100 Elwood Davis Road ♦ North Syracuse, NY 13212 ♦ USA

# SonnetLab Current Data Exporter






**Specialists in High-Frequency Electromagnetic Software**
(315) 453-3096 Fax: (315) 451-1694 http://www.sonnetsoftware.com

# Introduction

The current exporter Matlab toolbox is designed to give Sonnet users the ability to export current data information from Sonnet. All current data is exported as an easily readable CSV file.

The current exporter framework may be used as a standalone utility or used in conjunction with Sonnet's Matlab toolbox (SonnetLab). The interoperability layer between SonnetLab and the current exporter framework simplifies the process of extracting current data from projects already loaded in Matlab.

The current exporter framework includes a data reader which will load current data file information into Matlab structures. This simplifies the process of using exported current data for Matlab calculations.

The SonnetLab current exporter can be used to do any of the following:

- Export current data from an entire layout
- Export data from a particular region of a layout (Ex: rectangular region)
- Export current data from one/several/all metallization levels of a stackup
- Export current data from any set of analysis frequencies
- Specify the port excitations for any/all ports of the layout
- Import current data from CSV files into Matlab
- Plot current data to a Matlab figure

# Requirements

The current exporter framework requires Sonnet 13 and SonnetLab version 4.0 or later. The current exporter may be used with any of the Sonnet Suite levels. Before using the current exporter users will need to add the folder of scripts to their Matlab path.

# General Instructions

There are two steps required in order to export current data from Sonnet projects. First the user must make a current data output configuration file and second they must execute a method that performs the request.

Current data output configuration files specify how current data should be outputted. Output configuration files specify options such as the region of a layout that current data should be returned from, the excitation of ports, etc.

A single current data output configuration may be used for multiple iterations of a project design. For example if a user is optimizing a design and is generating a large number of similar project files they may be able to use a single output configuration for the entire set of project files.

The advantage of output configuration files is that a project's current data may be exported with a variety of options without the project being re-simulated.

## *Design an Export Configuration File*

One or more current data exports may be completed simultaneously using SonnetCurrentRequest objects. Adding multiple current data request configurations to a SonnetCurrentRequest object may come in handy when users would like to export data from a circuit using several sets of port excitation settings or if they would like to export data from multiple regions of a circuit. Most users will not need to interact with request configuration objects directly; the SonnetCurrentRequest provides methods for adding current data export configurations to a request object.

The simplest way to add an export configuration to a request is the following (which utilizes no optional arguments):

```
Request.addExport(theFilename,theLabel,theRegion,theType,thePorts,theFr
equency,theGridX,theGridY);
```

The most complicated form of the method is the following which uses all available optional arguments.

```
Request.addExport(theFilename,theLabel,theRegion,theType,thePorts,theFr
equencies,theGridX,theGridY,theLevel,theComplex,theParameterName,thePar
ameterValue)
```

The filename should be the desired name of the CSV file. The label is a unique identifier specified by the user to make it easier to identify exports (Ex: 'Original Design'). The region should be one of the following three types:

1. Empty Matrix ([ ]) – Indicates that data from the entire box area should be returned.
2. Rectangle – Exports data from a rectangular region of the box.
   Rectangles objects can be instantiated with the following command:

   ```
   SonnetRectangle(theLeft,theRight,theTop,theBottom)
   ```

   The arguments specify the planes that bound the rectangle. The left and right edges are maximum and minimum X coordinates and the top and bottom edges are the maximum and minimum Y coordinates.
3. Line – Exports data along a vertical or horizontal line within the box.
   Line objects can be instantiated with the following command:

   ```
   SonnetLine(isVertical,thePosition)
   ```

   The first argument is a Boolean that specifies if a vertical line or a horizontal line is desired. If the line is vertical then thePosition is the desired X value for the line; if the line is horizontal then thePosition is the desired Y value for the line.

The type should be either 'JX', 'JY', or 'JXY' to indicate the type of data that should be returned. The ports argument should be a vector of SonnetJxyPort() objects. The SonnetJxyPort() constructor takes the following arguments:

```
SonnetJxyPort(thePortNumber,theVoltage,thePhase,theResistance,theReacta
nce,theInductance,theCapacitance)
```

The port number selection indicates which port(s) will have the specified voltage, phase, resistance, reactance, inductance, and capacitance values. The port selection may be either an integer to indicate an individual port or 'All' to indicate that all ports should have the specified settings.

Alternatively JXYPorts can be generated based on a project's existing geometry port. The JXYPort's values for resistance, inductance, capacitance, reactance, and inductance will be copied from the geometry port. The JXYPort will have values of zero for its voltage and phase. An example of how to construct a JXYPort using a geometry port is presented below:

```
SonnetJxyPort(aGeometryPort)
```

The frequency selection should be a vector of frequency values that data should be returned for (units are Hz).

The X and Y grid sizes specify the resolution of the output. The grid size indicates the separation between sample points in the specified direction. For example a grid size of two will result in data being available at 2 mil separated positions (1, 3, 5, 7, 9 …). The sequence of points always starts at half the grid size; that is why in the two mil size example starts at one.

Users can optionally specify what metallization level(s) should be exported. The optional level argument may be in one of the following formats:

1. Empty Matrix ([ ]) – Indicates that data from all metallization levels should be outputted.
2. Integer – Indicates that only data from one metallization level should be outputted (Ex: 4 will only output data from metallization level four).
3. [StartLevel, EndLevel] – Indicates a range of levels that data should be exported for (Ex: [0, 2] will export data from metallization levels zero, one and two).

Another optional argument for addExport() is a Boolean that indicates whether the current data should be exported as a complex number. A value of true indicates that the data should be complex and a value of false indicates that the data should not be complex.

The final pair of optional arguments must be either both specified or neither specified. These arguments allow users to specify particular values for the project's parameters. The specified parameter values will be used for the current calculation but the original project file will remain unmodified. Simulation data must be available for the specified parameter value(s); this can easily be accomplished by using a parameter sweep to simulate the project. The parameter names should be either in the form of a vertical vector of strings or a cell array. A vertical vector of strings can be generated using Matlab's strvcat method (it is unnecessary to use strvcat when only one parameter is being modified). The parameter values should be either a vector of numeric values or a cell array. The $n^{th}$ value of theParameterValue should correspond to the parameter specified by the $n^{th}$ value of theParameterName.

Once the user has added all the desired export configurations to the request the user may save it as an XML file using the following command:

```
Request.write('Filename.xml');
```

The XML file can be used to describe the output configuration of any number of files. To get the current data for a particular circuit the user simply needs to call the ExportCurrents method and specify the XML configuration file and the Sonnet project. Please see the section titled "Export Data Using an Output Configuration File" for information regarding how users can obtain current data for a project using an output configuration. Output configurations can be directly passed to Sonnet project objects in Matlab; see the section titled "Integration with SonnetLab" for more information regarding how to use an existing output configuration file with a SonnetLab project object.

## *Export Data Using an Output Configuration File*

Existing output configuration files can be used to describe the output configuration of any number of Sonnet project files. To get the current data for a particular circuit the user simply needs to specify the XML configuration file and the Sonnet project.

```
JXYCurrentExport(RequestObjectOrFile, ProjectObjectOrFile);
```

The request file object can either be the name of an appropriate XML file on the hard drive or a SonnetCurrentRequest object that exists in Matlab. The project may either be the name of a Sonnet project file on the hard drive or a Sonnet project object existing in Matlab.

JXYCurrentExport() can optionally also accept a version number which indicates which version of Sonnet should be used to generate the current data file. The version of Sonnet should be at least thirteen. Current data exports are not available with Sonnet version 12.

```
JXYCurrentExport(RequestObjectOrFile,ProjectObjectOrFile,Version)
```

JXYCurrentExport() will temporarily modify the project such that Sonnet only simulates the project at the frequencies specified by the request object/file. The JXYCurrentExport() method will also temporarily enable current calculations for the project. The Sonnet project file will be reverted back to its original settings after the current data has been exported.

## *Read Current Output Data*

When Sonnet exports JXY data it does so to a CSV file. The CSV file is a numeric grid with the data from individual exports displayed separately. The current exporter framework includes a reader which will read the current data into Matlab and store the data in an array of structures. The current data reader can be called to parse a data file with the following command:

```
CurrentData=SonnetCurrentReader(theFilename);
```

Sonnet will export the current data for each level and each frequency value separately. The current data reader will make a separate structure for every combination of levels and frequency values; users can access the data they want by finding the structure in the array that contains data from the desired level and frequency. A description of each of the fields in the output structure is presented in the table below:

| Field Name | Description |
|---|---|
| **Label** | The user specified name for the export (Ex: 'Original Design') |
| **ProjectFilename** | The file name of the project (Ex: 'Project.son') |
| **DataFilename** | The name of the CSV file that the export data was extracted from (Ex: 'Project.csv') |
| **Frequency** | The frequency value for this export in Hz (Ex: 1e10) |
| **Level** | The metallization level number that generated the data (Ex: 0) |
| **Type** | The type may be 'JX', 'JY' or 'JXY' to specify the type of current data stored |
| **Data** | Stores the current data. The value at Data(X, Y) is the current information at the location (XPosition(X), YPosition(Y)). |
| **XPosition** | Stores the X coordinate values where current data values exist. |
| **YPosition** | Stores the Y coordinate values where current data values exist. |

For example suppose we wanted to extract the current data at location (45, 88.5) (The current exporter does not use an inverse grid like Sonnet does. A Y value of zero is at the bottom of the box). This can be accomplished by finding the indexes of XPostion and YPosition which correspond to the desired location and then access the data field at those indexes.

```
>> find(CurrentData(1).XPosition==45)
ans =
    23
>> find(CurrentData(1).YPosition==88.5)
ans =
   454
>> CurrentData(1).Data(23,454)
ans =
   21.3429
```

Because the current data is computed at a particular resolution, data will not be available at every possible coordinate value in the box range. For example if there is one X value per mil of a 100 mil wide box then there will be current data values at locations 1, 2, 3, 4, …, 45, 46, 47, …, 98, 99, 100; but there will be no data at location 45.5 or 46.7. If a user would like to receive current data at an uncalculated point they may either increase the current export resolution or use Matlab's interpolation routines to arrive at a close value.

## *Integration with SonnetLab*

The current exporter can be called from SonnetLab with SonnetLab's built in exportCurrents method. There are two ways to use the exportCurrents() method: the first is to specify an output configuration file that specifies the current export options and the second way is to pass the appropriate values to the method such that it will build a temporary output configuration file.

The syntax for calling the exportCurrents() with a preexisting output configuration file is the following:

```
aData=Project.exportCurrents(Filename);
```

The current data will be exported to a file with the same name as the project but with the extension ".csv" rather than ".son". The current data will be automatically read by the method and return an array of current data structures as described in the section titled "Read Current Output Data".

Instead of specifying an output configuration file users may instead specify values directly. When taking this approach the exportCurrents() method will take the following arguments.

```
1) Region - Must be either a SonnetLine object, a SonnetRectangle
    object or []. If the region is [] then the currents for the entire
    layout will be outputted.
2) Type - Must be either 'JX','JY', or 'JXY'.
3) Ports - The ports should be either a vector of JXYPort objects or a
    matrix that stores the voltage and phase values for each port. The
    user only has to define values for ports that have non-zero voltage
    or phase values. When using a matrix the data must be formatted as
    follows: [ PortNumber, Voltage, Phase;
               PortNumber, Voltage, Phase; ... ]
4) Frequencies - The frequency values. Values should be specified in
    the same units as the project.
5) X Grid Size - This determines the separation between sample points
    in the X direction.  A grid size of two will result in data being
    available at 2 mil separated positions.
6) Y Grid Size - This determines the separation between sample points
    in the Y direction.  A grid size of two will result in data being
    available at 2 mil separated positions.
7) (Optional) Level - Specifies what metallization level(s) should be
    outputted. The level selection should be [] if all levels should be
    outputted. The level value should be a single number (Ex: 4) if only
    one level should be outputted. If a range of levels should be
    outputted then the level value should be a vector in the form of
    [startLevel, endLevel].
8)  (Optional) Complex - Should be either true or false. If the user
    would like to specify values for parameters they may use the last
    two arguments.
9)  (Optional) ParameterName - Should be a vertical vector of strings
    (use strvcat) that represent the names of the parameters whose
    values will be specified in the tenth argument.
10) (Optional) ParameterValue - Should be an equal length vector as
    ParameterName which provides a value for every specified parameter
    such that ParameterValue(n) is the value for ParameterName(n,:).
    Simulation data must already exist for the specified values for the
    variables. This is often used to select a particular value set used
    by a parameter sweep.
```

The syntaxes for the SonnetLine constructor, the SonnetRectangle constructor and the SonnetJxyPort constructor are presented in the section labeled "Design an Export Configuration File."

When exporting current data using the SonnetLab method the user may import the values for resistance, inductance, capacitance, reactance, and inductance from each of the existing ports in the project. The user will still need to supply the method with a matrix that specifies the voltage and phase values that should be used to excite the ports. The matrix should be in the following form:

```
[ PortNumber, Voltage, Phase;
  PortNumber, Voltage, Phase;
  PortNumber, Voltage, Phase;
  PortNumber, Voltage, Phase; ... ]
```

The arguments for exportCurrents() will effectively make a SonnetCurrentRequest object and use it to export the current data. The label field of the output structure will be the project's filename. Using the exportCurrents() method in the above manner may be simpler than creating a custom output configuration file for a single Sonnet project.

# Additional Framework Tools

The current exporter framework has several additional tools that demonstrate ways in which users can use the current exporter and provide additional functionality.

## *Current Data Plot*

The JXYPlot() method provides users with a function that will graphically plot a layout's current data in a Matlab figure. Areas of the figure that have a high amount of current will appear as dark red and areas with a small amount of current appear as dark blue. The magnitude of the current data has the units of Amps/Meter. Users may interact with the figure using the standard Matlab figure tools including the data cursor. The data cursor will make it simpler to find the magnitude of the current data at a particular point in the figure. The current plotter does not have support for complex current data.

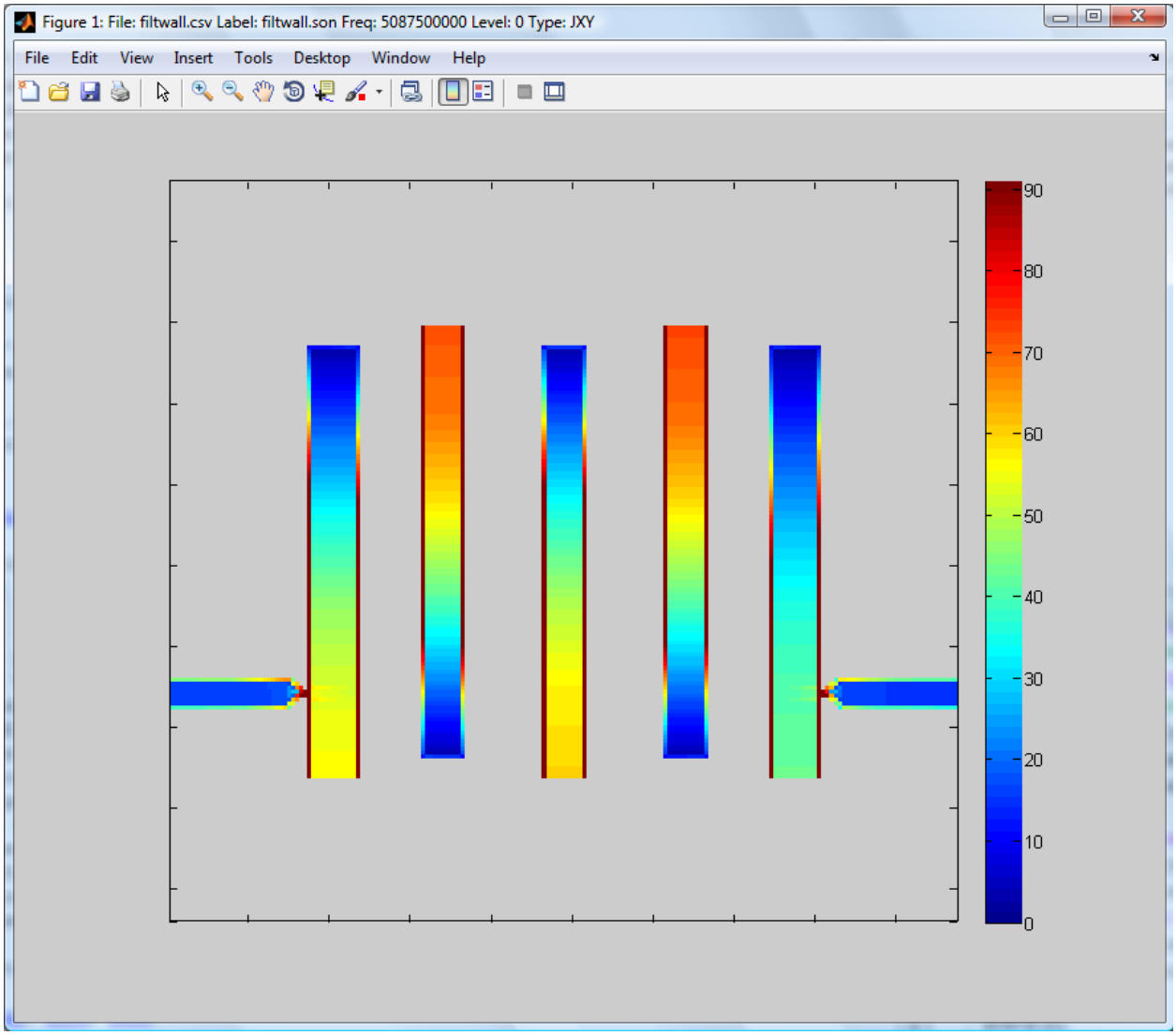
Figure 1: File: filtwall.csv Label: filtwall.son Freq: 5087500000 Level: 0 Type: JXY
File Edit View Insert Tools Desktop Window Help
90
80
70
60
50
40
30
20
10
0