motion toward higher indices of the normal axis. The slider bar in the orientation area to the right of the plot shows the current location.

ZOOM WINDOW

As an alternative to the entries in the *Set slice plane* dialog, you can interactively change plot limits in the normal plane using the mouse. Choose the command and move the mouse pointer into the plot area. The status bar enters coordinate mode. It shows the current mouse position in the plot. Use the left button to pick one corner and then move the mouse to create a view box. Click the left button again, and the plot regenerates. On any coordinate operation, press the *F1* key if want to enter values from the keyboard. Note that the normal plane box in the orientation area to the right of the plot shows the dimensions of the slice plane and the outline of the current zoomed view.

ZOOM IN

Enlarge the plot about the current view center.

EXPAND VIEW

Expand the plot about the current view center.

GLOBAL VIEW

Enlarge the plot boundaries to show the entire normal plane.

PAN

When the plot is zoomed, you can use this operation to shift the current view center. Use the mouse to define relative start and end points for the shift.

10.2 Setting slice plot properties

The commands in the *Plot settings* popup menu are used to set the plot style and mouse options.

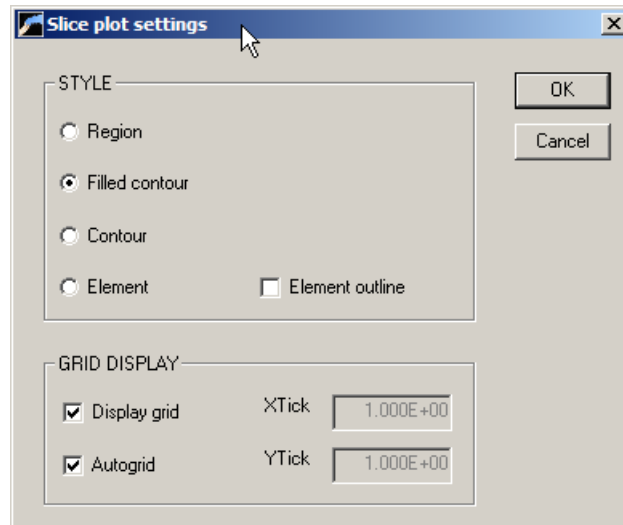


Figure 25: Slice plot settings dialog.

PLOT STYLE

This command brings up the dialog of Fig. 25 for setting properties of the slice plot. In the *Style* group, the *Region* plot type is a cross-section view of the mesh element divisions color-coded by region. The *Filled contour* plot (the default) shows discrete areas of the slice color-coded by the selected plot quantity. The *Contour* plot consists of lines a constant values of the plot quantity. The *Element* plot shows the division of the slice plane into elements with color-coding by the average value of the plot quantity in the element. For this option, the check box *Element outline* determines whether element facets are included. displayed. The four fields in the *Grid* group determine the nature of grid lines superimposed on the plot. Uncheck the *Display grid* box to remove the display. In the *Autogrid* mode, **Aether** chooses good values for the horizontal and vertical grid intervals. When *Autogrid* is unchecked, the *XTick* and *YTick* boxes are active. Enter the desired values for intervals.

PLOT QUANTITY Choose the quantity for color coding in *Filled contour* and *Element* plots or lines for *Contour* plots. The choices are the same as those discussed in Chap. 9. In the RF mode, the interpretation of the quantities depends on whether the display mode is set to *Amplitude* or *Reference phase*. In the latter case, the appearance of the plot depends on ϕ_{ref} .

PLOT LIMITS Set the range for color coding of filled contour, contour and element plots. In the *Autoscale* mode, **Aether** sets the limits for the plotted quantity to the full range in the slice. Uncheck the *Autoscale* box to set values manually. In this mode, you can compare different slices or view variations in regions of weak fields. The *Log intervals* option may be useful for radiation calculations where there is a large change in the displayed quantity. This setting may be applied only to positive-definite quantities like $|\mathbf{E}|$.

NUMBER OF CONTOURS Set the number of lines in *Contour* plots and intervals in *Filled contour* plots.

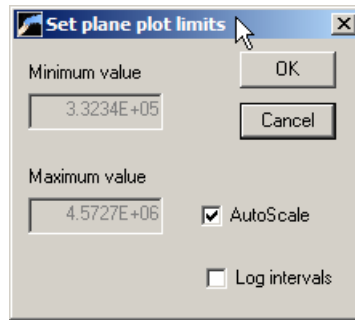


Figure 26:

TOGGLE GRID DISPLAY

Activate or deactivate the superposition of a reference grid on the plot. The grid properties may be set in the *Plot style* dialog.

TOGGLE SNAP MODE

Mouse coordinates for commands such as *Zoom window*, *Pan*, and *Line scan* may be entered in two modes. In the normal mode, the returned position corresponds to the mouse position on the screen. In the snap mode, the program picks a point at an even interval close to the mouse position. The returned point depends on the value of the parameter *DSnap*. For example if $DSnap = 0.1$ and the mouse is at position (6.2345,-5.6113), the returned position is (6.2000,-5.6000). The status bar displays the actual or snapped position of the mouse.

SET SNAP DISTANCE

Change the value of *DSnap* from the default value determined by the program.

TOGGLE REFERENCE/AMPLITUDE REFERENCE PHASE

These commands, discussed in Chap. 9, are present when an *RF* mode solution has been loaded. They determine how the complex-number quantities are displayed.

10.3 Analyses in a slice

You can determine field values at points and along scan lines with the commands of the *Analysis* popup menu.

POINT CALCULATION

This command is useful to make quick checks of fields in the solution volume. After you click the *Point calculation* command, move the mouse into the plot area. The mouse pointer changes to a cross-hair pattern and the status bar enters coordinate mode. Click the left button to specify a point or press the *F1* key to enter the coordinates from the keyboard. Note that mouse coordinates will shift between discrete values if snap mode is active. **Aerial** calculates the field quantities at the corresponding point in the normal plane. Figure 27 shows the information

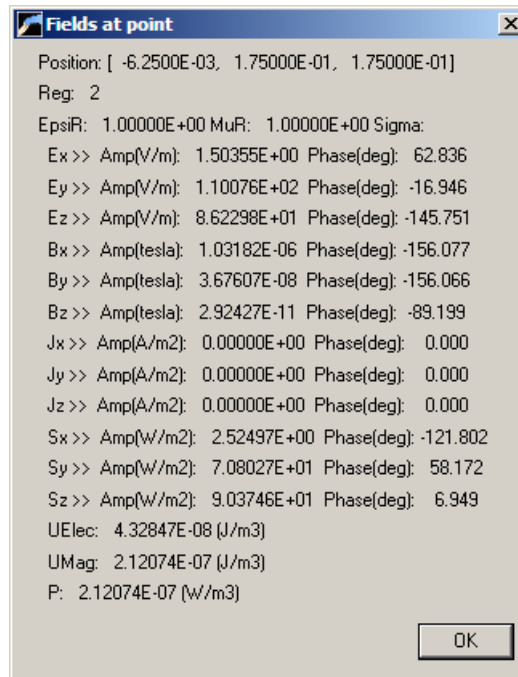


Figure 27: Information display point calculation

display. The results are also recorded if a data file is open. In the RF mode, numbers may be displayed and recorded in amplitude/phase or real/imaginary format (see the *Number format* command).

LINE SCAN

Line scans are one of the most useful **Aerial** features. After clicking on the command, supply two points with the mouse to define a scan line (or press the *F1* key to enter coordinates manually). The snap mode is useful in this application (for example, you may want the scan to extend from 0.000 to 5.000 rather than 0.067 to 4.985.) The program computes a series of values of field quantities in the normal plane at equal intervals along the line. Complete information is recorded if a data file is open. In the *RF* mode, numbers may be recorded in amplitude/phase or real/imaginary format (see the *Number format* command). **Aether** also makes a screen plot of the currently-selected quantity versus distance along the scan and activates the *Scan plot* display of Fig. 28. In the RF mode, plotted values follow settings controlled by the *Toggle amplitude/phase* and *Reference phase* commands.

SET SCAN QUANTITY

With this command you can pick the quantity that will be displayed in screen and exported plots of line scans. Pick the quantity from the list box and click *OK*. The available quantities are the same as those discussed in Chap. 9. In the *RF* mode, the calculated values for the plot depend on settings controlled by the *Toggle amplitude/phase* and *Reference phase* commands.

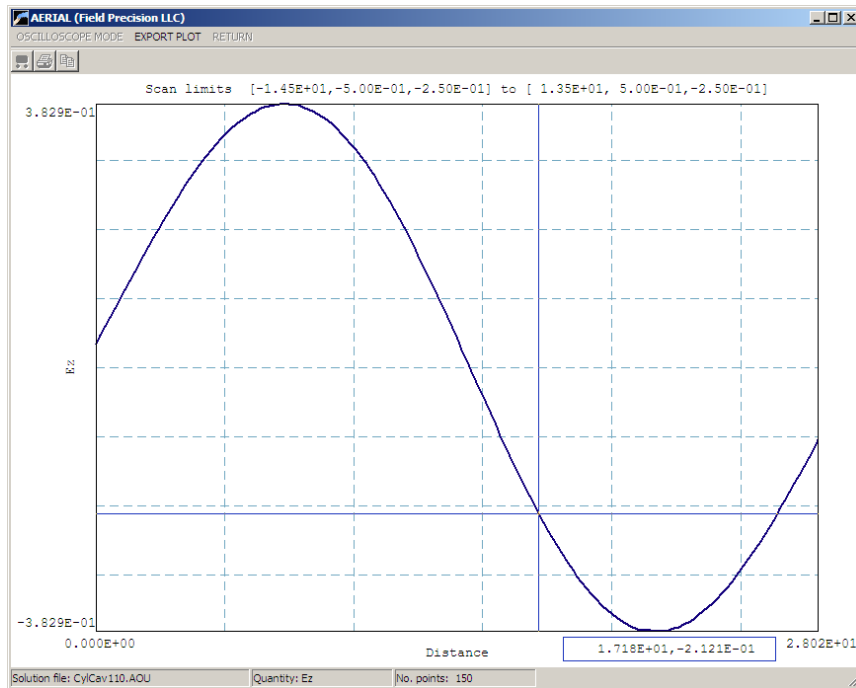


Figure 28: Scan plot display.

SET NUMBER OF SCAN POINTS

This command sets the number of line scan points in the screen plot and data file listing. The default value is 100 and the maximum number is 500.

In addition to the standard *Export plot* options, the *Scan plot* menu of Fig. 28 contains the following command:

OSCILLOSCOPE MODE

In oscilloscope mode, a scan plot assumes characteristics of a digital oscilloscope. **Aerial** superimposes a cross-hair pattern on the graph. Plot values at the intersection are displayed in the information window. Move the marker along the plot by moving the mouse. If you click the left mouse button at a point, the program records information when a data file is open. Press the right mouse button to exit the oscilloscope mode.

Click *Return* to exit the scan plot and return to the slice plot mode.

The function of the *Solution integrals* command was discussed in Sect. 8.2. The command is active when a data record has been opened. The following analysis commands are unique to the slice mode.

LINE INTEGRAL

This command is active when a data record is open. Supply two points with the mouse to define a line (or press the *F1* key to enter coordinates manually). **Aerial** computes the integrals $\int \mathbf{E} \cdot d\mathbf{l}$ and $\int \mathbf{H} \cdot d\mathbf{l}$ between the points. The first quantity may be useful to determine the effective voltage between points or to normalize a resonant mode calculation. The calculation returns a

real-number quantity for *Pulse* mode solutions and a complex-number quantity for *RF* mode solutions.

CIRCUIT INTEGRAL

As in the *Zoom window* command, define a box by specifying two corner points with the mouse or press *F1* to enter values from the keyboard. **Aerial** returns the quantities $\oint \mathbf{E} \cdot d\mathbf{l}$ and $\oint \mathbf{H} \cdot d\mathbf{l}$, where the circuit integrals are taken around the sides of the box. This command may be useful to find enclosed magnetic flux or displacement current.

The remaining set of commands appears only in the *RF* analysis mode.

NORMALIZE SOLUTION Multiply all field values by a specified factor. This command is useful for normalizing resonant solutions to a desired field level.

SAVE SOLUTION Saved the normalized solution in the standard Aether output format for an *RF* mode solution. For a supplied field prefix, the file is saved with the name `FPrefix.AOU`.

COMPLEX NUMBER FORMAT Set the format for recording complex-number field quantities in response to commands such as *Scan* or *Write matrix file*. The choice are amplitude and phase or the real and imaginary parts of the complex numbers.

The final two commands may be used to identify incident and reflected traveling wave components in transmission lines and waveguides. The commands function only in the *RF* analysis mode. The assumption is that the solution volume includes a section with uniform dimensions that supports waves with wavelength λ .

SET PORT WAVELENGTH

Use this command to set the values of the wavelength λ in the port. Enter the dimensions in units set by *DUnit*. The wavelength equals the vacuum value λ in a transmission line, but is generally longer in a waveguide. It may be necessary to employ a numerical calculation to find λ in a waveguide with a complex shape.

R/I PORT WAVE TOOL

In response to this command, **Aerial** finds incident and reflected traveling-wave components in a port using complex electric field values calculated at two positions along the direction of wave propagation. The wavelength in the transmission line or waveguide must be specified before using this command. Use the mouse or keyboard to specify two locations. The propagation vector must be in the slice plane but need not be parallel to the other axes. The routine displays values for incident and reflected components of electric field on the screen and makes the following entry if a data record is open:

Wave decomposition into incident and reflected parts

```
Point 1
  X:  7.00005E-02
  Y: -2.50000E-01
  Z: -3.95000E+01
Point 2
  X:  7.00005E-02
  Y: -2.50000E-01
  Z: -3.35000E+01
EpsiR:  1.00000E+00
MuR:    1.00000E+00
Lambda:  3.21800E-02 (m)
(L2-L1):  6.00000E-03 (m)
(L2-L1)/Lambda:  0.1865
|ExI|:   4.11814E+01  |ExR|:   7.10560E+01
|EyI|:   1.50825E+05  |EyR|:   1.11912E+05
|EzI|:   3.38532E+02  |EzR|:   3.78687E+03
```

10.4 Vector tools

The plots discussed in Sect. 10.2 show the magnitude of electromagnetic quantities. The commands of this section enable the display of the directions of the vector quantities **E**, **H** and **B**. For RF mode solutions, it is important to note that the commands function only in the *Reference phase* display mode at a specific value of ϕ_s . The vector tools can be used in any plot style (*Region*, *Filled contour*, *Contour* and *Element*).

VECTOR QUANTITY

Choose **E**, **H** or **B**. In the RF mode, you may need to change ϕ_s to display non-zero values of a chosen quantity.

FIELD PROBE

The field probe (shown in Fig. 29) is entertaining and informative. When you click on the tool and move the cursor into the slice-plot area, it changes to a semi-transparent probe that rotates about the blue pivot point to show the direction of the chosen field quantity. The probe display can be combined with field-line traces and scatter plots. The status bar at the lower-left shows the coordinates and magnitude of the vector quantity at the pivot point. The probe disappears inside metal regions. Click the right mouse button or press the *ECS* key to exit the probe mode.

FIELD LINE AT POINTS

Use this command to add projected lines of **E**, **H** or **B** to the plot (Fig. 29). The program enters coordinate entry mode when you click the command. Move the mouse to a point in the solution volume and click the left button. **Aerial** calculates the three-dimensional path of a field line that passes through the point and plots the projection in the slice plane. You can continue to add up to 100 lines. Click the right mouse button or press *ECS* to exit coordinate

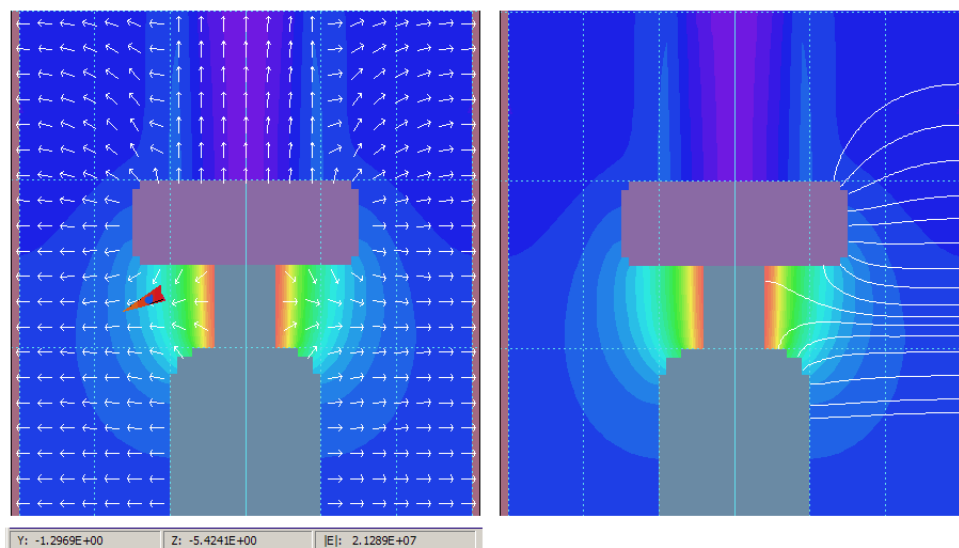


Figure 29: Slice vector tools. Left: field probe and vector scatter plot. Right: field line traces.

mode. The lines disappear if you change slice planes or the view in a plane. Also, the lines are not included in plot exports to a printer or plot file Use a screen capture utility to record them. It is important to recognize the nature of the plot. The lines are three-dimensional curves projected to the plane. They may be difficult to interpret if the line does not lie close to the slice plane. For full three-dimensional field line plots, see the *Field line plot file* command in Chap. 11.

SCATTER PLOT Add vector arrows to the current plot (Fig. 29). The lines are redrawn if you change slices or the view in a slice and they are included in exports to a printer or plot file.

REMOVE VECTORS

Remove vectors from the current plot and turn off the scatter plot mode.

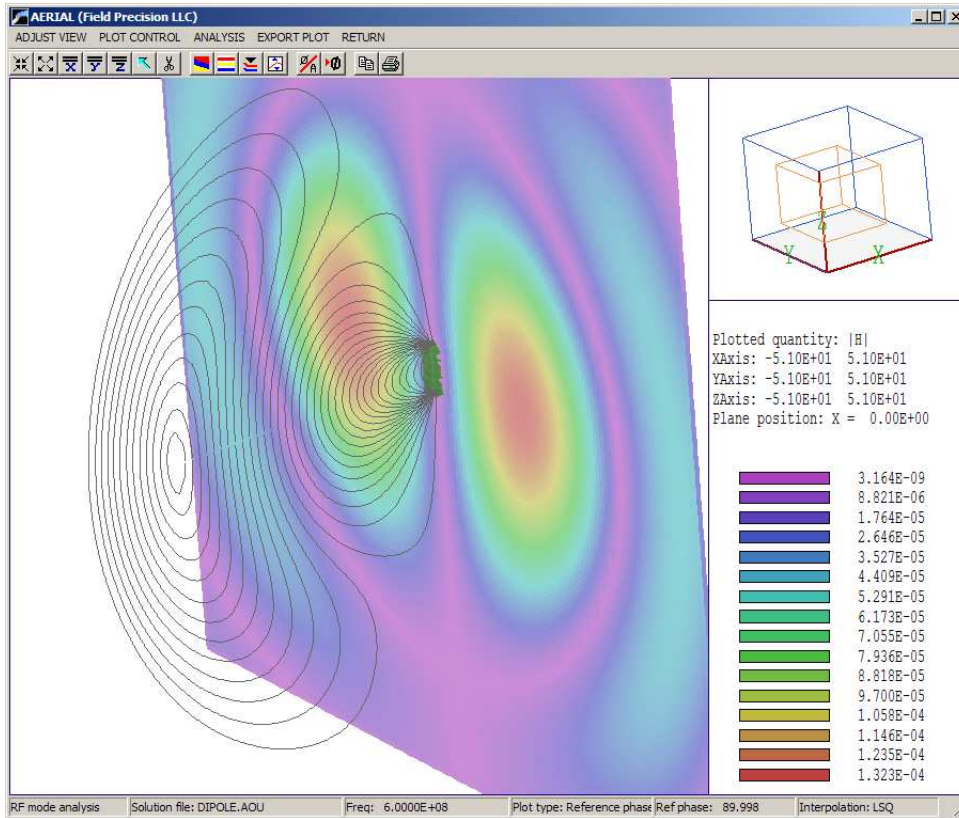


Figure 30: **Aerial** working environment for surface plots. The plot shows color-coding by $|H|$ in a slice plot and three-dimensional lines of E .

11 Aerial – surface plots

Surface plots are three-dimensional views of the solution space (Fig. 30). Four types of information may be superimposed: 1) region boundaries color-coded by region number, 2) region boundaries with color-coding by a computed quantity, 3) computed quantities in a slice plane normal to one of the Cartesian axes and 4) lines of E or B . Surface plots are created directly from the mesh and preserve true spatial scaling.

The method to control the three-dimensional display with the mouse is identical to that used in **Geometer** and **MetaMesh**. Figure 31 shows the active areas of the screen. The central zone (A) is used for zooming in (left button) and out (right button). Hold down the left mouse button in zones B, C, D and E to walk around the object. Hold down the right mouse button in zones B, C, D and E to move the viewpoint to the right, upward, to the left and downward. Note that changes are reflected in the orientation box in the upper-right portion of the screen. The plot is updated when you release the mouse button. You can control additional aspects of the three-dimensional view with the commands of the *Adjust view* popup menu.

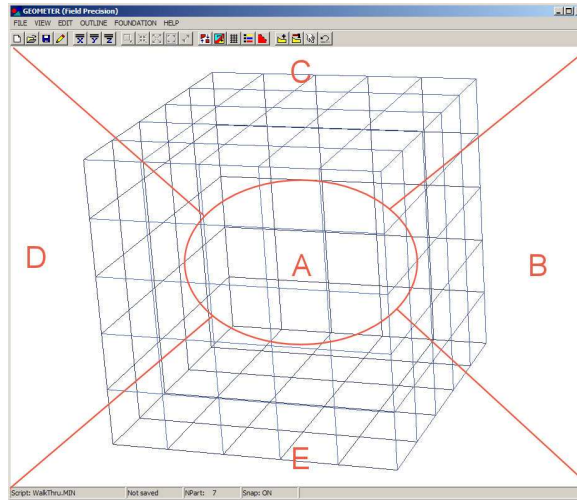


Figure 31: Active areas for mouse control of the 3D view.

SET SURFACE VIEW

This command brings up a dialog (Fig. 32) where you can set specific view angles, displacements and the relative distance to the viewpoint, $DView$. The parameter $DView$ controls perspective. Set it to a large value ($DView \gg 1.0$) for an orthographic view. The minimum value is 1.5.

+X VIEW

+Y VIEW

+Z VIEW

Rotate to viewpoints from the $+x$, $+y$ or $+z$ directions. Origin shifts are not affected

DEFAULT VIEW

This command is useful if you loose your orientation after several rotations and translations. The view is returned to the default: $\theta_x = -30^\circ$, $\theta_y = 0^\circ$ and $\theta_z = 45^\circ$ with the origin at the center.

CENTER VIEW

Remove shifts by setting the origin to the center of the solution volume.

The commands of the *Plot control* popup menu control the appearance of the plot.

PLOT CONTROL

This command brings up the dialog of Fig. 33. The group of commands on the left-hand side controls the plot style. A three-dimensional plot is constructed from color-coded facets. There are two types of facets: 1) element facets on the boundary of a region with color coding by region number or the selected computed quantity and 2) rectangular facets comprising a slice plane with color-coding by the values of the computed field quantity. Slice plane and region boundary information may be superimposed. The *Facet style* radio buttons control whether the

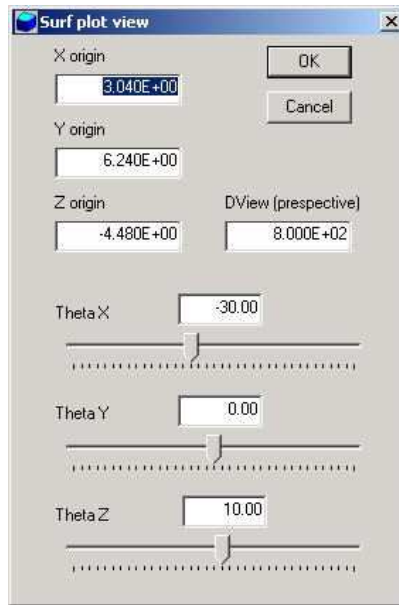


Figure 32: Dialog to set the three-dimensional view.

facets of region boundaries are plotted as solid plates (hidden surface) or wireframe outlines. The *Include facet boundaries* check box determines whether the boundaries of region facets are plotted in the hidden-surface mode. (Note that the facets of the slice plane are always plotted as solids with no outline). **Aerial** plots a reference grid along the boundaries of the solution volume when the *Include reference grid* box is checked. The group of commands on the right-hand side controls the slice plane. The *Include slice plane* box determines whether a slice plot is included. The radio buttons determine the axis normal to the slice plane. You may move the plane along the chosen axis by entering values in the box or moving the slider. In the default mode, the slice plot covers the full normal plane of the solution volume. The remaining fields may be used to set limits in the normal plane.

REGION DISPLAY

The command brings up the dialog of Fig. 34 where you can pick region boundaries to include in the plot. Depress a button in the *Display* column to activate a region. The buttons in the column marked *Field* determine the presentation style for boundary facets. If the box is unchecked, **Aerial** colors facets by region number. If one or more buttons are depressed in the *Field* column, color coding is by the current field quantity. The information window shows the correspondence between color and the quantity. Note that the field quantity is calculated at the center of the element adjacent to the facet **outside** the chosen region. If a boundary separates two regions with different values of ϵ_r , μ_r and/or σ , the field values depend on which region is chosen for display.

PLOT QUANTITY

Pick the quantity that determines color coding in the slice plane and on region boundaries under the *Field* option.

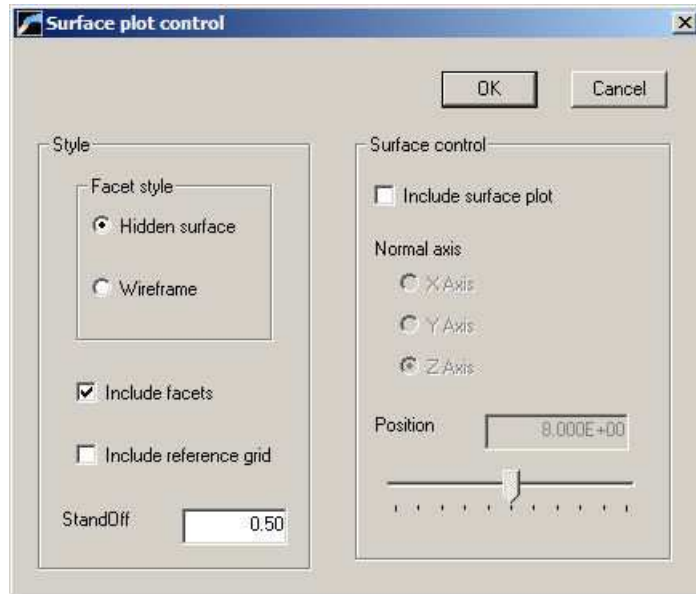


Figure 33: Surface plot control dialog.

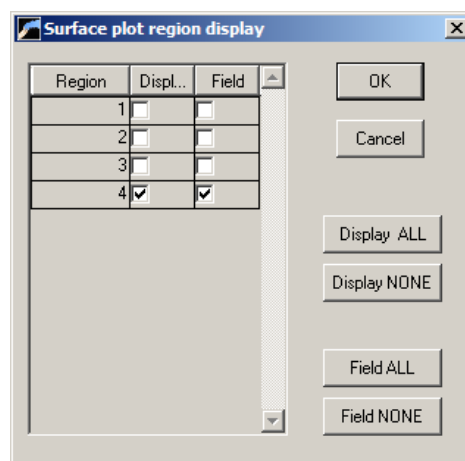


Figure 34: Region display dialog.

PLOT LIMITS

By default, **Aerial** calculates limits on the plotted quantity for both slice plane and region facets and sets the color coding to cover the full range. You can limit the range manually with this command.

REGION CUT PLANES

In a hidden surface plot, internal geometric details or the normal plane plot may be obscured by surrounding region boundaries. This command brings up a dialog that allows you to adjust the displayed portions of region surfaces along the x , y and z axes. **Aerial** does not display facets that lie outside the limits. With this feature you can create cutaway views. The default is that cut limits are set equal to the dimensions of the solution volume so that all facets are included.

The following commands control the superposition of three-dimensional field lines on surface plots. Note that they are active only for *Reference phase* plots in the *RF* mode.

FIELD LINE PLOT FILE

You can add electric or magnetic field lines to three-dimensional plots with this command. To use it, you must prepare a text file that contains any number of point coordinates. **Aerial** reads the file and adds field lines that pass through each point. The file consists of data lines, where each line contains three real numbers (x , y , z) separated by any of the standard delimiters. Enter the coordinates in units set by *DUnit*. The file may contain comment lines that start with an asterisk and should end with the *ENDFILE* command. Lines are not plotted for target points outside the solution volume.

FIELD LINE CUT PLANES

Use this command to set limits in three-dimensional space where field lines will be plotted.

FIELD LINE QUANTITY

The choices are the electric field (**E**), magnetic field (**H**) and magnetic flux density (**B**).

RECORD FIELD LINE PLOT

Use this command to record calculated paths of field lines. **Aerial** prompts for the name of an output text file with a name of the form *FNAME.FLP*. A file extract is shown below. The command functions only if a file of starting points has been opened with the *Field line plot file* command.

```
Magnetic field line plots for file: WGuide010.002
  DUnit(unit conversion factor):  1.00000E+02
```

```
Start point
  X:  8.00000E+00
  Y:  2.50000E-01
  Z:  3.00000E+00
```

Forward

8.00000E+00	2.50000E-01	3.00000E+00
8.00599E+00	2.50385E-01	3.00003E+00
8.01197E+00	2.50768E-01	3.00006E+00
8.01796E+00	2.51151E-01	3.00009E+00
8.02395E+00	2.51532E-01	3.00011E+00
8.02994E+00	2.51912E-01	3.00014E+00

CLOSE FIELD LINE PLOT

Close the field line plot file and remove field lines from the current plot.

12 Aerial – script operation

You can write scripts to call analysis commands automatically. **Aerial** script files have names of the form **SPREFIX.SCR**. An automatic analysis can be initiated from the program main menu using the *File/Run script* command. It is also possible to run **Aerial** in the background. Enter the following line from the command prompt or include it in a batch file:

```
\prospath\aerial SPREFIX
```

The script and any solution files that will be loaded must be available in the current directory.

This chapter describes the command set of **Aerial** scripts. The commands listed below are shown in symbolic form along with an example of how they might appear in a script.

INPUT FileName

INPUT = KLYSTRON.AOU

INPUT = TLINE75.002

Close the current solution file and load a binary file available in the current directory. **Aerial** automatically determines the analysis mode (*Pulse* or *RF*) from the file prefix. A data file must be loaded before performing most script operations.

OUTPUT FPrefix

OUTPUT: WGUIDE1

If necessary, close the current data record file and open a new one. The file has a name of the form **FPrefix.DAT**. A data file must be opened before performing most script operations.

The following four commands function only when an RF solution has been loaded.

FORMAT [Phase Complex]

NFORMAT = Complex

This command sets the format for writing complex-number values in response to the *Point*, *Line*, *Path* and *Matrix* commands. Under the *Phase* option, **Aerial** writes the amplitude and phase (in degrees) of field quantities. Under the *complex* option, the program writes the real and imaginary parts of field quantities.

NORMALIZE NFactor

NORMALIZE: 5.6687

Multiply field values in the currently-loaded file by a real-number normalization factor. This command may be useful for adjusting resonant-mode solutions to a standard amplitude or changing units. The command affects only the values in memory, not the original file.

SAVE FPrefix**SAVE = SolenoidNorm**

Use this command to save normalized values or to make a backup. The file is saved with the name `FPrefix.AOU`.

RITOOOL Xs Ys Zs Xe Ye Ze Lamba**RITOOOL = (0.00 0.00 25.00) (0.00 0.00 27.5) 32.28**

Resolve a general wave in a transmission line or waveguide into incident and reflected parts. The quantity λ is the wavelength in the guide. The two field-calculation points should lie along the direction of wave propagation and be separated by a distance of about $\lambda/4$. Enter all position and length values in units set by *DUnit*.

The remaining commands function in both the *Pulse* and *RF* modes.

NSCAN NScan**NSCAN = 100**

Set the number of intervals for line scans. The default value is $NScan = 100$, the maximum value is $NScan = 1000$.

POINT xp yp zp**POINT = (0.00, 0.05, 4.67)**

Perform field calculations at a point and write the result to the data file. Enter coordinates in the units set by the value of *DUnit* used in the currently-loaded solution.

LINE xp1 yp1 zp1 xp2 yp2 zp2**LINE = (0.00, 0.00, 15.00) (12.00, 0.00, 15.00)**

Perform $(NScan+1)$ calculations along a line in space and write the results to the data file. Enter coordinates in units set by *DUnit*.

PATH PathName**PATH = CircleRad12.PTH**

It is often useful to make a scan of field calculations along paths other than straight lines. With this versatile command, you can make a formatted list of field values along any path in three-dimensional space. The file contains a set of data lines. Each line contains three real numbers in any valid format, the coordinates of a calculation point (x,y,z) . Enter values in units set by *DUnit*. The list must terminate with the *EndFile* command. You may include comment lines (that begin with an asterisk) and blank lines. The file may contain any number of calculation points. The set of calculated values is recorded in the currently-opened data file. The output table lists the components of the electric field, magnetic field and current density along with material properties.

INTEGRALS

Write the integrals of energy, power and power flux shown in Table 8.2 to the data file.

LINEINT xp1 yp1 zp1 xp2 yp2 zp2

LINEINT = (0.00, 0.00, 15.00) (12.00, 0.00, 15.00)

Record the line integrals $\int \mathbf{E} \cdot d\mathbf{l}$ and $\int \mathbf{H} \cdot d\mathbf{l}$ between the start and end points. Enter coordinates in units set by *DUnit*. Both points must be within the solution volume.

MATRIX FPrefix XMin XMax NX YMin YMax NY ZMin ZMax NZ

MATRIX WGUIDE 1.00 1.00 10.00 2.00 2.00 12.00

Write computed field values at an array of locations to a data file in text format. The real-number parameters x_{min} and x_{max} are the limits for calculations along the x axis. Enter the values in units set by *DUnit*. **Aerial** computes values at (N_x+1) evenly spaced points along x . The total number of recorded values is $(N_x+1)(N_y+1)(N_z+1)$. In the RF mode, the number format may be changed with the *NFormat* command. The file *FPrefix.MTX* is created in the current directory.

ENDFILE

Terminate the analysis. You may add annotating text in any format after *EndFile*.

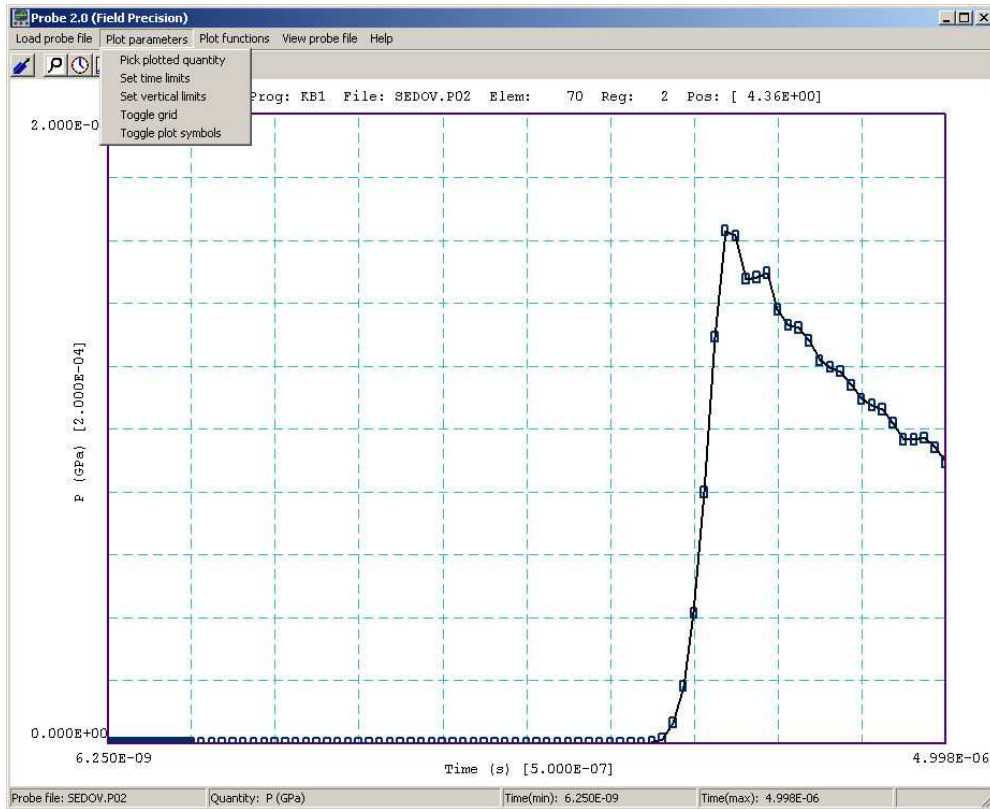


Figure 35: Probe screen shot.

13 Probe – history file plot utility

13.1 Introduction

Probe is the universal plotting program for all Field Precision initial-value solution codes. You can set from 1 to 20 probes by specifying positions in the solution program command script. The probes record quantities in an element or at a node as a function of time. The resulting text files have names of the form `FPREFIX.P01,...`, `FPREFIX.P12`, where `FPREFIX` is the run prefix.

Table 7 shows the standard probe file format. The first section is a header that contains the following information:

- Generating program name.
- Dimensionality of the generating program (1, 2 or 3).
- The spatial position of the probe (from 1 to 3 quantities).
- The index of the element sampled by the probe.
- The region number of the element.

- Conversion factors for the probe position and the recorded quantities.
- Labels for the recorded quantities.

Although the solution programs and their output files employ SI units (meters, kilograms,...), the graphical analysis displays often use practical units to make it easier to visualize results and to facilitate automatic grids. **Probe** multiplies file quantities by the conversion factors during the loading process. Note that the quantity *DConv* and conversion factors for positions are equal to *DUnit*, a variable used in many solution programs. After four lines of label information, the remainder of the file consists of data lines. Each line contains the time (in seconds) and one or more element or node quantities. Real numbers are recorded in E15.6 format.

13.2 Loading data files

When you start **Probe** the only active menu option is *Load probe*. Plotting and analysis functions become active when a probe file has been opened. The program displays a dialog showing all files with suffixes of the form P01,...,P12. Pick a file to analyze and click *OK*. Changing directories in the dialog will change the working directory of the program. If the load is successful, **Probe** creates a default plot of the data (Fig. 35).

The status bar at the bottom of the window contains the name of the probe file, the current plot quantity, and the temporal range of data. The default plot shows the first quantity recorded in the probe file over the full range of time. The horizontal and vertical scales are chosen so that the plot fits on the screen and the grid lines are automatically adjusted so that they lie on even values of the plotted quantity with easily recognized intervals (*e.g.*, 0.02, 0.05, 0.10, ...). The grid intervals are shown in parentheses next to the labels of the horizontal and vertical axes. The title line at the top of the plot shows the following information: generating program, probe file name, element number, region number and position. This information is recorded in hardcopy plots to help you archive your data.

13.3 Plot settings

The commands of the *Plotsettings* menu control the quantities, ranges and appearance of the plot. The screen plot automatically updates whenever you make a change.

PLOTTED QUANTITY

A dialog shows a list of element quantities included in the probe file. Highlight your choice and click *OK*.

TIME LIMITS

By default **Probe** shows the full time-span recorded. You can narrow the range by supplying values for the minimum and maximum time. Uncheck *Autoscale* in the dialog and supply maximum and minimum values. To return to the full range, check the *Autoscale* box.

Table 7: Example of the **Probe** file format

Field Precision probe file

Program: Aether

NDimen: 3

XPosition: 4.200000E-02

YPosition: 0.000000E+00

ZPosition: 2.000000E-02

ElementNo: 79244

RegionNo: 3

NQuant: 10

DConv: 1.000000E+02

QConv1: 1.000000E+00

QConv2: 1.000000E+00

QConv3: 1.000000E+00

QConv4: 1.000000E+00

QConv5: 1.000000E+00

QConv6: 1.000000E+00

QConv7: 1.000000E+00

QConv8: 1.000000E+00

QConv9: 1.000000E+00

QConv10: 1.000000E+00

QConv11: 1.000000E+00

QConv12: 1.000000E+00

QLabel1: Ex (V/m)

QLabel2: Ey (V/m)

QLabel3: Ez (V/m)

QLabel4: Bx (tesla)

QLabel5: By (tesla)

QLabel6: Bz (tesla)

QLabel7: Jx (A/m2)

QLabel8: Jy (A/m2)

QLabel9: Jz (A/m2)

QLabel10: Sigma (S/m)

Time	Ex	Ey	Ez	...
(s)	(V/m)	(V/m)	(V/m)	...
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	...
3.105910E-07	1.724805E-02	1.474984E-01	0.000000E+00	...
...				

VERTICAL LIMITS

In the default mode **Probe** picks a scale to display the full range of the plotted quantity. You can narrow or expand the range by supplying minimum and maximum values. The program returns to full range if you check the *Autoscale* box or if you change quantities using *Pick plotted quantity*.

TOGGLE GRID

Switch between grid and and fiducial lines in the plot.

TOGGLE PLOT SYMBOLS

Include or remove symbols to mark the recorded points.

13.4 Plot functions

The commands of the *Plot functions* menu activate the *Oscilloscope mode* of the program and also send plots to hardcopy devices or plot files.

OSCILLOSCOPE MODE

When you issue this command, **Probe** simulates a digital oscilloscope. As shown in Fig. 36, the mouse cursor changes to a cross-hair pattern when it is inside the plot window. The program adds movable fiducial lines to mark the current point. You can drag the fiducial lines along the time axis by moving the mouse. A box at the bottom of the plot shows values of the time and plotted quantity at the current position. If you press the left mouse button, the program displays a box with the following information about the current point:

- Time, t .
- Value of the plotted quantity, $V(t)$.
- Derivative of the plotted quantity, $dV(t)/dt$.
- Definite integral of the plotted quantity, $\int_0^t V(t')dt'$.

The definite integral is taken from the time of the first recorded value in the probe file to that of the current point. You can find integrals between points by subtracting values. The information is also copied to the Windows clipboard. Other functions of the program are deactivated in the *Oscilloscope mode*. Press the right mouse button or the *Esc* key to return to normal program operation.

SMOOTH DISPLAY

Use this command one or more times to smooth the currently-displayed trace. Smoothing applies to the screen display and exported plots, but does not affect the data values in the probe file.

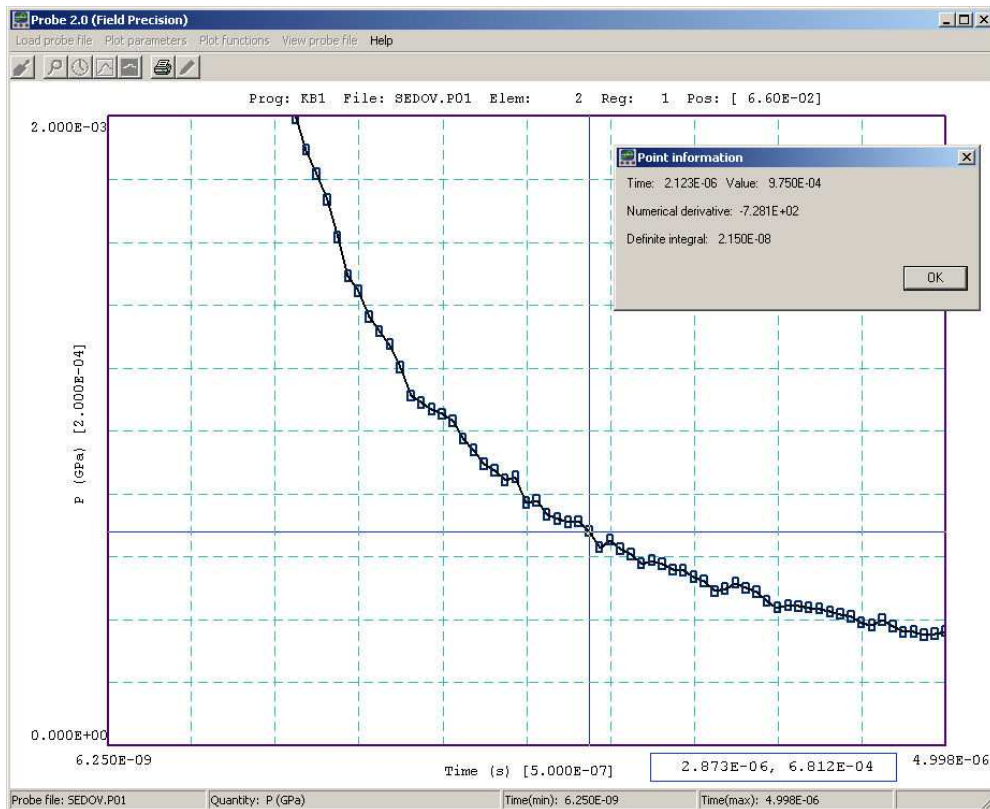


Figure 36: **Probe** in the *Oscilloscope mode*.

DEFAULT PRINTER

Probe can port copies of the plot to any installed Windows printer. The program sends output to the default printer, so be sure to select the correct device using the *Settings/Printer* function of Windows before making the plot.

PLOT FILE (EPS)

PLOT FILE (BMP)

PLOT FILE (PNG)

Send the plot to a file in the following formats: Encapsulated PostScript, Windows Bitmap or Portable Network Graphics. The program prompts for a file prefix and then creates a file with the names FPREFIX.EPS, FPREFIX.BMP or FPREFIX.PNG.

COPY TO CLIPBOARD

Copy the plot to the clipboard in in Windows MetaFile format.

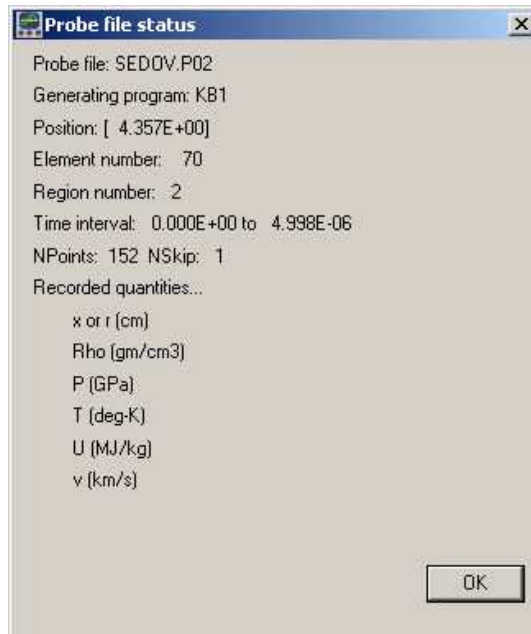


Figure 37: **Probe** file-information message box.

13.5 Information

PROBE FILE INFORMATION

Display information on the probe file in a message box (Fig 37). The quantity *NSkip* in line 7 is used for long files. There is no reason to store more than 1000 points for plots on typical screens and hardcopy devices. When there are less than 1000 data lines, **Probe** loads all points (*NSkip* = 1). When the file contains 1000 to 2000 data lines, the program loads every second point (*NSkip* = 2), and so forth. In this way the **Probe** can handle probe files of any length without exceeding memory limits.

VIEW PROBE FILES

Load a probe file into the internal editor so you can inspect the data directly. The editor runs in read-only mode so that you cannot change the file. Exit the editor to return to program operation.

PROBE MANUAL

Show this document in your default PDF viewer.

13.6 Using Probe with Aether

In response to the *History* command, **Aether** writes standard Probe files in all modes. The recorded quantities are the instantaneous values of E_x , E_y , E_z , B_x , B_y , B_z , J_x , J_y , J_z and σ during the time-domain part of the solution. The current density represents the sum of drive

Table 8: Initial section of **Probe** file created in the *Res* mode

```
Field Precision probe file
Program: Aether
NDimen: 3
XPosition: -5.500000E-02
YPosition:  0.000000E+00
ZPosition:  0.000000E+00
ElementNo:  647683
RegionNo:   2
NQuant: 1
DConv: 1.000000E+00
QConv1: 1.000000E+00
QLabel1: H(f)
```

Frequency (Hz)	H(f)
0.000000E+00	4.848690E-02
2.335408E+07	3.876171E-02
4.670816E+07	4.083475E-02
7.006224E+07	4.953769E-02
...	

and ohmic currents. The time variation of conductivity is of interest in runs where the *SigMod* command appears.

In the *Res* mode, **Aether** writes special files in response to the *Probe* command. They have names of the form *RunName.P51*, ..., *RunName.P55*. During the run, **Aether** records values of time and a chosen field quantity. At the end, the files contain the Fourier transform of the quantity. Table 13.6 illustrates the file format. You can view the results with **Probe**. Note that the horizontal axis labeled *time* in the program should be interpreted as the frequency in Hz.

14 MagWinder – defining applied current

14.1 Program function

Input files of *current elements* may be used in **Aether** to determine drive current densities. Magnet coils are represented numerically by dividing them into a large set of short segments. Sometimes drive coils may be simple, but often they follow complex paths in three-dimensional space. **MagWinder** provides an interactive environment where you can build magnet windings step-by-step. Several features of **MagWinder** help in the task of current-element generation:

- A comprehensive set of parametric models for common coil configurations (solenoids, helices,...).
- Versatile graphical displays to show the state of the assembly.
- Interactive dialogs to modify the geometry, position and orientation of components.

A **MagWinder** session involves three steps: 1) specify a set of parametric models to define one or more drive coils, 2) divide the coils into elements and 3) record the results in a file for input to **Aether**. The program creates two types of text data files:

- A list of model parameters containing all necessary information to regenerate elements. The file has a name of the form `FPREFIX.CDF` (**C**oil **D**efinition **F**ile). You can reload the script into **MagWinder** to make changes in geometry or element lengths.
- The **Aether** element file `FPREFIX.WND`, which represents a specific embodiment of the parametric models. Section [14.13](#) reviews the file organization.

It is important to save the `CDF` file if you want to repeat a calculation or to change an assembly. The script can regenerate an element file, but the `CDF` file cannot be recovered from the contents of the element file. **MagWinder** may be used as a command-line utility with an existing `CDF` file with command of the form:

```
[PATH\]MAGWINDER FPREFIX <Enter>
```

The next sections in this chapter describe how to create `CDF` scripts. The easiest approach is to employ the interactive features of **MagWinder** (Sections [14.2](#) through [14.9](#)). You can also build a script and make changes directly with a text editor. Sections [14.10](#) through [14.12](#) describe the script syntax. Either way, the process is easier if you understand the definitions of a few terms:

- **Element**. A differential element of current familiar from introductory electromagnetism courses. An element extends from vector position \mathbf{x}_s to \mathbf{x}_e . It has length $dl = |\mathbf{x}_e - \mathbf{x}_s|$ and carries a current dI . For accuracy, the length of elements should be shorter than the distance from the coil to the magnetic field calculation point.

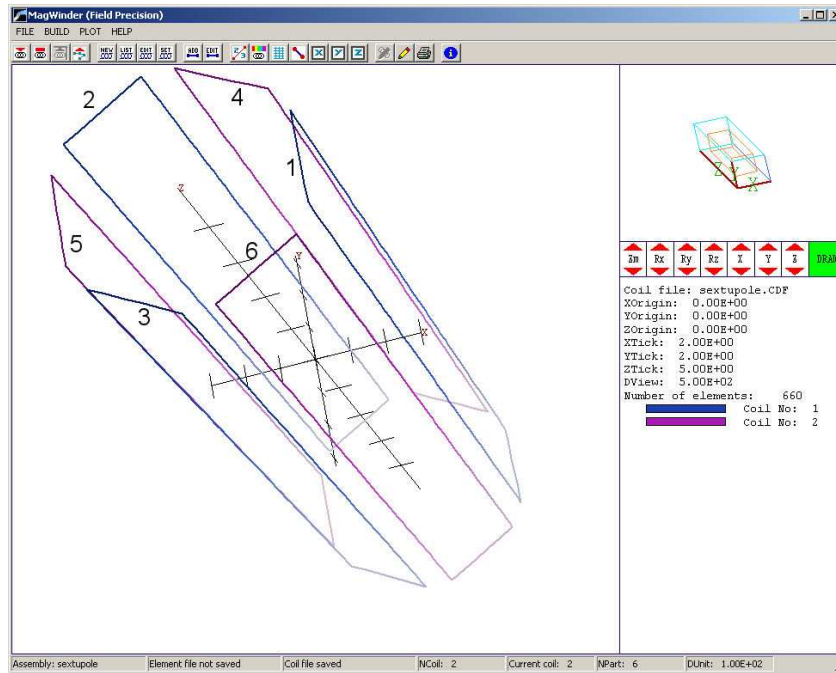


Figure 38: Working environment of **MagWinder** showing the **SEXTUPOLE** example. The loop numbers have been added for clarity.

- **Part.** A basic building block consisting of wires with a specified shape, position and orientation. The shape is defined by the choice of a model. **MagWinder** has fifteen models ranging from simple (line segment between points, circular coil,...) to complex (solenoid with radial thickness, helical coil,...).
- **Coil.** A set of associated parts that carry the same current. For physically-meaningful solutions, the parts should form self-connected circuits.
- **Assembly.** The collection of coils that defines all drive currents for the **Aether** calculation.

For reference, an assembly may contain up to 150 coils. The maximum number of parts to build all coils is 2500. The assembly may contain up to 2,000,000 current elements.

14.2 Starting an assembly

To create a new CDF file, run **MagWinder** and choose the *File/New coil assembly* command. In the dialog, supply a descriptive name that will be used as the default prefix for the CDF and element files. The name should have length from 1 to 50 characters and may not include standard **AMaze** delimiters (*i.e.*, use underscores rather than spaces). You must choose dimensional units that will be used for positions and length. For custom units, pick *Other*. In this case, a dialog will prompt for a conversion factor $DUnit$. The factor equals the number of units per meter. For example, to work in units of feet, set $DUnit = 3.281$. Finally, you have the option to specify a default global element length $DsGlobal$. The length applies to parts in coils that do not have individual values of Ds .

When you exit the dialog, **MagWinder** updates the status bar to show the assembly name and the unit conversion factor. Note that the plot window is initially blank. Plots are generated from elements, and there are no elements until you define at least one coil with at least one part.

We shall use the example shown in Fig. 38 to illustrate the procedure. Six rectangular loops are arranged to generate a sextupole field over a 50 cm length along z . To follow the example in the following sections, start a new assembly with the name **SEXTUPOLE** using dimensions of cm and setting $DsGlobal = 1.0$.

NEW COIL ASSEMBLY

Initialize all variables and begin a new assembly. **MagWinder** issues a prompt if current work has not been saved.

14.3 Adding a coil

Use the *New coil* command to add a coil to an assembly. The command activates the dialog of Fig. 39. Coils are numbered from 1 to 250 as they are added and assigned default names such as **COIL001**. You can change the name to a more descriptive title up to 80 characters in length. You must supply a value for the current – this value applies to all parts that constitute the coil. Optionally, you can enter a local value Ds for element length that applies only to components of the coil. This feature is useful to represent small components in a large assembly.

Entries in the group of boxes in the lower half of the dialog are used to set global values of shifts and rotations that apply to all parts of the coil. These operations are performed after individual shifts and rotations of parts within a coil. With this feature, you can construct a complex set of parts and then move it as a rigid body. Section 14.5 gives a detailed discussion of positioning operations for coils and individual parts.

For the **SEXTUPOLE** example, we shall use a strategy that minimizes the number of positioning operations and reduces the chances of error. Rather than flip the loops to reverse the field direction, we shall use the same relative orientation for all loops and divide them into two sets (*i.e.*, coils) with positive and negative current values. Loops 1, 3 and 5 have $I = +10,000$ A while loops 2, 4 and 6 have $I = -10,000$ A. Accordingly, we set up the first coil with parameters shown in Fig. 39. Click the *New coil* command again, and give the second coil the name **Negative** and assign current $I = -10,000$ A. After you exit the dialog, the status bar shows that the assembly contains two coils with no parts or elements.

NEW COIL

Add a coil to the assembly. The new coil becomes the current coil for editing. You can assign a name, current, local element length and global rotations and shifts that affect all components of the coil. These parameters may be changed using the *Edit coil* command.

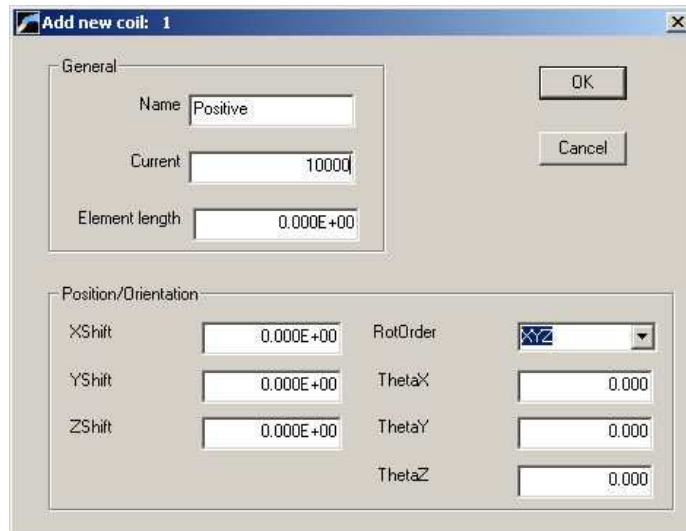


Figure 39: New coil dialog.

14.4 Adding a part

Coil editing operations (such as adding parts) are applied to the *current coil*. You can change the current coil with the *Set current coil* command. The resulting dialog shows a list of defined coils. To choose a coil, press one of the buttons on the right-hand side and then click *OK*.

The *Add part* command brings up the dialog of Fig. 40. The entries in the top group define the model type and parameters. The entries in the bottom group determine the orientation and position of individual parts within a coil. Regarding the top group, all boxes for real-number and integer parameters are initially inactive. The first step is to pick a model type for the part in the drop-down list box at the top. When a model is specified, **Magwinder** activates appropriate parameter fields and adds descriptive labels. Section 14.6 describes the geometries and parameters of available models.

To continue the **SEXTUPOLE** example, click on *Set current coil* and choose the first coil (**Positive**). To define the first rectangular loop, click on *Add part* and choose the **RECTANGLE** model. Enter **LOOP01** in the *Name* field. The loop has narrow dimension 5.0 cm and long dimension 50.0 cm. Enter the following values to define two corners of the rectangle: $x_{c1} = -25.0$, $y_{c1} = -2.5$, $x_{c2} = 25.0$ and $y_{c2} = 2.5$. Click *OK* to exit the dialog. The plot shows a rectangle with the desired dimensions centered in the x - y plane. We must make changes from the default position and orientation to place it in the assembly.

ADD PART

Open a dialog to create a new part in the current coil. Specify the model type and fabrication, position and orientation parameters.

EDIT PART

Pick a part in the current coil and change any of the parameters.

SET CURRENT COIL

Pick the current coil for editing operations and the addition of new parts.

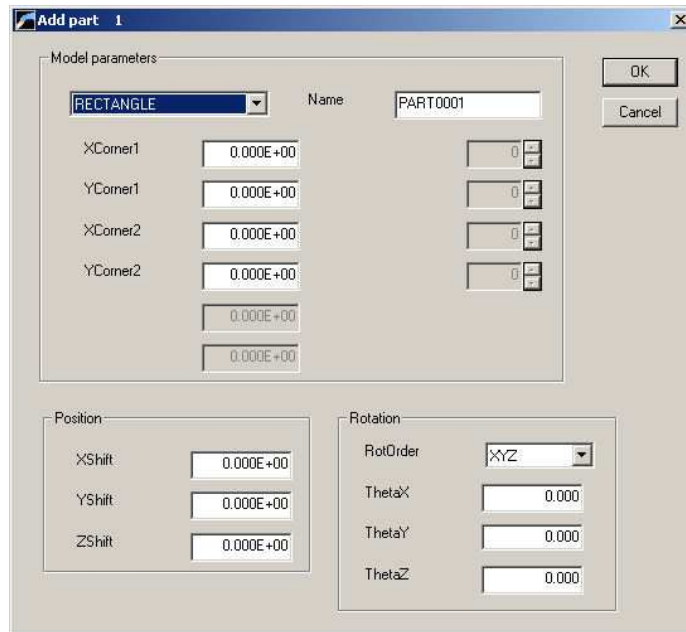


Figure 40: Add part dialog.

14.5 Position and orientation of parts and coils

You can rotate and shift individual parts within a coil and also apply global positioning operations that affect all parts of a coil. The positioning procedure is similar to the operations necessary to construct a physical coil:

- Fabricate the part on the workbench by specifying the model and the associated dimensions. The *workbench frame* is equivalent to the default orientation and position of the model.
- Rotate the part in three dimensions so that it has the the correct orientation relative to the coil.
- Move the part from the workbench to its position in the coil.
- When the coil is complete, you have the option to rotate or shift it.

We can choose items with different features (models) from a standard parts bin and fabricate them by setting dimensional parameters. Simple shapes like line segments may connect any two points in the workbench frame. Complex parts like helices have specific orientations and positions with respect to the workbench. The order of operations is important. In **MagWinder**, rotations always precede shifts.

ROTORDER

You can achieve any orientation in space by making rotations about the x , y and/or z Cartesian axes of the workbench coordinate system. Rotations are not commutative; therefore, the final orientation depends on the order. Use the pull-down list box to specify the order in which rotations are applied. The default is **XYZ**.

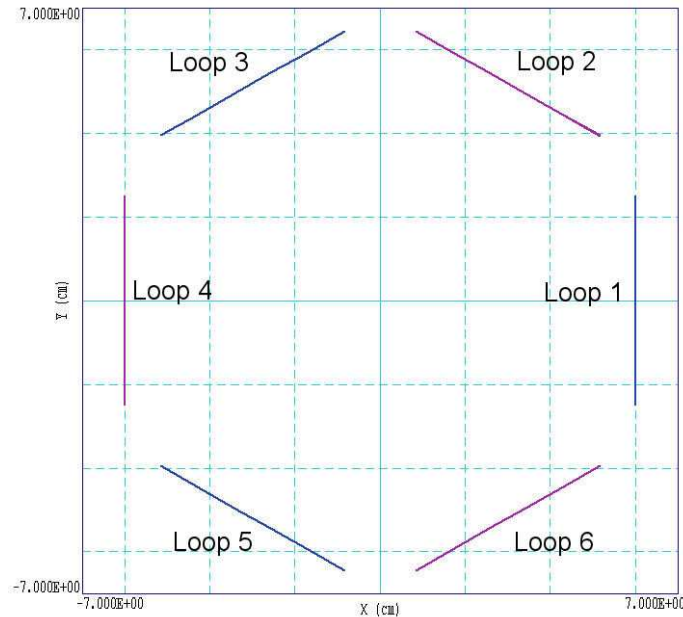


Figure 41: Positions of the six loops in the SEXTUPOLE example. View normal to the z axis.

THETAX
THETAY
THETAZ

Rotation angles about the x , y and z axes. Enter values in degrees. Negative angles are allowed.

XSHIFT
YSHIFT
ZSHIFT

Apply shifts along x , y and/or z to move the part from the workbench to its position in the coil.

To illustrate, we shall position the first loop in the SEXTUPOLE example. Click the *Edit part* command and press the button next to LOOP01. **MagWinder** displays the dialog of Fig. 40. Note that fields contain all currently-defined values. To make the long axis of the rectangle point along z , apply a rotation $\theta_y = 90.0^\circ$ by entering the number 90.0 in the THETAY field. Click *OK* and check the plot to confirm that the part has the correct orientation. Figure 41 shows a scaled view of the configuration in a plane normal to z . The goal is to position the six loops at 60° intervals at a radius $R = 6.0$ cm. To move LOOP01 to its final position, use the *Edit part* command and enter the value 6.0 in the XSHIFT field.

14.6 Part models

The standard parts bin contains fourteen models with which you can construct most practical magnet coils. Enter lengths in units specified by *DUnit* and angles in degrees. All models are defined with respect to the workbench coordinate frame. Their positions and orientations in the simulation space may be adjusted by specifying local or global shifts and rotations.

LINE

The line segment is the simplest and most versatile part. It is a vector that connects any two points in the workbench coordinate system. The model involves six parameters: the coordinates of the start point (x_s, y_s, z_s) and the end point (x_e, y_e, z_e) . A positive value of current flows from the start point to the end. **MagWinder** attempts to cut lines into equal segments with length less than or equal to Ds . For short lines, the minimum number of segments is 2.

RECTANGLE

This model consists of four line segments that define a rectangular loop. A rectangle lies in the x - y plane of the workbench coordinate system at $z = 0.0$. The model has four parameters to define two corners of the rectangle in the x - y plane: (x_1, y_1) and (x_2, y_2) . Current flows in the sense of positive rotation. (*i.e.*, if you point the thumb of your right hand along z , positive current flows in the direction of your fingers.) As with lines, the minimum number of segments per side is 2.

CIRCLE

This model defines a circular coil in the x - y plane of the workbench at $z = 0.0$. The single parameter is the radius R . The circle is centered at position $(x = 0.0, y = 0.0)$. Current flows in the direction of positive rotation. The minimum number of elements in a circle is 12.

ELLIPSE

This command creates an elliptical coil in the x - y plane of the workbench at $z = 0.0$ that follows the curve:

$$\left(\frac{x}{R_x}\right)^2 + \left(\frac{y}{R_y}\right)^2 = 1. \quad (126)$$

The two real-number parameters are R_x and R_y . Positive current flows in the direction of positive rotation. The minimum number of elements is 12.

ARC

The model requires three real-number parameters: R , θ_s and θ_e . The arc has radius R and lies in the x - y plane of the workbench with center point at $x = 0.0, y = 0.0$. The start angle (relative to the x axis) is θ_s and the end angle is θ_e . Positive current flows in the direction of positive rotation.

HELIX

This model is useful to create a circular coil or twisted wire pairs. The model involves four real-number parameters: R , Z_s , Z_e and $Pitch$. The helix winds along z and has a circular projection of radius R in the x - y plane centered at the origin. The helix extends along z from Z_s to Z_e . The quantity $Pitch$ is the axial distance for a full revolution. Therefore, the number of turns is

$$N = \frac{|Z_e - Z_s|}{Pitch}. \quad (127)$$

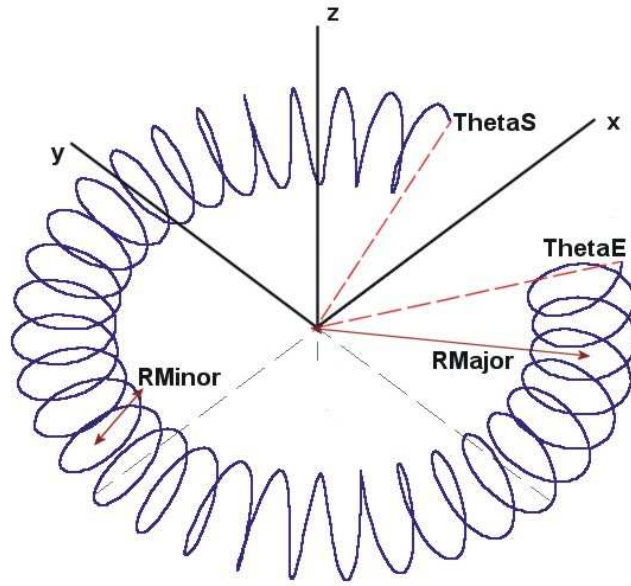


Figure 42: Parameters used in the TORUS model.

By convention, a helix starts at position $(R, 0.0)$ in the x - y plane at $z = Z_s$. The end position in the x - y plane is determined by N . Enter a value in the **ThetaZ** field to change the start position. The minimum number of elements per turn is 12. By default, the helix has a positive sense of rotation as it moves from Z_s to Z_e .

TORUS

The torus (Fig. 42) involves five real-number parameters R_{maj} , R_{min} , θ_{pitch} , θ_s and θ_e and the optional parameter θ_{init} . The toroidal winding creates an azimuthal field. The torus is centered at position $(0.0, 0.0, 0.0)$ with major radius R_{maj} in the x - y plane. Figure 42 defines the parameters R_{min} and θ_{pitch} . The pitch angle (in degrees) is the rotation about the z axis per turn. A TORUS starts at angle θ_s (in the x - y plane relative to the x axis) and ends at θ_e . Current flows in the direction of positive rotation. The minimum number of elements per turn is 12. By default, the winding starts at an angle of 0.0° relative to the plane of the minor radius. You can change this value with the parameter θ_{init} .

TORUSR

This model (Fig. 43) creates an object consisting of a set of loops with rectangular cross section arranged to form a torus. The model involves five real-number parameters (R_{in} , R_{out} , L , θ_s and θ_e) and one integer parameter N . The torus is centered at position $(0.0, 0.0, 0.0)$ with inner radius R_{in} and outer radius R_{out} in the x - y plane. The parameter L is the coil height along z . The part consists of N loops that cover the range from θ_s to θ_e in the x - y plane.

The next group of models is useful for constructing volumetric coils with non-zero cross-section. The models automatically create parallel sets of elements with transverse spacing on

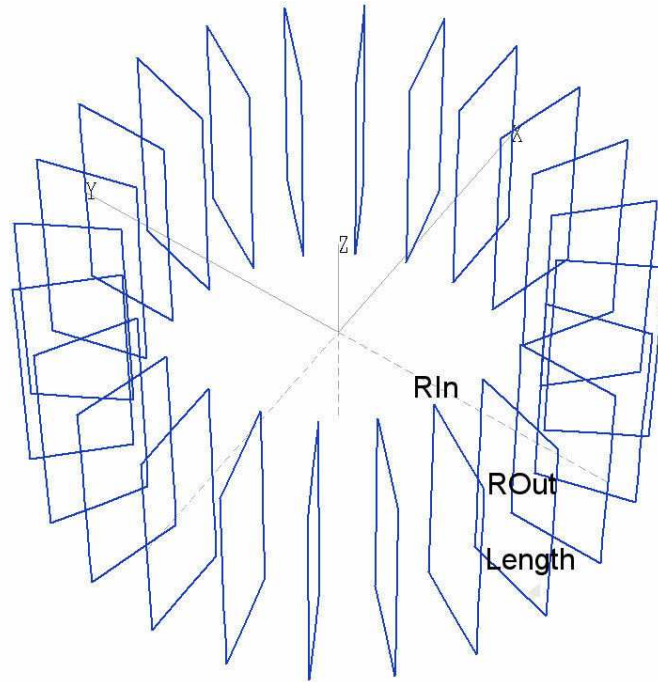


Figure 43: Parameters of the TORUSR model. The example contains 24 loops from 0.0° to 345.0° .

the order of *DsGlobal* or *Ds*. All volumetric assemblies have a fixed orientation and position in the assembly coordinate system. Note that volumetric models may generate large numbers of elements. You can ensure that solutions are efficient by using volumetric assemblies only where they are necessary. For instance, suppose you want to find fields near the surface of a circular coil with a rectangular cross section. It is not necessary to model the entire circular coil using the ELBOWR model if you only need to know fields at a representative position. Instead, define a short azimuthal section of the coil with the ELBOWR model and then fill in the remainder with the ARC model. Because of the $1/R^2$ variation in the Biot-Savart integral of **Aether**, the contribution from a distant coil segment has little dependence on the coil cross section.

SOLENOID The solenoid model (Fig. 44) generates elements to represent a solenoid with radial thickness. It involves four required real-number parameters R_{min} , R_{max} and L and three optional integer parameters N_R , N_Z and N_C . The quantity R_{min} is the inner radius, R_{max} is the outer radius and L is the axial length. The solenoid consists of a set of nested circular loops centered in the x - y plane and arrayed along z . The wires extend from $-L/2$ to $L/2$ in the z direction. There are three optional integer numbers: N_R (number of radial layers), N_Z (number of axial layers) and N_C (number of azimuthal segments in each circular coil). If the numbers are not specified, **MagWinder** makes a choice based on the value of *DsGlobal* or *Ds*. The coil current I_c equals the total number of A-turns in the SOLENOID. Therefore, the current per circular loop is $I_c/(N_r \times N_z)$.

BAR

A BAR is a set of straight wires parallel to z that fill a rectangular cross section in the x - y plane. The model involves three required real-number parameters (L_x , L_y and L_z) and two optional

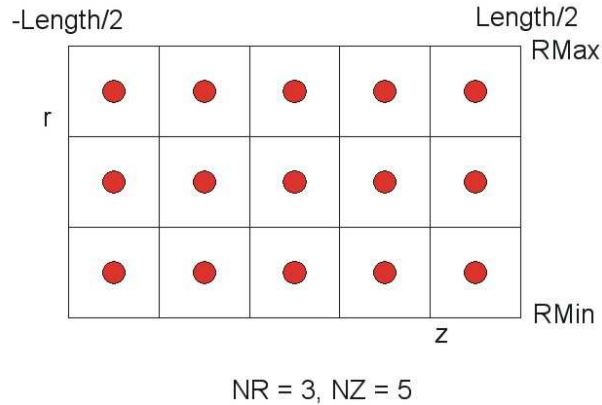


Figure 44: Arrangement of wires in the SOLENOID model.

integer parameters (N_x and N_y). The required parameters are the cross section dimensions (L_x and L_y) and the axial length L_z . The BAR extends from $z = -L_z/2$ to $z = L_z/2$ in the workbench frame. The number of elements along z is determined by *DsGlobal* or *Ds*. You have the option to specify the number of parallel wires by entering values for N_x and N_y . The BAR carries a total current equal to the coil current I_c . Therefore, the current per wire is $I_c/(N_x \times N_y)$.

ELBOWR

The ELBOWR model is an elbow with a rectangular cross section. The elbow is equivalent to a SOLENOID that extends over a limited angular range. There are four required real-number parameters: R_{min} , R_{max} , L and θ . The quantities R_{min} , R_{max} and L are identical to those used for the SOLENOID. The quantity θ is the angular extent (in degrees) in the x - y plane. The ELBOWR extends from the x axis of the workbench frame to θ . There are three optional integer parameters: N_r (number of wires along r), N_z (number of wires along z) and N_θ (number of elements along the arc). Figure 45 shows a combination of the BAR and ELBOWR models to define the drive coil for a C-magnet. The ELBOWR carries a total current equal to the coil current I_c . Therefore, the current per wire is $I_c/(N_r \times N_z)$.

ROD

A ROD is similar to a BAR. The difference is that it has an approximately circular cross section. There are two required real-number parameters: R (radius) and L (length). In this model **MagWinder** sets up a set of parallel line currents in a hexagonal pattern in the x - y plane. The hexagons have scale size on the order of *DsGlobal* or *Ds*. The total current of all wires in the ROD equals I_c .

ELBOWC

The ELBOWC model represents an elbow with a circular cross section. It involves three real-number parameters: R_{out} (outer radius), R_{in} (inner radius) and θ (angular extent in the x - y plane). The ELBOWC model starts at the x axis and extends to θ . You can combine ELBOWC and ROD models to create a continuous set of parallel wires. The total current of all wires in the ELBOWC equals I_c .

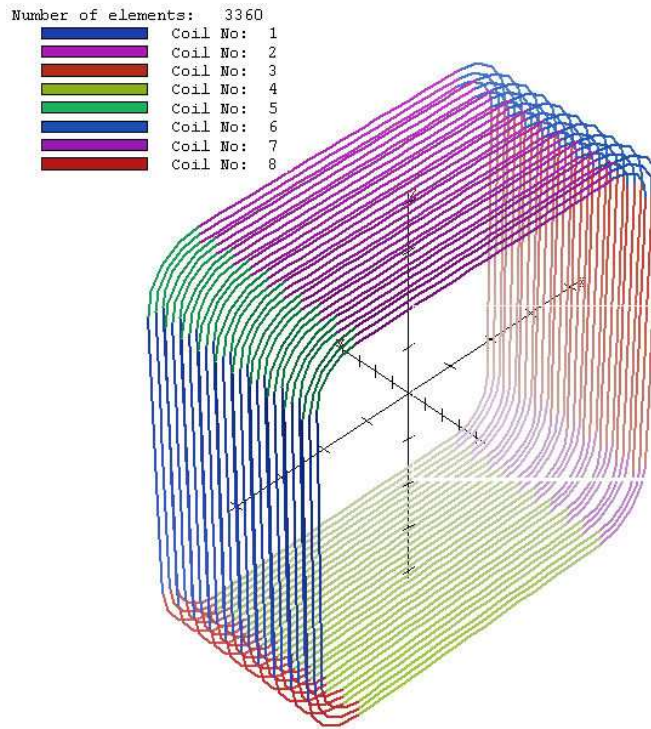


Figure 45: Magnet winding composed of four BAR models and four ELBOWR models.

POLYNOID

A POLYNOID (Fig 46) is similar to a SOLENOID. The difference is that the individual coils are polygons rather than circles. There are four required parameters. The real-number parameters R_{min} , R_{max} and L are identical to those of the SOLENOID. The integer N_s is the number of polygon sides. There are three optional parameters: N_r (number of radial layers), N_z (number of axial layers) and N_e (number of elements per side). The coil current I_c equals the total number of A-turns in the POLYNOID. Therefore, the current per circular loop is $I_c/(N_r \times N_z)$.

14.7 Editing coils and parts

Use the *Edit coil* command to change coil parameters. Changes apply to all existing and future parts associated with the coil. The command brings up the dialog of Fig. 39. The fields contain the present parameter values for the coil. The *Coil parameters* command provides a quick alternative to change selected coil properties.

COIL PARAMETERS

This command brings up the dialog of Fig. 47. The grid has a row for each coil in the assembly. You can modify the entries in the white boxes (name, current and local element width). The *Display* check box determines whether the coil is displayed in 2D and 3D views. (Note that this setting does not affect whether the coil is recorded in the CDF and element files.) Make a choice in the *Select* column and the *Current coil* button to pick the current coil. You can delete a coil and its associated parts by using the *Select* column and the *Delete coil* button.

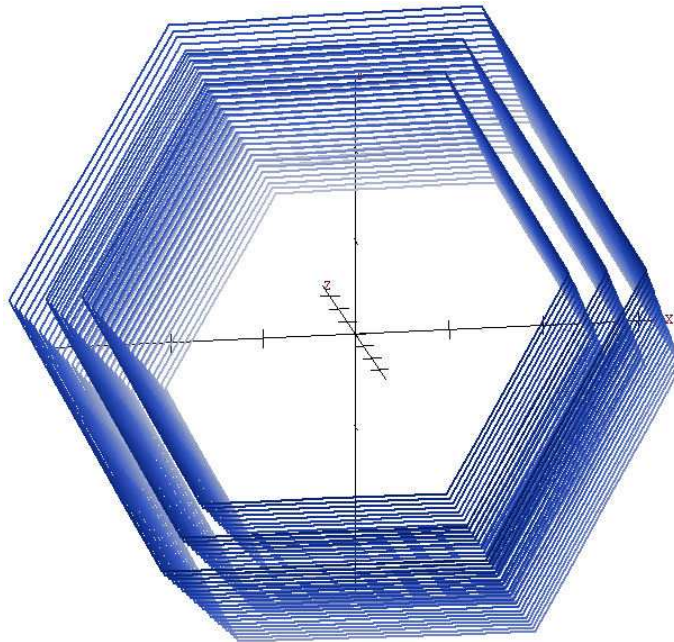


Figure 46: POLYNOID with $R_{max} = 6.0$, $R_{min} = 4.0$, $L = 8.0$, $N_s = 6$, $N_r = 3$ and $N_z = 20$.

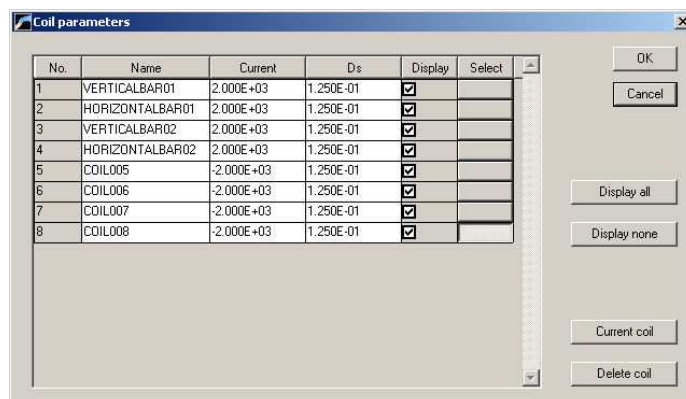


Figure 47: Coil parameters dialog

You can change any part in the current coil with the *Edit part* command. To modify parts in a different coil, first use the *Set current coil* command. The following commands change the properties of parts in an assembly.

DELETE PART

Remove a part from the current coil. **MagWinder** displays a list of available parts. Press the button on the right-hand side for the part you want to delete.

COPY PART

Select a part from the current coil and copy it to a temporary buffer.

CUT PART

Remove a part from the current coil and copy it to a temporary buffer.

PASTE PART

Use the contents of the temporary buffer to create a part in the current coil.

As an example, to move a part to a different coil, cut it from the current coil, change the current coil and then use the paste operation. Note that we have not included copy, cut and paste operations for entire coils because it is easier to perform the operations by direct editing of the CDF file.

The editing operations are particularly useful for the **SEXTUPOLE** example because the assembly is composed of six similar parts with different orientations. Before making further changes, it is a good idea to save the present state of the assembly. Use the *Save coil file* command to record your work. Then, make sure that the current coil is set to **POSITIVE**. Use the *Copy part* command to put **LOOP01** in the buffer. Then use the *Past part* command twice to add two parts to the coil. Note that the plot does not change because the three parts have identical shapes and positions.

Three parts are available when you click the *Edit part* command. Choose the second part. In the dialog, change its name to **LOOP03**. Inspection of Fig. 41 shows that we need to rotate the part by 120° and to move it to the appropriate position. Change the following fields: $ThetaZ = 120.0$, $XShift = 6.0 \cos(120^\circ) = -3.000$, and $YShift = 6.0 \sin(120^\circ) = 5.196$. The loop appears in the correct position when you exit the dialog. Similarly, edit the third part. Assign the name **LOOP05** and apply rotations and displacements for 240° .

To complete the assembly, we need to add parts to the **NEGATIVE** coil. Change the current coil and then use the paste operation three times to create three parts in the second coil. Use the *Edit part* command as before with the following names and rotations: **LOOP02** (60°), **LOOP04** (180°) and **LOOP06** (300°). The plot should be similar to Fig. 38. Save the completed assembly with the *Save coil file* command.

14.8 Displaying an assembly

MagWinder creates plots in two modes:

- Three-dimensional plots (with perspective and shading) provide an overview of the assembly.
- Two-dimensional plots are useful for precise work.

In the 3D mode (Fig. 38), the display is divided into four sections: 1) the main plot area, 2) the orientation area (top-right), 3) the information area (bottom-right) and 4) the control area (middle-right). The orientation area shows the current viewpoint relative to the Cartesian axes. To change the viewpoint, move the mouse cursor into the control area where it changes to a cross-hair pattern. Press the left mouse button in any of the red-arrow areas to make the following changes:

- **Zm.** Zoom or expand the view. The approximate size the view volume is shown in the orientation area.
- **Rx, Ry, Rz.** Rotate about the x , y or z axes.
- **X, Y, Z.** Shift the view point along the x , y or z axes.

Move the mouse cursor into the **Draw** area and click the left button to update the main plot. You can also update by clicking the right mouse button anywhere inside the control region.

Two-dimensional plots (Fig. 41) are created by projecting elements to a chosen Cartesian plane. With the correct window size, the plots preserve true scaling. If you move the mouse cursor into the main plot area, the cursor changes to a cross-hair pattern and **MagWinder** displays coordinate values of the location on the bottom-left part of the window.

The following commands control the plot display.

TOGGLE 2D/3D

Switch between the 2D and 3D plot modes.

DISPLAYED COILS

You can suppress the display of coils for clarity. Use the buttons in the grid box of the dialog to change the display status of individual coils. Note that all coils are recorded in coil and element files, independent of their display status.

GRID CONTROL

Use this command to suppress display of a grid or to change grid parameters from the defaults. The dialog options depend on whether the current plot mode is 2D or 3D. Three-dimensional plots show the Cartesian axes relative to an origin. You can control the following functions in the *Grid control* dialog: 1) suppress the axis display, 2) set the axis origin, 3) suppress the tick display and 4) set values for the tick intervals. In the 2D mode, you can suppress the grid and/or specify grid intervals in the horizontal and vertical directions.

ENDPOINT DISPLAY

Elements are normally plotted as thick lines. In the endpoint mode, **MagWinder** marks the

start and end points of all elements. The mode is useful to check whether element sizes are appropriate.

ARROW DISPLAY

Use this command to show the orientation of current elements in 2D plots. The display is useful to check the polarity of elements of a part when rotations have been applied. Elements are displayed as sperm cells, swimming in the direction of the head. The endpoint displayed is deactivated in the arrow plot mode.

XNORMAL

YNORMAL

ZNORMAL

These commands change the projection plane in the 2D plot mode. In the 3D mode, the commands change the viewpoint to $+x$, $+y$ or $+z$.

INITIALIZE DISPLAY

This command performs the following functions in the 3D mode: 1) set the viewpoint to a reference with 45° azimuth and elevation, 2) set the zoom factor to include the whole assembly and 3) center the display. The command has no effect in the 2D mode.

DEFAULT PRINTER

Send the current plot (2D or 3D) to the default Windows printer. Be sure to set the desired printer before running **MagWinder**.

PLOT FILE (EPS)

PLOT FILE (BMP)

PLOT FILE (PNG)

Send the current plot (2D or 3D) to a plot file in the current directory. Supply a prefix **FPREFIX**. Plots may be created in the following formats: Encapsulated PostScript (**FPREFIX.EPS**), Windows Bitmap (**FPREFIX.BMP**) or Portable Network Graphics (**FPREFIX.PNG**).

COPY TO CLIPBOARD

Copy the current plot (2D or 3D) to the clipboard in Windows MetaFile format.

14.9 Additional MagWinder features

LOAD COIL FILE

Load a CDF file for viewing or editing. You can modify existing coils and parts or add new ones.

SAVE ELEMENT FILE

Create a WND file for input to **Aether**.

EDIT COIL FILE

Use the internal text editor to view or to modify the currently-loaded CDF file. If you make changes, be sure to reload the file with the *Load coil file* command.

VIEW ELEMENT FILE

Use the internal editor in read-only mode to inspect any WND file.

EDIT FILE

Use the internal editor to view or to modify any file.

MAGNUM MANUAL

Display this manual in your default PDF viewer. The file `magnum.pdf` must be in the same directory as `magwinder.exe`.

14.10 Structure of the coil definition file

The **MagWinder** script is a text file that provides a succinct description and permanent record of a coil assembly. You can build scripts using the interactive environment of **MagWinder** or compose them directly with a text editor. Sometimes, the editing approach is more efficient. For example, suppose the assembly consisted of 50 identical coils (each composed of several parts) in a linear array. It is relatively easy to copy the text for the coil section, paste it 50 times and modify *ZShift* values to create the array.

This section covers general features of the CDF file. The script consists of a set of commands processed in sequence. The file follows the standard rules for Field Precision scripts. Blank lines are ignored. You can include any number of comment lines that begin with an asterisk (*). **MagWinder** ignores all information after the *EndFile* command, so you can add annotations in any format. Commands are analyzed with a free-form parser. Sets of characters are grouped into words separated by the following delimiters: Space [], Comma [,], Tab, Colon [:], Equal sign [=], Left parenthesis [(] and Right Parenthesis [)]. Delimiters may be used in any combination, so you have considerable latitude to choose the appearance of your script. For example, the following commands have the same meaning:

```
Fab 2.0 0.5 10.0 0.0 350.0
Fab = (2.0, 0.5) (10.0, 0.0, 350)
Fab: 2.0, 0.5, 10.0, 0.0, 350
```

The CDF file consists of a *Global* section and up to 250 *Coil* sections. Each *Coil* section may contain any number of *Part* sections (as long as the total number of parts does not exceed 2500). Table 9 shows the general file layout.

The *Global* section must appear at the beginning of the CDF file. The following two commands may appear in any order between the commands *Global* and *End*. They are shown in symbolic form and in the form that they might appear in the script.

Table 9: Structure of the CDF file

```
GLOBAL
  (Global commands)
END
COIL
  (Coil 1 commands)
  Part
    (Part 1 commands)
  End
  Part
    (Part 2 commands)
  End
  Part
    (Part 3 commands)
  End
  ...
END
COIL
  (Coil 2 commands)
  Part
    (Part n commands)
  End
  ...
END
...
ENDFILE
```


DUNIT DUnit**DUnit = 1.0E6**

MagWinder works internally in SI units (lengths in meters). It is often convenient to enter dimensions in alternate units. The quantity *DUnit* is a factor to convert input dimensions to meters. It equals the number of units per meter. For example, to enter dimensions in inches, set *DUnit* = 39.37. If the command does not appear, the default is *DUnit* = 1.0.

DS DsGlobal**DS = 0.1**

The quantity *DsGlobal* is the approximate length and diameter of current elements in units set by *DUnit*. Smaller values of element length give more accuracy but result in longer run times. It is good practice to set element lengths explicitly. In the absence of a specification, **MagWinder** will try to pick reasonable values for different part models. When a *Ds* command appears in a *Coil* section, the new value replaces *DsGlobal* for all parts of the coil.

14.11 Coil commands

A coil is a set of wires carrying the same current. **Aether** will calculate the magnetic field corresponding to any set of wires, connected or unconnected. It is your responsibility to ensure that the coil definitions are physically correct. *Coil* sections follow the *Global* section. Commands for a coil section appear between the commands *Coil* and *End*. A *Coil* section may contain any number of *Part* sections and the following commands.

CURRENT Current**CURRENT = -125.0**

A *Current* command must appear in each *Coil* section. The parameter *Current* is the amplitude of the coil current in amperes.

DS DsCoil**DS = 0.100**

The *Ds* command sets the approximate length of current elements for the present coil. It is useful, for example, if you have a large assembly that contains small coils. If the *Ds* command does not appear, the default *DsGlobal* will be used.

NAME CoilName**NAME = Orbit_Correction_Coil_015**

MagWinder numbers coils in the order they appear in the file. You can also assign a descriptive name (from 1 to 80 characters). The name will be displayed if you load the CDF file into the **MagWinder** interactive environment.

ROTATE ThetaX ThetaY ThetaZ [RotOrder]**ROTATE 0.0 90.0 45.0 ZXY**

This command specifies global rotation angles for all parts of the coil about the Cartesian axes

of the assembly coordinate system. Rotations can be performed about the x , y or z axes. The parameters $ThetaX$, $ThetaY$ and $ThetaZ$ are the respective angles in degrees. The optional string parameter $RotOrder$ controls the order in which rotations are performed. The default is XYZ .

SHIFT XShift YShift ZShift
SHIFT -10.0 0.0 5.65

This command controls global translations of all parts of the coil. The parameters $XShift$, $YShift$ and $ZShift$ are the components of the displacement vector. Enter the displacements in the units set by $DUnit$.

To clarify, rotations and shifts are performed in the following order:

1. An individual part is rotated relative to its default position in the workbench frame to the proper orientation in the coil.
2. The part is shifted to its position in the coil.
3. Global rotations are applied to coil parts relative to their position in the coil.
4. Global shifts are applied to the coil parts.

14.12 Part commands

A *Coil* section may contain any number of *Part* sections. A *Part* section has the form:

```
PART
  (Part commands)
END
```

The following commands may appear within a *Part* section.

ROTATE ThetaX ThetaY ThetaZ [RotOrder]
ROTATE 0.0 90.0 45.0 ZXY

This command specifies local rotation angles to orient the part in the coil. Rotations are performed about the x , y or z axes. The parameters $ThetaX$, $ThetaY$ and $ThetaZ$ are the respective angles in degrees. The optional string parameter $RotOrder$ controls the order in which rotations are performed. The default is XYZ .

SHIFT XShift YShift ZShift
SHIFT -10.0 0.0 5.65

This command controls local translation of the part to its position in the coil. The parameters $XShift$, $YShift$ and $ZShift$ are the components of the displacement vector. Enter the displacements in the units set by $DUnit$.

Table 10: FAB command parameters

Type	P1	P2	P3	P4	P5	P6	P7
LINE	x_s	y_s	z_s	x_e	y_e	z_e	
RECTANGLE	x_{c1}	y_{c1}	x_{c2}	y_{c2}			
CIRCLE	R						
ELLIPSE	R_x	R_y					
ARC	R	θ_s	θ_e				
HELIX	R	Z_s	Z_e	Pitch			
TORUS	R_{maj}	R_{min}	θ_{pitch}	θ_s	θ_e	$[\theta_{init}]$	
TORUSR	R_{in}	R_{out}	L	θ_s	θ_e	N_c	
SOLENOID	R_{min}	R_{max}	L	$[N_r]$	$[N_z]$	$[N_\theta]$	
BAR	L_x	L_y	L_z	$[N_x]$	$[N_y]$		
ELBOWR	R_{min}	R_{max}	L	θ	$[N_r]$	$[N_z]$	$[N_\theta]$
ROD	R	L					
ELBOWC	R_{min}	R_{max}	θ				
	R_{min}	R_{max}	L	$[N_s]$	$[N_r]$	$[N_z]$	$[N_\theta]$

TYPE

Set the model for the part. The options (discussed in Section 14.6) are LINE, RECTANGLE, CIRCLE, ELLIPSE, ARC, HELIX, TORUS, TORUSR, SOLENOID, BAR, ELBOWR, ROD, ELBOWC and POLYNOID. In addition, the LIST type (described below) must be defined by direct entry in the script.

FAB

Set real-number and/or integer parameters for the model. The number of entries and the interpretation of values depends on the model type. Table 10 summarizes the choices.

The LIST part type is useful if you cannot form a shape from the available models or if you have your own mathematical specification for a path in three-dimensional space. In this case, the *Type* command takes the form:

```

TYPE List
  x0 y0 z0
  x1 y1 z1
  ...
  xN yN zN
END

```

The effect is to create a set of N contiguous elements from (x_0, y_0, z_0) to (x_N, y_N, z_N) . Each element carries the coil current in the direction from the start point to the end. Enter the coordinates in units set by *DUnit*. Each line contains three real numbers in any valid format separated by any of the valid delimiters (i.e., space, comma, tab, ...). The list may contain any number of data lines as long as the maximum number of elements (2,000,000) is not exceeded. A script may contain a maximum of 12 lists.

Table 11: Initial section of a MagWinder current-element file

Magnum Current Element File (Field Precision, Albuquerque NM)

NCoil: 2
NElem: 660

Coil No	I (A)
1	1.0000E+04
2	-1.0000E+04

Coil No	XStart (m)	YStart (m)	ZStart (m)
1	5.9980E-02	-2.5000E-02	2.5000E-01
1	5.9981E-02	-2.5000E-02	2.4000E-01
1	5.9982E-02	-2.5000E-02	2.3000E-01
1	5.9983E-02	-2.5000E-02	2.2000E-01
1	5.9983E-02	-2.5000E-02	2.1000E-01

...

	XEnd (m)	YEnd (m)	ZEnd (m)	I (A)
	5.9981E-02	-2.5000E-02	2.4000E-01	1.0000E+04
	5.9982E-02	-2.5000E-02	2.3000E-01	1.0000E+04
	5.9983E-02	-2.5000E-02	2.2000E-01	1.0000E+04
	5.9983E-02	-2.5000E-02	2.1000E-01	1.0000E+04
	5.9984E-02	-2.5000E-02	2.0000E-01	1.0000E+04

...

14.13 Structure of the current element file

Table 11 shows a portion of the current-element FPREFIX.WND created by **MagWinder** for input to **Aether**. The header lists the number of coils (*NCoil*) and the number of elements (*NElem*). The coil section lists the current in amperes for each coil. The element section contains *NElem* data lines. Each line contains the associated coil number, the coordinates of the start and end points (in meters) and the element current. Note that the current of an element may be less than the coil current if the element is constructed from a volumetric model such as a SOLENOID. Current flows from the start point to the end point.

15 Output file formats

This chapter documents the format of **Aether** space files, binary records of field quantities in elements of the solution volume. In the *Pulse* mode, the program may create a series of space files at specified times. The files have names of the form `RUNNAME.001`, `RUNNAME.002`,.... The suffix numbers are in chronological order. The initial header quantities are I_{max} , J_{max} and K_{max} . The 4-byte integers are the maximum mesh node numbers along x , y and z . The header concludes with two 8-byte real (double precision) numbers: $DUnit$ (the length unit conversion factor) and t (the recording time in seconds).

The next set of $(I_{max}+1)$ 8-byte real numbers are node positions along the x axis: x_0, x_1, x_2, \dots . They are followed by $(J_{max}+1)$ values for the y axis node positions and $(K_{max}+1)$ values for the z axis. Element field values listed in the remainder of the file. Values are arranged according to the following loop through elements of the solution volume:

```
DO K=1,KMax
  DO J=1,JMax
    DO I=1,IMax
      . . .
    END DO
  END DO
END DO
```

The following quantities are recorded for each element:

- N_{reg} . The region number of the element (4-byte integer)
- E_x, E_y, E_z . Components of the electric field in V/m (8-byte real).
- H_x, H_y, H_z . Components of the magnetic field in A/m centered in space and time (8-byte real).
- J_x, J_y, J_z . Components of the current density in A/m². The time-centered values represent the sum of drive and ohmic current density (8-byte real).
- ϵ, μ and σ . The dielectric permittivity ($\epsilon = \epsilon_0 \epsilon_r$), magnetic permeability ($\mu = \mu_0 \mu_r$) and electrical conductivity (σ) in the element (8-byte real).

For each element, the file contains one 4-byte integer and twelve 8-byte real numbers.

The single output file created from an *RF* mode solution has a name of the form `RUNNAME.AOU`. The header is similar to that of the *Pulse* mode. The difference is that the RF frequency f_0 (in Hz) is recorded instead of the time. The node section and element loop order are the same. The following quantities are recorded for each element:

- N_{reg} (4-byte integer).
- $E_x[\text{real}], E_x[\text{imag}], E_y[\text{real}], E_y[\text{imag}], E_z[\text{real}], E_z[\text{imag}]$ (8-byte real).

- $H_x[\text{real}], H_x[\text{imag}], H_y[\text{real}], H_y[\text{imag}], H_z[\text{real}], H_z[\text{imag}]$ (8-byte real).
- $J_x[\text{real}], J_x[\text{imag}], J_y[\text{real}], J_y[\text{imag}], J_z[\text{real}], J_z[\text{imag}]$ (8-byte real).
- ϵ, μ and σ .

The real and imaginary parts of the field and current density components are included, giving the amplitude and phase. For each element, the file includes one 4-byte integer and twenty-one 8-byte real numbers.

Index

- absorbing boundary, [16](#), [35](#), [44](#)
- Aerial, [6](#), [44](#)
 - analysis script, [59](#), [86](#)
 - data file, [60](#)
 - loading solutions, [59](#)
 - making plot files, [69](#)
 - matrix file, [64](#)
 - numerical analysis, [74](#)
 - plane plot type, [66](#)
 - plane versus slice plot, [66](#)
 - plot quantities, [67](#)
 - printing plots, [69](#)
 - slice plot type, [71](#)
 - surface plot controls, [80](#)
 - surface plot type, [80](#)
 - text editing, [60](#)
- Aerial commands
 - Circuit integral, [77](#)
 - Close data file, [60](#)
 - Complex number format, [77](#)
 - Copy to clipboard, [69](#)
 - Create matrix file, [64](#)
 - Create script, [60](#)
 - Default printer, [69](#)
 - Default view, [81](#)
 - Edit data file, [60](#)
 - Edit file, [62](#)
 - Edit script, [60](#)
 - Expand view, [72](#)
 - Field line at points, [78](#)
 - Field line cutplanes, [84](#)
 - Field line plot file, [84](#)
 - Field line quantity, [84](#)
 - Field probe, [78](#)
 - Format, [86](#)
 - Global view, [72](#)
 - Input, [86](#)
 - Instruction manual, [62](#)
 - Integrals, [87](#)
 - Jump, [72](#)
 - Line, [87](#)
 - Line integral, [63](#), [76](#)
 - Line scan, [75](#)
 - LineInt, [88](#)
 - Load named view, [69](#)
 - Load solution file, [59](#)
 - Matrix, [88](#)
 - Normalize, [86](#)
 - Normalize solution, [64](#), [77](#)
 - NScan, [87](#)
 - Number format, [64](#)
 - Number of contours, [73](#)
 - Open data files, [60](#)
 - Oscilloscope mode, [76](#)
 - Output, [86](#)
 - Pan, [72](#)
 - Path, [87](#)
 - Plot limits, [68](#), [73](#), [84](#)
 - Plot quantity, [67](#), [73](#), [82](#)
 - Plot style, [67](#), [73](#)
 - Point, [87](#)
 - Point calculation, [74](#)
 - R/I port wave tool, [77](#)
 - Record field line plot, [84](#)
 - Reference phase, [69](#)
 - Region cut planes, [84](#)
 - Region display, [82](#)
 - Remove vectors, [79](#)
 - Rotate plot, [68](#)
 - Run script, [59](#)
 - Save, [87](#)
 - Save named view, [69](#)
 - Save plot file, [69](#)
 - Save solution, [65](#), [77](#)
 - Scan quantity, [75](#)
 - Scatter plot, [79](#)
 - Set number of scan points, [76](#)
 - Set plane, [67](#)
 - Set port wavelength, [77](#)
 - Set series, [59](#)
 - Set slice plane properties, [71](#)
 - Set snap distance, [74](#)
 - Set view dialog, [81](#)
 - Slice normal, [72](#)
 - Solution file information, [59](#)
 - Solution integrals, [62](#)

- Step, [72](#)
- Surface plot control, [81](#)
- Toggle grid display, [74](#)
- Toggle reference/amplitude, [69](#), [74](#)
- Toggle snap mode, [74](#)
- Vector quantity, [78](#)
- Zoom in, [72](#)
- Zoom window, [72](#)

Aether, [6](#)

- absorbing boundary, [16](#)
- algebraic expressions, [20](#), [40](#)
- classes of solutions, [4](#)
- computational modes, [7](#)
- convergence, [48](#), [57](#)
- current element file, [96](#), [116](#)
- current sources, [40](#), [55](#)
- file format, [117](#)
- learning, [7](#)
- magnetic field calculation, [39](#)
- material properties, [42](#)
- MetaMesh rules, [32](#)
- output files, [6](#)
- probe, [48](#)
- pulse mode, [32](#)
- resonant mode, [46](#), [50](#), [55](#)
- RF mode, [52](#)
- script example, [36](#), [49](#), [56](#)
- script format, [36](#)
- space files, [44](#)
- stable time step, [15](#)
- standard pulses, [20](#), [41](#)
- steps in a solution, [5](#)
- tables, [18](#), [41](#)
- time files, [44](#)
- time variations, [18](#)
- using Probe, [94](#)

Aether commands

- AbsLayer, [44](#)
- Courant, [39](#)
- Dt, [33](#), [38](#)
- DTime, [35](#), [44](#)
- DUnit, [32](#), [37](#)
- EddyCurrent, [39](#)
- Epsi, [42](#)
- Freq, [49](#), [55](#)
- History, [45](#)
- Interp, [38](#)
- Jx,Jy,Jz, [40](#)
- Mesh, [37](#)
- Metal, [44](#)
- Mode, [37](#)
- Mu, [43](#)
- NHStep, [45](#)
- NPeriod, [55](#)
- NStep, [44](#)
- Probe, [50](#)
- RunTime, [38](#)
- SetTime, [45](#)
- SFile, [40](#)
- Sigma, [43](#)
- SigMod, [43](#)
- SMod, [35](#), [41](#), [55](#), [57](#)
- Source, [49](#)
- SymBound, [38](#)
- TMax, [33](#), [38](#)
- Vacuum, [44](#)

- algebraic expressions, [20](#), [40](#)
- complex number, [29](#)
- conformal mesh, [10](#)
- continuity, equation of, [9](#)
- Courant stability condition, [15](#), [33](#)
- current density, [10](#)
 - defining, [40](#)
 - spatial variation, [40](#)
- current element, [96](#)
- delimiters
 - standard, [36](#)
- dielectric permittivity, [9](#), [42](#)
 - spatial variation, [43](#)
- eddy current limit, [39](#)
- electric field
 - difference equation, [12](#)
 - energy, [30](#), [62](#)
- electrical conductivity, [9](#), [42](#)
 - spatial variation, [43](#)
 - time variation, [43](#)
- electromagnetic energy, [31](#)
- element
 - definition, [5](#)

- FETD method, 4
- finite-element method, 5, 10
- Fourier transform, 7, 25, 46
 - properties, 26
- frequency-domain solutions, 4, 29
- Gaussian pulse, 20
- Geometer, 5
- HiPhi
 - input file, 7
- impedance match, 16
- magnetic field
 - difference equation, 14
 - energy, 30, 62
- magnetic permeability, 9, 42
 - spatial variation, 43
- MagWinder, 6
 - adding parts, 99
 - assembly, 97
 - assembly display, 108
 - begin assembly, 97
 - coil, 97
 - coil file structure, 111
 - editing parts, 106
 - element, 96
 - element file, 33, 40
 - functions, 96
 - model parameters, 115
 - output files, 96
 - part, 97
 - part models, 101
 - part orientation, 100, 114
 - part position, 100
 - volumetric parts, 103
- MagWinder commands
 - Add part, 99
 - Arc, 102
 - arrow display, 110
 - Bar, 105
 - Circle, 102
 - Coil parameters, 106
 - Copy part, 108
 - Current, 113
 - Cut part, 108
 - Delete parts, 108
 - Displayed coils, 109
 - Ds, 113
 - DUnit, 113
 - Edit coil file, 111
 - Edit part, 99
 - ElbowC, 105
 - ElbowR, 105
 - Ellipse, 102
 - Endpoint display, 110
 - Fab, 115
 - Grid controls, 109
 - Helix, 103
 - Initialize display, 110
 - Line, 102
 - List, 115
 - Load coil file, 110
 - Name, 113
 - New coil, 98
 - New coil assembly, 98
 - Paste part, 108
 - Polynoid, 106
 - Rectangle, 102
 - Rod, 105
 - Rotate, 114
 - Save element file, 110
 - Set current coil, 99
 - Shift, 114
 - Solenoid, 104
 - Toggle 2D/3D, 109
 - Torus, 103
 - TorusR, 103
 - Type, 115
 - View element file, 111
- materials
 - absorber, 44
 - linear, 9
 - metal, 44
 - properties, 9, 35, 42
 - vacuum, 44
- Maxwell equations
 - differential, 9
 - integral, 10
 - linear, 9
- mesh generation, 5
- MetaMesh, 5

- regions, [42](#)
 - using with Aether, [32](#)
- modulated functions, [23](#)
- Nyquist limit, [50](#)
- Ohm's law, [10](#)
- parametric pulses, [20](#)
- phase, definition, [29](#)
- phasor, [29](#)
- plot quantities, [67](#)
- Plot views, saving, [69](#)
- power density, [30](#), [62](#)
- power spectral density, [26](#)
- Poynting vector, [30](#), [62](#)
- Probe, [44](#), [89](#)
 - file format, [89](#)
 - file information, [94](#)
 - graphical environment, [90](#)
 - loading files, [90](#)
 - oscilloscope mode, [92](#)
 - plot export, [93](#)
 - plot settings, [90](#)
 - smooth display, [92](#)
 - use with Aether, [94](#)
- pulse
 - frequency spectrum, [27](#)
- Pulse mode, [7](#), [32](#)
- raised cosine function, [23](#), [50](#)
- real-number format, [36](#)
- regular mesh, [10](#)
- Res mode, [7](#), [46](#)
- resonance
 - frequency, [46](#)
- RF mode, [7](#), [52](#)
- sinc function, [22](#)
- Slice plot styles, [73](#)
- surface resistance, [16](#)
- tabular functions, [18](#), [41](#)
 - periodic, [41](#)
- time-domain solutions, [4](#)
- Vector tools, [78](#)
- voltage integral, [77](#)
- volume resistivity, [42](#)