File SFFILES.DOC contains the following bookmarks. Select the topic you are trying to find from the list and double click the highlighted text.

AutomeshInputFile
BinarySolutionFile
CCDTLcell_PMIfile
CommentsInFiles
DTLcell_PMIfile
EGUNdataFile
FCEfileForForce
FieldPlotsInPandira
FieldPlotsInPoisson
FilesAndFilenameConventions
FilesForOtherCodes
FileSummaryTable
FishScan_TBL
HardCopyOfGraphicsScreens
InputFilesForCodesAfterAutomesh
OpeningInputFiles
OUTAUT.TXT
OUTFIS.TXT
OUTFOR.TXT
OUTPAN.TXT
OUTPOI.TXT
OutputTextFiles
OUTSF7.TXT
OUTWSF.TXT
PandiraInputForParmela
Parmela2DfieldMaps
PlotFiles
PlotFilesInSF7
PoissonInputForParmela
RegisteredFileTypes
SEGFileForSFO
SF7_InputFile
SF7inputForParmela
SFOouputFile
T36File
TAPE35
TAPE35.TXT
TAPE36
TAPE37
TBETA_TBL
TRANSIT_TBL
TransitTimeFileForParmila
UserSuppliedCurveForSF7
WSFplot.PRF

File SFFILES.DOC Table of Contents

I. Files and Filename Conventions

This section describes the input and output files associated with the Poisson Superfish codes. In the Introduction we discuss general properties of the files, naming conventions, and methods for launching programs. Later sections contain some details about the file contents.

A. Introduction

The most important file associated with a Poisson Superfish problem is the Automesh input file, which defines the problem geometry and actions taken by later programs. Section B gives a brief discussion of this file. The Automesh chapter discusses the contents of the input file in more detail. The automated tuning programs create an Automesh input file for each problem.

Some files created by Automesh and subsequent codes will have names based upon the Automesh input file. For example, if you start Automesh with input file PROB1.AM, then the name of the solution file will be PROB1.T35. Default input filenames for postprocessors SFO, SF7, Force, and SF8 follow this pattern. These codes replace the extension of the Automesh input file with the default extension appropriate to the code. The code looks for this file first. If that file cannot be found, the code then looks for the original filename you supplied. This search order is just the opposite in Automesh and Autofish. They first look for the original filename as entered (with or without an extension), and then they add the default extension if there was none to start with.

Using fixed filenames for some output files and temporary files minimizes needless clutter on the disk. Filenames of the form OUTxxx.TXT are standard output files. Program Fish writes OUTFIS.TXT, Poisson writes OUTPOI.TXT, and so on. Some temporary files (e.g., TAPE36 and TAPE37) contain data saved for use later by the same program. For example, Automesh saves data in file TAPE36, and rereads it after processing the user-supplied boundary regions.

1. Summary of the input and output files

Table I-1 lists files used by the codes. Each code’s default filename extension corresponds to the extension included as part of a filename in the Input files column. For example, AM is the default extension for Automesh input files. The string XXXX represents the root name of an Automesh input file, and nn refers to a sequence number added by the code. Names in parentheses are files created by the program for later use by itself or another program. Names in *italics*, such as the WSFplot *CurveFile*, either have no default name or the code creates the name. Programs create the name of the *TAPE40* file from the current disk and directory names. Files that have a default extension, but whose root name need not be the same as the Automesh input file appear in the table as Filename.EXT, where EXT is the default extension.

The binary solution file appears in Table I-1 as XXXX.T35. Programs that read and write the solution file read and write a text file in the local directory named TAPE35.INF. This file lists, for each type problem (Superfish or Poisson), the names of the last solution files created by Automesh and the last solution file accessed by any other code. Another file, SF.PRF in the LANL directory, lists similar information for every Poisson Superfish program. Each TAPE35.INF contains local information about the directory it is in, while information in SF.PRF includes drive and path information. In the next section, we will discuss the order in which codes may access these files and how they use the information in the files upon starting up.

Table I-1. Files used in Poisson Superfish.

Program	Input files	Output files
Automesh,	XXXX.AM, XXXX.T36	OUTAUT.TXT, XXXX.T35, (TAPE36, TAPE37)
Autofish	XXXX.AF, XXXX.T36, XXXX.SEG	OUTAUT.TXT, OUTFIS.TXT, XXXX.T35, XXXX.SFO, XXXX.PMI, (TAPE36, TAPE37, TAPE40)
Fish	XXXX.T35	OUTFIS.TXT, FishScan.TBL , XXXX.T35, (TAPE40)
CFish	XXXX.T35	OUTFIS.TXT, FishScan.TBL , XXXX.T35, (TAPE40)
Poisson	XXXX.T35	OUTPOI.TXT, OUTPOI.TBL, XXXX.T35, XXXX.PO7, EFLD.QKP, BFLD.QKP
Pandira	XXXX.T35,	OUTPAN.TXT, OUTPAN.TBL, XXXX.T35, XXXX.PA7, EFLD.QKP, BFLD.QKP, (TAPE40)
SFO	XXXX.SEG, XXXX.T35	Transit.TBL, TBETA.TBL, XXXX.T35, XXXX.SFO, XXXX.PMI, TBETAnn.TBL
SF7	XXXX.IN7, XXXX.T35	OUTSF7.TXT, XXXX.nn.TBL, XXXXnn.EGN, XXXXnn.T7
Force	XXXX.FCE, XXXX.T35	OUTFOR.TXT
WSFplot	XXXX.T35, (WSFplot.PRF), CurveFile	OUTWSF.TXT, WSFplot.PRF, numerous hardcopy graphics file types
SegField	Filename.SGF, Filename.SFO as named in SGF file	Tablplot file named in the input SGF file
SFOtable	Filename.SFT, series of files Filename.SFO as named in SFT file	Tablplot file named in the input SFT file
SF8	Filename.IN8, two T35 solution files as named in IN8 file	OUTSF8.TXT
List35	XXXX.T35	XXXX.TXT
Quikplot	Filename.QKP	numerous hardcopy graphics file types
Tablplot	Filename.TBL	numerous hardcopy graphics file types
ABCfish (tuning codes)	Filename00.ABC, where ABC stands for CCL, DTL, etc.	Filename01.ABC, Filename00.LOG, plus files XXXX.AM, XXXX.T35, XXXX.SFO for each problem solved. May also include XXXX.PMI files.

2. Naming files

Poisson Superfish input files can have long filenames and long directory names in the complete path to a file. Names of files reported in standard output files generally include the complete path. However, there is some dependence on how the code is started. If you start from a Windows tool (e.g., My Computer, Explorer, etc.), then output files will report long filenames. If you start a code from a batch file or a DOS tool (including CMD.EXE, COMMAND.COM), then some output files will use DOS 8-character abbreviations for long filenames and long directory names.

Spaces in filenames and directory names are allowed, but not recommended. Never use filenames that contain trailing spaces. If a name contains spaces and you wish to supply that name to a program in a batch file, the name must be enclosed in double quotation marks and separated from other command-line parameters by spaces.

Many users have become accustomed to working with the old DOS “8.3” filename and extension conventions. If you plan to switch back and forth between Windows tools and DOS tools, we recommend keeping filenames short. When working with the tuning programs, use filename prefixes of 6 characters or less so that after a two-digit sequence number has been appended the filename does not exceed 8 characters.

3. Registered file types and right-button menus

The Setup program registers several file types for convenience in starting Poisson Superfish programs from Windows tools such as My Computer or Explorer. (Registry keys under HKEY_CLASSES_ROOT define the properties of the file types.) Registered extensions appear in Table I-2 along with the program that starts by double-clicking on the file. For example, double click on a file with extension AM to launch Automesh automatically.

Notice that Automesh and Autofish input files (extensions AM and AF) are really the same file. Setting the extension to one or the other value determines which code starts upon double clicking the file. You can use the Run command to launch either program on an input file with either of the two extensions.

Holding the right mouse button down on files with extensions in Table I-2 brings up a menu of possible actions. For most file types, the menu includes the Open, Edit, and Print commands. The Open selection launches the appropriate code. The Edit option brings up the file in Notepad. The Print selection uses Notepad to print the file. For Poisson Superfish solution files (.T35), the menu includes a Run command for each solver program (see Table I-3), an option to list the contents (using the List35 program), and the Interpolate option, which runs the SF7 field interpolator. The Open command and double-clicking on the T35 file starts the plotting code WSFplot.

Table I-2. Registered file types.

Extension	Registered name	Opens program
.AM	Automesh input	Automesh
.AF	Autofish input	Autofish
.T35	Poisson Superfish solution	WSFplot
.SEG	SFO input	SFO
.IN7	SF7 input	SF7
.IN8	SF8 input	SF8
.FCE	Force input	Force
.CCL	CCLfish control file	CCLfish
.CDT	CDTfish control file	CDTfish
.ELL	ELLfish control file	ELLfish
.DTL	DTLfish control file	DTLfish
.MDT	MDTfish control file	MDTfish
.RFQ	RFQfish control file	RFQfish
.SCC	SCCfish control file	SCCfish
.QKP	Quikplot input	Quikplot
.TBL	Tablplot input	Tablplot
.SGF	SegField input	SegField
.SFT	SFOtable input	SFOtable

Table I-3. Right mouse button menu for T35 files.

Menu item	Description
Open	Start plotting code WSFplot.
Compute Force	Start the Force program (runs on Poisson problems only).
Create List	Start List35 to make listing of solution file arrays.
Interpolate (SF7)	Start the field interpolator SF7.
Postprocess (SFO)	Start the postprocessor SFO.
Run Fish	Start the real-array Superfish solver.
Run CFish	Start the complex-array Superfish solver.
Run Poisson	Start the static-field solver Poisson.
Run Pandira	Start the static-field solver Pandira, required for permanent-magnet problems.

4. Opening input files on startup

There are several ways to start a Poisson Superfish program using tools commonly available under Windows. You can

1. double-click an input file of one of the registered types,
2. right mouse click on a registered file and select the program from the menu ,
3. go to Start, Programs, Poisson Superfish and select from the menu,
4. use the Run tool on the Start menu,
5. run the program from a batch command file, or
6. type the program name at the command prompt in a CMD or MS-DOS window.

Method 1 and method 2 always sends the name of a file to the program being launched. Methods 4, 5, and 6, have the potential for supplying one or more filenames to the program since you add the name of a file to the command line. Method 3 never supplies any additional information when starting the program. With all these options available, the user should be aware of the steps that each code follows when starting up. We divide the discussion into two sections: with and without available input file information.

a. *Starting with an input filename*

If you double-click or right-click on a registered file type (method 1 or 2), there is no ambiguity and the program will open that file. These actions supply complete drive and path information to the program, in addition to the name of the file. The only thing left for some codes to decide is whether to also open another related file.

Program Automesh reads only a single input file with extension .AM (or .AF), Solver programs Fish, CFish, Poisson, and Pandira read only the Poisson Superfish solution file started by Automesh. Thus, for Automesh and the solver programs, there is nothing further to decide if it starts up with a supplied filename. Postprocessors SFO, SF7, and Force read the Poisson Superfish solution file written by one of the solvers plus an optional input text file. You can start the postprocessor by supplying either then name of the solution file (extension .T35) or the name of the input text file (extensions .SEG, .IN7, .FCE).

For example, suppose Automesh started with file PROB1.AM and created solution file PROB1.T35, which one of solver codes then updated with the field solution data. The user can now start SFO by double-clicking on file PROB1.SEG. Program SFO recognizes this file as the input text file automatically opens the corresponding solution file PROB1.T35. The code displays an error dialog box if it cannot find the .T35 file. Another way to start SFO is to right click on file PROB1.T35 and select menu item Postprocess (SFO). In this case, SFO recognizes the file as a Poisson Superfish solution file and automatically attempts to open file PROB1.SEG. If SFO cannot find the .SEG file, it does not diagnose an error because the file is optional. Instead, the code proceeds using default settings.

b. *Starting with more than one input filename*

When using methods 4, 5 and 6 with the postprocessors SFO, SF7, and Force, the user can supply the names of two input files. One file is the Poisson Superfish solution file and other file is an input text file containing additional data or instructions for the code. If you use the standard naming convention, where both files have the same name plus their respective registered extensions (e.g. PROB1.T35 and PROB1.SEG for SFO), then there is no need to supply both filenames. The program will find both files automatically with only one of the filenames on the command line.

If the input text file does not follow the standard naming convention, or if one of the files is in a different directory, then you must supply both filenames on the command line. In this case, the best practice is to supply complete file specifications including the path and extension. If the files are in the same directory, the path information may not be required, but only if the directory containing the files is the current directory.

c. Starting without an input filename

If you start a Poisson Superfish code from the Windows Start menu (method 3), the code has no information about input filenames. Without a filename on the command line, programs follow a hierarchy of procedures to either open a file automatically or assist the user in selecting a file. The action taken depends upon the type of program. We discuss each category of programs separately.

Automesh and Autofish

Automesh and Autofish will display the standard Open dialog window. Both codes will display all .AM and .AF files found in the last directory where the code was run. If the last input file still exists, it will appear as the default in the Filename: box.

Solver codes Fish, CFish, Poisson, Pandira

Solver codes first check for the existence in the startup directory of file TAPE35.INF. If this file exists, the code reads from it the name of the last Poisson Superfish solution file generated by Automesh in that directory. The INF file stores filenames for both type problem (Superfish and Poisson) so that, for example, program Fish will not try to open a Poisson problem. If the solution file exists and was created recently enough, the code opens it. Variable AutoOpenAgeLimit in file SF.INI sets the maximum age of files opened in this manner. The default is 15 minutes.

If the solution file found in TAPE35.INF exists, but is older than the age limit, the code will display the standard Open dialog window in that directory. The solution file will appear as the default in the Filename box.

If file TAPE35.INF does not exist in the startup directory, or if last Poisson Superfish solution file listed in TAPE35.INF does not exist, then the code next tries the directory where the Automesh last created a solution file (of the correct type, either Poisson problem or Superfish problem). If that solution file exists and was created recently enough, the code opens it. If the last solution file exists, but is older than the age limit, the code will display the standard Open dialog window in that directory with the last solution file as the default selection.

Failing all these attempts to locate (or browse for) a file created recently by Automesh, the code next tries the directory where the solver code itself last ran. In this case, the code will not open a file automatically. Instead it will display the standard Open dialog window in that directory with the last solution file as the default selection (if it still exists).

Postprocessors SFO, SF7, Force

Postprocessors first check for the existence in the startup directory of file TAPE35.INF. If this file exists, the code reads from it the name of the last Poisson Superfish solution file written by one of the solver programs. The INF file stores filenames for both type problem (Superfish and Poisson). Program Force only considers files written by Poisson and Pandira. Programs SFO and SF7 consider files written by any solver. If the solution file exists and was created recently enough (see the previous section), the code opens it.

If the solution file found in TAPE35.INF exists, but is older than the age limit, the code will display the standard Open dialog window in that directory. The solution file will appear as the default in the Filename box.

If file TAPE35.INF does not exist in the startup directory, or if last Poisson Superfish solution file listed in TAPE35.INF does not exist, then the code next tries the directory where Fish, CFish, Poisson, or Pandira last wrote a solution file (Poisson or Pandira only for Force). If that solution file exists and was created recently enough, the code opens it. If the last solution file exists, but is older than the age limit, the code will display the standard Open dialog window in that directory with the last solution file as the default selection.

Failing all these attempts to locate (or browse for) a recently solved problem, the code next tries the directory where the postprocessor itself last ran. In this case, the code will not open a file automatically. Instead it will display the standard Open dialog window in that directory with the last solution file as the default selection (if it still exists).

WSFplot and List35

The procedure for WSFplot and List35 is similar to the procedures for solvers and postprocessors. WSFplot or List35 first checks for the existence in the startup directory of file TAPE35.INF. If this file exists, the code reads from it the name of the last Poisson Superfish solution file accessed by any other program, including itself. If a solution file exists and was created recently enough (see the previous section), the code opens it.

If the solution file found in TAPE35.INF exists, but is older than the age limit, the code will display the standard Open dialog window in that directory. The solution file will appear as the default in the Filename box.

If file TAPE35.INF does not exist in the startup directory, or if last Poisson Superfish solution file listed in TAPE35.INF does not exist, then the code next tries the directory the most recently used solution file. If that solution file exists and was created recently enough, the code opens it. If the last solution file exists, but is older than the age limit, the code will display the standard Open dialog window in that directory with the last solution file as the default selection.

Failing all these attempts to locate (or browse for) a recently used solution file, the code next tries the directory where it last ran. In this case, the code will not open a file automatically. Instead it will display the standard Open dialog window in that directory with the last solution file as the default selection (if it still exists).

Other programs

All other programs, which includes plotting codes Tablplot and Quikplot, tuning programs, and utility programs, go to the directory where the program last ran. Tuning programs and utility programs (SFOtable, SegField, FScale) start the standard Open immediately. Plotting codes Quikplot and Tablplot open the main display window and wait for the user to either open a file for plotting or open a file for editing.

5. Using comments to document input files

We recommend using descriptive comments in the input files. Comments can occur on any line after a semicolon (;) or exclamation mark (!). (An exception is the problem description that appears before the first REG namelist in the Automesh input file.) Programs ignore blank lines and comments.

6. A word about text editors and output-file fonts

The Poisson Superfish codes generate text files best displayed with a fixed (non-proportional) font. Some output files use a few symbol characters in the MS-DOS English character set (code-page 437). These files will not be displayed correctly by applications such as Microsoft Write or Microsoft Word. Although Windows Notepad is an ASCII text editor, for some reason it does not use code-page 437 and, therefore, does not display correctly the line-drawing characters.

B. Automesh input file

Program Automesh reads sets of namelist variables named REG (for region), PO (for point), and MT (for material table). The Automesh documentation includes lists of all the possible entries in each namelist section. You can include descriptive comments anywhere in this file. Comment indicators are the semicolon (;) or exclamation mark (!).

There are two possible extensions for the Automesh input file (see Table I-1). The choice depends upon which code you want to start by double clicking on the input file. For Automesh the default extension is AM, and for Autofish it is AF. You can launch Automesh automatically by double-clicking on a file with the AM extension. Double-clicking on files with extension AF launches Autofish automatically. If you right click on these files, the menu includes options to Open the file or Edit the file. The Open section launches the appropriate code (Automesh for AM files, Autofish for AF files). The Edit option brings the file up in Notepad. The right click menu also includes an option to start the other program.

Both programs, Automesh and Autofish, will open either input file in command mode. For example, suppose you start Automesh with the command:

```
Automesh        PROB1
```

If file PROB1 exists in the current directory, Automesh will open it as the input file. If file PROB1 does not exist, but file PROB1.AM does exist, then PROB1.AM becomes the input file. If neither of these files exists, but file PROB1.AF exists, Automesh will open PROB1.AF. Program Autofish looks for the input files in the order PROB1, PROB1.AF, and PROB1.AM. If two or more of these files exist, you can type the complete filename, if necessary. The tuning programs create Automesh input files with the AM extension. These files contain comments that describe the variables in the first REG namelist.

Other Poisson Superfish codes use the Automesh input filename as the root name for their own default input files as described in the next section..

C. Other input files

The default input filename consists of the name of the Automesh input file plus a particular extension for each code. The Poisson Superfish solver programs (Fish, CFish, Poisson, and Pandira) read only the T35 file written by program Automesh. The postprocessors (SFO, SF7, Force) generally will read an input file. Program SFO has default settings that it will use in the absence of input-file data. Program SF7 opens a dialog box if there is no input file. Force can operate without an input file if the force object consists of a single complete region (the code will prompt for it).

In any input file, you can include descriptive comments after a semicolon (;) or exclamation mark (!). The following subsections describe the input files for the solver programs and postprocessors. They include a few short examples.

1. T36 file, modifying the Automesh-generated boundaries

The T36 file is an optional input file for Automesh. This file provides a means for the user to enter distortions of the problem boundary calculated by a thermal/structural mechanical analysis code (e.g. ANSYS, ABAQUS, COSMOS, ALGOR, etc.). The T36 file is a text file containing node displacement data, and it has the same name as the Automesh input file, but with extension T36. For example, if you start Automesh with input file PROB1.AM, the mesh modification file is PROB1.T36.

Program Automesh reads the T36 file if ModT36 = 1 in the first REG namelist of the Automesh input file. The code applies the displacements to the coordinates found in temporary file TAPE36.

There are three possible formats for lines in the T36 file. All lines should use the same format.

- Two integers followed by two real numbers specify the logical coordinates K,L of the node to modify and the new X,Y coordinates of the node. (Dimensions of X,Y are the same as in the original Automesh input file.)
- Four real numbers specify the original X,Y coordinates followed by the new X,Y coordinates. (All dimensions are the same as in the original Automesh input file.)
- Two real numbers specify the displacements to apply to original coordinates. Displacements use the same dimensions as coordinates in the original Automesh input file.)

For the last format, consecutive lines in the T36 file correspond to consecutive boundary points in the TAPE36 file. If the first boundary point to be modified is not the first one in TAPE36, then the T36 file defines the first point using the FirstNode keyword, which can appear anywhere in the file. For example, if the first modified node's original coordinates are X = 5.5 and Y = 1.0, then the following line should appear in the file:

```
FirstNode = 5.5,1.0
```

For the other two format options, the order of the coordinates in the T36 file is not important.

2. SEG file, input file for program SFO

Input to SFO consists of a Poisson Superfish solution file (extension T35) and an optional input filename with extension SEG. Both files have the name of the Automesh input file plus the extension. You can double click on either of these files to start SFO. For more details about startup options refer to section I.A.4.

The tuning programs create SFO input files with comments that describe entries in the file. Figure I-1 is a sample SEG file created by DTLfish for a drift-tube linac cell. The first line sets the drift-tube stem radius to 0.635 cm. The second line sets the phase length of the half cavity for the synchronous particle to 180 degrees. This is the correct value for DTL cell of length 1βλ, where βλ is the distance traveled in one rf period by a particle of velocity βc.

```
StemRadius = 0.635              ; Stem radius in cm
GeometryPhaseLength = 180       ; Phase length of the problem geometry in degrees
FieldSegments                   ; Start of list of segments included in power and field calculations
5 6 7 8 9                       ; Segment numbers
–10 –11 –12
13 14 15 16 17 18 19 20 21 22 23 24
EndData                         ; End of FieldSegments list
End                             ; End of input file
```

Figure I-1. SFO input file written by DTLfish.

3. IN7, input file for program SF7

Input to SF7 consists of a Poisson Superfish solution file (extension T35) and an optional input filename with extension IN7. Both files have the name of the Automesh input file plus the extension. You can double click on either of these files to start SFO. For more details about startup options refer to section I.A.4.

Here is a sample IN7 file that generates a Tablplot input file containing fields interpolated at 200 points along Y = 0 from X = 1.0 cm to X = 2.5 cm:

```
line   plot
1.0    0      2.5    0
200
end
```

In the absence of the IN7 input file, SF7 opens a Windows dialog box where you can enter input data.

4. *CurveFile*, an input file for program SF7

Program SF7 can interpolate fields along user-supplied curves. After a CURVE command in SF7, the code requests the name of the *CurveFile*, which contains a list of coordinates on which to interpolate the fields. This file has no default name. You must supply the file’s name and extension or the complete path if the file is in a different directory. The coordinates of each point on the curve must appear on the same line in the file. Like other input files, you can include descriptive comments in this file.

Suggestions for creating data for a *CurveFile* include using the boundary points along segments generated by Automesh and using the field-contour feature in WSFplot. A complete list of boundary-point data appears in file OUTAUT.TXT after running Automesh. WSFplot writes coordinates along field contours to file OUTWSF.TXT if you include the following settings in the [WSFplot] section of SF.INI:

```
[WSFplot]
CreateOutputTextFile = Yes
WriteFieldContours = Yes
```

5. FCE file, input file for Force

Input to Force consists of a Poisson Superfish solution file (extension T35) and an optional input filename with extension FCE. Both files have the name of the Automesh input file plus the extension. You can double click on either of these files to start SFO. For more details about startup options refer to section I.A.4.

The following file contains the entries needed to run Force:

```
2
0.00 2.00 count
end
```

The first line is the region number of the force-element material. The next line (or several lines) will contain physical (X,Y) coordinates will define the force element. Coordinates use as many lines as necessary, though usually any single point on the region boundary is sufficient. The word COUNT (or any abbreviation) on the last line of coordinates instructs the program to count the number of coordinate-pair entries. The last line stops the program after the first task is completed.

D. Files for sharing data among codes

Some files (e.g., TAPE36, TAPE37) contain data saved for use later by the same program. Automesh creates file TAPE36, and sometimes the temporary file TAPE37. The codes pass information to other programs through the binary solution file. Automesh rereads TAPE36 and is the first code to write the binary solution file. All of the solver codes (Fish, CFish, Poisson, and Pandira) read and write the solution file. Postprocessor SFO also reads and writes the solution file, but the other postprocessors (SF7, Force, and WSFplot) only read data from it.

1. The Poisson Superfish binary solution file

The binary solution file is unformatted binary file containing input variables, mesh information, and the solution data. We sometimes refer to the binary solution file by its historical name TAPE35. The default extension is T35. For example, if you start Automesh with input file PROB1, then the codes read or write file PROB1.T35.

Programs that read and write the solution file also use the text file TAPE35.INF, which contains the names of the last used and last generated solution files for each type of problem (Superfish or Poisson). The TAPE35.INF file allows starting the codes in sequence without having to type the name of the solution file on the command line.

Automesh creates the original copy of the solution file and writes to it the problem title, the list of problem constants, a complete description of the mesh geometry, and data supplied in the MT namelist sections of the input file. Programs Fish, CFish, Poisson, and Pandira read this file and append to it the Superfish or Poisson solution, consisting of either the rf magnetic field H_z or H_ϕ (for Superfish problems), the vector potential A_z or the product rA_ϕ (for Poisson or Pandira magnet problems), or the scalar potential (for Poisson or Pandira electrostatic problems) at each mesh point. Because of symmetry, the vector potential can have only one nonzero component. For magnet problems, Poisson and Pandira store a single number at each mesh point: A_z for X,Y problems or rA_ϕ for cylindrically symmetric R,Z problems.

Postprocessors SFO, SF7, Force, SF8, and WSFplot read information in the binary solution file and display the results in a variety of useful formats. SFO updates the solution file with the calculated field normalization for later use by program SF7.

The utility program List35 reads the binary solution file to produce a text file with extension TXT that contains tables of the data.

Because of a change in compilers used to generate the codes, solution files written by Poisson Superfish version 7 are different from those written by version 6. Version 7 programs that read the solution file will automatically translate version 6 files to the new format. This translation is irreversible, you cannot run a version 6 code on a version 7 binary solution file.

2. TAPE36, boundary-point data file

TAPE36 is a text file created by Automesh, which the code rereads after processing all the regions in the input file. No other code uses TAPE36. In older versions of the code, TAPE36 was read by LATTICE. (In UNIX version 4 and earlier, this file was known as TAPE73.) TAPE36 contains the complete logical path along every boundary segment in the problem. The last section contains information about the location in the mesh of the ends of each segment.

Automesh erases file TAPE36 after it finishes generating the mesh unless setting SaveTAPE36 = Yes appears in the appropriate section of file SF.INI

After Automesh has created the TAPE36 file, the user can modify the mesh boundaries to reflect distortions computed by a thermal or structural analysis code. The user supplies the mesh displacement data in the optional T36 file.

3. TAPE37, temporary Automesh file

Automesh may create a TAPE37 file to store data from one or more regions that it will eventually write in file TAPE36. The program creates this file when the problem contains special boundary regions that must appear at the end of file TAPE36.

Ordinarily, the code erases TAPE37 before stopping. However, if Automesh crashes or stops with an error, it may leave a TAPE37 file in the current directory. You can safely delete the file, but there is no need to do so since the code will delete it the next time it runs in the same directory.

E. Poisson Superfish output text files

Automesh and the Poisson Superfish solvers and postprocessors all write an output text file that summarizes the progress and results of a computation. The filenames start with the three letters OUT and have the TXT extension. Usually the last three letters in the output filename are in the first three letters of the program name. For example, Automesh write the file OUTAUT.TXT. The exception is CFish, which writes the same file OUTFIS.TXT as program Fish.

1. OUTAUT.TXT, the Automesh output file

Automesh writes a text file called OUTAUT.TXT, which logs the operations carried out by the program. One table lists boundary points entered in the PO namelist sections. Another table that repeats the boundary segment data reflects the new segment numbering scheme caused by breaks in the mesh size at X and Y line regions.

Columns labeled Fwd and Bkw in the second table give the number of boundary points along the logical path between the segment end points. The code tries both directions and picks the path with the fewest number of points. OUTAUT.TXT also lists logical (K,L) and physical (X,Y) coordinates along the entire logical path as they appear in the temporary file TAPE36 used by Automesh. These points appear in separate tables for each region. At the end of a logical path for a region, the code indicates the number of fixed boundary points and the total number of points along the boundary. Fixed points include points from the PO namelist lines plus intersections of the boundary with X and Y line regions. For Superfish problems, OUTAUT.TXT lists logical boundary segment end points for the SFO program’s calculation of power and frequency shifts. Automesh includes this data in the binary solution file for later use by SFO. OUTAUT.TXT also contains information about the region boundaries, end-point segment numbers, and relaxation parameters for the mesh. At the end is a list of problem constants. If problems occur setting up the boundaries or in generating the mesh, OUTAUT.TXT will usually contain detailed information that may help you to resolve the problem.

2. OUTFIS.TXT, the Fish and CFish output file

Fish and CFish each write a text file called OUTFIS.TXT. File OUTFIS.TXT contains a complete list of problem constants followed by a log of each iteration of the solution. A short list at the end of the file gives the variables calculated by the code.

3. OUTPOI.TXT and OUTPAN.TXT, Poisson and Pandira output files

Poisson writes an output file called OUTPOI.TXT, and Pandira writes output file OUTPAN.TXT. These text files contain a complete list of problem constants followed by a log of each iteration of the solution.

The file may include a table of the potential and field components and their derivatives along the symmetry axis for cylindrical geometries.

4. OUTFOR.TXT, the Force output file

Program Force writes a text file called OUTFOR.TXT. This file contains a complete list of problem constants followed by a list of the logical-coordinate path around the force element. Calculated forces appear at the end of the file. For Cartesian coordinates (ICYLIN = 0), the code reports the X and Y components of the force and the magnitude of the resultant force. These quantities have dimensions of force per unit length. For convenience, the code reports the results in N/m, kg/m, and pounds/inch. For Cartesian coordinates, the code also reports information about torque or moments about the origin. This data appears in terms of the line-of-action parameters r and θ, where |r| is the perpendicular distance of the line of action from the origin (X,Y = 0,0), and θ is direction of the force (θ = 0 is along X, θ = 90 is along Y). A positive value for r corresponds to a positive torque, which means a vector along +Z in a right-handed coordinate system.

For problems with cylindrical symmetry (ICYLIN = 1), the abscissa is the R axis and the ordinate is the Z axis. The code reports the radial and longitudinal force components per radian, the net force per radian, and the total force in the z direction. The radial component of the total force is zero because of the cylindrical symmetry. Again, the code reports these force in several units (N, kg, and lb).

5. SFO output file

Program SFO writes a text file that has the name of the Automesh input file plus the extension SFO. For all problems, the SFO output includes a complete list of the problem constants and the computed field values for each specified boundary segment. For Superfish problems SFO writes a table of the electric field versus longitudinal position along the line specified by variable LINT. The default for LINT is 1, which means that the integral is along the row of mesh points on the beam axis. SFO version 4 and earlier versions calculated E_z between the first and second row of mesh points when LINT = 1. This feature caused a small mesh-size-dependent error in the problem's normalization of the fields. See Field interpolation from the solution arrays for more information.

For Superfish problems with cylindrical symmetry, the SFO output file includes a section that summarizes the transit-time factor data for the cavity or for each cell in a multicell cavity. This summary follows all the detailed segment data. Most of this transit-time factor data also appears in the PMI file (for the latest version of Parmila) if you have activated the ParmilaData option in SF.INI. For compatibility with older versions of Parmila, this section also includes the SFDATA line for Parmila. See the discussion of the SFDATA line in the SFO documentation.

A summary of cavity data appears near the end of the output file for Superfish problems. After this summary another table lists the total power and frequency-shift data for each segment that was included in the calculation. For negative segment numbers supplied in the input SEG file, another table lists the results of Slater-perturbation calculations of power and frequency shifts for drift-tube stems.

6. OUTSF7.TXT, the SF7 output file

Program SF7 writes a text file that contains tables of interpolated fields requested by the Line, Arc, Grid, Parmela and Curve commands. The precision of the field data may be selected with SF.INI variable DecimalPlaces. If configuration variable SF7_Table in file SF.INI is equal to Expanded, then OUTSF7.TXT also will include detailed information for each point about the fit performed by the field interpolator. Refer to the section on Field interpolation from the solution arrays for more information.

7. OUTWSF.TXT, the WSFplot output file

Program WSFplot does not create an output file unless variable CreateOutputTextFile = Yes in the [WSFplot] section of file SF.INI. The file OUTWSF.TXT contains a list of the equipotential values plotted by WSFplot. The contours are identified by number in the order WSFplot draws them on the screen. Each output line also includes the symbol and units associated with the contour value.

If variable WriteFieldContours = Yes in SF.INI, then OUTWSF.TXT also will include the coordinates of the plotted contour lines.

F. Hard copy graphics files

Program WSFplot and the plotting programs Quikplot and Tablplot include various hard copy output drivers You can use these files to produce high-resolution pictures on compatible printers or plotters.

G. Input files for plotting programs

Some Poisson Superfish codes produce input files for program Tablplot. These files provide a quick way to view some of the data computed by the solvers or postprocessors.

1. OUTPOI.TBL and OUTPAN.TBL, Tablplot files of interpolated fields

In Poisson and Pandira, if variables KMIN, KTOP, LMIN, LTOP, XMINF, XMAXF, YMINF, and YMAXF were configured to interpolate the fields along a line, then the code also writes the fields that appear in OUTPOI.TXT or OUTPAN.TXT to a Tablplot file. The same data appears in the OUTPOI.TXT file or the OUTPAN.TXT file. The codes do not write this file if the interpolation region is a rectangular area. You can change the default ordinate arrays in OUTPOI.TBL or OUTPAN.TBL file with any text editor, or you can use the Tablplot menu to select different arrays to plot. Double-click on file OUTPOI.TBL to start Tablplot.

2. FishScan.TBL, Fish and CFish frequency-scan output file

FishScan.TBL is a Tablplot input file containing the results of mode searches and frequency scans. Fish and CFish create or update this file automatically when you use variable NSTEP to step the frequency. To step the frequency, enter values for NSTEP, DELFR, and FREQ (or FREQS) in the first REG namelist in the Automesh input file, where NSTEP is the number of steps, DELFR is the frequency step, and FREQS is the starting frequency. Program Fish or CFish will step the frequency by the indicated amount and write a line to file FishScan.TBL after each cycle. The plot file contains headings and axis labels defined in Table I-4 for Fish and Table I-5 for CFish. Some of these quantities also appear in file OUTFIS.TXT. The chapter on RF Field Examples discusses a frequency scan example.

Table I-4. Frequency scan data written by Fish.

Heading	Definition
FREQ	Frequency f (in MHz).
k^2	Square of the wave number $k^2 = (\omega/c)^2$.
$D(k^2)$	Resonances are at roots of $D(k^2)$ with slope = –1.
dH_1	Field error at the driving point.

Table I-5. Frequency scan data written by CFish.

Heading	Definition
FREQ	Frequency f (in MHz).
k^2	Square of the wave number $k^2 = (\omega/c)^2$.
$Re[D(k^2)]$	Real part of $D(k^2)$.
$Im[D(k^2)]$	Imaginary part of $D(k^2)$.
$Abs[D(k^2)]$	Magnitude of $D(k^2)$.
$Re[dH_1]$	Real part of the driving-point field error.
$Im[dH_1]$	Imaginary part of the driving-point field error.
$Im[dH_1]/Abs[dH_1]$	Ratio as indicated.

Use program Tablplot to plot the results. If the solution array is real, then frequency is the initial abscissa and $D(k^2)$ is the initial ordinate. If the solution array is complex, then the default initial axis are $Im[dH_1]$ versus $Re[dH_1]$ and the plot includes labels at every point giving the frequency corresponding to the point. If you want the initial plot to be $D(k^2)$ versus frequency set variable ScanFormatComplex = No in the [CFish] section of SF.INI.

You can run Fish or CFish repeatedly on the same problem, each time adding more data to an existing FishScan.TBL file. If an existing FishScan.TBL file has a different problem name or drive-point location, or if you switch between Fish and CFish, then the code creates a new file. A backup file called FishScan.OLD holds data from previous runs. This file gets overwritten each time you use the frequency scan option. Rename file FishScan.TBL for long-term storage. The FishScan configuration variable allows you to control whether programs Fish, CFish, CCLfish, CDTfish, DTLfish, ELLfish, MDTfish, RFQfish, and SCCfish write to file FishScan.TBL. For example, you can configure the codes also to write the FishScan.TBL file during resonance searches by setting FishScan to Always. Insert the following line in the [Global] section of SF.INI:

```
FishScan = Always
```

You also can use different settings for FishScan by entering separate lines in the code-specific sections of SF.INI. For example, you can prevent the code CFish from ever writing the FishScan.TBL file by setting the configuration parameter FishScan to Never in the [CFish] section of SF.INI:

```
[CFish]
FishScan = Never
```

When you use NSTEP to scan a frequency range, Fish and CFish do not write to the binary solution file. The reason for this is that the final frequency after the scan is most likely not a resonant mode. To write the fields for a particular frequency (so you can display them with WSFplot for example), run Fish or CFish with MAXCY =1 and FREQ set equal to the desired frequency in the Automesh input file.

3. Transit.TBL, transit-time data versus longitudinal position

Transit.TBL is a Tablplot input file containing the integrands of the transit-time integrals. SFO creates this file if ITFILE is equal to 1. Table I-6 lists the headings and axis labels for data columns in file Transit.TBL. In the table, the wave number $k = 2\pi/\beta\lambda$, and Z_{rel} is the ratio of the distance from the center of the cell Z_c to the distance traveled by a particle of velocity βc in one rf period:

$$Z_{rel} = \frac{z - Z_c}{\beta\lambda}.$$

Use program Tablplot to plot the data. This file gets overwritten each time you use the ITFILE option. Rename the file for long-term storage.

Table I-6. Transit-time data in file Transit.TBL.

Heading	Column data	Description
z	z	Longitudinal coordinate.
E_z	E_z	Longitudinal component of the electric field.
T	$E_z \cos[k(z\text{-}Z_c)]$	Integrand of the T integral.
Tp	$Z_{rel} E_z \sin[k(z\text{-}Z_c)]$	Integrand of the T′ integral.
Tpp	$Z_{rel}^2 E_z \cos[k(z\text{-}Z_c)]$	Integrand of the T″ integral.
S	$E_z \sin[k(z\text{-}Z_c)]$	Integrand of the S integral.
Sp	$Z_{rel} E_z \cos[k(z\text{-}Z_c)]$	Integrand of the S′ integral.
Spp	$Z_{rel}^2 E_z \sin[k(z\text{-}Z_c)]$	Integrand of the S″ integral.

4. TBeta.TBL, transit-time data versus particle velocity

TBeta.TBL is a Tablplot input file containing the transit-time integrals as a function of β, the particle velocity. SFO creates this file if IBETA is equal to 1 or 2. Table I-7 lists the headings and axis labels for data columns in file TBeta.TBL.

Table I-7. Transit-time data in file TBeta.TBL.

Heading	Description
Beta	Particle velocity relative to light.
Zc	Longitudinal reference position in integrals.
delZ	$Z_{ec} - Z_{gc}$.
T	T integral.
Tp	T′ integral.
Tpp	T″ integral.
S	S integral.
Sp	S′ integral.
Spp	S″ integral.
T	Overall transit-time factor Abs(T + iS).

For multicell cavities, SFO produces a separate file for every cell. The filenames include two integers that identify the cell number. For example, if the CellData table had 3 lines, then SFO will create files TBeta01.TBL, TBeta02.TBL, and TBeta03.TBL.

The value of IBETA affects Z_c used in the transit-time integrals. If IBETA = 1, then Z_c corresponds to the nominal gap center Z_{gc} for the cavity or cell. The nominal center is ZCTR for whole-cavity calculations (no CellData table), or ZCENTER from the CellData table for multicell cavities. If IBETA = 2, then Z_c corresponds to Z_{ec}, the electrical center of the cell computed by SFO. The column labeled delZ is the calculated difference between Z_{ec} and the nominal gap center Z_{gc}. This column reports the same value regardless of which position is used for Z_c in the transit-time integrals. You can always tell these cases apart by inspecting the Z_c and S columns. When $Z_c = Z_{gc}$ the entire column of Z_c entries will have the same constant value. When $Z_c = Z_{ec}$ the entire column of S entries will be zero or very nearly zero.

Use program Tablplot to plot the transit-time data in file TBeta.TBL (or the TBetaxx.TBL files for multiple cells). These files get overwritten each time you use the IBETA option. Rename the files for long-term storage.

5. Plot files created by SF7

SF7 can create Tablplot input files for displaying field components listed in the OUTSF7.TXT file. Select this option by including the keyword PLOT on the Line, Arc, or Curve line when entering SF7 data. Tablplot filenames have extension TBL. SF7 writes the Tablplot files to the current directory. The filenames consist of the first several characters of the Automesh input filename plus a sequence number (starting with 1). Use SF.INI setting RootNameLength in the SF7 section of SF.INI to control the number of characters in the filename.

The files written by SF7 contain the same data columns that appear in OUTSF7.TXT plus the necessary title and label sections required by Tablplot. The default setting in the Tablplot file is to plot E_z and E_r (or E_x, E_y; B_x, B_y; B_r, B_z as appropriate) versus Z (or X). If the Z (or X) coordinate does not change, the code changes the abscissa to R (or Y).

6. Files that store current screen settings

WSFplot creates a file called WSFdata.PRF that it reads itself the next time the code starts. WSFdata.PRF contains data describing all the current screens that you have set up during a WSFplot session. Upon restarting the code on the same problem (even after changing the geometry and rerunning the solver program), WSFplot starts with the last screen it displayed during the previous session.

WSFdata.PRF is a binary file. You cannot edit the file. If you want to make the program start over with its initial settings, simply delete the WSFdata.PRF file from the current directory.

Quikplot and Tablplot write similar files named Quikplot.PRF and Tablplot.PRF.

H. Files for other programs

Besides the plot files discussed in the previous section, some Poisson Superfish codes produce input files for other computer codes. Most of the other codes were developed and are currently maintained at Los Alamos National Laboratory. The EGUN code was developed by W. B. Herrmannsfeldt at SLAC.

1. Data files for program Parmela

If you are a licensed user of Parmela, you can use programs Poisson, Pandira, and SF7 to generate Parmela input files containing 2-D field maps. Parmela is an electron linac design code whose name comes from the phrase "Phase and Radial Motion in Electron Linear Accelerators." The recommended method for producing these files uses the SF7 postprocessor. Programs Poisson and Pandira include the ability to generate the 2-D maps for static magnetic or electrostatic fields.

The various field maps have formats illustrated by the Fortran code fragments shown below. In these files, all length dimensions are in cm regardless of the setting of variable CONV in the Automesh input file. Note that the last number on the lines containing the limits of coordinates R and Z is the number of increments, not the total number of entries for each coordinate. Also, notice that the order of the radial and longitudinal components differs between static field maps and rf field maps. This difference has its origin in the conventions used in Poisson Superfish codes for problems with cylindrical coordinates. For Poisson and Pandira (static) problems, the codes interpret input coordinates (X,Y) as (R,Z) cylindrical coordinates. For Superfish (rf) problems, input coordinates (X,Y) become (Z,R) in cylindrical coordinates.

The following code fragment writes a 2-D electrostatic field map for the Poisson line in Parmela. The electric field components are in V/m.

```
      open(33,file='2D_ES.TXT')
```

```
write(33,*)rmin,rmax,nmr-1
write(33,*)zmin,zmax,nmz-1
do j=1,nmz
   do i=1,nmr
      write(33,*)Er(i,j),Ez(i,j)
   enddo
enddo
close(33)
```

The following code fragment writes a 2-D static magnetic field map for the Poisson line in Parmela. The magnetic field components are in Gauss.

```
open(33,file='2D_MAG.TXT')
write(33,*)rmin,rmax,nmr-1
write(33,*)zmin,zmax,nmz-1
do j=1,nmz
   do i=1,nmr
      write(33,*)Br(i,j),Bz(i,j)
   enddo
enddo
close(33)
```

The following code fragment writes a 2-D rf field map for the CField or TRWCField line in Parmela. Variable freq is the rf frequency in MHz. The electric field components Ez and Er and the electric field magnitude E are in MV/m and the magnetic field H is in A/m.

```
open(33,file='2D_RF.TXT')
write(33,*)zmin,zmax,nmz-1,freq
write(33,*)rmin,rmax,nmr-1
do j=1,nmz
   do i=1,nmr
      write(33,*)Ez(i,j),Er(i,j),E(i,j),H(i,j)
   enddo
enddo
close(33)
```

EFLD.QKP, BFLD.QKP, and RFFLD.QKP are Quikplot input files for viewing some of the data in the 2-D maps created by SF7, Poisson, and Pandira. The codes write files EFLD.QKP for electrostatic problems, BFLD.QKP for magnetostatic problems, and (SF7 only) RFFLD.QKP for rf cavity problems. These plot files contain two curves. One curve is the quantity $B_z(z)$ or $E_z(z)$ along the line $X = X_{min}$ on the interpolation grid. This line should correspond to r = 0. The other curve is the ratio $B_r(z)/r$ or $E_r(z)/r$ evaluated two grid intervals from X_{min}. If the variation of B_r or E_r is nearly linear in the neighborhood of the origin, this quantity allows you to estimate B_r anywhere near the origin. Comment lines in the plot file tell the value of r for each curve.

a. SF7-interpolated fields for Parmela

For problems with cylindrical symmetry, postprocessor SF7 can create file *rootname*nnn.T7 for Parmela, where *rootname* is the name of the Automesh input file and nnn is a sequence number of one or more digits. The default length of the *rootname* is 6 characters, but you can instruct SF7 to use more characters (up to the full length of the name) by increasing variable RootNameLength in the SF7 section of SF.INI. For example, if you start Automesh with input file PROB.AM or PROB.AF, then SF7 will create files PROB1.T7, PROB2.T7, etc.

For rf problems, SF7 normalizes the fields to EZERO = 1.0 MV/m, which is expected by Parmela in most cases. Parmela reads a single T7 file containing a 2-D rf field map after a CField line in the Parmela input file. Two data files are needed in Parmela for traveling-wave accelerator problems. Parmela reads two T7 file containing 2-D rf field maps at different phases after a TRWCField line in the Parmela input file. It may be necessary to use a different normalization for one of these files. You can turn off the 1-MV/m normalization in SF7, if needed.

b. Poisson- and Pandira-interpolated fields for Parmela

Poisson and Pandira write input files for Parmela containing 2-D static field maps. The filename is based upon the Automesh input file's root name. For example, if you start Automesh with input file PROB1, then Poisson and Pandira will write file PROB1.T7. (The ParmelaFields setting in SF.INI for Poisson and Pandira replaced program POI7, which was previously distributed with Parmela for this purpose. Although still available, the ParmelaFields setting for Poisson and Pandira has been superseded by the addition of this capability in postprocessor SF7.)

Poisson and Pandira will generate a Parmela input file for the following conditions:

1. The problem must have cylindrical symmetry (ICYLIN = 1),
2. REG namelist variables XMINF, XMAXF, YMINF, and YMAXF must specify a valid range of physical coordinates,
3. REG namelist variables KTOP and LTOP must both be 2 or greater,
4. The ParmelaFields setting in SF.INI must be Yes.

KTOP and LTOP are the number of interpolation steps in the R and Z directions. By default, the codes write warning messages if ParmelaFields = Yes, but one of the other conditions listed above is not satisfied. You can disable these warning messages by setting ParmelaWarning = No in SF.INI. You can set the ParmelaFields option in either the [Global] section or the individual code sections of SF.INI. For example, to set only Poisson to generate Parmela input files, use the following lines:

```
[Poisson]
ParmelaFields = Yes

[Pandira]
ParmelaFields = No
```

2. Input file for the electron-gun code EGUN

For static field problems solved by Poisson or Pandira, postprocessor SF7 creates file *rootname*nnn.EGN in response to the EGUN command, where *rootname* is the name of the Automesh input file and nnn is a sequence number of one or more digits. The default length of the *rootname* is 6 characters, but you can instruct SF7 to use more characters (up to the full length of the name) by increasing variable RootNameLength in the SF7 section of SF.INI.

For example, if you start Automesh with input file PROB.AM, then SF7 will create files PROB1.EGN, PROB2.EGN, etc. Each line of the EGUN data file contains four entries. The first two numbers are integer node numbers on a square-mesh rectangular grid in the X and Y (or R and Z) directions. The last two numbers are the field components B_x and B_y, (or B_r and B_z) for magnetostatic problems, or the components E_x and E_y, (or E_r and E_z) for electrostatic problems.

Program EGUN computes trajectories of charged particles in electrostatic and magnetostatic focusing systems including the effects of space charge and self magnetic fields. Starting options include Child's Law conditions on cathodes of various shapes. Either rectangular or cylindrically symmetric geometry may be used. Magnetic fields may be specified using arbitrary configurations of coils, the output of a magnet program such as Poisson, or an externally calculated array of the axial fields.

The code has approximately 5000 lines of C source code. It is used at more than 100 sites. A 125-page user's guide, examples, and an on-line help facility are available. This code is maintained by Glen Herrmannsfeldt and W.B. Herrmannsfeldt. It is being distributed, for a fee, by:

W. B. Herrmannsfeldt
Stanford Linear Accelerator Center
Mail Stop 26, Group: TSP
2575 Sand Hill Road
Menlo Park, CA 94025
Phone: 415-926-3342
Email: wbhap@slac.stanford.edu

3. PMI file, transit-time data for program Parmila

The name Parmila comes from the phrase "Phase and Radial Motion in Ion Linear Accelerators." Parmila is a multi-particle code most often used to design drift-tube linacs. The PMI file written by SFO is a short text file containing information used in the ion-linac design code Parmila (the 1996 version written by H. Takeda). The traditional version of Parmila that has been in use since the 1960s cannot use all the information in the PMI file. Instead, use the SFDATA line found for each cell in the transit-time factor summary of the SFO output file.

The PMI file has the name of the Automesh input file plus the extension PMI. The data written in the PMI file also appears in the transit-time factor summary section of the SFO output file. This section includes some extra columns of data that SFO computes from the data found in the PMI file. Writing the PMI file directly can save time searching for the data in longer SFO output file.

Use settings in the [SFO], [Autofish], [DTLfish], [CCLfish], [CDTfish], and [MDTfish] sections of SF.INI to generate a PMI file. You can set the ParmilaData option in either the

[Global] section or the individual code sections of SF.INI. For example, to set only SFO to generate Parmila data files, use the following lines:

```
[Global]
ParmilaData = No

[SFO]
ParmilaData = Yes
```

The PMI file includes properties of the entire cavity followed by lists of parameters for each cell containing an accelerating gap. Cavity properties are the resonant frequency in MHz, the particle β, the average axial electric field E_0 for the entire cavity in MV/m, the cavity transit-time factor T, and the normalization length L used for calculating E_0.

The next entry is a table that lists the following quantities for each cell: the lower limit Zstart of the transit-time integrals, the electrical center of the cell Zcenter, the upper limit Zend of the transit-time integrals, the normalization length Lnorm, the average electric field E_0, and the ratio of the gap length to βλ. Following this table are individual tables for each cell that give the transit-time integrals T, T′, T′′, S, S′, and S′′. The two columns of entries for each integral are the separate integrals from Zstart to Zcenter and from Zcenter to Zend. If Zcenter is the true electrical center of the cell the two terms listed for S will be of equal magnitude, but opposite sign.

Shunt impedance data in MΩ/m appears in a separate section. This table has only a single entry for DTLs. For CCLs and CCDTLs the table has additional entries to allow Parmila to calculate the cavity power including the effect of the coupling slot.

The normalization length Lnorm will equal the difference Zend – Zstart unless the cavity included an extension of the bore tube on one or both ends. For a single-cell cavity (for example, a DTL or CCL cell), you can supply a value for CLENGTH to define E_0 using a length that differs from the actual cavity length. The information about the cavity type on the second line of the example files comes from the tuning program (CCLfish, CDTfish, DTLfish, ELLfish, MDTfish, or SCCfish) that generated the Automesh input file. For the examples shown here, DTLfish and CDTfish were the originating programs. The tuning codes include a REG namelist variable from Table I-8 in the Automesh input file to indicate the cavity type for Parmila. If none of the indicators appear, then Automesh or Autofish would identify the cavity type as “Unknown RF Cavity.” Program DTLfish sets DTL = 1 and MDTfish sets DTL equal to the number of cells or rf gaps in the problem. CCLfish and SCCfish both set CCL = 1. CDTfish sets CCDTL equal to the number of rf gaps in a coupled-cavity drift-tube linac cavity. Program ELLfish sets SCCAV = 1, and program ELLCAV sets SCCAV equal to the value of NumberOfCells that appears in the ELLfish control file.

For single-cell cavities, the cavity-type setting is not important. The fact that the PMI file contains the line “Cavity type: Unknown RF Cavity” will not affect the results. The table created by ReadPMI will be the same type used for drift-tube linac (DTL) structures.

Table I-8. Automesh namelist parameters that indicate the cavity type.

Variable	Possible Values
DTL	0 = not a DTL problem; >0 indicates the number of DTL cells.
CCL	0 = not a CCL problem; 1 indicates a CCL cavity.
CCDTL	0 = not a CCDTL problem; >0 indicates the number of gaps.
SCCAV	0: not a SCCAV problem; >0: total number of cells in a superconducting cavity.

a. Example PMI file for a DTL half cell

Figure I-1 shows the PMI file for a DTL half cell of length 2.29918 cm. This is the first cell generated by the RGDTL sample file in directory Examples\CavityTuning\DTL. In this case, the cell length is equal to the normalization length. SFO calculated the particle beta from this length, and it integrated the fields from Zstart = 0.0 cm to Zend = 2.299176 cm.

```
Input data for Parmila; originating program = DTLfish.
Cavity type: DTL
Design frequency =          425.000 MHz
Resonant frequency =        424.998 MHz
Design beta =               0.651884E-01
Particle beta =             0.651884E-01
Cavity E0 =                 1.00000 MV/m
Transit-time factor =       0.814641
Normalization length =      2.29918 cm

Cell    Zstart(cm)   Zcenter(cm)   Zend(cm)    Lnorm(cm)   E0(MV/m)   g/bl
1       0.000000     0.000000      2.299176    2.299176    1.000000   0.167694

Cell 1       Beta = 0.651884E-01      dZctr = 0.0000000
X            X1              X2
T            0.4073204       0.4073204
Tp           0.0263057       0.0263057
Tpp          0.0023273       0.0023273
S            -0.2218192      0.2218192
Sp           -0.0256786      0.0256786
Spp          -0.0040116      0.0040116

Shunt impedance data:
Coupling(%) Z(MOhm/m)
0.000        68.6884
```

Figure I-1. A sample PMI file for a single-gap problem.

b. Example PMI file for a 3-gap CCDTL

For multiple-cell cavities, the cell boundaries are contained in the CellData table in the SFO input file. The first and last cells of a multiple-cell cavity may use a normalization length that is less than the difference between the integration limits for the cell. Figure I-2 is the PMI file for half of a three-cell cavity. The first gap in the Superfish problem is the right half of the cavity’s middle gap. The overall length of the problem geometry was 8.888780 cm, and the normalization length was 5.88878 cm. SFO integrated the fields for the half cell between Zstart = 0.0 cm and Zend = 2.355512 cm and for the full cell (including the bore-tube extension) between Zstart = 2.355512 cm and Zend = 8.888780 cm. For the first of these cells, the value of Lnorm corresponded to Zend – Zstart, but not for the second cell.

```
Input data for Parmila; originating program = CDTfish.
Cavity type:  2-DT CCDTL
Design frequency =          700.000 MHz
Resonant frequency =        699.995 MHz
Design beta=                0.140000
Particle beta =             0.140000 (final value for multicell problem)
Cavity E0 =                 2.74864 MV/m
Transit-time factor =       0.727634
Normalization length =      7.49481   cm

Cell  Zstart(cm)  Zcenter(cm)  Zend(cm)   Lnorm(cm)   E0(MV/m)  g/bl
1     0.000000    0.000000     2.997925   2.997925    2.323363  0.149999
2     2.997925    5.995849     9.494811   4.496887    3.032151  0.149999

Cell 1      Beta = 0.140000  dZctr = 0.0000000
X           X1               X2
T           0.3635943        0.3635943
Tp          0.0354617        0.0354617
Tpp         0.0012893        0.0012893
S           -0.2493864       0.2493864
Sp          -0.0220400       0.0220400
Spp         -0.0065870       0.0065870

Cell 2      Beta = 0.140000  Zctr = -0.0000765
X           X1               X2
T           0.3642570        0.3638313
Tp          0.0353851        0.0353454
Tpp         0.0013272        0.0012968
S           -0.2492756       0.2492172
Sp          -0.0221530       0.0221080
Spp         -0.0065549       0.0065220

Shunt impedance data:
Coupling(%) Z(MOhm/m)
0.000       66.5005
1.000       65.1405
2.000       63.8350
3.000       62.5808
4.000       61.3750
5.000       60.2147
6.000       59.0975
```

Figure I-2. A sample PMI file for a multicell problem.

I. Sample batch command files

The Examples subdirectories contain many sample input files for running the Poisson Superfish codes. Chapters XVI though XIX have more information about the sample problems. Each subdirectory includes a simple batch file that runs the codes in the proper order.