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

AllocationOfMemory	AllowYregInGap
BatchFileForSamplePrograms	BETAinSFO
BetterSFOIntegration	BoundaryConditions
BrowsingForFiles	CavityModes
CodeHistory	CodeReferences
CodeSummary	ComplexFish
ComplexTape35Arrays	ComputeStoredEnergy
ControlVariables	ConvergenceCriteriaHistory
CutoffTerm	DecimalPlaces
DeleteNoRingFiles	DirichletBoundary
DisplayPreferences	DocumentationDirectory
DownloadingFilesByFTP	DrivePointChecking
EnhancedNamelist	ExamplesDirectory
ExitCodes	FieldContourWriting
FieldInterpolatorDescription	FieldInterpolatorHistory
FieldInterpolatorVariables	FieldsOnDirichletSegmentsInSFO
Filenames	FilesForCodeDevelopers
FishScanFile	GeometricBeta
GradedBeta	HardcopyDriver
INCLUDEfiles	InterpolationOnNodes
KilpatrickReporting	LaheyFortran
LimitingMemory	LineWidths
List35Added	LogicalPath
LongProblemTitles	MaterialDataArrays
MaterialDataEntry	MeshStructure
MultiDTLCells	NeumannBoundary
ParmelaAndEgunDataFiles	ParmelaFieldMaps
ParmilaData	Performance
PermittivityAndPermeabilityArrays	PlottingCodeSFiniSettings
PoissonVariables	PrecisionOfVariables
PrintBGammaTables	ProgramLimits
RealTape35Arrays	References
RootFinderHistory	SampleReadingCFishSolution
SampleReadingPoissonSolution	SaveTAPE35
SaveTAPE36	SF7ComplexData
SF7DefaultSettings	SF7FitResults
SFINIreadFailure	SFINIsettings
SFINIsearchOrder	SFPlotFiles
ShowWarning	SlaterFrequencyShiftOption
SoftwareInstallation	SolutionFileExtension
StoringTempData	StructureModes
SubroutineInterpolate	SubroutineRDTape35
SuperfishVariables	SurfaceResistance

TechnicalSupport	TemporaryDataArrays
Terminology	The1987Manuals
ThirdNearestNeighbors	ThirdNearestNeighborPreference
TuningCodePreferredNormalization	TuningCodeSummary
TuningRingLimit	UnlimitedLineRegions
UnlimitedMeshPoints	UpdatesOnInternet
WarningMessages	Winteracter
WSFplot_OutputFile	Zcenter

File SFINTRO.DOC Table of Contents

II. Poisson Superfish software installation ____ 9

A. Machine requirements ____ 9
1. Performance issues ____ 9

B. Installation directory and subdirectories ____ 10
1. Subdirectory LANL\Docs, documentation files ____ 10
2. Subdirectory LANL\Examples, sample input files ____ 10
3. Subdirectory LANL\DeveloperFiles, files for code developers ____ 11

C. Other files in the LANL directory ____ 11

D. The Lahey/Fujitsu Fortran compiler and *Winteracter* library ____ 12

E. Technical support and code updates ____ 12
1. Electronic mail support address: Superfish@lanl.gov ____ 13
2. Email discussion group ____ 13
3. Obtaining updated installation files ____ 13
4. Email notification of new features and bug fixes ____ 14

F. Files provided for program developers ____ 14
1. Using the dynamic link library ____ 15
a. Function INTERP ____ 17
b. Functions I_Variable and R_Variable ____ 17
c. Function GetTitle ____ 18
d. Multiple DLLs ____ 18
e. Example files ____ 19
2. Calling Fortran LF95 library routines ____ 20
a. Batch files for running the sample programs ____ 20
b. LF95.Fig, configuration file ____ 21
c. INCLUDE file T35ARRAY.INC ____ 21
d. Fortran modules ____ 21
e. RDTape35 and RDRECORD1, subroutines for reading solution files ____ 32
f. Interpolate, subroutine for interpolating fields ____ 36

III. Introduction to Poisson Superfish ____ 44

A. History of the development of Poisson Superfish ____ 45
1. Code development under LAACG ____ 45
2. Features added in version 5 ____ 46
a. User interface improvements ____ 47
b. Automesh improvements ____ 48
c. RF solver improvements ____ 49
d. New field interpolator ____ 51
e. Memory management and program limits ____ 52
f. Postprocessor improvements ____ 53
g. New programs ____ 54
3. Features added in version 6 ____ 55
a. Source code organization ____ 55
b. Installation program ____ 56
c. Program WSFplot ____ 56

d. Programs Quikplot and Tablplot 56
e. Unlimited number of material tables and error checking of material data 56
f. Force code improvements 57
g. Field interpolator improvements 57
h. Program MK_SFINI 58
i. Tuning-code companion programs 58
4. Features added in version 7 58
a. Conversion of binary solution files to version 7 format 58
b. Automesh improvements 58
c. Larger problems supported 59
d. Numbered warning messages 59
e. Dynamic link library containing the field interpolator 59

B. References 59
1. Published references to Poisson Superfish 59
2. Other references 60

C. Terminology 61
1. Superfish codes 61
2. Solver program 61
3. Tuning program 61
4. Superfish problem or Poisson problem 61
5. Postprocessor 62
6. Plotting program 62
7. SOR 62
8. Binary solution file 62
9. Configuration file or initialization file 62
10. Boundary conditions 62
11. Beta, the symbol β 63
12. Beta-lambda, the product βλ 63
13. Cavity 63
14. CCDTL 63
15. CCL 63
16. Cell 63
17. Coupling 64
18. Coupling cell or coupling cavity 64
19. DTL 64
20. Gamma, the symbol γ 64
21. Gap 64
22. Geometric β 64
23. Graded β 65
24. Kilpatrick field 65
25. Modes 65
a. Cavity modes 65
b. Structure modes 66
26. Post coupler 66
27. Quality factor Q 66
28. RFQ 66
29. Shunt impedance 66
30. Skin depth 67
31. Transit-time factor 67

D. Summary of the Poisson Superfish codes 68
1. Autofish 69
2. Automesh 69
3. Lattice 69
4. Fish 69
5. CFish 69
6. Poisson 70
7. Pandira 70
8. WSFplot 70
9. SFO 70
10. SF7 71
11. Force 71
12. DTLfish 71
13. DTLCells 72
14. CCLfish 72
15. CCLCells 72
16. CDTfish 72
17. ELLfish 72
18. ELLCAV 73
19. MDTfish 73
20. MDTCells 73
21. RFQfish 73
22. SCCfish 73
23. Quikplot 73
24. Tablplot 74
25. List35 74
26. SFOtable 74
27. SegField 74
28. Beta 74
29. Kilpat 74
30. ConvertF 74
31. FScale 74
32. SF8 75
33. MK_SFINI 75

E. Problem variables 76
1. Problem variables for Fish and CFish 76
2. Problem variables for Poisson and Pandira 79

F. Configuring Poisson Superfish features: SF.INI 83
1. InternalArrays setting 86
2. Saving file TAPE36 86
3. Drives for scratch files and storing scratch data in memory 86
4. Limiting memory available to certain programs 87
5. Forcing programs to always browse for an input file 88
6. Setting the binary solution file extension 88
7. Turning on and off selected warning messages 88
8. Setting the hardcopy graphics type 92
9. Setting initial display preferences for WSFplot 93
10. Setting the interval between contour lines in WSFplot 93

11. Saving the WSFplot field contours 95
12. Choosing the plotting code preference file 95
13. Configuring program Quikplot and Tablplot 95
14. Setting the filename length for SF7 output files 96
15. Setting precision of field data printed in output text files 96
16. Including field-interpolator fitting data with SF7 results 97
17. Configuring SF7 to write real and imaginary CFish fields 97
18. Setting default values for the SF7 dialog window 97
19. Controlling field normalization in SF7 when generating Parmela data 98
20. Controlling some behavior of the field interpolator 98
21. Setting initial display preferences for Beta utility 100
22. Writing the file FishScan.TBL in Fish or CFish 100
23. Create file CircleFit.QKP during CFish resonance 101
24. Writing Poisson or Pandira field data for program Parmela 101
25. Writing permeability tables in Poisson and Pandira 102
26. Calculating stored energy in Poisson and Pandira 102
27. Configuring particle velocity settings for SFO and tuning programs 102
28. Calculating fields on Dirichlet boundaries in SFO 103
29. Writing transit-time data for Parmila in SFO 103
30. Saving the binary solution files during tuning program sessions 104
31. Setting the preferred normalization method in tuning codes 104
32. Making CDTfish compute transit-time factors at the geometric center 105
33. Setting a maximum tuning-ring thickness in CCLfish and CDTfish 105
34. Deleting or saving no-ring Superfish files in CCLfish and CDTfish 106
35. Controlling line regions through the drift tube in DTLfish and MDTfish 106
36. Computing Slater-Perturbation Frequency Shifts in WSFplot and SF7 106

G. Precision of variables in Poisson Superfish 107

H. Field interpolation from the solution arrays 108

1. Problems with Cartesian (XY) coordinates use functions 17 to 25 111
   a. Function 17: general function for interior non-boundary points 111
   b. Function 18: vertical Neumann boundary 111
   c. Function 19: horizontal Neumann boundary 111
   d. Function 20: vertical Neumann and horizontal Neumann boundaries 111
   e. Function 21: vertical Dirichlet and horizontal Dirichlet boundaries. 111
   f. Function 22: vertical Dirichlet boundary 112
   g. Function 23: horizontal Dirichlet boundary 112
   h. Function 24: vertical Neumann and horizontal Dirichlet boundaries 112
   i. Function 25: vertical Dirichlet and horizontal Neumann boundaries 112
2. Electrostatic problems with cylindrical symmetry use functions 27 to 29 112
   a. Function 27: general function for interior non-boundary points 113
   b. Function 28: horizontal Dirichlet boundary 113
   c. Function 29: horizontal Neumann boundary 113
3. Magnetostatic problems with cylindrical coordinates use functions 37 to 39 113
   a. Function 37: interior non-boundary points 114
   b. Function 38: horizontal Dirichlet boundary 114
   c. Function 39: horizontal Neumann boundary 114
4. RF problems with cylindrical coordinates use functions 57 to 68 114
   a. Function 57: general function for interior non-boundary points 114
   b. Function 58: horizontal Neumann boundary 115
   c. Function 59: vertical Neumann boundary 115

d. Function 60: horizontal Neumann and vertical Neumann boundaries 115
e. Function 61: horizontal Dirichlet and vertical Dirichlet boundaries 115
f. Function 62: horizontal Dirichlet boundary at nonzero R 115
g. Function 63: vertical Dirichlet boundary 115
h. Function 64: horizontal Neumann and vertical Dirichlet boundaries 115
i. Function 65: horizontal Dirichlet and vertical Neumann boundaries 116
j. Function 66: near R = 0 and no other boundaries 116
k. Function 67: near R = 0 and a vertical Neumann boundary 116
l. Function 68: near R = 0 and a vertical Dirichlet boundary 116

I. Poisson Superfish program limits 116

J. Temporary data arrays used by Fish, CFish, and Pandira 117

1. Using a scratch file 118
2. Forcing the codes to use a scratch file instead of memory 118
3. Deleting unerased scratch files 118

II. Poisson Superfish software installation

This is version 7 of Poisson Superfish, the standard version supported by the Los Alamos Accelerator Code Group (LAACG). This section describes the software installation for Poisson Superfish version 7. The installation files are distributed by FTP (see the section "Obtaining updated installation files on Internet ").Los Alamos programs distributed by the LAACG run under Windows 2000 and Windows XP.

The original version of Poisson Superfish was written by R. F. Holsinger, with theoretical assistance from Klaus Halbach. This version is maintained by James H. Billen. For questions or for other information regarding these programs contact:

James H. Billen
Los Alamos National Laboratory
Mail Stop H824
Los Alamos, NM 87545
Phone: 505-667-6627
Fax: 505-665-9998
Email: JBillen@lanl.gov

A. Machine requirements

The codes run under Windows NT, Windows 2000, and Windows XP on Pentium or equivalent processors. We compile the programs with the Lahey/Fujitsu Fortran compiler LF95. The computer must have sufficient memory to run the largest code. See Program limits for more information. Refer to the READ.ME file on distribution disk 1 (or in the d:\LANL directory where d: is the installation drive for the codes on your computer) for specific information about your distribution.

1. Performance issues

Programs Fish, CFish, all of the tuning codes, and Pandira use a solver algorithm that involves inverting a tridiagonal matrix. The procedure generates N_{max} square block matrices of dimension $N_{min} \times N_{min}$, where N_{max} is the larger of KMAX and LMAX and N_{min} is the smaller. In the original paper on Superfish (see References), Halbach and Holsinger estimated the time T for inverting all of the block matrices:

$$T = T_1 N^2 R \,,$$

where N = (KMAX + 2)(LMAX + 2) = K_2L_2, and R is the minimum of K_2/L_2 and L_2/K_2. The number they quoted for T_1 on a CDC7600 computer in the mid 1970s was ~0.75 µs. Early versions of the code could solve problems with $N < 32{,}768$. This limit was related to the way array elements stored multiple pieces of data about the corresponding mesh point. For problems of comparable size on a 1-GHz Pentium III in 2001, T_1 = 0.015 µs or about 50 times faster than the CDC7600. However, for larger problems (e.g. N ~ 90,000) one will observe T_1 = 0.052 µs. The reason for this performance degradation is that on large problems the processor's onboard cache is too small, forcing the system to fetch more array elements from RAM, which is much slower than the cache memory.

The 1-GHz machine in the above example had a 256K cache. Of course, machines with a smaller cache will show degraded performance at lower values of N than machines with a larger cache. Another factor that will influence the size at which degradation occurs is the aspect ratio R. Square problems (with $R \approx 1$) will suffer the slowdown at smaller N than meshes that are elongated in either direction.

B. Installation directory and subdirectories

LANL, an abbreviation of Los Alamos National Laboratory, is the recommended installation directory for Los Alamos computer codes. The Setup program creates this directory and the subdirectories listed in Table II-1. The LANL directory contains the Readme files, Changes files and executable files.

Table II-1. Disk directories.

Directory	Description
LANL	Root for Los Alamos code distribution.
LANL\DeveloperFiles	Files provided for code developers.
LANL\Docs	Documentation files.
LANL\Examples	Sample input files.

1. Subdirectory LANL\Docs, documentation files

The Setup program installs the documentation files in directory LANL\Docs. File SFTOC.DOC includes a detailed list of the files that make up this manual, as well as links to the other files.

2. Subdirectory LANL\Examples, sample input files

The Examples directory created by the Setup program contains subdirectories in Table II-2. This documentation contains instructions for running these example files. It includes a brief description of the input files and sample plots of the computed fields.

Table II-2. Subdirectories directly under LANL\Examples.

Subdirectory	Description
CavityTuning	Automated tuning programs for accelerator cavities.
Electrostatic	Poisson and Pandira electrostatic problems.
Magnetostatic	Poisson and Pandira magnetostatic, including permanent-magnet, problems.
PlottingCodes	Examples for several general-purpose plotting codes.
RadioFrequency	Fish and CFish radio frequency cavity and waveguide problems.

The input files use enhancements in the latest version including entry of all problem variables in the Automesh input file, line-by-line comments, and the use of more "line regions" to divide the mesh into coarse and fine sections. To reduce the size of the distribution files, these subdirectories do not include any output files. If you are viewing this document on-line using Microsoft Word or Word Viewer, you can double click a directory name to jump directly to a section in which you are interested.

Long-time users of the codes may recognize sample problems from the 1987 manuals. (Some problem titles refer to the section in these older documents.)You should not expect perfect agreement between numerical results from the present code version and the data published in the 1987 manuals. Several factors may produce small differences.

- Machine precision. Most examples in the 1987 manuals were probably run on CRAY machines, but some of the results may be from VAX runs. There are significant differences among the various combinations of single and double precision on the CRAY and VAX and double precision on the 80x86 and Pentium families of processors.
- Differences in mesh size. Program Automesh may produce a mesh that differs slightly from the mesh generated by older versions of the code. Often the mesh spacing in older example problems is too coarse. We have added line regions to some problems and simply decreased the mesh size in others.
- Errors in PO namelist data. We have corrected some typographical errors that we noticed in the data printed in the 1987 manuals. These changes may alter the problem geometry slightly.

3. Subdirectory LANL\DeveloperFiles, files for code developers

Programmers may wish to write their own postprocessors for Poisson or Superfish problems. Subdirectories under LANL\DeveloperFiles\PoissonSuperfish contain Lahey/Fujitsu Fortran source files, Fortran libraries, and dynamic link libraries (DLLs) for several language systems. These files give code developers access to the Poisson Superfish field interpolator for computing fields from the solution written by programs Fish, CFish, Autofish, Poisson, Pandira, and the tuning programs.

C. Other files in the LANL directory

The LANL directory also contains the files in Table II-3. The Microsoft Word template and custom dictionary files are discussed in section I. How To Use This Document.

Table II-3. Miscellaneous files in the LANL directory.

File	Description
SF.INI	Configuration file for Poisson Superfish codes.
ReadMePoissonSuperfish.txt	Text file containing instructions for starting the installation program, and other information about the installation.
License.txt	End-User License Agreement for Los Alamos software.
LANLHELP.DOT	Microsoft Word template file used with Los Alamos documentation.
LANLHELP.DIC	Microsoft Word custom dictionary file used with Los Alamos documentation.

D. The Lahey/Fujitsu Fortran compiler and *Winteracter* library

Poisson Superfish codes are compiled with Lahey Computer System's Fortran 95 compiler plus ***Winteracter***, which is a Win32 user-interface and graphics package for Fortran 90 and Fortran 95 developers written by Interactive Software Services Ltd. The ***Winteracter*** package provides the graphics hardcopy options in Poisson Superfish plotting codes. The resulting programs run under Windows, NT, 2000 , and XP. At the time of this manual's revision date, we were using Lahey/Fujitsu Fortran LF95, version 5.7d and ***Winteracter***, version 5.10f.

The installation program installs the run-time error file LF95.EER in the LANL directory. It will not overwrite files in existing LF95 Lahey/Fujitsu Fortran directories.

If you wish to use the files provided for code developers, you may purchase the Fortran 95 compiler and ***Winteracter*** from:

Lahey Computer Systems, Inc.
P. O. Box 6091
Incline Village, NV 89450
Phone: 800-548-4778
Fax: 702-831-8123
http://www.lahey.com/

The ***Winteracter*** vendor prefers that customers in the United States purchase their product through Lahey Computer Systems. Potential customers in other countries may wish to contact the vendor directly:

Interactive Software Services Ltd.
Westwood House
Littleton Drive
Huntington
Staffs WS12 4TS
United Kingdom
Phone: +44 (0)1543 503611
Fax: +44 (0)1543 574566
http://www.winteracter.com/

E. Technical support and code updates

James H. Billen and Lloyd M. Young maintain the Poisson Superfish codes. We would appreciate hearing from you if you discover a bug or if you cannot find the information you need to run a code in this documentation. We are also interested in your suggestions for improvements we might make in the codes. Several features now in the codes were originally suggested by Poisson Superfish users.

If your code distribution is more than a few months old, the problem you are encountering may already be fixed. The latest installation files are usually available 24 hours a day via our FTP server.

1. Electronic mail support address: Superfish@lanl.gov

We will be happy to try running your problem with the latest version of the codes. We might be able to suggest changes in the input file. If there is a bug in the codes, we will try to fix it. Please take the time to document the difficulty as thoroughly as possible. Note the revision date of the program when it starts and save all the input files in use when the problem occurred.

We prefer to correspond by electronic mail because it is fast and there is no need to retype the input files. Please send email to Superfish@lanl.gov. In your email message, include a description of the symptoms. Either attach relevant input files or include them in the body of the email message. Please do not include output files unless we have specifically asked for them.

2. Email discussion group

Users who wish to discuss the use of Poisson Superfish with other code users can subscribe to the mailing list PoissonSuperfish_Forum@lanl.gov. Participation is entirely voluntary. This discussion group is not moderated. However, only subscribers may post messages to this list and only registered users of Poisson Superfish may subscribe. The character string "[SF]" appears in the subject line of messages posted to PoissonSuperfish_Forum@lanl.gov for those who wish to filter messages in their mail application. Posters to this list agree to adhere to the guidelines included in the welcome message.

To subscribe to the mailing list, send mail to <listmanager@maillist.lanl.gov> with the following command in the body of your email message:

```
subscribe PoissonSuperfish_Forum
```

3. Obtaining updated installation files

You can download the latest copies of Los Alamos code distributions by FTP (Internet's File Transport Protocol). We maintain two FTP servers with the same files, directories, and log-in procedures. At least one of these FTP servers is usually available 24 hours a day. If you are behind a fire wall at your institution, set the FTP program for "passive transfers" and be sure that any institutional or personal fire wall will accept connections from the servers. Connect to one of the following computers:

LAACG1.LANL.GOV	(IP address 204.121.24.18), or
LAACG2.LANL.GOV	(IP address 204.121.24.19)

Log in as follows:

User name:	SFUSER
Password:	ftpsuperfish

The password is case sensitive. You must use lower case. Please note that this is not a web site. Anonymous connections via web browsers will not work. Try the following connection from your web browser:

```
FTP://SFUSER:ftpsuperfish@LAACG1.LANL.GOV/
```

If this method does not work, we recommend using an FTP tool such as WS_FTP (available from http://www.ipswitch.com). See the _READ_ME.1ST file on the FTP server for more information about downloading files.

When you log in as SFUSER, directory PoissonSuperfish\Version7 contains the latest distribution of Poisson Superfish. If your software has trouble “seeing” directories or files, check whether you can set the host type in your FTP software. The host computer runs Windows NT. Try setting the host type to NT. When downloading files, be sure to set the transfer mode to “binary” or equivalent in your FTP software. Most of the files are not ASCII files and they cannot be transferred as ASCII.

4. Email notification of new features and bug fixes

We maintain an electronic mailing list of registered users of Poisson Superfish. Registered users receive notification of new features and bug fixes by email. If you obtained your original copy of the codes from another user, you may wish to register with us as a new user. To register as a Poisson Superfish user, please include your postal mailing address, phone and fax numbers in an email message to Superfish@lanl.gov.

You can check the FTP server occasionally to see if there have been any updates to the codes. The file Changes.SF7 lists changes, additions, and bug fixes in reverse chronological order for the Poisson Superfish version 7. Table II-4 lists other Changes files with similar information for the other Los Alamos codes:

Table II-4. Files listing code changes.

File	Lists changes, additions, and bug fixes in:
Changes.SF7	Poisson Superfish, version 7.
Changes.PM	Parmela and related codes (Electron Linac Design Codes).
Changes.PMI	Parmila and related codes (Ion Linac Design Codes).
Changes.RFQ	Radio Frequency Quadrupole Design Codes.
Changes.TR	TRACE 2-D and TRACE 3-D codes.

F. Files provided for program developers

We provide files for several language systems that allow users to call the Poisson Superfish field interpolator from their own programs. Table II-5 lists the directories for the supported compilers. Most Fortran compilers support linking with object modules made with one or more of these languages, which provides a way for your Fortran code to call the DLL. Each directory contains a dynamic link library (DLL), the object module used to create the DLL, an import library, and (if available) an example program that calls the DLL. The files and sample programs described in this section are provided “as is” without additional documentation except that found here. Los Alamos National Laboratory cannot provide consulting for source-code developers.

The LF95 compile and link operation creates an import library for each version of the DLL. However, the library may be incompatible with your language system. Please refer to documentation of mixed language programming for the compiler you are using. Most languages provide instructions for creating an import library from the DLL and some additional information. The additional information may consist of the object module used to create the DLL or a skeleton program that you can create yourself. A skeleton program contains each function statement including argument calling sequences and declarations, a dll_import (or equivalent) statement, and an end statement, but no other source code. In case it makes a difference in your language system, all the functions names in the DLL use upper case letters (even though, for readability, we do not use all caps in this document).

In some programming languages, the calling program must refer to the functions by their "mangled" name. You can check the exported DLL function names by running Microsoft program DumpBin.exe, which is distributed with many compilers.. For example, the command (executed in a CMD.EXE window)

```
dumpbin     /exports     fld_msvc.dll
```

finds the following function names in the DLL for Microsoft Visual C++:

```
_GETTITLE@4
_INTERP@36
_I_VARIABLE@4
_R_VARIABLE@4
```

The number following the "@" character is the number of bytes in the argument list for the function. We have provided the output from DumpBin.EXE in the Readme_xxx.txt files found in the Table II-5 directories (xxx).

Table II-5. Contents of DeveloperFiles\PoissonSuperfish subdirectories

Directory	Description
Abs	DLL for Absoft Fortran version 8.0 and later.
BC	DLL for Borland C++, version 5.0 and later (also works with Borland C).
BD	DLL for Borland Delphi, version 2.0 and later.
CVF	DLL for Compaq (Intel) Visual Fortran (developed with version 6.6b).
LF90	DLL for Lahey Fortran 90, version 2.01 and later.
LF95	DLL for Lahey/Fujitsu Fortran 95 (5.0 and later) and source files and Fortran library for Lahey/Fujitsu Fortran 95 (version 5.7).
MSVB	DLL for Microsoft Visual Basic, version 4.0 and later.
MSVC	DLL for Microsoft Visual C++, version 2.0 and later.

If you have Lahey/Fujitsu Fortran LF95 version 5.7 plus the ***Winteracter*** development tools, you also may link your codes with library LANLSF.Lib found in the LF95 directory. Using the DLL should be simpler and is the recommended approach. The DLL works with any version of LF95, but library routines are compatible only with LF95 version 5.7.

1. Using the dynamic link library

The DLL contains four functions. To interpolate the fields at point (X,Y), the user supplies to function INTERP the name of the Poisson Superfish solution file and the coordinates X and Y. The function returns the interpolated solution value (H for rf problems, A for magnet problems, V for electrostatic problems), and the interpolated X and Y field components. For magnet problems, the function also returns derivatives of the field components. Functions I_Variable and R_Variable return integer and real values, respectively, of Poisson Superfish problem variables. The only input parameter is the 8-byte character string containing the name of a problem variable listed in Table III-3 or Table III-4. Function GetTitle returns the ten 80-character title lines for the problem.

Figure II-1 contains a skeleton Fortran 90 program listing for the DLL functions, showing the calling sequence and variable declarations. Create an equivalent listing in the language you are using to serve as a skeleton program for creating an import library.

```
INTEGER (KIND=4) FUNCTION INTERP(SolutionFile,Xvalue,Yvalue,&
Solution,Xcomponent,Ycomponent,DBYDY,DBYDX,DBXDY)
IMPLICIT NONE
DLL_EXPORT INTERP
CHARACTER (LEN=256),INTENT(IN)      :: SolutionFile
REAL (KIND=8),INTENT(IN)            :: Xvalue,Yvalue
REAL (KIND=8),INTENT(OUT)           :: Solution,Xcomponent,Ycomponent
REAL (KIND=8),INTENT(OUT)           :: DBYDY,DBYDX,DBXDY
END FUNCTION INTERP

INTEGER (KIND=4) FUNCTION I_VARIABLE(VariableName)
IMPLICIT NONE
DLL_EXPORT I_VARIABLE
CHARACTER (LEN=8),INTENT(IN)        :: VariableName
END FUNCTION I_VARIABLE

REAL (KIND=8) FUNCTION R_VARIABLE(VariableName)
IMPLICIT NONE
DLL_EXPORT R_VARIABLE
CHARACTER (LEN=8),INTENT(IN)        :: VariableName
END FUNCTION R_VARIABLE

INTEGER (KIND=4) FUNCTION GETTITLE(Title)
IMPLICIT NONE
DLL_EXPORT GETTITLE
CHARACTER (LEN=80),DIMENSION(10),INTENT(OUT) :: Title
END FUNCTION GETTITLE
```

Figure II-1. Fortran 90 declarations for DLL functions.

Please note the length of character variables in the argument lists. When creating mixed-language DLLs, the Lahey/Fujitsu Fortran 95 compiler does not allow character-type arguments with variable length. If the string length supplied to the DLL function differs from the declared length, the results may be unpredictable.

Functions in the dynamic link library can write diagnostic information that may be of use in developing your program to file DiagDLL.txt. To request the diagnostic information, set REG variable DIAGDLL = 1 in the Automesh input file used to create the solution file. The information consists of the function name followed by the input and returned values of calling parameters if no error occurred. If an error occurred, then function INTERP writes the value of the error code (see Table II-6), and functions I_Variable and R_Variable write information about the type of problem encountered.

a. Function INTERP

When calling the INTERP function, the first calling parameter is a 256-byte string containing the name of the Poisson Superfish solution file for which you want interpolated fields. The next two calling parameters are the X and Y coordinates of the interpolation point. Variables Xvalue and Yvalue have the same units used in the Automesh input file. Function INTERP checks whether the supplied file is the last solution file opened. If so, it calls the field interpolator immediately. If not, the function calls the field interpolator after successfully opening and reading the new file. (In order to retrieve values of problem variables using functions I_Variable or R_Variable, or to retrieve the problem title using function GetTitle, you must first call INTERP at least once to open the solution file.)

The returned function value (see Table II-6) indicates success or failure of the operation. Negative return values indicate errors detected by the INTERP function and three-digit positive values are error codes returned by Poisson Superfish routines called by the INTERP function. (There may be more errors detected than listed here. Please report any errors not included in Table II-6.)

Table II-6. Function INTERP return values.

Value	Description
0	The operation completed successfully.
–1	The solution file does not exist.
–2	Unable to allocate solution arrays, usually not enough memory.
–3	The solution does not include the point X,Y.
–4	The DLL could not determine the solution file version.
–5	The DLL cannot read version 6 solution files.
207	An error occurred reading the first record from the Poisson Superfish solution file.
208	Solution arrays have not been properly declared. Please report this error.
209	The dimension of at least one solution array is too small. Please report this error.
210	Mesh point arrays are not large enough. Please report this error.
216	Unable to read triangular mesh data (possibly incompatible code versions).
217	The solution file for a Superfish problem contains only mesh data and no solution arrays.
218	The solution file for a Poisson problem contains only mesh data and no solution arrays.
220	Reached the end of the solution file unexpectedly. Rerun problem stating with Automesh.
221	An error occurred reading a record from the solution file. Please report this error.
222	The Poisson Superfish solution file is obsolete.

b. Functions I_Variable and R_Variable

Function I_Variable or R_Variable requires only the calling parameter VariableName, which is an 8-byte character string containing the name of a problem variable listed in

Table III-3 for Superfish problems or Table III-4 for Poisson problems. If the variable exists, its value is returned as the function value. Function I_Variable returns a 4-byte integer, and R_Variable returns an 8-byte real value (i.e., double precision). If the requested variable does not exist (for example, if you request a Superfish variable from a Poisson solution file), then I_Variable returns the value −999999 and R_Variable returns the value −1.0D+99. You can call either function for any problem variable. The R_Variable function always returns the variable value. However, if you request an integer value of a real variable and the result would be outside the range of a 4-byte integer, then I_Variable returns −999998. I_Variable returns the value −999997 if the requested variable is a real variable that has the "indefinite" value −1.0D+99, which means that it has never been set by any of the Poisson Superfish codes. These functions will write information about these possible outcomes to file DiagDLL.txt, if requested. To request this diagnostic information, set REG variable DIAGDLL = 1 in the Automesh input file used to create the Poisson Superfish solution file.

Before using one of these functions for the first time, the calling program must first call function INTERP to open the Poisson Superfish solution file. These functions do not check to see if a file has been read. If no file has been read, the function return the initialized default value of the variable.

Unless the calling program has access to dimensions from another source, the code will not know the limits of the problem geometry on the first call to INTERP. The code can obtain these dimensions later by calling R_Variable. For the first call, use any value for Xvalue and Yvalue on the INTERP calling line. The function reads the file and retains the data whether or not point Xvalue, Yvalue is in the problem geometry.

c. Function GetTitle

Function GetTitle requires only the calling parameter Title, which is a 80-byte character array of dimension 10. Before using function GetTitle, the calling program must first call function INTERP to open the Poisson Superfish solution file. The function returns ten title lines read from the Automesh input file. If the problem title contained fewer lines, the remaining lines are blank filled (ASCII character 32).

d. Multiple DLLs

Subroutine INTERP in the dynamic link library will automatically switch between different Poisson Superfish solution files whenever the name of the file on the calling line changes. However, switching between files very often will cause an application to run very slowly. To avoid this problem, an application can keep open up to 4 different solution files simultaneously if the program is linked with additional DLLs provided separately in WinZip file MultiDLL.ZIP on the LAACG FTP servers. A text file in each subdirectory listed in Table II-5 lists the additional files provided for each language system.

An application can open an unlimited number of different solution files sequentially using only one DLL. This approach works well if the application only needs to open each solution file once (or just a few times) and interpolate fields at many points while the file remains open. Multiple DLLs are useful if the application needs to switch back and forth many times between solution files, interpolating only a few points each time the file is opened. Opening the solution file takes at least 20 times longer than interpolating fields at a point once the file is open.

After downloading file MultiDLL.ZIP, extract the additional files to the appropriate subdirectory. The ZIP file includes subdirectory names, so you can extract all the files to their proper subdirectories, if desired. Or, you may extract only the files you need. For example, for a Microsoft Visual C++ application, extract the following additional files to the MSVC subdirectory:

Dynamic link libraries:	fld1_msvc.dll,	fld2_msvc.dll,	fld3_msvc.dll
Object modules:	fld1_msvc.obj,	fld2_msvc.obj,	fld3_msvc.obj
Import libraries:	fld1_msvc.lib,	fld2_msvc.lib,	fld3_msvc.lib

Each DLL contains the same the four functions as the original file fld_msvc.dll, but the functions all have the number 1, 2, or 3 appended to their names. For example, file fld1_msvc.dll contains functions INTERP1, I_Variable1, R_Variable1, and GetTitle1. Use as many additional DLLs as needed in your application. Be sure to declare all the new function names following the Fortran 90 example (or equivalent for other language systems) shown in Figure II-1. Link each numbered DLL in the same manner used for the original DLL.

e. *Example files*

We would like to provide an example main program that calls the DLL for each language system. However, we only have direct experience with the LF90 and LF95 compilers and limited experience with the Absoft compiler. Our colleague Harunori Takeda has developed the example for Borland C.

We are most grateful to Tibor Kibedi of the Australian National University for helping with debugging of the DLL and for developing the example for Compaq (Intel) Visual Fortran. We also thank Dr. Kibedi for suggesting the implementation of multiple DLLs, which is now (June, 2004) available for all the supported language systems. In April, 2004, Dr. Kibedi notified us that this compiler is now called Intel Fortran Compiler. Please let us know if you have any experience using the CVF DLL with newer versions of the Intel Fortran Compiler.

We thank Peter de Castro of the Thomas Jefferson National Accelerator Laboratory for supplying sample programs that use the DLL provided for Microsoft Visual C++ (in directory MSVC).

If a directory does not yet include an example, and you develop one for that language (or if you can suggest improvements for an existing example), please send us your example files by email (Superfish@lanl.gov). We will add your example to the code distribution with an acknowledgment in this section and in your example files. (Please let us know if you do not wish such an acknowledgment. Likewise, let us know if you would be willing to answer questions from other code users.)

The DLL example files for LF90, Absoft Fortran, and Borland C produce the same results as file Sample.F90 in the LF95 directory. The example reads the H-magnet solution file

HTEST1.T35 to create a simple Quikplot input file to display B_y versus x. The LF90 and LF95 example uses an identical file named Sample_DLL.F90 that can be compiled under either language. The example for Compaq (Intel) Visual Fortran requests X and Y coordinates and writes the interpolated results and the screen and to file OUT.TXT. The example for Microsoft Visual C++ reads solution file Septum.T35 (from Automesh input file SEPTUM.AM in directory Examples\Magnetostatic\Septum), interpolates the fields at (X,Y) = (5,5), and writes the results to file OUTPUT.TXT.

2. Calling Fortran LF95 library routines

Directory DeveloperFiles\PoissonSuperfish\LF95 contains Fortran-callable object modules for reading the binary solution file and for interpolating fields. The two subroutines in the library are RDTape35 and Interpolate. Subroutine RDTape35 reads the Poisson or Superfish solution file written by codes Automesh, Autofish, Fish, CFish, Poisson, Pandira, SFO and the tuning programs. All Poisson Superfish codes use subroutine RDTape35 to read the binary solution file. We also provide the source code for RDTape35 for programmers who wish to customize their application. Subroutine Interpolate is the Poisson Superfish field interpolator, which is used by all codes that compute field values from the solution arrays. Table II-7 lists files that are available for use in your own LF95 codes.

Source file Sample.F90 uses the H-magnet example file HTEST1.AM to create a simple Quikplot input file to display B_y versus x. File Sample2.F90 uses the coaxial waveguide example file COAXWG.AM to create a Tablplot input file to display the real part of the complex field components E_z and E_r versus z.

Table II-7. Files for use with LF95 main programs.

File	Description
RunSampl.Bat	Batch file for creating and running the sample programs.
LANLSF.Lib	Fortran library containing Interpolate, RDTape35 and modules.
Sample.F90	Source file using the interpolator on a Poisson problem.
Sample2.F90	Source file using the interpolator on a CFish problem.
T35Array.Inc	INCLUDE file used by codes with complex arrays.
*.mod	Fortran modules used by library routines and sample programs.
SF_RDT35.F90	Subroutine for reading the binary solution file.
SF_RDR1.F90	Subroutine for reading the first record from the solution file.
LF95.Fig	Lahey/Fujitsu Fortran configuration file.

a. Batch files for running the sample programs

Batch file RunSampl.Bat compiles and links Sample.F90 and Sample2.F90 with the field interpolator and with the RDTape35 subroutine in library LANLSF.Lib. The batch file has been written for the Lahey/Fujitsu Fortran compiler LF95 available from Lahey Computer Systems, Inc.

After compiling both sample programs, the batch file prepares to run SAMPLE.EXE, which reads the solution file from the H-Magnet\HTEST1 example. The H-Magnet directory is a subdirectory of LANL\Examples\Magnetostatic. If RunSampl.Bat has already run in the DeveloperFiles directory, solution file HTEST1.T35 will already exist. The batch file skips running the codes, but, if necessary, it will run Automesh and Poisson to create the solution file. Program SAMPLE creates file HTEST1.QKP, and the command Quikplot HTEST displays the data in HTEST1.QKP.

After exiting the Quikplot program, the batch file prepares to run Sample2.EXE, which reads the solution file from the CFish\COAXWG example. The CFish directory is a subdirectory of LANL\Examples\RadioFrequency. If necessary, the batch file will run Automesh, CFish, and SFO to create the solution file COAXWG.T35 and normalize the fields. Program Sample2 creates file COAXWG.TBL, and the command Tablplot COAXWG displays the data in COAXWG.TBL. After exiting the Tablplot program, the batch file stops.

b. *LF95.Fig, configuration file*

We provide file LF95.Fig shown in Figure II-2 as an example of a configuration file for Lahey's LF95 Fortran compiler. The ***Winteracter*** subroutines need routines contained in the libraries listed on the last several lines. To compile the sample programs, the LF95.Fig file in the BIN subdirectory of your own LF95 version 5.7 installation must include similar settings. Change the directory as required for your ***Winteracter*** installation.

```
# LF95.Fig
# If the linker cannot find the libraries, add path information
# in this file or add the path to the LF95 library directory to
# the LIB environment variable (e.g. LIB=C:\LF9557\Lib).
 -tpp  -f95   -swm 3125,3205,2315,3100
 -nap -nchk -ncover -ng -o1 -npca -nsav -nstchk -ntrace -nzero
 -mod c:\wint\lib.l9m
 -maxfatals 5
 -wo  -win
 -lib c:\wint\lib.l9m\winter.lib
 -lib comdlg32
 -lib winmm
 -lib winspool
```

Figure II-2. Example LF95.Fig file.
This example assumes that the *Winteracter* software has been installed in directory WINT on drive C.

c. *INCLUDE file T35ARRAY.INC*

File T35Array.Inc contain the statement "USE ComplexTape35Arrays" for use with subroutine RDTape35 and the example files Sample2.F90 and Sample2.F90. (Some Poisson Superfish codes require the statement "USE RealTape35Arrays" instead.)

d. *Fortran modules*

The files with extension .MOD are Fortran 95 module files (needed at compile time) that correspond to object modules (needed at link time) contained in the library file LANLSF.Lib. Table II-8 gives a brief description of these modules. Subroutines Interpolate and RdTape35 (whose object modules also are contained in LANLSF.Lib) use these modules. A few of the modules also appear in USE statements in the sample programs Sample.F90 and Sample2.F90. The compiler will also expect to find your Winteracter modules in the location specified by the -mod and -lib switches in file LF95.Fig (see Figure II-2).

In the next few subsections section we describe briefly the type of data in some of these modules. We provide definitions for some of the variables in the modules.

Table II-8. Fortran modules.

Module name	Description
PoissonSuperfishData	Problem variables used in Poisson Superfish codes.
InterpolatedFields	Variables used by the Interpolate subroutine.
RealTape35Arrays	Solution-file arrays for a real potential.
ComplexTape35Arrays	Solution-file arrays for a complex potential.
MaterialData	Variables related to material properties.
SF_CTRL	Control variables used in all Poisson Superfish codes.
SF_Menued	Variables used in Winteracter resource file.
LineParserData	Free-format line parser and its variables and arrays.
FormatData	Subroutines and data that help format output lines.
FileInformation	Subroutines that obtain file information from the operating system.

SF_CTRL, control variables

Most programs and subroutines in the Poisson Superfish codes share a number of control variables in Fortran module SF_CTRL. Main programs define the parameters upon starting. Other routines make decisions or write meaningful messages based upon the parameter values. Table II-9 lists some of the parameters from this module that may be of interest to code developers.

Table II-9. Variables module SF_CTRL.

Variable	Value	Used	Description
IGNOREB	False		Line parser ignores blank lines if true.
NIN	5	R	Input unit number.
NOUT	–1	R,I	If positive, NOUT is the unit number of an output file.
KERROR	0		Error code reported by last routine to detect an error..
KPROBR	–1	R	KPROB value required by solver codes (–1 for others).
COM	Blank		Character buffer.
DATAFILE	Blank	R	Filename character buffer.
LCODE	Blank	R	Code currently running under Autofish or a tuning program.
MAINCODE	Blank	R	Name of the main program.
RFCODE	False	R	True for rf solver codes including tuning codes.
MAGCODE	False		True for solver codes Poisson and Pandira.
POSTPROC	True	R	True for postprocessors SFO, SF7, WSFplot, and Force.
TUNING	False		True for tuning codes only.
COMBINED	False		True for Autofish and tuning codes.
CHECK35	True	R	If true, RDTape35 verifies the length of supplied arrays.
FINDEF	–1.0D+99		Initial value of certain variables.
FTOLER	1.0D+84		Tolerance for comparisons with FINDEF.

The initial values in the second column are those set by codes that call subroutine Interpolate in the library LANLSF.Lib. These initial values mimic the postprocessors SFO and SF7, except that the character variables MAINCODE and LCODE are blank. If a code calls only RDTape35, but not Interpolate, the values in SF_CTRL will be undefined. Subroutine RDTape35 uses variables marked with letter R in the third column; subroutine Interpolate uses only the output-file unit number NOUT (marked with letter I), and then only if the interpolator fails and needs to report an error. If your program opens an output file for messages, you can set the value of NOUT to the appropriate unit number to allow RDTape35 and Interpolate to write to this file.

Table II-9 refers to two REAL (KIND=8) variables FINDEF and FTOLER. The codes use FINDEF to initialize certain variables so that codes can later tell if the user has supplied a value (or if a code has calculated a value). Variable FTOLER, the tolerance in comparisons of a number with FINDEF, is about 8 times the spacing between adjacent 8-byte real numbers in the vicinity of FINDEF.

ComplexTape35Arrays and RealTape35Arrays

Each Poisson Superfish code uses one of the two modules ComplexTape35Arrays or RealTape35Arrays. The two modules are nearly the same except that some arrays and variables are declared as COMPLEX (KIND=8) in one module and REAL (KIND=8) in the other module. Program CFish and the postprocessors SFO, SF7, and WSFplot require the complex version of the module. All other programs use the real version. Like all other modules and subroutines in the Poisson Superfish codes, the Tape35Arrays modules include the statement

```
      IMPLICIT NONE
```

and all variables are declared and initialized. Table II-10 lists arrays in modules ComplexTape35Arrays and RealTape35Arrays. The types of variables are as follows:

- Most variables starting with letters A-H and O-Z are REAL (KIND=8), except as noted elsewhere.
- Most variables starting with letters I-N are INTEGER (KIND=4), except as noted elsewhere.
- Variables AVECTOR, REPS, and FMU are COMPLEX (KIND=8) in the ComplexTape35Arrays module and REAL (KIND=8) in the RealTape35Arrays module.
- Variable ComplexFields is LOGICAL(KIND=4).
- Variable TPNAME is CHARACTER (LEN=20).
- Variables IREGUP, IREGDN, and KFILT are INTEGER(KIND=2).
- Variables KSCAT and NGO are INTEGER(KIND=1).

The calling program must allocate the correct set of arrays before calling RDTape35. A code might use only some of the arrays. The caller also sets the last element in each array to a particular value so that subroutine RDTape35 can verify that a long enough array has been allocated. The Sample2.F90 and Sample2.F90 programs illustrate this procedure.

Program Automesh determines the array dimensions listed in Table II-11. The Description column in Table II-11 mentions which Poisson Superfish codes use arrays of each dimension, and notes specifically whether the field interpolator requires the arrays of that dimension. Programmers writing their own postprocessor must provide all of the arrays dimensioned ITOT1 in Table II-10 even if their code does not call Interpolate. Programmers need only define other arrays actually required by their program. Arrays that the calling code actually requires must be declared large enough to hold all the data read by RDTape35. Placeholder arrays can be a single variable.

Table II-10. Arrays in ComplexTape35Arrays and RealTape35Arrays.

Array	Description
XX(ITOT1)	X coordinate.
YY(ITOT1)	Y coordinate.
IRLAX(ITOT1)	Ordered set of pointers to the other array elements.
IREGUP(ITOT1)	Material number for a point's upper triangle.
IREGDN(ITOT1)	Material number for a point's lower triangle.
KSCAT(ITOT1)	Dirichlet boundary-point indicator.
KFILT(ITOT1)	Current filament indicator.
NGO(ITOT1)	Indicates 1, in empty space; 2, on interface; or 3, in iron.
AVECTOR(ITOT1)	The solution array.
SOURCE(ITOT2)	Original AVECTOR array (Poisson and Pandira).
GTU(ITOT3)	Reluctivity γ for a point's upper triangle (Poisson and Pandira).
GTL(ITOT3)	Reluctivity γ for a point's lower triangle (Poisson and Pandira).
JNDEX(NWTOT)	Ordered set of pointers for coupling calculations (Poisson).
JUP(NWTOT)	Pointers to the upper triangle associated with JNDEX (Poisson).
MAT(0:NAREG)	Material numbers for each region.
DEN(0:NAREG)	Current densities for each region.
AVECINI(NBSPL)	Initial values of imaginary part of complex AVECTOR array.
NBDRIVE(NBSPL)	Pointers to locations in AVECTOR for the AVECINI array.
NBENDS(2,NASEG)	Pointers to starting and ending points in IROWBOUBD for segments.
IROWBOUND(NABOUND)	List of mesh-point indices from the IRLAX array along boundaries.
TPVALUE(INFOSIZE)	Tuning program setup parameter values.
TPNAME(INFOSIZE)	Tuning program setup parameter names.
REPS(0:MAXTABLE)	Reciprocal of relative permittivity.
FMU(0:MAXTABLE)	Relative magnetic permeability.

Reluctivity in arrays GTU and GTL is the reciprocal of the permeability: $\gamma = 1/\mu$. Program Automesh sets up the order of calculations performed in the solver programs and stores the results in array IRLAX. Automesh also stores iron-region coupling order information in arrays JNDEX and JUP for Poisson problems.

PoissonSuperfishData, problem variables

The PoissonSuperfishData module contains problem variables listed in Table III-3 for Superfish problems and in Table III-4 for Poisson problems. Table II-12 defines several other variables associated with the problem. Unlike previous versions of Poisson Superfish, the individual variables are not equivalenced to array elements. For convenience, codes that read and write the solution file copy the variables into arrays before writing the file and extract the variables from the arrays when reading the file.

Table II-11. Variables in ComplexTape35Arrays and RealTape35Arrays.

Dimension	Description
ComplexArray	Logical variable set false in the real version, true in complex version.
ITOT1	Total number of mesh points. Arrays dimensioned ITOT1 in Table II-10 are used in all Poisson Superfish codes. Subroutine RDTape35 always returns these arrays.
ITOT2	Dimension of the SOURCE array. Used in solver codes and postprocessors except Fish and the tuning programs. All calling programs that use Interpolate must retrieve the SOURCE array as illustrated in Sample.F90 and Sample2.F90.
ITOT3	Dimension of the reluctivity arrays GTU and GTL. Used in Poisson and Pandira.
NBSPL	Number of fixed-potential points. Used in CFish and WSFplot.
NASEG	Total number of segments, including those created by the addition of line regions. Used in SFO and tuning codes.
NABOUND	Total number of boundary points. Used in SFO and tuning codes.
NAREG	Number of regions. Arrays dimensioned NAREG in Table II-10 are used in all Poisson Superfish codes. All calling programs that use Interpolate must retrieve arrays MAT and DEN as illustrated in Sample.F90 and Sample2.F90.
INFOSIZE	Number of tuning code parameters written in the SFO output file. Used in SFO and tuning codes.
NWTOT	Dimension of Poisson coupling arrays.
MAXTABLE	Upper dimension of material table arrays in module MaterialData. Set by Automesh.

Table II-12. Variables in module PoissonSuperfishData.

Variable	Description
NTITLE	Number of title lines supplied in TITLE.
IPROBSIZE	Number of bytes in the input file.
TITLE(10)	Array of 80-character title lines for the problem.
ORIGINATINGCODE	Name of the program reading or creating the input file.
PROBTYPE	The type of problem being solved.
PROBFILE	The complete path to the problem file read by Automesh.
PROBDATE	Date and time of the problem file read by Automesh.

The array CONVDATA in the PoissonSuperfishData module contains variables that have dimensions of length, area, or volume converted to cm, cm^2, or cm^3. The codes use these parameters instead of the variables in Table III-3 or Table III-4. Original variables retain the dimensions supplied by the user in the parameter CONV. Each variable has the same name as the original name with the letters “CM” appended. Several variables in CONNVDATA are certain fractions of DXMINCM and DYMINCM, which are the smallest X and Y mesh intervals in the geometry. Variables DX1PC, DY1PC are respectively 1% of the minimum values of the mesh intervals in the problem, DX001PX, DY001PC are two orders of magnitude smaller, etc. Poisson Superfish codes use these variables as tolerances in comparisons between coordinates.

Table II-13. Variables in module InterpolatedFields.

Variable	Type	Description
H	C8	Fitted result for H_z or H_ϕ in rf field problems.
EX [EZ]	C8	Electric field component E_x or E_z for rf field problems.
EY [ER]	C8	Electric field component E_y or E_r for rf field problems.
AFIT	R8	Fitted result for A_z or A_ϕ in magnet problems.
BX [BR]	R8	Magnetic field component B_x or B_r for magnet problems.
BY [BZ]	R8	Magnetic field component B_y or B_z for magnet problems.
VFIT	R8	Fitted result for V in electrostatic problems.
EX_ES [ER_ES]	R8	Electric field component E_x or E_r for electrostatic problems.
EY_ES [EZ_ES]	R8	Electric field component E_y or E_z for electrostatic problems.
DBYDY	R8	Magnetic field partial derivative $\partial B_y/\partial y$.
DBYDX [DBZDR]	R8	Magnetic field partial derivative $\partial B_y/\partial x$ or $\partial B_z/\partial r$.
DBXDY [DBRDZ]	R8	Magnetic field partial derivative $\partial B_x/\partial y$ or $\partial B_r/\partial z$.
FIELDINDEX	R8	Index $n = (r/B_z)*(dB_z/dr)$.
PP(12)	C8	Fitted coefficients for the fit to H, A, or V.
PPMAX	R8	Magnitude of the largest term in PP.
PP_RATIO	R8	Ratio of the highest-order PP to the maximum-magnitude PP.
PP_CUTOFF	R8	Interpolator stops adding terms if PP_RATIO < PP_CUTOFF.
CHIS	R8	Normalized Chi-squared for the PP fit.
CHISDF	R8	Normalized Chi-squared per degree of freedom for the fit.
XEDIT	R8	X coordinate of the interpolation point.
YEDIT	R8	Y coordinate of the interpolation point.
INREGION	L4	True if XEDIT,YEDIT is in the problem geometry.
NEUMANN	L4	True if a Neumann boundary condition was imposed.
DIRICH	L4	True if a Dirichlet boundary condition was imposed.
NFUNC	I4	Function number used in the fit to H, A, or V.
NTERMS	I4	Number of terms in the polynomial fit.
NEARESTIROW	I4	Index number of the mesh point closest to XEDIT,YEDIT.
IROWREF	I4	Index number of the reference mesh point KPOINT(1).
MATNUM	I4	Material number of the triangle containing XEDIT,YEDIT.
NPO	I4	Number of mesh points used in the fit.
KPOINT(19)	I4	Indices of the mesh points used in the fit.

InterpolatedFields, variables used in the field interpolator

Table II-13 lists some of the variables declared in the InterpolatedFields module. The module includes other variables of no particular interest to a user of the field interpolator. In the Type column, C8 stands for COMPLEX (KIND=8), R8 stands for REAL (KIND=8), I4 stands for INTEGER (KIND=4), and L4 stands for LOGICAL (KIND=4). Programs that call the field interpolator obtain the fitted results from variables in this module. For convenience, some variables have equivalent names, which appear in square brackets [ ].

MaterialData, Fortran module

The MaterialData module contains declaration statements for materials used in all types of Poisson Superfish problems. All of the arrays are allocatable, and must be allocated before a program can reference the arrays. Program Automesh allocates the material data arrays according to requirements of the input file. Other codes must allocate arrays of the same size so that subroutine RDTape35 can read the data from the material-data records in the solution file. After calling subroutine RDRECORD1, the array dimensions are available as two variables in the PoissonSuperfishData module. MAXMAT is the highest material number and MAXTABLE is the maximum number of material tables. The example files Sample2.F90 and Sample2.F90 both include the sequence of steps necessary to allocate the material arrays.

Most of the data are for magnetic materials used in Poisson and Pandira. A routine that calls RDTape35 can access the material data by including the following USE statement immediately after the PROGRAM or SUBROUTINE statement:

```
      USE MaterialData
```

Superfish problems use only two of the arrays (EPSMT and FMUMT). The rest are for use in Poisson and Pandira problems. There are different types of arrays distinguished by how the arrays are indexed. Some arrays are indexed by material ID number and some are indexed by material number. Table II-14 lists the material arrays. The Shape column refers to Table II-15 which lists the rank and extents of the arrays. Note the relationship among region numbers, material numbers, and material ID numbers. Automesh assigns region numbers sequentially to the boundary defined after each REG namelist. The array MAT, which is indexed by region number, stores the material numbers. Array MATTBLI is indexed by material number and stores the material ID numbers, which are pointers to locations in other arrays (e.g., GAMPERMT).

Arrays EPSMT(MAXTABLE) and FMUMT(MAXTABLE) are the complex permittivity and permeability for Superfish problems. The index is a material number from the solution-file array MAT(0:NREG) found either in module ComplexTape35Arrays or module RealTape35Arrays. For Superfish problems, there is no distinction between material number and material ID number. Data from EPSMT and FMUMT must be transferred to two other arrays, which are used in postprocessors. One array is the reciprocal of the permittivity REPS(0:MAXTABLE) and the other is the permeability FMU(0:MAXTABLE). These arrays are in module ComplexTape35Arrays or in module RealTape35Arrays. Array elements REPS(0) and FMU(0) are equal to zero for convenience in making comparisons in some codes. Values in the REPS and FMU elements with a nonzero index are dimensionless numbers, relative to the material properties of air $1/\varepsilon_0$ and μ_0. The example file Sample2.F90 shows how to transfer the data from the solution-file arrays to the REPS and FMU arrays.

Table II-14. Material arrays indexed by material ID number.

Variable(s)	Shape	Description
BT,GT	1	Tables of B,γ data. Each table has 201 points.
EPSMT	2	Complex permittivity ε for Superfish problems.
FMUMT	2	Complex permeability μ for Superfish problems.
AEASYMT	3	Easy axis direction in degrees for an anisotropic or permanent-magnet material.
GAMPERMT	3	Relative reluctivity perpendicular to the easy axis.
X0AMT, Y0AMT	3	Coordinates of the center of the circle used with the option that allows a variable direction of the easy axis.
PHIAMT	3	Angle in degrees for the above option.
AMULTMT	3	Polar-angle multiplier for the above option.
HCEPTMT	3	H_c intercept for anisotropic or a permanent-magnet material.
BCEPTMT	3	B_r intercept for anisotropic or a permanent-magnet material.
GAMMAMT	3	Reluctivity for magnet problems with fixed permeability.
MTID	3	Material table ID number.
MTLENGTH	3	Length of (B,γ) material tables BT and GT stored in the binary solution file.
MTYPEMT	3	Original form of the user-supplied data in arrays BT and GT. A value of 1 means the original data was of the form (B,γ), 2 means (B,μ), and 3 means (B,H). Automesh converts the data of types 2 and 3 to (B,γ).
STACKI	4	Stacking factors for each material.
MATTBLI	4	Material ID number for each material. For Poisson problems, MATTBLI can point to any one of the internal BT,GT tables or to a user-supplied table.
MSHAPE	4	Material property indicators for each material. MSHAPE = –1 corresponds to fixed permeability, MSHAPE = 0 to variable permeability, MSHAPE =1 to an anisotropic material with fixed permeability, and MSHAPE = 2 to an anisotropic material with variable permeability for the easy axis.

Table II-15. Shape of arrays in Table II-14.

Shape	Allocation rank and extent
1	(201,MINTABLE:MAXTABLE)
2	(MAXTABLE)
3	(MINTABLE:MAXTABLE)
4	(0:MAXMAT)

LineParserData, Fortran module

The LineParserData module contains declaration statements used by subroutine BREAK0, which is a general purpose line parser used in many Los Alamos programs. Code developers who use the library LANLSF.Lib are welcome to call this routine in their applications. We provide here a brief description of the line parser subroutine. Another subroutine BREAK1 parses the command line. Here is the calling line for these subroutines:

```
CALL BREAK0([iUnit, IsolatedSign, TrailingSign, IgnoreComments])
CALL BREAK1
```

where the brackets indicate that all calling parameters (see Table II-16) are optional. Subroutine BREAK1 has no calling parameters. The line to parse is in character variable

JCHR. In BREAK0, if iUnit is not a positive number or is not present, then the calling program has already placed the string to parse in JCHR. If iUnit is positive, the code reads a line of text into JCHR from the file connected to that unit number. If IgnoreComments is present and .TRUE., then the code strips off all characters starting with the first semicolon (;) or exclamation mark (!) before parsing the remaining line. You can select other comment indicators by setting variable CommentCharacter, which is a 2-character string. For example, to make the slash and backslash the comment indicators insert the following line in the calling program before calling the line parser:

```
        CommentCharacter='/\'
```

If you want only one character as the comment indicator, put it in both places as the code always looks for both characters in the string. For example, to make only the semicolon the comment indicator, use the following assignment:

```
        CommentCharacter=';;'
```

Variable OriginalLine always contains the original value of the string JCHR and OriginalLength is the length of string OriginalLine (up to the trailing blanks). Variable ParsedLine is another copy of the final string JCHR that is parsed, and ParsedLength is the length of string ParsedLine, which is the same as NCHR unless the code uses NCHR to indicate an error. The default value for IgnoreComments is .FALSE.

Variables IsolatedSign and TrailingSign tell BREAK0 how to deal with isolated algebraic signs, i.e., plus (+) or minus (–) characters separated from other characters by blanks or by other nonblank delimiters. By default, BREAK0 identifies an isolated sign or trailing sign as a one-character-long string. Table II-16 lists the other choices. The default value of TrailingSign is the value of IsolatedSign, so you need only supply the TrailingSign calling parameter to select different treatment for an isolated sign at the end of the parsed string. Any value for IsolatedSign or TrailingSign other than 1, 0, –1, and –2, has the same meaning as the default setting of 2.

Table II-17 describes variables in module LineParserData. Most variables are set by the subroutine for use in the calling program. Elements in the character array ITYPE indicate the type of returned parameter. If ITYPE = “R” or “I” the corresponding locations in arrays ANUMB and NUMB contain REAL (KIND=8) and INTEGER (KIND=4) values, respectively. Variable NUMB contains the integral part of the real number if the number is not actually an integer. If ITYPE is blank, then the parameter is neither an integer nor a real number. The following code fragment shows an example in which the code gets a number from the fourth parameter on a line read from unit 15. The calling program places the result in the REAL (KIND=8) variable PARAM, but the entry in the text file might have been an integer or a real number:

```
        CALL BREAK0(15)
        IF(ITYPE(4).NE.' ')PARAM=ANUMB(4)
```

The subroutine returns REAL (KIND=8) values in array ANUMB for all numerical entries (including integers). It sets ITYPE to R for entries containing a decimal point and for entries expressed in scientific notation (e.g., –1.2E03, 2.34E–6, +2.34D–6, 250E+1).

A number in scientific notation has a “D” or “E” (in upper or lower case) preceded by at least one numeral and followed by an optional algebraic sign and at least one numeral. A plus or minus sign may be the first character of a number. The single letter “D” or “E” is character variable. A parameter that is neither integer nor real is, by default, a character string.

Table II-16. Meaning of the line-parser calling parameters.

Parameter	Description
iUnit > 0	Parses a line read from the file connected to unit IUN .
iUnit =≤ 0 (or not present)	Parses the string JCHR supplied by the calling program.
IgnoreComments = .TRUE.	Strip off comments (after semicolon or exclamation mark) before parsing.
IgnoreComments = .FALSE.	Default setting, calling parameter not required.
IsolatedSign = 2	Default setting, identify an isolated + or – character as a string.
IsolatedSign = 1	Return integer value –1 for isolated – character and +1 for isolated + character.
IsolatedSign = 0	Return integer value zero for an isolated + or – character.
IsolatedSign = –1	Ignore blanks between an isolated + or – character and a following valid number (integer or real). Apply the algebraic sign to following numeral entry, provided that entry does not begin with an algebraic sign. If the next parameter is not a number, then identify the + or – character as a string.
IsolatedSign = –2	Flag an isolated + or – character as an error, returning NCHR = –93 with J1CHR equal to the location of the character in string JCHR.
TrailingSign = 2, 1, 0, –2	Same meaning as IsolatedSign for these values.
TrailingSign = –1	Same meaning as TrailingSign = 0 or IsolatedSign = 0.

The next code fragment retrieves a character variable from the second parameter on the line (you also can get the character representation of integer or real parameters in this way):

```
CALL BREAK0(15)
I1=MAX(1,LOCCHAR(2))
I2=MAX(I1,I1+NUMCHAR(2)-1)
CHR=JCHR(I1:I2)
```

By default, BREAK0 recognizes one or more spaces, a comma, or an equal sign as delimiters between parameters. It ignores consecutive blanks. Consecutive nonblank delimiters imply missing numeral zeros, which BREAK0 automatically inserts into the NUMB array. A leading nonblank delimiter means that the first parameter is an integer zero. A trailing nonblank delimiter means that the last parameter is an integer zero. For example, these two character strings are equivalent:

```
  ,10,;,word=STRING,   ,20,30,
0 10 0 0 word STRING 0 20 30 0
```

Table II-17. Line-parser variables available to the calling program.

Variable	Description
MaxNPBK	Maximum number of entities in a parsed string, currently 2000.
MaxJCHR	Maximum number of characters in a parsed string, currently 8000.
NPBK	Number of returned parameters.
NCHR	Number of characters found in the string JCHR, or an error indicator.
JCHR	Character string (maximum length MaxJCHR) to be parsed.
OriginalLine	Original value of string JCHR.
OriginalLength	Length of string OriginalLine.
ParsedLine	Copy of string actually parsed (same as final value of JCHR).
ParsedLength	Length of string ParsedLine.
CommentCharacter	A two-character-long string containing comment indicators: “;!”.
ITYPE(MaxNPBK)	Array that identifies parameter type in arrays NUMB and ANUMB.
NUMB(MaxNPBK)	Array of returned integers.
ANUMB(MaxNPBK)	Array of returned real numbers.
LOCCHAR(MaxNPBK)	Array of starting locations in JCHR of returned character variables.
NUMCHAR(MaxNPBK)	Array of lengths of returned character variables.
NDELIM	Number of nonblank delimiters in the DELIM array.
DELIM(32)	Character array containing nonblank delimiters.

You may change the nonblank delimiters in array DELIM before calling BREAK0. Up to 32 nonblank delimiters may be active at once. Set NDELIM equal to the number of elements in DELIM that you wish the code to use.

Table II-18 lists the error conditions returned by subroutine BREAK0 in variable NCHR. The calling code can check NCHR and issue appropriate error messages. The condition described by NCHR = –93 can only happen if the calling line contained one or both optional parameters IsolatedSign = –2 or TrailingSign = –2.

Table II-18. Meaning negative values of NCHR.

NCHR	Description
–91	Error occurred reading from file connected to unit iUnit.
–92	Reached the end of the file connected to unit iUnit.
–93	Line has an entry error (isolated sign) at character J1CHR.

e. RDTape35 and RDRECORD1, subroutines for reading solution files

Subroutine RDTape35 in library LANLSF.Lib reads the Poisson Superfish binary solution file written by codes Automesh, Autofish, Fish, CFish, Poisson, Pandira, SFO and the tuning programs. RDTape35 calls subroutine RDRECORD1 to read the first record of the file. We provide the source code for these subroutines in files SF_RDT35.F90 and SF_RDR1.F90 for programmers who wish to customize their application. In most cases, the compiled versions in LANLSF.Lib should suffice.

Postprocessors SF7 and SFO are both stand-alone programs and also subroutines called by other programs. When used as subroutines (e.g., in the tuning programs), these codes deal only with real solution arrays. When used as stand-alone programs, they work with complex solution arrays. RDTape35 fills the appropriate arrays depending upon which module (ComplexTape35Arrays or RealTape35Arrays) appeared in the USE statements at compile time. Programs Sample2.F90 and Sample2.F90, which are discussed below, assume that the solution arrays are complex.

The calling line for RDTape35 is:

CALL RDTape35

RDTape35 will not fill certain arrays in module ComplexTape35Arrays if the allocated array size is too small to hold the entire array stored in the binary solution file. If CHECK35 in module SF_CTRL is true, then RDTape35 verifies that arrays are large enough. The code diagnoses an error and returns to the caller if an array is too small. To perform this checking, the code requires that the calling program set the last value of each array according to Table II-19. The parameter FINDEF is found in declaration statements in module SF_CTRL. The Sample.F90 and Sample2.F90 programs use this feature. Subroutine RDTape35 performs this checking by default. If you prefer not to use it, your code must set CHECK35 to false.

Table II-19. Settings for array verification.

Variable type	Setting of last array element
COMPLEX (KIND=8)	CMPLX(FINDEF,0.0D0)
REAL (KIND=8)	FINDEF
INTEGER (KIND=1)	–99
INTEGER (KIND=2)	–99
INTEGER (KIND=4)	–99
CHARACTER	all blank

Using the solution-file arrays

Poisson Superfish codes allocate the array AVECTOR(ITOT) to store the magnetic field H_z or H_ϕ for RF modes, the vector potential A_z or the product rA_ϕ for Poisson and Pandira magnet problems, or the scalar potential V for electrostatic problems. The array can be either complex or real, but we will assume it is complex in this section.

We now discuss some of the tasks that a program can perform with the arrays in the binary solution file. Suppose you want to get the field and coordinates for logical point (K,L) in the mesh. Calculate the array index and get the parameters as follows:

```
I = L*IMAX+K+1
XVAL = XX(I)
YVAL = YY(I)
HCMPX = AVECTOR(I)
```

where IMAX = KMAX +2, and HCMPX is the complex magnetic field. Variable KMAX is the number of mesh nodes in the horizontal direction. See the source code in files Sample.F90 and Sample2.F90 for some examples. A point I in the logical mesh is surrounded by six nearest neighbors, numbered 1 through 6 in Figure II-3.

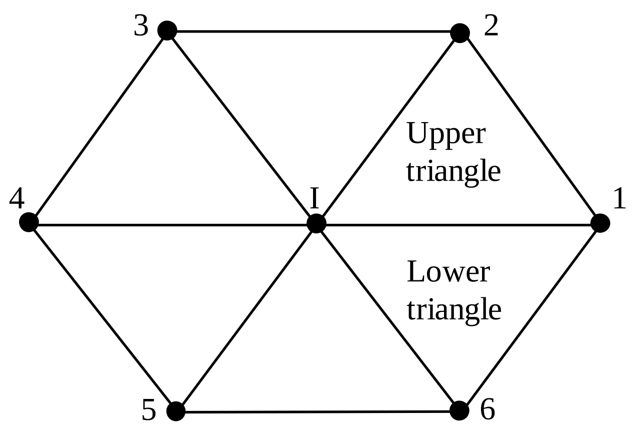

Figure II-3. The six triangles surrounding a mesh point.
Point I in the logical mesh is surrounded by six nearest neighbors, numbered 1 through 6. Associated with every point I is an upper triangle and a lower triangle.

If L is the logical row of point I, then points 5 and 6 are on row L−1, points 1 and 4 are also on row L, and points 2 and 3 are on row L+1. Each point has an upper and a lower triangle in the logical mesh. Points I, 1, and 2 form the upper triangle of point I. Points I, 1, and 6 form its lower triangle. If IREGUP(I) is zero, then the upper triangle is not in the field region. If IREGDN(I) is zero, then the lower triangle is not in the field region. Nonzero entries in IREGUP and IREGDN are material numbers from the solution-file array MAT(0:NREG). If K is the logical column of point I, then point 1 is on column K+1 and point 4 is on K−1. Column numbers for the other four points depend on whether point I's row number is even or odd. This code fragment calculates the array indices I1 through I6 for points 1 through 6:

```
NODD = MOD(I/IMAX,2)
I2 = I+IMAX+NODD
I6 = I2–2*IMAX
I1 = I+1
I3 = I2–1
I4 = I–1
I5 = I6–1
```

Example code fragment that reads the binary solution file

RDTape35 calls the subroutine RDRECORD1 in file SF_RDR1.F90 to read only the first record of the binary solution file. A program that will later call RDTape35 should first call RDRECORD1, which retrieves the variables listed in Table II-11. The program then allocates the necessary arrays before calling RDTape35. The following code fragment illustrates this procedure for variables ITOT and NREG. In this example, we have replaced error messages and other error handling statements with comment lines.

```
      PROGRAM MAIN
      USE PoissonSuperfishData
      USE MaterialData
      INCLUDE ComplexTape35Arrays
      USE SF_CTRL
      IMPLICIT NONE
!    Declare and initialize local variables here.
!
!    Open the TAPE35 file to get the size of the problem arrays.
!    RDTape35 automatically rewinds unit 35 and rereads record 1.
!
      TAPE35='PROB1.T35'
      IF(.NOT.FEXIST(TAPE35))THEN
!    Error handling here.
      ENDIF
      OPEN(35,FILE = TAPE35,FORM = ‘UNFORMATTED’)
      CALL RDRECORD1
      IF(KERROR.GT.0)THEN
         IF(KXT35.EQ.1)THEN
!    Error handling here.
         ENDIF
      ENDIF
!
!    Allocate the arrays. Material arrays start on line 4 of the ALLOCATE
!    statement. The statement given here will work for any problem. For
!    Superfish problems, lines 5 through 12 can be omitted. For Poisson and
!    Pandira problems, line 4 can be omitted. If an omission is chosen, the
!    corresponding assignments below must also be omitted.
!
      ALLOCATE(XX(ITOT),YY(ITOT),AVECTOR(ITOT),SOURCE(ITOT),&    ! 1
      IRLAX(ITOT),IREGUP(ITOT),IREGDN(ITOT),KSCAT(ITOT),&        ! 2
      KFILT(ITOT),NGO(ITOT),MAT(0:NREG),DEN(0:NREG),&            ! 3
      EPSMT(MAXTABLE),FMUMT(MAXTABLE),&                          ! 4
      BT(201,MINTABLE:MAXTABLE),GT(201,MINTABLE:MAXTABLE),&      ! 5
      AEASYMT(MINTABLE:MAXTABLE),GAMPERMT(MINTABLE:MAXTABLE),&   ! 6
      X0AMT(MINTABLE:MAXTABLE),Y0AMT(MINTABLE:MAXTABLE),&        ! 7
      PHIAMT(MINTABLE:MAXTABLE),AMULTMT(MINTABLE:MAXTABLE),&     ! 8
      HCEPTMT(MINTABLE:MAXTABLE),BCEPTMT(MINTABLE:MAXTABLE),&    ! 9
      GAMMAMT(MINTABLE:MAXTABLE),MTID(MINTABLE:MAXTABLE),&       ! 10
      MTLENGTH(MINTABLE:MAXTABLE),MTYPEMT(MINTABLE:MAXTABLE),&   ! 11
      MATTBLI(0:MAXMAT),MSHAPEMT(0:MAXMAT),STACKI(0:MAXMAT),&    ! 12
      STAT=IEE)
!
!    Error handling here (check the error status IEE)
!    Before calling the subroutine, set the last element of declared
!    arrays to values required in RDTape35 to verify the required length.
!
      xx(itot)=findef
      yy(itot)=findef
      avector(itot)=cmplx(findef,0.0d0)
      irlax(itot)=-99
      iregup(itot)=-99
      iregdn(itot)=-99
      kscat(itot)=-99
      kfilt(itot)=-99
      ngo(itot)=-99
      mat(nreg)=-99
      den(nreg)=findef
      source(itot)=findef
      iccc=2
      itot1=itot
      itot2=itot
      itot3=1
      nareg=nreg
      nwtot=1
      naseg=1
      nabound=1
      nbspl=1
      epsmt(maxtable)=findef
      fmumt(maxtable)=findef
      bt(201,maxtable)=findef
      gt(201,maxtable)=findef
      aeasymt(maxtable)=findef
      gampermt(maxtable)=findef
      gammamt(maxtable)=findef
      x0amt(maxtable)=findef
      y0amt(maxtable)=findef
      phiamt(maxtable)=findef
      amultmt(maxtable)=findef
      hceptmt(maxtable)=findef
      bceptmt(maxtable)=findef
      mtlength(maxtable)=-99
      mtypemt(maxtable)=-99
      stacki(maxmat)=findef
      mattbli(maxmat)=-99
      mshapemt(maxmat)=-99
      CALL RDTape35
!    End of code fragment.
```

This code does not use some solution-file arrays. The code sets the appropriate variables from Table II-11 to 1 before calling RDTape35.

f. Interpolate, subroutine for interpolating fields

Subroutine Interpolate is the Poisson Superfish field interpolator. All of the codes that compute field components from the solution arrays use this routine. Another section in this document provides more information about the field interpolator. The two sample programs. Sample2.F90 and Sample2.F90 emulate the behavior of the postprocessor SF7 when reading the binary solution file. The calling routine must define the interpolation point XEDIT,YEDIT in module InterpolatedFields. Before calling the interpolator, the main program first allocates appropriate arrays listed in Table II-10, then calls RDTape35 to read the solution arrays. The calling line for the field interpolator is:

CALL Interpolate(IROW)

where the calling parameter IROW points to mesh point XX(IROW),YY(IROW), known to be near the interpolation point XEDIT,YEDIT. The value –1 for the calling parameter indicates that the nearest point in the solution array is unknown. If the first parameter is not a positive number from 1 to ITOT, then Interpolate will find the nearest point. You can supply a value of IROW even if it might not be the closest point. The code will determine that is it not the nearest point and proceed to find it. This procedure is often faster than starting from no information at all (IROW = –1). The actual nearest point is returned as NEARESTIROW in the InterpolatedFields module. Thus, the loop in Sample.F90 discussed below might be more efficient if it were written as follows:

```
JROW=-1
XINC=XMAXG/200.0D0
DO J=0,200
   XEDIT=DBLE(J)*XINC
   CALL INTERPOLATE(JROW)
   JROW=NEARESTIROW
WRITE(45,"(2E15.7)")XEDIT/CONV,BY
ENDDO
```

Variable IROWREF is the index of the reference point, the mesh point around which Interpolate finds the first and second nearest neighbors. The reference point will usually be equal to NEARESTIROW except near boundaries. Near boundaries, the code may choose a first nearest neighbor of NEARESTIROW for IROWREF in order to maximize the number of points (NPO) used in the fit. This step ensures that the fit includes all of the first nearest neighbors of the reference point. Because some of the second nearest neighbors are unavailable near boundaries, the interpolator may add selected third nearest neighbors until the total number of neighboring points reaches 18.

Sample.F90, interpolating fields for a Poisson problem

Program Sample.F90 uses Fortran modules Winteracter, PoissonSuperfishData, ComplexTape35Arrays (file T35Array.inc has statement "USE ComplexTape35Arrays"), InterpolatedFields, and SF_CTRL. The Winteracter module, which comes with the ***Winteracter*** software, does not appear explicitly because it is included on a USE statement in module SF_CTRL. File Sample.F90 in the LF95 subdirectory contains some additional comments that have been omitted from the following listing:

```
PROGRAM SAMPLE
USE PoissonSuperfishData
USE MaterialData
INCLUDE 't35array.inc'
USE InterpolatedFields
USE SF_CTRL
USE SF_Menued
IMPLICIT NONE
INTEGER (KIND=4) :: j=0,iee=0
REAL (KIND=8) :: xinc=0.0
CALL WInitialise(' ')
WINDOW%FLAGS=SysMenuOn+HideRoot
CALL WindowOpen(WINDOW)
call WDialogLoad(IDD_Information)  ! Dialog may be needed in RDTape35
```

```
!
!    Open the TAPE35 file to get the size of the problem arrays.
!    RDTape35 automatically rewinds unit 35 and rereads record 1.
!
      TAPE35='HTEST1.T35'
      IF(.NOT.FEXIST(TAPE35))THEN
         call WMessageBox(OKOnly,StopIcon,CommonOK,' The file '//&
         trim(tape35)//' does not exist.','Error')
         call IOsExitProgram(' ',1)
      ENDIF
      OPEN(35,FILE=TAPE35,FORM='UNFORMATTED',ACTION='READ')
      CALL RDRECORD1
      IF(KERROR.GT.0)THEN
         IF(KXT35.EQ.1)THEN
            call WMessageBox(OKOnly,StopIcon,CommonOK,' An error'//&
            ' occurred reading the first record from file '//&
            trim(tape35)//'.','Error')
         ENDIF
         call IOsExitProgram(' ',1)
      ENDIF
!
!    Allocate the arrays, and call RDTape35.
!
      ALLOCATE(XX(ITOT),YY(ITOT),AVECTOR(ITOT),SOURCE(ITOT),&
      IRLAX(ITOT),IREGUP(ITOT),IREGDN(ITOT),KSCAT(ITOT),&
      KFILT(ITOT),NGO(ITOT),MAT(0:NREG),DEN(0:NREG),&
      BT(201,MINTABLE:MAXTABLE),GT(201,MINTABLE:MAXTABLE),&
      AEASYMT(MINTABLE:MAXTABLE),GAMPERMT(MINTABLE:MAXTABLE),&
      X0AMT(MINTABLE:MAXTABLE),Y0AMT(MINTABLE:MAXTABLE),&
      PHIAMT(MINTABLE:MAXTABLE),AMULTMT(MINTABLE:MAXTABLE),&
      HCEPTMT(MINTABLE:MAXTABLE),BCEPTMT(MINTABLE:MAXTABLE),&
      GAMMAMT(MINTABLE:MAXTABLE),MTID(MINTABLE:MAXTABLE),&
      MTLENGTH(MINTABLE:MAXTABLE),MTYPEMT(MINTABLE:MAXTABLE),&
      MATTBLI(0:MAXMAT),MSHAPEMT(0:MAXMAT),STACKI(0:MAXMAT),&
      STAT=IEE)
      IF(IEE.NE.0)THEN
         call WMessageBox(OKOnly,StopIcon,CommonOK,'There is not'//&
         ' enough memory available to to start the program.','Error')
         call IOsExitProgram(' ',1)
      ENDIF
!
!    Before calling the subroutine, set the last element of arrays
!    to values required in RDTape35 to verify the array lengths.
!
      XX(ITOT)=FINDEF
      YY(ITOT)=FINDEF
      AVECTOR(ITOT)=CMPLX(FINDEF,0.D0)
      IRLAX(ITOT)=-99
      IREGUP(ITOT)=-99
      IREGDN(ITOT)=-99
      KSCAT(ITOT)=-99
      KFILT(ITOT)=-99
      NGO(ITOT)=-99
      MAT(NREG)=-99
      DEN(NREG)=FINDEF
      SOURCE(ITOT)=FINDEF
      ICCC=2
      ITOT1=ITOT
      ITOT2=ITOT
      ITOT3=1
      NAREG=NREG
      NWTOT=1
      NASEG=1
      NABOUND=1
      NBSPL=1
      BT(201,MAXTABLE)=FINDEF
      GT(201,MAXTABLE)=FINDEF
      AEASYMT(MAXTABLE)=FINDEF
      GAMPERMT(MAXTABLE)=FINDEF
      GAMMAMT(MAXTABLE)=FINDEF
      X0AMT(MAXTABLE)=FINDEF
      Y0AMT(MAXTABLE)=FINDEF
      PHIAMT(MAXTABLE)=FINDEF
      AMULTMT(MAXTABLE)=FINDEF
      HCEPTMT(MAXTABLE)=FINDEF
      BCEPTMT(MAXTABLE)=FINDEF
      MTLENGTH(MAXTABLE)=-99
      MTYPEMT(MAXTABLE)=-99
      STACKI(MAXMAT)=FINDEF
      MATTBLI(MAXMAT)=-99
      MSHAPEMT(MAXMAT)=-99
      CALL RDTape35
      IF(KERROR.GT.0) CALL IOsExitProgram(' ',KERROR)
!
!    Open a text file and write By versus X to a Quikplot data file.
!
      OPEN(45,FILE=‘HTEST1.QKP’,ACTION='WRITE')
      WRITE(45,’(A)’)’; File HTEST1.QKP’,&
      ‘; Sample Quikplot file using the Poisson Superfish field’//&
      ’ interpolator’,‘ ’&
      ’Center’,’Title’,&
      ‘By versus X from Poisson/Pandira problem: '//TRIM(title(1)),' ',&
      ’XLabel’,’X Position (cm)’,&
      ’YLABEL’,’By (Gauss)’,’ ‘,&
      ’Data’
      YEDIT = 0.0D0
      XINC=XMAXG/200.0D0
      DO J=0,200
         XEDIT=DBLE(J)*XINC
         CALL INTERPOLATE(-1)
         WRITE(45,”(2E15.7)”)XEDIT/CONV,BY
      ENDDO
      WRITE(45,'(A)')'EndData',' ','EndFile'
      CLOSE(45)
      call WMessageBox(OKOnly,InformationIcon,CommonOK,&
      'File HTEST1.QKP has been created.'//char(13)//&
      'Double-click on this file to run'//char(13)//&
      'Quikplot and view By versus x.','Program Completed')
      CALL IOsExitProgram(' ',0)
      END PROGRAM
!
!    End of Sample.F90.
!
```

Sample2.F90, interpolating fields for a CFish problem

Program Sample2.F90 uses Fortran modules Winteracter, PoissonSuperfishData, ComplexTape35Arrays (file T35Array.inc has statement "USE ComplexTape35Arrays"), InterpolatedFields, MaterialData, and SF_CTRL. The Winteracter module comes with the ***Winteracter*** software. File Sample2.F90 in the LF95 subdirectory contains some additional comments that have been omitted from the following listing:

```
      PROGRAM SAMPLE2
      USE PoissonSuperfishData
      USE MaterialData
      INCLUDE 't35array.inc'
      USE InterpolatedFields
      USE SF_CTRL
      USE SF_Menued
      IMPLICIT NONE
      CHARACTER (LEN=100)  :: messagebuffer=' '
      INTEGER (KIND=4) :: j=0,iee=0,nm=0,nregion=0
      REAL (KIND=8) :: xinc=0.0
      CALL WInitialise(' ')
      WINDOW%FLAGS=SysMenuOn+HideRoot
      CALL WindowOpen(WINDOW)
      call WDialogLoad(IDD_Information)  ! Dialog may be needed in RDTape35
!
!    Open the TAPE35 file to get the size of the problem arrays.
!    RDTape35 automatically rewinds unit 35 and rereads record 1.
!
      TAPE35='COAXWG.T35'
      IF(.NOT.FEXIST(TAPE35))THEN
        call WMessageBox(OKOnly,StopIcon,CommonOK,' The file '//&
        TRIM(TAPE35)//' does not exist.','Error')
        call IOsExitProgram(' ',1)
      ENDIF
      OPEN(35,FILE=TAPE35,FORM='UNFORMATTED',ACTION='READ')
      CALL RDRECORD1
      IF(KERROR.GT.0)THEN
        IF(KXT35.EQ.1)THEN
          call WMessageBox(OKOnly,StopIcon,CommonOK,' An error'//&
          ' occurred reading the first record from file '//
          trim(tape35)//'.','Error')
        ENDIF
        CALL IOsExitProgram(' ',1)
      ENDIF
!
!   Allocate the arrays and call RDTape35.
!
      ALLOCATE(XX(ITOT),YY(ITOT),AVECTOR(ITOT),IRLAX(ITOT),&
      IREGUP(ITOT),IREGDN(ITOT),KSCAT(ITOT),KFILT(ITOT),&
      NGO(ITOT),MAT(0:NREG),DEN(0:NREG),&
      EPSMT(MAXTABLE),FMUMT(MAXTABLE),&
      REPS(0:MAXTABLE),FMU(0:MAXTABLE),&
      STAT=IEE)
      IF(IEE.NE.0)THEN
        call WMessageBox(OKOnly,StopIcon,CommonOK,'There is not'//&
        ' enough memory available to to start the program.','Error')
        call IOsExitProgram(' ',1)
      ENDIF
!
!   Before calling the subroutine, set the last element of arrays
!   to values required in RDTape35 to verify the array lengths.
!
      XX(ITOT)=FINDEF
      YY(ITOT)=FINDEF
      AVECTOR(ITOT)=CMPLX(FINDEF,0.D0)
      IRLAX(ITOT)=-99
      IREGUP(ITOT)=-99
      IREGDN(ITOT)=-99
      KSCAT(ITOT)=-99
      KFILT(ITOT)=-99
      NGO(ITOT)=-99
      MAT(NREG)=-99
      DEN(NREG)=FINDEF
      ICCC=2
      ITOT1=ITOT
      ITOT2=1
      ITOT3=1
      NAREG=NREG
      NWTOT=1
      NASEG=1
      NABOUND=1
      NBSPL=1
      EPSMT(MAXTABLE)=FINDEF
      FMUMT(MAXTABLE)=FINDEF
      CALL RDTape35
      IF(KERROR.GT.0) CALL IOsExitProgram(' ',KERROR)
!
!   Transfer the material data to the REPS and FMU arrays for use
!   in the field interpolator.
!
      REPS(0)=0.0D0
      FMU(0)=0.0D0
      REPS(1)=1.0D0
      FMU(1)=1.0D0
      IF(NMATR.GT.0)THEN
        DO NREGION=1,NREG
```

```
          NM=MAT(NREGION)
          IF(NM.LE.1)CYCLE
          IF(ABS(DBLE(EPSMT(NM))-FINDEF).LT.FTOLER.OR.&
          ABS(DBLE(FMUMT(NM))-FINDEF).LT.FTOLER)THEN
            write(messagebuffer,1100)trim(tape35),nm
1100        format(/' ---ERROR--- File ',a,' does not include material',&
            ' data for'/13x,'material ID',i3)
            call WMessageBox(OKOnly,StopIcon,CommonOK,messagebuffer,&
            'Error')
            call IOsExitProgram(' ',1)
          ENDIF
          REPS(NM)=1.0D0/REPSMT(NM)
          FMU(NM)=FMUMT(NM)
        ENDDO
      ENDIF
!
!   Open a text file and write fields to a Tablplot data file.
!
      OPEN(45,FILE='COAXWG.TBL',ACTION='WRITE’)
      WRITE(45,'(a)')'; File COAXWG.TBL',';  Sample Tablplot file'//&
      ' using the Poisson Superfish field interpolator',' ',&
      'Center','Title',&
      'Er and Ez versus Z from CFish problem:',TRIM(title(1)),&
      ’SubTitle’,TRIM(title(2)),' ',&
      'MArkersize -0.4',&
      ’Ylabel’, 'Ez,Er (MV/m)',&
      'ABscissa 1',&
      'ORdinate 2 3',’ ‘,&
      'Titles','Z','Ez','Er','H','EndTitles',' ',&
      'AxisLabels','Z (cm)','Ez (MV/m)','Er (MV/m)','H (A/m)',&
      'EndLabels',' ',&
      'CurveLabels','Z','Ez','Er','H','EndLabels',’ ‘,&
      'Data',&
      ';  Z     Ez      Er      H',&
      '; (cm)  (MV/m)  (MV/m)  (A/m)'
      XINC=XMAXG/200.0D0
      YEDIT=2.5D0
      DO J=0,200
        XEDIT=DBLE(J)*XINC
        CALL INTERPOLATE(-1)
!
!   Scale the field components to MV/m and A/m. Write only the real part.
!
        EZ=EZ*ASCALE*1.0D-4
        ER=ER*ASCALE*1.0D-4
        H=25.0D9*H*ASCALE/(PI*CLIGHT)
```

```
           WRITE(45,"(F10.5,1P3E14.6)")XEDIT/CONV,DBLE(EZ),DBLE(ER),DBLE(H)
         ENDDO
         WRITE(45,'(A)')'EndData',' ','EndFile'
         call WMessageBox(OKOnly,InformationIcon,CommonOK,&
         'File COAXWG.TBL has been created.'//char(13)//&
         'Double-click on this file to run'//char(13)//&
         'Tablplot and view Ez and Er versus z.','Program Completed')
         CALL IOsExitProgram(' ',0)
         END PROGRAM
!
!    End of Sample2.F90.
!
```

III. Introduction to Poisson Superfish

Poisson and Superfish are the main solver programs in a collection of programs for calculating static magnetic and electric fields and radio-frequency electromagnetic fields in either 2-D Cartesian coordinates or axially symmetric cylindrical coordinates. The programs generate a triangular mesh fitted to the boundaries of different materials in the problem geometry. The collection of codes is most often called the Poisson Superfish group of codes. The RF solvers Fish and CFish iterate on the frequency and field calculation until finding a resonant mode. Poisson finds static fields by successive over relaxation. Pandira calculates static fields by a direct matrix inversion method. The codes write a variety of information on the resulting solution. A brief summary of the Poisson Superfish codes appears later in this document.

The original Poisson Superfish codes were developed by Ronald F. Holsinger and Klaus Halbach. This version is supported by Lloyd M. Young and James H. Billen. See the History and References sections for more information.

This document (LA-UR-96-1834) contains all the information you need to run the Poisson Superfish programs on your Windows-based computer. Two other publications that were written in 1987 for an older version of these codes running on VAX and UNIX platforms are no longer available. These documents are:

- Reference Manual for the POISSON/SUPERFISH Group of Codes, LA-UR-87-126,
- User's Guide for the POISSON/SUPERFISH Group of Codes, LA-UR-87-115.

The Reference Manual contained John L. Warren's treatment of a number of physics topics listed in Table III-1. We have reproduced the sections mentioned in Table III-1 in essentially their original form as reference material to the physics in the Poisson Superfish codes. Most of the information from the 1987 Reference Manual appears in the last several chapters. The discussion of harmonic analysis has been incorporated into the chapter on Poisson and Pandira. The last column in Table III-1 refers to the chapter in this document. The present code distribution includes other material besides that listed in Table III-1. We have updated all of the example problems from the 1987 publications to use features in the latest version of the codes. The examples are included in the LANL\Examples subdirectories.

Table III-1. Topics from the 1987 Reference Manual

Reference Manual section	Topic	Chapter
Part B, Chapter 13.1	Theory of electrostatics and magnetostatics.	XX
Part B, Chapter 13.2	Properties of static magnetic and electric fields.	XXI
Part B, Chapter 13.5	Boundary conditions and symmetries.	XXII
Part B, Chapter 13.6	Numerical methods in Poisson and Pandira.	XXIII
Part C, Chapter 13.1	RF cavity theory.	XXIV
Part B, Chapter 13.3	Harmonic analysis.	VIII

A. History of the development of Poisson Superfish

The Poisson, Pandira, and Superfish codes derive from a diffusion calculation code written by Alan Winslow at Lawrence Livermore National Laboratory. In the 1960s, Jim Spoerl at Lawrence Berkeley Laboratory (LBL) modified the code to develop the TRIM programs MESH and FIELD for magnetostatic problems. MESH constructed an irregular mesh to fit the geometry of the magnet and FIELD solved Poisson's equation for the potential function over the mesh.

Ronald Holsinger, Klaus Halbach, and other associates at LBL made extensive changes to the codes and introduced the use of conformal transformations. They renamed the codes Lattice and Poisson and they introduced the plotting program TEKPLOT. Klaus Halbach recalls that the LBL effort began in 1966. With continuing theoretical assistance from Halbach, Holsinger improved the existing codes and wrote the codes Force and MIRT. Holsinger worked on these codes while he was at several institutions: LBL, the Swiss Institute for Nuclear Research (SIN), and the European Center for Nuclear Research (CERN). Holsinger worked at Los Alamos National Laboratory from 1975 to 1977 developing the codes Automesh, Pandira, and Superfish, again with Klaus Halbach's continued support and with additional theoretical support from Robert Gluckstern. Holsinger left Los Alamos in 1977 and continued to update and maintain the programs until 1982.

1. Code development under LAACG

Since 1986, the Los Alamos Accelerator Code Group (LAACG) has received funding from the U.S. Department of Energy to maintain and document a standard version of these codes. The LAACG started in Accelerator Technology division's theory group AT-6, led by Richard K. Cooper. Richard Cooper headed the LAACG until Dominic Chan took over in the early 1990s. When group AT-6 was dissolved, the LAACG under Dominic Chan moved to group AT-7. In 1992, when Chan assumed group leader duties for AT-7, Robert Ryne took over leadership of the LAACG. Richard Cooper and several other members of the Code Group opted for a retirement incentive late in 1993. In 1994, AT division merged with the former Medium Energy Physics (MP) division, which ran the LAMPF (Los Alamos Meson Physics Facility) accelerator. The new division was known as Accelerator Operations and Technology. The remaining members of the LAACG joined the linac technology group AOT-1 led by Jim Stovall.

In 1997, another reorganization placed most of AOT division in a new LANSCE (Los Alamos Neutron Science) division. The LAACG, still led by Robert Ryne, stayed in the same group, renamed LANSCE-1. In 1999, several members of the LAACG moved temporarily to a new SNS project division to design and build the linac for the Spallation Neutron Source. Lloyd Young, Harunori Takeda, and James Billen continue to support the core group of accelerator design codes, now partially underwritten by the SNS because of the project's heavy use of the codes. Jim Stovall assumed the LAACG leadership role in 2000 when Robert Ryne left Los Alamos.

LAACG released Poisson Superfish version 3 in the Spring of 1992, and version 4 in the Fall of 1992. Several LAACG members who have worked with the Poisson Superfish codes through version 4 are now retired from Los Alamos National Laboratory, including Richard K. Cooper, Mary T. Menzel, Grenfell Boicourt, Therese Barts, Jean Merson, Gary Rodenz, and John L. Warren. Part of the work contributed by this team was to convert all the codes to standard Fortran 77. John Warren led the documentation effort that resulted in two volumes: the Reference Manual and the User's Guide.

Until her retirement from Brookhaven National Laboratory in the early 1990s, Judith Colman distributed a set of PC codes based upon latest LAACG standard version. We are also aware of at least one commercial PC version of Superfish that was based upon LAACG version 3.

Version 4.12 of Poisson Superfish was the last revision of the codes maintained for use on UNIX and VAX VMS platforms. (A future release may support the Linux operating system.) Version 4.12 is still available, but active support of it stopped in 1993. Version 4, which developed concurrently with version 5 (see next paragraph), incorporated a few of the version 5 features. For example, version 4 also removed the 32,767 mesh-point limit, but a compile-time limit still applied. Version 4 also employed the Superfish root finder from version 5.

Version 5 of Poisson Superfish ran on the 80x86 family of processors of under DOS. Version 5 originated in 1985 when Lloyd M. Young adapted the LAACG Cray version (known as version 3 at the time) to run on IBM-PC compatible computers. Therese Barts incorporated into version 3 some features that Lloyd Young had added to the PC codes. A major revision of the PC codes began in 1992 when James H. Billen and Lloyd M. Young removed the previous 32K limit on the number of mesh points. The codes allocated memory for data arrays at run time, so a computer's available memory is the only limit on the maximum problem size. Lloyd Young wrote the Los Alamos CFish code following the method described by M. S. de Jong and F. P. Adams. Version 5 was continuously supported and developed between 1992 and 1999. The code MIRT, which was available in earlier LAACG distributions was not included in version 5.

We released version 6 for the Windows platform in January, 1999. Development continued for about four years using Lahey Fortran LF90 compiler plus the ***Winteracter*** development tools. Support for version 6 stopped in April, 2003. In December, 2002, we released version 7 to beta testers and in April, 2003 we released version 7 to all users. Version 7 uses the Lahey/Fujitsu LF95 compiler (version 5.7) and the ***Winteracter*** package.

Every year, the code group is pleased grant the requests of instructors at universities and at special accelerator schools (USPAS, CERN) to use LAACG codes (Poisson Superfish, Parmela, and Trace 3-D) in the classroom.

2. Features added in version 5

For version 5, Billen and Young made improvements to the user interface, and introduced several new codes, including VGAPLOT, several other general-purpose plotting codes, and rf cavity tuning programs In April, 1997, the separate program LATTICE was retired. All of the functions previously performed by LATTICE were incorporated into

Automesh. The following sections describe changes and additions that were included in version 5:

- Consistent use of the saturation magnetization in Poisson and Pandira,
- Source code for reading the solution file, for program developers,

a. User interface improvements

Initialization file SF.INI

Since August, 1994, Poisson Superfish codes read the initialization file SF.INI , which contains configurable options, run-time limits, and other preferences.

Robust selection of problem type

In older versions of the codes, the first character of the title indicated the type of problem to solve: a blank character for Poisson or Pandira problems; a nonblank character for Superfish problems. Version 5 and later provides a robust method for selecting the problem type. Variable KPROB in the first REG namelist now serves this purpose. KPROB is 1 to define a Superfish problem or 0 to define a Poisson or Pandira problem.

Drive-point checking in Superfish problems

For Superfish problems, the codes try to ensure the existence of a valid drive point at several points in the solution of a problem. See Drive point requirements in Fish and CFish for more information.

Multi-line problem descriptions

Problem descriptions of up to ten 80-character title lines are supported in all the codes. We added this feature in February, 1995. These title lines appear before the first REG namelist in the Automesh input file. The binary solution file stores the title lines and each code prints the problem description in its respective output file.

Enhanced namelist input in Automesh

Program Automesh reads namelist-like entries, but the code does not actually use the Fortran implementation of namelist. Instead, Automesh parses every line and checks for possible errors in the input parameters. This feature results in better diagnostic messages and can help save time debugging an input file. You can use either a dollar sign ($) or an ampersand (&) as the namelist delimiter. Comments can appear on any line in the input file after a semicolon (;) or exclamation mark (!).

The first REG namelist in the Automesh input file allows entry all problem variables used by the solver programs. Up to ten 80-character title lines can appear before the first REG namelist. The PO namelist also includes new features. Automesh can connect points with straight lines, circular arcs, elliptical arcs, or hyperbolic curves. The MT namelist, which was introduced in November, 1996, includes all the parameters necessary to define material properties. The POA namelist, which was introduced in June, 2003, sets the potential to a supplied value at one point in the mesh.

It is not necessary to indicate the number of regions (NREG) or the number of points in each region (NPOINT) since these parameters are calculated in the codes. You can simulate the end of the file (for example, to exclude the last region in the file) with the keyword STOP. Automesh will not read beyond the STOP line. Synonyms for STOP are ENDFile and ENDOFFILE.

Exit error codes

Poisson Superfish programs set the exit error code to the value reported by any error that cause the code to stop. Normal exist has error code zero. The error code can be used in batch jobs. Control programs that run stand-alone versions of the Poisson Superfish programs can use this code to diagnose problems with the run or to produce meaningful error messages. A list of error codes appears at the end of sections VI through XII for each program.

Tuning code integration

The automated tuning programs CCLfish, ELLfish, CDTfish, DTLfish, MDTfish, RFQfish, and SCCfish are fully integrated with the Superfish routines. These programs use an internal variable to report errors that would stop the stand-alone Superfish programs. Detailed messages appear in the Superfish code output files when a routine fails.

b. Automesh improvements

Program Automesh writes a list of logical coordinates along boundaries. Integer coordinates K,L describe the logical overlay mesh and X,Y are the physical coordinates. Older Automesh routines do little checking for consistency among points, sometimes resulting in triangles for which it calculates a negative or zero area. A “negative area” means that the three points are out of their logical order resulting in a triangle that overlaps other mesh triangles.

Boundary point assignment

Program Automesh imposes constraints on the selection of logical coordinates for boundary points, intersections of line regions with the boundary, and points along the connecting path. The code defines each segment along its entire length. For example, Automesh tries to avoid using different logical rows for the same Y coordinate, or different logical columns for the same X coordinate. When possible, all K (or L) values for the logical path along a vertical (or horizontal) line are identical. The path does not jog back and forth between logical rows or columns. This requirement and other similar tests prevent most “negative-area-triangle” crashes.

Lines and arcs

In June, 1994, we implemented a powerful meshing algorithm for lines and arcs in Automesh. When assigning logical coordinates K,L to physical points X,Y, the code transforms line and arc segments into a “logical plane” in which the mesh is uniform in both directions. This approach does a better job than before especially for regions where the mesh triangles in the X,Y plane have a large aspect ratio. We have not yet adapted the new method to hyperbolic segments. On hyperbolic segments the code still assigns points in the physical plane so it is prudent to make the mesh fairly uniform in that part of the geometry.

Elliptical arc segments

In July, 1995, we added elliptical arc segments to Automesh. The user supplies either both semiaxes of the ellipse, or one of the semiaxes and the aspect ratio.

Overlapping regions

In April, 1996, we reorganized the way Automesh assigns logical points and traces the logical path along boundary segments. The code first reads all user-supplied fixed points (in the PO namelist sections). It scans all the regions and looks for segment end points that will appear on other boundaries. The code then adds these physical points in all the regions that share the common line or arc segment. These changes automatically prevent many potential problems with overlapping and adjacent regions. You need not traverse common lines or arc segments in the same direction. It also is no longer necessary to manually insert boundary points from regions that appear later in the input file. See Overlapping and adjacent regions in the Automesh section for more information.

PO namelist variable NEW eliminated

Some Automesh versions earlier than version 5 referred to a PO namelist variable called NEW. If NEW was +1 or –1, the code (supposedly) would not allow points on the logical path for a region to coincide with any previous region's path (except for the end points for NEW = –1). According to the 1987 Reference Manual, part C, chapter 3, section 3, page 22, the NEW option was a fix for certain types of logical mesh "glitches." Version 5 and later versions do not support this variable because it conflicts with the extensive checking for consistency among points that Young and Billen added to Automesh.

c. RF solver improvements

Convergence criteria

Resonances occur at roots of the $D(k^2)$ function where the slope equals –1. For convergence to a resonant frequency, the present version of Superfish requires that the $D(k^2)$ function is small enough and has the correct slope. The convergence requirements are:

$$\frac{\left|D\left(k^2\right)\right|}{k^2} < \text{EPSIK}$$

$$\frac{\left|\delta\left(k^2\right)\right|}{k^2} < \text{EPSIK}$$

$$-1.02 < \frac{dD}{d\left(k^2\right)} < -0.98$$

where the default value of EPSIK is 0.0001. The notation $dD/d(k^2)$ refers to the first derivative (or slope) of the $D(k^2)$ function. The term $\delta(k^2)$ is the proposed change in k^2 calculated before the next evaluation of $D(k^2)$. RESIK is equal to the quantity $|D(k^2)|/k^2$. The automated tuning programs relax requirements on the slope for some early runs on a problem. The discussion of the Fish and CFish root finder contains more information.

Version 3 and earlier versions used only the second of the above criteria. The early codes did not actually require that $D(k^2)$ itself be small and it did not check for a negative slope. Though the additional criteria originated in version 5, version 4.12, which for a time was supported in parallel with version 5, also used the above criteria. Failure to check the slope can be a problem when the starting frequency is far from a root where the slope is positive.

Robust root finder

In the Superfish code, resonances occur at roots of the $D(k^2)$ function where the slope equals –1 [see the paper by K. Halbach and R. F. Holsinger, "SUPERFISH - A Computer Program for Evaluation of RF Cavities with Cylindrical Symmetry", Particle Accelerators 7 (1976) 213-222.], where k is the wave number corresponding to frequency f ($k = \omega/c = 2\pi f/c$). Subroutine FROOT attempts to locate the slope = –1 root using various polynomial approximations. The root finder used in Superfish for many years was not very robust. Occasionally, it would not converge to the nearest mode and it sometimes could not find a mode at all.

In August, 1992 Lloyd Young and James Billen identified several of the deficiencies in subroutine FROOT. The old routine stored $D(k^2)$ and k^2 at each iteration in two arrays. Under some conditions, the code swapped the new point with a previous point. The arrays were neither chronological nor were they necessarily sorted by k^2. Nevertheless, the code calculated derivatives of $D(k^2)$ using adjacent points in the arrays. Not only did the old algorithm fail to use points close in k^2 to one another for the slope, but it never updated the slope of previous points when more data became available. As a result, the derivatives reported by the code were almost always wrong. Fortunately, the code made very little use of the derivatives that it calculated, so most of the time Superfish converged to a resonance near the initial frequency. Young and Billen wrote a new, more robust version of the root-finder subroutine for version 5. The section on Fish and CFish includes a detailed description of the present root finder. The improvements include:

1. The code takes a smaller first step toward resonance. Lacking other information, it assumes it is near the slope = –1 root and extrapolates a slope = –1 line to $D(k^2) = 0$. But, if the resulting step in k^2 is large, it limits the step to 1% of the present k^2. This change has two important consequences. First, the next point gives much more reliable information of the local slope of the curve. Second, it greatly reduces the chances of missing a mode where the function has a narrow local minimum.
2. The code sorts the available data by k^2 and recalculates first and second derivatives after every iteration. The first derivative is a weighted average of the slope obtained from the points on either side.
3. Before making any polynomial approximations for the next value of k^2, the code analyzes the available data to place upper and lower bounds on the location of the desired root. New values for k^2 that lie out of bounds are not accepted.

4. After evaluating the function at least four times, the code chooses three points for the parabola formula. Carefully chosen points speed convergence to the root. The root finder selects any point that is a lower or upper bound. It always uses the last chronological point if it has not already been selected as an upper or lower bound. Any other point must be adjacent to or between the bounds. If there is a choice, the algorithm selects the point whose slope is closest to -1.
5. The new algorithm abandons most of the polynomial approximations in the old routine. It keeps a 3-point parabola formula, but it does more evaluation of the properties of the solution. For example, if all the $D(k^2)$ values are positive, a parabola with positive curvature may have no real roots. The minimum of this parabola is likely to be very near the desired root of $D(k^2)$. The same is true if the real root with a negative slope is below the lower bound. In these cases, the code selects the minimum of the parabola for the next k^2.
6. In place of the previous polynomial approximations, the new version uses a parabola defined by two points plus a slope of -1 at $D(k^2) = 0$. These constraints result in four equations with four unknowns. The equations can be reduced to a single quadratic equation for k^2, and so has two solutions. The code selects a solution based upon known properties of the $D(k^2)$ function. It rejects solutions above the upper bound or below the lower bound.
7. If after two iterations the current k^2 is near a slope = +1 root, then the code makes an arbitrary 0.5% step toward smaller k^2. This approach is simple and quickly results in better information about the shape of the function.
8. If both parabolic approximations fail to find a new k^2 consistent with the upper and lower bounds, then the new code uses one of several procedures. If the direction to the root is known, it simply takes a step toward the root. If upper and lower bounds have been established, it tries the midpoint between the bounds. These and other procedures produce new information about the function at the next iteration.

CFish root finder

Certain modifications of the root finder are needed for complex problems solved by CFish. The root finder in CFish works correctly with lossy materials, but it must complete at least 3 cycles in order to test for convergence. To compute $D(k^2)$, Fish and CFish first calculate the quantity $\delta H_1 = H_{Drive} - H_1$ where H_{Drive} is the driving field and H_1 is the field implied by the six neighboring mesh points surrounding the driving point. Thus, δH_1 is the field error at the driving point. When the solution array is complex then δH_1 is complex. For real problems $D(k^2)$ is proportional to δH_1. For complex problems, δH_1 near an isolated resonance lies on a circle in the complex plane. Resonance occurs at the point on the circle closest to the origin. $D(k^2)$ for a point δH_1 in the complex plane is the perpendicular distance from the point to the line containing both (0,0) and the center of the circle. $D(k^2)$ versus k^2 has a negative slope at the intersection of the circle with this line.

d. New field interpolator

In early 1993, Billen and Young wrote a new field interpolation algorithm for the version 5 postprocessors SFO and SF7. The old interpolator did such a poor job of point selection that the resulting fields along a line often had large fluctuations in electric and magnetic field from point to point. In September, 1994 the new technique was extended to Poisson and Pandira static magnetic and electric field problems.

The new algorithm first finds the mesh triangle that contains the physical point X_0,Y_0 at which it will calculate the fields. It fits one of several polynomial functions of X and Y to all the first and second nearest neighbors of the closest mesh point. Near boundaries in some types of problems, the interpolator may add third nearest neighbors until the total number of neighboring points reaches 18. Some of these functions satisfy either Neumann or Dirichlet boundary conditions on or near vertical or horizontal boundaries.

The code distribution includes a linkable version of the field interpolator for code developers.

e. Memory management and program limits

The original Poisson Superfish codes were developed when memory was scarce, so they used a method of bit shifting and masking to store multiple pieces of information in one computer word. For example, the INDEX array allotted 5 bits for region numbers, 15 bits for the mesh-point counter, and one or two bits for some other items. In all, the INDEX array stored seven items for each point. Elimination of these mask and shift operations was the primary motivation for version 5, first released in 1992. The codes use a four-byte integer array for the mesh-point counter, two-byte integer arrays for region numbers, and several one-byte integer and logical arrays for other information previously coded into the INDEX array.

Mesh-point arrays

The codes allocate memory for arrays that store information about each mesh point. Thus the number of mesh points is limited only by the computer memory. UNIX version 4.12 has a compile-time limit on the number of mesh points. Several other parameters have maximum values that are fixed at run time according to settings in SF.INI. Maximum values for these parameters are limited only by available memory. The Program limits section provides more details about these settings.

Temporary arrays

While solving the tridiagonal matrix problem, Fish, CFish, and Pandira store a data matrix for each row, which they later read back in reverse order. The codes that allocate memory for temporary data storage are Autofish, CCLfish, CDTfish, ELLfish, DTLfish, MDTfish, RFQfish, SCCfish, Fish, CFish, and Pandira. If possible, these programs use available memory for these temporary data arrays thus reducing considerably the computation time for some problems.

Unlimited line regions

Poisson Superfish runs faster on large complicated geometries if the mesh is fine only where it needs to be. Starting with version 5, Automesh allows for an unlimited number of divisions in the mesh. These divisions are called line regions. In version 6, the user set variables in SF.INI to determine the maximum number of X and Y line regions used by Automesh, Autofish, and MDTfish. Starting with version 7, the codes pre-scan the input files to determine the required settings.

f. Postprocessor improvements

Field interpolation along boundary segments

For rf problems with cylindrical symmetry, the postprocessor SFO uses a better field-interpolation algorithm for E_z than the one in earlier versions of Poisson Superfish. Starting with version 5, the algorithm calculates E_z in twice as many mesh triangles as the old version. Postprocessor SFO uses the new method when integrating E_z to obtain the field normalization scale factor, and for the transit-time integrals T, TP, TPP, S, SP, and SPP. The old method suffered from errors at locations where the mesh size changed in the longitudinal direction. The new method eliminates these errors by associating the correct z coordinate with each interpolated field value.

Multiple DTL cells

The postprocessor SFO supports multiple drift-tube linac (DTL) cells. SFO calculates for all cells transit-time integrals, power, and frequency shifts. Each cell can have single or multiple stems and post couplers. Cells also can have different phase lengths. SFO includes Slater-perturbation calculations of the power losses on stems and post couplers.

Surface resistance options

There are several options for specifying how SFO determines the surface resistance R_s for rf power calculations. The default setting is to compute R_s from the room-temperature resistivity of copper. The code will correct the resistivity for other temperatures and for the effects of the material's magnetic permeability. You can enter your own values for the reference resistivity and temperature, the temperature coefficient of resistivity, and the permeability. An alternative method for normal conducting materials computes the surface resistance from a user-supplied value of the bulk resistivity and ignores the temperature correction (but still includes the effect of magnetic permeability).

Another option computes R_s for a given temperature of a superconducting material, which is characterized by a critical temperature and a residual resistance. If none of these options applies to your problem, you can specify the surface resistance itself.

Finally, SFO allows entry of the surface resistance for any of the individual segments in the problem geometry. For each boundary segment, all of the options mentioned above are available. This feature allows computation of power, Q, and other figures of merit for cavities built of different materials. Another example is a beam pipe attached to a superconducting cavity that has a normal-conducting section in a region of low field.

Peak surface electric field (Kilpatrick factor)

Many people find it useful to relate the peak surface electric field for Superfish solutions to the Kilpatrick field that corresponds to the cavity's resonant frequency. SFO calculates the Kilpatrick field and reports the ratio of the maximum surface field to the Kilpatrick field in the output summary. The code distribution also includes the stand-alone utility program Kilpat for calculating the Kilpatrick field as a function of frequency.

Input files for Parmela and Egun

Postprocessor SF7 can create input files for other programs. For rf-field problems with cylindrical symmetry, SF7 write field data to a file for the Parmela program. For static field problems solved by Poisson or Pandira, SF7 creates a file for the Egun program.

g. New programs

Version 5 introduced several new codes. The VGAPLOT code replaces an older platform-dependent plotting code. The CFish solver, tuning codes, general-purpose plotting codes, the List35 utility are entirely new.

VGAPLOT program

VGAPLOT was an entirely new plotting code that replaced TEKPLOT from previous versions. VGAPLOT displays the physical coordinates of the cursor location in the plot window and the field components at that location. The field-display window includes the logical coordinates of the nearest mesh point, the material number of the triangle containing the cursor, and the number of the fitting function used to interpolate the fields.

CFish, a complex version of the rf solver Fish

Starting with version 5, the code distribution included program CFish, which is a version of Fish that uses complex rf fields. CFish can help design RF windows, and it calculates power losses in dielectric and magnetic materials. Postprocessors are compatible with both real and complex versions of the code.

Lloyd Young wrote the Los Alamos CFish code following the method described by M. S. de Jong and F. P. Adams in the AECL internal report “Seafish, A 2-Dimensional Complex Helmholtz Equation Solver.” See also the paper by M. de Jong, F. Adams and R. Hutcheon, “Computation of RF Fields for Applicator Design,” Journal of Microwave Power and Electromagnetic Energy 27, No. 3, 136 (1992). [In the microwave power industry, the term “applicator” appears to refer to waveguide transitions and other devices that deliver the rf power to a load.]

Autofish program

Version 5 included the code Autofish, which is a combined version of Automesh, Fish, and SFO in a single executable program.

Tuning programs

Table III-2 lists several automated tuning programs. These programs set up input data for the Superfish codes, run the codes, and then analyzes the results. The tuning program adjusts the geometry and reruns the Superfish codes, iterating until the cavity resonates at the desired frequency.

Table III-2. The tuning programs.

Tuning code	Description
CCLfish	Tunes symmetric coupled-cavity linac (CCL) cells.
CDTfish	Tunes coupled-cavity drift-tube linac (CCDTL) cavities.
DTLfish	Tunes symmetric drift-tube linac (DTL) cells.
ELLfish	Tunes elliptic cavities, for superconducting applications.
MDTfish	Tunes multiple-cell drift-tube linac cavities.
RFQfish	Tunes radio-frequency quadrupole cavities.
SCCfish	Tunes side coupling cavities for coupled-cavity linac structures.

General purpose plotting codes

Several of the codes write plot files for programs Tablplot or Quikplot. Programs Fish and CFish automatically produce and update a plot file called FishScan.TBL containing the results of mode searches and frequency scans. Program SFO includes an option to write the plot file Transit.TBL of the transit-time integrands for both single-cell and multiple-cell problems. Program SF7 writes interpolated fields along arcs, lines, or user-defined curves to plot files.

List35, a utility program for examining binary solution files

List35 permits the user to examine the contents of the binary solution file. List35 writes a text file named after the Automesh input file, but with extension TXT. The unformatted binary solution file contains mesh and solution arrays and other variables. Automesh writes a complete description of the mesh geometry. The Fish, CFish, Poisson, and Pandira codes append the solution at each mesh point, consisting of the magnetic field for rf problems, vector potential for magnet problems, or scalar potential for electrostatic problems. Postprocessor programs SFO, SF7, Force, SF8, and WSFplot read information in the binary solution file and display the results in a variety of useful formats.

3. Features added in version 6

In January, 1999, we released version 6 of Poisson Superfish. Version 6 retains most of the features added to version 5 between 1992 and 1999. The main change between versions 5 and 6 was the switch to a Fortran 90 compiler, which we used to create Windows programs with the usual program menus and pop-up dialog boxes. Our goal was to make the codes more robust and easier to maintain, and to make them easier to use in the Windows environment.

Extensive testing showed that version 5 and 6 codes give the same answers on a given problems. Where possible, we check the code results against solutions to problems that we can solve analytically. Some of these problems are included in the LANL\Examples subdirectories.

a. Source code organization

Development of version 6 involved a major overhaul of the codes. The organization of the source code for the Fortran 90 codes is substantially different from version 5. For example, nearly all named COMMON blocks were eliminated in favor of a smaller number of Fortran 90 modules.

Starting with version 6.03 in March, 2001 the codes use the robust programming practice that declares IMPLICIT NONE in every program unit (main program, subroutine, function, module, etc.). Poisson Superfish program units declare explicitly the type and kind of all variables and arrays. In addition, the code assigns initial values to all variables and arrays where they are declared.

b. Installation program

Version 6 uses a Setup program developed using the InstallShield commercial product. 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.) A separate section on Files and filename conventions describes all the files used in these codes.

c. Program WSFplot

We retired program VGAPLOT in September, 2001 replacing it with WSFplot, a full-featured Window plotting program. New features include several hard copy choices, arrow and circle plots to show field amplitude and direction, and numbered axes. Several new variables in SF.INI choose default startup options. The user can change most parameters from the program menu.

d. Programs Quikplot and Tablplot

The general-purpose plotting codes Quikplot and Tablplot are full-featured Windows programs in version 6. These codes include the same large selection of hardcopy drivers and other features in WSFplot. The codes also include a text editor with which the user can edit the current input file or other text files. The program named Colmplot has been eliminated as the new Tablplot program includes all of its capabilities.

e. Unlimited number of material tables and error checking of material data

The MT namelist in the Automesh input file defines material properties for magnetostatic, electrostatic, and radio-frequency problems. The solver codes read the binary file created by Automesh that contains all the material data. For rf problems, the data are the real or complex permittivity and permeability. For magnet problems, a material can use a fixed value of the reluctivity, a table of (B,γ) data, or data for anisotropic and permanent-magnet materials. Automesh performs extensive checking of material-data input and offers advice when it detects insufficient or inconsistent entries.

Since August, 2001 the maximum number of materials has been limited only by computer resources. In version 6, Automesh set the maximum number of materials and material tables at run time according to settings in file SF.INI. Starting with version 7, Automesh determined the required setting directly from the input file. Codes then allocate arrays for storing the material data. Material 0 corresponds to unmeshed regions, and material 1 is reserved for open space and coil regions. The codes allows as many user-defined materials as needed, and there are none of the past restrictions on material numbers. For example, in magnet problems any material number 2 or higher can use either a fixed permeability, a variable permeability linked to an internal or user-supplied table, or the straight-line B-H curves associated with an anisotropic or permanent-magnet material.

f. Force code improvements

In April, 2001for version 6.04, we made several changes in the Force program. A new example directory Examples\Magnetostatic\ForceTest contains two problems that verify the balance of forces between a current carrying wire loop and a piece of iron.

Pop-up menu

The Force program no longer requires the coordinates of a starting point if the user supplies the region number via the pop-up menu. In this case, the code computes the force on the entire region.

Identifying boundary types

The code in Force that determines the type of boundary along the path of integration has been improved. The previous method would sometimes fail to correctly identify the path around an iron region, which would result in no force calculation.

Input units

Force now deals correctly with input units other than centimeters.

Field interpolation

Along iron-space boundaries, Force now uses the standard Poisson Superfish field interpolator rather than an interpolation based upon the three nodes of one mesh triangle.

g. Field interpolator improvements

We improved performance of the field interpolator by modifying the section of code that tries to pick a reference point, which affects how many nearby nodes are included in the fit to harmonic polynomials. The original code counted the number of the surrounding 6 nodes that are actually in the problem geometry. If the result was 6, then it defined the node closest to the point of interest as the reference point. This approach proved to be unsatisfactory in magnet problems containing iron of fixed or variable permeability. Near an iron-space boundary, the above test always finds 6 nodes in the problem geometry. As a result, the polynomial fit may use only 11 to 13 points instead of the maximum possible 19 points. The new algorithm asks how many of the 6 surrounding triangles are in the material of interest. If the value is less than 6, then the code asks the same question of the nearby nodes and chooses as the reference point one of the nodes having the most triangles in the desired material. After collecting all the available first and second nearest neighbors of the reference point, if the total number of points is less than the full complement of 19 nodes, the interpolator may add some of the closest third nearest neighbors that are in the desired material. This scheme results in better fitted results as measured by the chi-squared per degree of freedom.

Configuration options in the SF.INI file allow user control over whether the field interpolator includes third nearest neighbor points in the fit. This feature allows users to investigate whether omitting third nearest neighbors results in more accurate interpolated fields. Generally, the accuracy will be better with the third nearest neighbors unless the mesh is very coarse in the region of interest.

h. Program MK_SFINI

MK_SFINI is a new utility program that creates a new copy of the SF.INI file. The installation program Setup.EXE runs this code to update or create file SF.INI. The user can run MK_SFINI to restore default settings or generate a new copy of SF.INI in any specified directory.

i. Tuning-code companion programs

Programs DTLCells and CCLCells are new companion programs to DTLfish and CCLfish, respectively. These codes will read the Parmila.OUT file for a linac design and generate a new DTLfish or CCLfish input file with each cell geometry interpolated from the original set of representative cells.

ELLCAV is a companion program to ELLfish that generates an Autofish input file for a multicell cavity. ELLCAV assumes that the internal and external cell have already been tuned by appropriate ELLfish sessions. The code reads the resulting tuned geometry from an ELLfish control file. This tool can be useful for finding the higher-order longitudinal modes of the cavity from which one can compute the cell-to-cell coupling.

4. Features added in version 7

In April, 2003, we released version 7 of Poisson Superfish. Version 7 retains all features from version 6. The main change between versions 6 and 7 was the switch to a Fortran 95 compiler that supports both the Windows and Linux operating systems. Under Windows, it appears that LF95-compiled codes perform substantially better than LF90-compiled codes. Fish and Pandira solvers invert the tridiagonal matrix in 20% to 40% of the time required in version 6. The magnitude of the improvement depends on the size of the mesh and on the size of the processor’s on-board cache memory.

a. Conversion of binary solution files to version 7 format

Lahey Fortran LF90 and Lahey/Fujitsu LF95 use different formats for sequential unformatted files such as the Poisson Superfish binary solution files. Users who have solutions files created by version 6 can use the files in version 7. Upon first encountering a solution file in the old format, Poisson Superfish codes translate it to the version 7 format. This translation is irreversible, you cannot run a version 6 code on a version 7 binary solution file.

b. Automesh improvements

Automesh now pre-scans the input file to find the coordinates of the problem boundaries. By requiring that the first region enclose the entire problem geometry, the code can perform more robust error checking on subsequent regions and ensure that boundary conditions at the edges are applied properly.

Also because Automesh pre-scans the input file, we were able to eliminate several SF.INI settings that defined limits on the number of line regions and materials.

c. *Larger problems supported*

There is no practical limit except disk space for the size of a scratch file if needed by programs Fish, CFish, Pandira, or the tuning codes. The previous limit under version 6, compiled by Lahey LF90 was 2 GB. The version 7 limit is 18 EB, nearly 10 billion times larger.

d. *Numbered warning messages*

All warning messages now have a number. The user can choose whether or not codes display a pop-up message for any or all warnings. (Warnings still appear in output text files.)

e. *Dynamic link library containing the field interpolator*

Version 7 includes files for several language systems that allow users to call the Poisson Superfish field interpolator from their own programs via a Windows dynamic link library (DLL).

B. References

We include two lists of reference material, both in chronological order. The first section contains references to work that specifically mention the Poisson Superfish codes or earlier codes crucial to the development of these codes. The second section lists general references to the theory of electricity and magnetism and also related computer programs. The lists include all of the references that appeared in the 1987 Reference Manual.

1. Published references to Poisson Superfish

Following are some of the published papers to work done on the Poisson Superfish codes. The first paper by Winslow describes a code for diffusion calculations from which the original LATTICE and Poisson codes were derived (see the history section).

1. A. Winslow, "Numerical Solution of the Quasilinear Poisson Equation in a Nonuniform Triangular Mesh." Journal of Computational Physics **2**, 149-172 (1967).
2. K. Halbach, "A Program for Inversion of System Analysis and Its Application to the Design of Magnets," Lawrence Livermore National Laboratory report UCRL-17436, CONF-670705-14 (1967); also Proceedings of the Second International Conference on Magnet Technology, Oxford, England, 47 (July 10-14, 1967).
3. K. Halbach, "Application of Conformal Mapping to Evaluation and Design of Magnets Containing Iron with Nonlinear B(H) Characteristics," Nuclear Instruments and Methods **64,** p. 278 (1968).
4. K. Halbach, "Calculation of the Stray Field of Magnets with Poisson," Nuclear Instruments and Methods **66,** p. 154 (1968).
5. K. Halbach and R. F. Holsinger, "SUPERFISH -- A Computer Program for Evaluation of RF Cavities with Cylindrical Symmetry," Particle Accelerators **7** (4), 213-222 (1976).

6. K. Halbach, R. F. Holsinger, W. E. Jule, and D. A. Swenson, “Properties of the Cylindrical RF Evaluation Code SUPERFISH,” Proceedings of the 1976 Proton Linear Accelerator Conference (September 14-17, 1976).
7. K. Halbach, “Design of Permanent Multipole Magnets with Oriented Rare Earth Cobalt Materials,” Nuclear Instruments and Methods **169**, p. 1-10 (1980).
8. L. Gluckstern, R. F. Holsinger, K. Halbach, and G. N. Minerbo, “ULTRAFISH - Generalization of SUPERFISH to m > 1,” Proceedings of the 1981 Linear Accelerator Conference, Santa Fe, New Mexico, USA (October 19-23, 1981).
9. S. O. Schriber, Ed., Chalk River Nuclear Laboratories report AECL-5677, pp. 122-128.
10. M. S. de Jong and F. P. Adams in the AECL internal report “Seafish, A 2-Dimensional Complex Helmholtz Equation Solver.”
11. M. S. de Jong, F. P. Adams and R. Hutcheon, “Computation of RF Fields for Applicator Design,” Journal of Microwave Power and Electromagnetic Energy **27**, No. 3, 136 (1992).
12. J. H. Billen and L. M. Young, “Poisson/Superfish on PC Compatibles,” Proceedings of the 1993 Particle Accelerator Conference, Vol. 2 of 5, 790-792 (1993).

2. Other references

1. A Stratton, Electromagnetic Theory, McGraw-Hill, (New York, 1941).
2. Maier Jr. and J. C. Slater, Journal of Applied Physics **23**, p. 68 (1952).
3. M. Morse and H. Feshbach, Methods of Theoretical Physics, McGraw Book Co. (1953).
4. L. Ginzton, Microwave Measurements, McGraw-Hill Book Co. (1957),
5. D. Jackson, Classical Electrodynamics, John Wiley, (New York, 1962).
6. C. Herewood in Linear Accelerators, edited by P. M. Lapostolle and A. L. Septier, North Holland Publishing Co., p. 28 (New York, 1970).
7. Carne et al. in Linear Accelerators, edited by P. M. Lapostolle and A. L. Septier, North Holland Publishing Co., p. 147-83 (New York, 1970).
8. A. Loew et al., “Computer Calculations of Traveling-Wave Periodic Structure Properties,” IEEE Transactions on Nuclear Science **NS26,** p. 3701 (1979).
9. M. Fomell, V. P. Jackowlev, M. M. Karliner, and P. B. Lysyansky, “LANS - A New Code for the Evaluation of the Electromagnetic Fields and Resonance Frequencies of Axisymmetrical Cavities,” Particle Accelerators **11**, p. 173-9 (1981).
10. C. Daikovsky, Yu I. Portugalov, and A. D. Ryabov, “PRUD - Code for Calculation of the Nonsymmetric Modes in Axial Symmetric Cavities,” Particle Accelerators **12**, p. 59-64 (1982).
11. Wilhelm, “CAVIT and CAV3D - Computer Programs for RF Cavities with Constant Cross Section or any Three Dimensional Form,” Particle Accelerators **12**, p. 139-45 (1982).

12. L. Gluckstern, R. D. Ryne, and R. F. Holsinger, “Numerical Programs for Obtaining Accurate Resonant Frequencies of Modes in Azimuthally Symmetric Electromagnetic Cavities,” Proceedings of the 1983 COMPUMAG Conference, Genoa, Italy.
13. Weiland, “TBCI and URMEL - New Computer Codes for Wake Field and Cavity Mode Calculations,” IEEE Transactions on Nuclear Science **NS30,** p. 2489-91 (1983).
14. L. Gluckstern, “RF Systems: Electromagnetic Fields in RF Cavities and Cavity Chains,” Physics of High Energy Particle Accelerators (Stony Brook Summer School, 1983) AIP Conference Proceedings No. 87 (New York, 1984).
15. A. Lowther and P. P. Silvester, Computer-Aided Design in Magnetics, Springer-Verlag (Berlin, 1986).

C. Terminology

This section includes definitions of a few terms should help users understand the Poisson Superfish codes and documentation. The authors hope to expand this section into a useful glossary as time permits.

1. Superfish codes

We sometimes use this term to refer to the entire distribution of codes including installation programs, mesh generation codes, solver codes, postprocessors, tuning codes, plotting codes, and other utilities. As we continue to update this documentation we are attempting to replace these references with the phrase “Poisson Superfish codes”

2. Solver program

The solver programs refer to Fish, CFish, Poisson, and Pandira. These programs calculate the solution to the problem set up by the code Automesh. Programs Autofish, CCLfish, CDTfish, DTLfish, ELLfish, MDTfish, RFQfish, and SCCfish all include a copy of the Fish solver.

3. Tuning program

The tuning programs are integrated code packages that tune certain types of resonant cavities to a desired frequency by adjusting a geometrical parameter. These programs include CCLfish, CDTfish, DTLfish, ELLfish, MDTfish, RFQfish, and SCCfish.

4. Superfish problem or Poisson problem

These terms refer to the two different classes of problems that you can solve using the Poisson Superfish codes. A “Superfish problem” is one in which the resulting solution consists of the radio-frequency or rf field distribution in a resonant cavity or waveguide. The solver programs for Superfish problems include Fish, CFish, Autofish, and the tuning programs CCLfish, CDTfish, ELLfish, DTLfish, MDTfish, RFQfish, and SCCfish. A “Poisson problem” refers to either a magnetostatic and electrostatic solution of Poisson’s equation. The solver programs for Poisson problems are Poisson and Pandira.

We often use the phrases “Fish and CFish” and “Poisson and Pandira.” When used as an adjective “Fish and CFish” refers to a “Superfish problem” and “Poisson and Pandira” refers to a “Poisson problem” in the context discussed above.

A CFish problem is a Superfish problem with complex fields and complex material properties. It must be solved using the CFish program.

5. Postprocessor

The postprocessors include the programs SFO, SF7, Force, and WSFplot. These programs read the solution file created by one of the solver programs and calculate various useful quantities. They all use a common field interpolator.

6. Plotting program

The plotting programs include the Poisson Superfish plotting program WSFplot and general purpose programs Quikplot and Tablplot.

7. SOR

The acronym SOR stands for successive over-relaxation. Automesh uses this numerical technique to optimize the size of mesh triangles. Poisson uses SOR to solve for the potential.

8. 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, though the actual filename is the name of the Automesh input file extension T35. This binary solution file contains information shared by all the Poisson Superfish programs.

9. Configuration file or initialization file

The Poisson Superfish codes read the file SF.INI for the settings of configurable options that affect program operation. The installation program put a copy of SF.INI with suggested settings in your LANL directory.

10. Boundary conditions

To compute a unique and well-behaved solution of Poisson’s (or Laplace’s) equation requires specification of the appropriate boundary conditions. The Poisson Superfish codes can accommodate two types of boundary conditions. One type, which is known as a Dirichlet boundary, in honor of P. G. L. Dirichlet (1805-1859), occurs when the value of the potential is specified at each boundary point.. The WSFplot contour lines are parallel to a Dirichlet boundary. For the other type, which is known as a Neumann boundary, in honor of Karl Gottfried Neumann (1832-1925), the value of the derivative, or rate of change, of the potential in the direction normal to the boundary is specified instead. The WSFplot contour lines are perpendicular to a Neumann boundary.

11. Beta, the symbol β

The symbol β has two very common uses in accelerator physics. Fortunately, the reader can easily deduce which one is meant from the context. In one context, the symbol β is the ratio v/c of the particle velocity to the speed of light. We often refer to β itself as the "velocity." Another use of β is the Twiss or Courant-Snyder ellipse parameter.

12. Beta-lambda, the product βλ

The product βλ is the distance traveled by a particle of velocity βc in one rf period of the resonant mode, whose frequency is f = c/λ. Linac designers often use βλ as a measure of length.

13. Cavity

A cavity refers to a resonant structure used to apply high electric fields near the beam axis to accelerate and focus the beam longitudinally. For the DTL, the resonant structure is a tank containing many cells or accelerating gaps. Cavities in the CCL and CCDTL structures include both accelerating cavities and coupling cavities.

14. CCDTL

An acronym for coupled-cavity drift-tube linac. The CCDTL is a hybrid structure that combines features of the DTL and CCL. This structure is like the CCL because the chain of accelerating cavities connect to one another though coupling cavities. Each individual accelerating cavity is a short drift-tube linac contains a small number of drift tube (usually only one or two). The structure's main advantages are that is allows placement of the transverse focusing magnets outside the rf structure and the short DTL cavities do not require post-couple stabilization. Without magnets inside the drift tubes, the drift-tube shape can be optimized for good shunt impedance. Since the mid 1990s Los Alamos National Laboratory has been working on the design of two high-current ion linacs that will use the CCDTL structure between ~10 MeV and 100 MeV proton energy.

15. CCL

An acronym for coupled-cavity linac, which consists of a chain of accelerating cavities connected together into a single resonant structure through coupling cavities. An example of a CCL is the LAMPF side-coupled structure. The acronym SCL is often used to refer to a side-coupled linac. This terminology distinguishes it from other types of CCL structures such as on-axis coupled structures.

16. Cell

A cell is a portion of a cavity containing a single gap. The term cell also refers to a half wavelength of the modulation in an RFQ accelerator. The terms cell and gap are often used interchangeably.

17. Coupling

The term coupling has been commonly used for two purposes: one refers to “coupling” the electromagnetic energy from a waveguide into a cavity, the other to the mutual inductance (e. g. for slots) or mutual capacitance (e. g. axial electric field) between cavities.

18. Coupling cell or coupling cavity

The nominally unexcited cavity between accelerating cavities for a structure operating in the $\pi/2$ mode. In the CCDTL, coupling cavities have the conventional orientation (with the cavity axis parallel to the beam axis) for integer multiples of $\beta\lambda$ between accelerating cavities. Coupling cavities turned on their side allow half-integral $\beta\lambda$ intervals between accelerating cavities.

19. DTL

An acronym for the drift-tube linac invented by Alvarez. The DTL is a graded-β structure that consists of a “tank” containing many cells, each cell slightly longer than the last one. Examples include the LAMPF 201.25-MHz DTL, and many others at laboratories around the world.

20. Gamma, the symbol γ

The symbol γ has several uses in accelerator physics and electromagnetic theory. One must deduce which use is meant from the context. In Parmila, the symbol γ is the relativistic factor:

$$\gamma = \frac{1}{\sqrt{1-\beta^2}},$$

where $\beta = v/c$ is the ratio of particle velocity to the speed of light. Another use of γ in Parmila is the Twiss or Courant-Snyder ellipse parameter.

In the Poisson and Pandira codes, the symbol γ refers to reluctivity, which is the reciprocal of the magnetic permeability μ.

21. Gap

The term gap refers to the longitudinal space between drift-tube noses across which the accelerating field appears in a resonant cavity. The Parmila beam-dynamics simulation transports particles across the accelerating gaps using an approximation called the drift-kick method. The terms cell and gap are often used interchangeably. A cell is a portion of a cavity containing a single gap.

22. Geometric β

The geometric β is a characteristic of a cavity or series of cavities related to the distance between the rf gaps. It refers to the velocity of a particle that is synchronous with the rf fields in that section of rf structure. Strictly speaking, the synchronous phase for a series of identical cavities is –90 degrees: the synchronous particle would not gain any energy. Building a series of identical symmetric cavities is an expedient introduced to save design and fabrication costs.

23. Graded β

The CCL and CCDTL structures can use "graded-β" cavities, in which successive cells change length to keep in step with the particle velocity βc. The CCL and CCDTL structures also can be assembled in segments of constant-β cavities.

24. Kilpatrick field

The Kilpatrick field is commonly used to describe the design field level for accelerating cavities. Kilpatrick observed experimentally that the breakdown field in resonant cavities was related to the frequency. Tom Boyd expressed Kilpatrick's result in a convenient formula. The formula relates frequency f in MHz to the Kilpatrick limit E_k in MV/m:

$$f = 1.643 E_k^2 \exp\left(\frac{-8.5}{E_k}\right).$$

Higher surface fields (to ~$2E_k$) are now fairly easily achieved. Program SFO reports the ratio of a cavity's maximum field to the Kilpatrick field in the output summary. The code distribution also includes a stand-alone utility program Kilpat for calculating the Kilpatrick field as a function of frequency.

25. Modes

The term mode refers to the standing waves in a resonant structure or in an individual resonant cavity. While it is beyond the scope of this document to discuss all the properties of these resonant modes, we present a few comments on the common use of the terminology in accelerator physics.

a. Cavity modes

Single-cavity modes are usually described as either transverse magnetic (TM) modes or transverse electric (TE) modes. The indices n, m, and p in the nomenclature TM_{mnp} and TE_{mnp} describe the field pattern in the cavity. For cylindrically symmetric pipe with end walls, index m is the number of full period variations in angle θ of the field components, index n is the number of axial-field zeros in the radial direction (excluding the zero at r = 0), and index p is the number of half-period variations in z of the fields.

With this definition in mind, the two most important modes in accelerator cavities are the TM_{010} mode used for acceleration in the DTL, CCDTL and CCL cavities and the TE_{210} quadrupole mode used in the RFQ cavity. Note that the TE_{210} mode cannot exist in an empty circular pipe because of the boundary conditions. Yet this nomenclature does accurately describe the RFQ cavity mode: there are no longitudinal zeros of the field over the length of the RFQ vanes. The ends of the RFQ vanes are undercut and tuned carefully to provide this mode over the length of the vanes.

b. Structure modes

Another type of nomenclature has developed over the years to describe the pattern of excitation in the individual cavities of a multi-cavity structure. In a structure consisting of a linear chain of N oscillators, there will be N modes of excitation of the cavities. In all N modes, every cavity has the same field pattern (e.g., TM_{010}). Only the level of excitation differs from one cavity to the next cavity. A series of modes in a structure (for a particular cavity mode) is often called a band or a passband.

We use the cavity-to-cavity phase shift to describe a structure mode. For example, “zero mode” or “0 mode” refers to a multi-cavity structure mode with zero phase shift from cell to cell. The DTL operates in zero mode. Likewise, the π mode has a 180-degree phase shift cell to cell. Superconducting cavities typically operate in the π mode.

A very important mode for linac structures is the $\pi/2$ mode, which has a 90-degree phase shift cell to cell. That is, every other cavity is unexcited. The CCL and CCDTL operate in the $\pi/2$ structure mode. This mode is preferred because of its very good stability compared to either 0-mode or π-mode operation.

26. Post coupler

The post coupler is a resonant field-stabilization device in DTL tanks. The post couplers serve the same purpose as the coupling cavities do in CCL and CCDTL structures. The posts, of electrical length $\lambda/4$, attach to the wall of the tank and protrude in toward the center of the drift tubes. They alternate from one side of the tank to the other, at azimuth angle of ± 90 degrees from the stems that support the drift tubes.

27. Quality factor Q

The quality factor is a figure of merit for a cavity given by the formula

$$Q = \frac{\omega U}{P} ,$$

where ω is the angular frequency ($2\pi f$), U is the cavity stored energy, and P is the power dissipated in the walls. For resonant frequencies in the range 100 to 1000 MHz, typical values are 10,000 to 50,000 for normal conducting copper cavities; 10^8 to 10^{10} for superconducting cavities.

28. RFQ

The radio-frequency quadrupole accelerator invented by Kapchinskiy and Tepliakov. Group AT-1 at Los Alamos built and tested the first RFQ in the West in the late 1970s. There are now many RFQs operating around the world.

29. Shunt impedance

Shunt impedance is an accelerator figure of merit the measures the amount of accelerating field per unit power expended. There are several ways of expressing this efficiency. The Parmila program uses the quantity called shunt impedance per unit length Z computed by the Superfish postprocessor SFO:

$$Z = \frac{E_0^2}{P/L},$$

where P/L is the power dissipated in the walls per unit length of accelerator structure, E_0 is the average axial electric field. Another useful concept is the effective shunt impedance per unit length ZT^2, which includes the square of the transit-time factor. Thus ZT^2 takes into account the velocity of the particles being accelerated. Typical values of ZT^2 for normal conducting linacs is 30 to 50 MΩ/m. The shunt impedance is not relevant for superconducting linacs.

30. Skin depth

For normal conductors, the rf surface resistance can be expressed in terms of the skin depth δ as

$$R_s = \frac{1}{\sigma\delta},$$

where σ is the dc conductivity, the reciprocal of the resistivity ρ. The skin depth is

$$\delta = \sqrt{\frac{2}{\sigma\mu\omega}},$$

where $\omega = 2\pi f$ and f is the resonant frequency, and μ is the permeability given by

$$\mu = \mu_r \mu_0,$$

where μ_r is the relative permeability of the material, and $\mu_0 = 4\pi \times 10^{-7}$ T-m/A. For normal conducting accelerating cavities the material of choice is usually copper, for which $\mu_r = 1$. The skin depth for copper at 400 MHz is about 3.3 μm and corresponds to a surface resistance $R_s = 5.2$ mΩ.

If a magnetic material were subjected to the rf magnetic fields, then one would need to take into account it relative permeability. A high permeability considerably reduces the skin depth and raises the surface resistance. For rf accelerator cavities, a typical magnetic field H_ϕ is of the order of 10^4 A/m, which corresponds to a magnetic induction B ~ 120 Gauss. Thus, magnetic materials generally do not reach saturation when subjected to the rf-cavity fields. A low-field value of the permeability for a magnetic material (in the linear portion of the material's B-H curve) is sufficient for computing the skin depth.

31. Transit-time factor

The transit-time factor accounts for the fact that the rf fields are changing in time as the particle traverses the cell. The simplest definition (where the position z = 0 is assumed to be in the center of the cell) is:

$$T = \frac{\int_{-L/2}^{L/2} E_z \cos\frac{2\pi z}{\beta\lambda} dz}{\int_{-L/2}^{L/2} E_z dz}$$

Note that the definition is independent of the synchronous phase and can be computed from the field distribution in the cavity as long as one knows the particle velocity β. If the center of the cell is not at z = 0, the expression for T has another term and it is no longer independent of the synchronous phase. The SFO chapter includes a more extensive discussion of the transit-time factor and related integrals. A useful approximation for the transit-time factor can be derived for the square-wave electric field distribution shown in Figure III-1. The transit-time factor is given by the equation

$$T = \frac{1}{I_0(2\pi a/\lambda)} \frac{\sin \pi g/\beta\lambda}{\pi g/\beta\lambda}$$

where $$I_0(x) \approx 1 + \frac{x^2}{4}.$$

The first term depends on the bore radius a, and the second term on the gap g. For typical values: a = 0.04λ, g = βλ/4, T ≈ (0.98) × (0.90) = 0.88. For g = βλ/2, the gap term is 2/π or about 0.64.

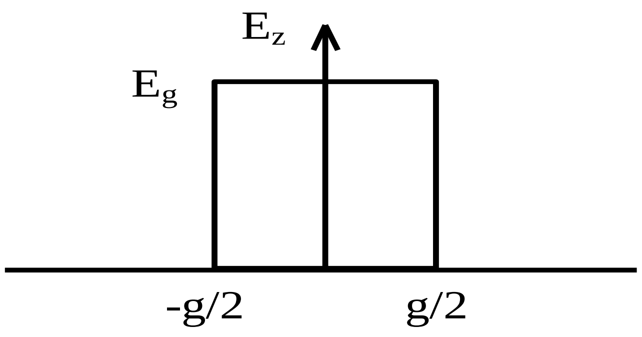

Figure III-1. Square-wave electric field distribution.

D. Summary of the Poisson Superfish codes

Here is a brief description of the programs. Detailed sections appear later in this document for the individual Poisson Superfish codes (Autofish, Automesh, Fish, CFish, Poisson, Pandira, the postprocessors WSFplot, SFO, SF7, and Force); the automated tuning programs (CCLfish, CDTfish, DTLfish, ELLfish, MDTfish, RFQfish, and SCCfish); the plotting programs (Quikplot and Tablplot); and the utility programs (List35, Beta, Kilpat, ConvertF, FScale, SF8, SegField, and SFOtable).

1. Autofish

For radio-frequency (rf) problems, Autofish combines the calculations done by Automesh, Fish, and SFO into a single program. It saves the extra load time and disk reads necessary to run all the codes separately. You cannot use Autofish for Poisson and Pandira problems.

2. Automesh

Whether you are solving a static magnetic field, a static electric field, or an rf problem, Automesh is always the first program to run. Automesh reads the namelist input file and sets up mesh data in file TAPE36, which is a text file containing all of the mesh points along boundary segments in the problem. Automesh writes this data and other information to file OUTAUT.TXT. All variables needed by the solver programs must appear in the first REG namelist section of the input file. After processing all the regions and material tables in the input file, Automesh reads the TAPE36 file that it wrote and generates the entire mesh for Fish, CFish, Poisson, or Pandira. After generating the mesh, Automesh erases TAPE36. The code fills in all the internal (non-boundary) points and optimizes the mesh by adjusting the size of the mesh triangles. Automesh produces the binary file with extension T35, which contains all the problem variables plus a complete description of the mesh geometry. File OUTAUT.TXT lists the problem variables and other diagnostic information in the event of a problem with the mesh.

3. Lattice

In older versions of Poisson Superfish, it was necessary to run program Lattice after Automesh. The functions previously performed by Lattice are now done in Automesh.

4. Fish

Fish is the radio-frequency field solver and runs after Automesh for problems that have KPROB = 1. The code reads the binary solution file started by Automesh and writes the solution array to the file. The solution for the electromagnetic field consists of the magnetic field at each mesh point. Fish writes the text file OUTFIS.TXT, which includes a list of the problem variables, material properties for each region, a record of the resonance search (or frequency scan), and a short list of quantities computed by the code during the run. If an error occurred, OUTFIS.TXT will usually include the error message and occasionally some helpful advice.

5. CFish

CFish is a version of Fish that uses complex variables for the rf fields, permittivity, and permeability. The code reads and writes the same files as Fish. Lloyd Young developed the CFish code following the method described by M. S. de Jong and F. P. Adams in the AECL internal report “Seafish, A 2-Dimensional Complex Helmholtz Equation Solver.” See also the paper by M. de Jong, F. Adams and R. Hutcheon, “Computation of RF Fields for Applicator Design,” Journal of Microwave Power and Electromagnetic Energy 27, No. 3, 136 (1992).

6. Poisson

Poisson is one of the static-field solvers and runs after Automesh for problems that have KPROB = 0. (The other static field solver is Pandira.) Poisson is a magnetostatic and electrostatic field solver that uses the method of successive over relaxation. The code reads the binary solution file started by Automesh and writes the solution to the file. The solution consists of the vector or scalar potential at each mesh point. If your problem has MODE = 0, you must supply the material properties in MT namelist sections in the Automesh input file. Poisson writes the text file OUTPOI.TXT, which includes a list of the problem variables, material properties for each region, a record of the successive over-relaxation progress, interpolated fields, results of a harmonic analysis if requested, and a short list of quantities computed by the code during the run. If an error occurred, OUTPOI.TXT will usually include the error message and occasionally some helpful advice.

7. Pandira

Pandira is one of the static-field solvers and runs after Automesh for problems that have KPROB = 0. (The other static field solver is Poisson.) Pandira is a magnetostatic and electrostatic field solver that uses a direct method for inverting the tridiagonal matrix. Unlike Poisson, Pandira can handle permanent magnet materials. The code reads the binary solution file started by Automesh and writes the solution to the file. The solution consists of the vector or scalar potential at each mesh point. If your problem has MODE = 0, you must supply the material properties MT namelist sections in the Automesh input file. Pandira writes the text file OUTPAN.TXT, which include the same information as OUTPOI.TXT written by Poisson. If an error occurred, OUTPAN.TXT will usually include the error message and occasionally some helpful advice.

8. WSFplot

WSFplot is the Poisson Superfish plotting program. You can run WSFplot after running Automesh to view the mesh and after running one of the solver programs to view the solution. If you run WSFplot after running the postprocessor SFO (on Superfish problems), then the field-display window will include the field normalization computed in SFO. Upon starting, WSFplot automatically plots the mesh and/or contours of constant A_z for rectangular magnet problems, rA_ϕ for cylindrically symmetric magnet problems, V for electrostatic problems, H_z for rectangular rf problems, or rH_ϕ for cylindrically symmetric rf problems. For complex rf fields, WSFplot plots the real and imaginary parts of H_z or rH_ϕ in different colors. Some settings in file SF.INI affect the behavior of the code at startup. Once the program is running, you can press the Esc key to redefine the plotting parameters.

9. SFO

SFO is a Poisson Superfish postprocessor. The code reads the solution from the binary solution file produced by the solver programs. The solution consists of the magnetic field (for rf problems), vector potential (for magnet problems), or scalar potential (for electrostatic problems) at each mesh point. For Superfish problems, SFO updates the solution file so that it contains the field-normalization information used by programs SF7 and WSFplot. SFO will read input data from a file if you supply the filename on the command line. If no filename appears on the command line, the code automatically reads a file with the same name as the Automesh input file, but with extension SEG, if such a file exists. If a problem requires tables of data (for example a list of segments on which to compute power), then you must supply these data in a SEG file. The recommended procedure for supplying other data is to include the appropriate entries in the Automesh input file.

SFO writes the text file named after the Automesh input file, but with extension SFO. This output file includes a list of problem variables and tables of field values along specified segments. For Superfish problems, the SFO output file includes power losses, transit-time-factor integrals, a summary of cavity data, and a short list of quantities computed by the code during the run. If an error occurred, the SFO output file will usually include the error message and occasionally some helpful advice.

10. SF7

SF7 is a Poisson Superfish postprocessor for interpolating fields on lines, arcs, rectangular grids, or user-supplied curves. The code reads the binary solution file produced by the solver programs. The solution consists of the magnetic field (for rf problems), vector potential (for magnet problems), or scalar potential (for electrostatic problems) at each mesh point. SF7 will read input data from a file if you supply the filename on the command line. If no filename appears on the command line, the code automatically reads a file with the same name as the Automesh input file, but with extension IN7, if such a file exists. In the absence of an input file, SF7 displays a dialog box where you can enter the required settings.

The code writes tables of interpolated field values in specified regions of the problem geometry to file OUTSF7.TXT. The same data appears in plot files for program Tablplot. Input options in SF7 writes 2-D field maps for input to program Parmela or EGUN.

11. Force

Force is a Poisson and Pandira postprocessor for computing the force on objects from the magnetic field. The code reads the binary solution file produced by Poisson or Pandira. The solution consists of the vector potential at each mesh point. Force will read input data from a file if you supply the filename on the command line. If no filename appears on the command line, the code automatically reads a file with the same name as the Automesh input file, but with extension FCE, if such a file exists. Force calculates the force on specified coil or iron regions. It writes file OUTFOR.TXT, which contains a table of the integration path around the force element followed by the calculated force.

12. DTLfish

DTLfish is a tuning program for drift-tube linac cells. This program sets up the geometry for drift-tube linac (DTL) cells for input to the Superfish codes. The DTL cell is a figure of revolution about the beam axis. DTLfish assumes a symmetric cell, and sets up Superfish runs for only half the cell. The symmetry plane is in the gap center between the two drift-tube noses. DTLfish runs the Superfish codes repetitively, varying the cell geometry to tune each cell to a specified frequency. The code tunes cells by adjusting either the tank diameter, drift-tube diameter, gap, or face angle.

13. DTLCells

DTLCells is a companion program for DTLfish that creates a new input file for DTLfish containing cell geometries interpolated from a set of representative cells. The code reads the Parmila.OUT file for a linac design and generates a DTLfish problem for each cell in a tank with a particular starting energy.

14. CCLfish

CCLfish is a tuning program for coupled-cavity linac cells. This program sets up the geometry for coupled-cavity linac (CCL) cells for input to the Superfish codes. The CCL cell is a figure of revolution about the beam axis. CCLfish assumes a symmetric cell, and sets up Superfish runs for only half the cell. The symmetry plane is in the gap center between the two noses. CCLfish runs the Superfish codes repetitively, varying the cell geometry to tune each cell to a specified frequency. The code tunes cells by adjusting either the cavity diameter, septum thickness, gap, or cone angle. You also can use this program to tune buncher cavities and iris-loaded cavities.

15. CCLCells

CCLCells is a companion program for CCLfish that creates a new input file for CCLfish containing cell geometries interpolated from a set of representative cells. The code reads the Parmila.OUT file for a linac design and generates a CCLfish problem for each unique cavity over a specified range of energies.

16. CDTfish

CDTfish is a tuning program for the coupled-cavity drift-tube linac (CCDTL). The CCDTL cavity is a figure of revolution about the beam axis. It resembles a CCL cavity, but with one or more internal drift tubes. CDTfish sets up either full cavities or half cavities with optional extensions of the bore tube on either side. The code tunes the cavity by adjusting either the cavity diameter or the drift-tube gaps. To correct for longitudinal asymmetries in the gap fields, the code uses information from the SFO transit-time-factor integrals to align the cell’s electrical centers. This program also will tune ordinary CCL cells that contain no internal drift tubes.

17. ELLfish

ELLfish is a tuning program for elliptic cavities, which are often used in superconducting applications. These cavities feature an elliptical segment near the bore radius. ELLfish can tune either a symmetric half cavity or an asymmetric full cell that includes an attached bore tube. The code tunes the cell by adjusting either the cavity diameter, outer radius of curvature, or the angle that the straight side of the cavity makes with the vertical.

18. ELLCAV

ELLCAV is a companion program for ELLfish for generating a multicell-cavity file from the internal and external cells have already been tuned by appropriate ELLfish sessions. This tool facilitates computing the higher-order longitudinal modes of the cavity from which you can compute the cell-to-cell coupling. Another use of the ELLCAV-generated input file is the creation of a family of PMI files for supplying transit-time-factor data to Parmila.

19. MDTfish

MDTfish is a tuning program for multicell drift-tube linacs. This program sets up the geometry for several cells of a drift-tube-linac tank for input to the Superfish codes. Individual drift-tubes have the same diameter and radii, but they can have different face angles. Output from MDTfish includes the distribution of E_0. The code tunes the cavity by varying either the tank diameter or the average gap. We recommend using DTLfish for designing the cells of a DTL tank. The main use of MDTfish is to restore the proper tuning to cells near the end of a DTL tank.

20. MDTCells

MDTCells is a companion program for DTLfish and MDTfish. The code converts a DTLfish input file containing multiple problems (cells) to a MDTfish input file.

21. RFQfish

RFQfish is a tuning program for radio-frequency quadrupole cavities. This program sets up the geometry for a cross section of an RFQ in Cartesian coordinates for input to the Superfish codes. The code assumes quadrupole symmetry and sets up the geometry for one quadrant. RFQfish tunes the cavity by varying the cavity volume or the shape of the vane tip. Several different tuning surfaces are available for adjusting the cavity volume.

22. SCCfish

SCCfish is a tuning program for side coupling cavities typically used in the construction of coupled-cavity linacs designed by the codes CCLfish and CDTfish. The side coupling cell is a figure of revolution and contains a central tuning post. SCCfish assumes a symmetric cell, and therefore sets up Superfish runs for only half the cell. The symmetry plane is in the center of the space between the two tuning posts. SCCfish tunes the cell by adjusting either the cavity diameter or the tuning post length.

23. Quikplot

Quikplot produces X-Y plots of data read from a text file. The default extension for input files is QKP. The code plots data on the screen and writes hardcopy graphics files in several popular formats. Quikplot can integrate the plotted data or fitted curve between specified limits.

24. Tablplot

Tablplot plots up to 10 columns of data versus any other column of data from a table of up to 16 related parameters read from a text file. The default extension for input files is TBL. The code plots data on the screen. Tablplot plots the transit-time data in files Transit.TBL and TBeta.TBL written by SFO. It also plots the result of field interpolations written by SF7.

25. List35

List35 is a utility program for examining the contents of binary solution files with extension T35. The historical name of the solution file is TAPE35. The program writes tables of data to a text file with extension TXT.

26. SFOtable

SFOtable reads families of SFO output files from DTLfish, CCLfish, ELLfish, CDTfish, and SCCfish sessions and makes a Tablplot input file for displaying cavity parameters. SFOtable scales parameters for fixed E_0, E_0T, peak electric field, or peak power density. E_0 is the average axial electric field and T is the transit time factor.

27. SegField

SegField reads an SFO output file and makes Tablplot input files of electric and magnetic fields and power density along boundary segments of Superfish problems. It scales parameters for a specified value of E_0T, where E_0 is the average axial electric field and T is the transit time factor.

28. Beta

Beta calculates particle velocities relative to the speed of light for a given kinetic energy. The default particle mass is that of a proton, but the user also can select the mass of particles $e^{\pm}$, H^-, D^+, D^-, or manually enter a rest-mass energy for some other particle.

29. Kilpat

Kilpat calculates the Kilpatrick field limit in MV/m for a given frequency entered in MHz.

30. ConvertF

ConvertF computes the difference between the frequency of a cavity measured in air at a given temperature and the frequency of the cavity operating under vacuum at a different temperature. Gases other air (e.g., nitrogen, argon, or helium) also may fill the cavity. The calculation includes the effect of relative humidity. The cavity material may be copper, aluminum, niobium, or iron.

31. FScale

FScale generates a new tuning-program control file, but for a different rf frequency than contained in the original file.

32. SF8

SF8 computes the coupling between two cavities given information about the coupling slot dimensions and properties of the two cavities. The program also computes the frequency shift in each cavity caused by the slot. SF8 is intended to aid in the construction of coupled-cavity linac structures.

33. MK_SFINI

MK_SFINI creates a new copy of the SF.INI file. The resulting file depends upon whether a copy of SF.INI already exists. If not, the new file contains the Poisson Superfish default settings for a new installation. If SF.INI already exist, then MK_SFINI retains current settings and adds missing entries from the default installation set.

E. Problem variables

The Poisson Superfish codes share a set problem constants and variables stored in the binary solution file. The variables contain information about problem setup and the optional calculations that you have selected. Most of these parameters appear in the first REG namelist section of the Automesh input file. Some are calculated quantities reported by the programs in their respective output files. Automesh assigns default values to all the variables not entered in the input file.

1. Problem variables for Fish and CFish

This section lists all the variables used by Superfish programs Fish, CFish, Autofish, and the automated tuning programs CCLfish, CDTfish, DTLfish, ELLfish, MDTfish, RFQfish, and SCCfish. The default value is the initial setting in Automesh, unless it is a calculated value or a required input parameter. Automesh initializes calculated values to zero.

Table III-3. Superfish variables.

Variable	Default value	Description
ALPHAT	3.93D–3	Temperature coefficient of resistance at TEMPR (C^{-1}).
ASCALE	4π D–8	Scaling factor for desired Ez integral or fields.
BETA	0	Particle velocity relative to light, used in SFO if KMETHOD = 1.
BETA1	0.10	Starting BETA for table of transit-time data.
BETA2	0.95	Ending BETA for table of transit-time data.
CAPK	Calculated	Phase change per unit length in SFO (XK0/BETA).
CCLDELK	1	Increment in the coupling-slot power table in SFO.
CCLMAXK	6	Highest coupling for coupling-slot power table in SFO.
CCLMINK	1	Lowest coupling for coupling-slot power table in SFO.
CLENGTH	0	User-supplied cavity length for defining E_0.
CLIGHT	Calculated	Speed of light ($2.99792458 \times 10^{10}$ cm/s).
CONV	1	Length conversion (number of cm/unit).
DBETA	0.05	BETA increment for table of transit-time data.
DELFR	0	Frequency step size (in MHz) for a resonance search.
DIAGDLL	0	If 1, DLL functions write diagnostics to file DiagDLL.txt.
DKSQ	Calculated	Change in k^2 after an iteration.
DPHI	180	Phase length (degrees) used in SFO if KMETHOD ≠ 1.
DSLOPE	1	Slope of the $D(k^2)$ function.
DSTOLER	0.02	Tolerance on DSLOPE for convergence.
DX1	Calculated	X mesh spacing in the first region.
DXMIN	Calculated	Minimum X mesh interval.
DYMIN	Calculated	Minimum Y mesh interval.
EMAX	Calculated	Peak electric field on specified boundary segments.

Table III-3. Superfish variables. (continued)

Variable	Default value	Description
ENERGY	Calculated	Problem geometry total stored energy in RF fields.
ENORM	1.0D6	Normalize to ENORM in V/m when NORM = 4.
EPS0	ε_0	Permittivity of free space (8.85418×10^{-12} F/m).
EPSIK	1.0D–4	Frequency convergence occurs when $\lvert D(k^2) \rvert / k^2 <$ EPSIK.
EPSO	1.0D–5	Convergence parameter in during mesh optimization.
ERG	Calculated	Stored-energy integral (H^2 r dr dz or H^2 dx dy).
EZERO	1.0D6	Normalize to E_0 in V/m when NORM = 0.
EZEROT	1.0D6	Normalize to E_0T in V/m when NORM = 1.
FMU0	μ_0	Permeability of free space ($4\pi \times 10^{-7}$ T-m/A).
FREQ	0	Resonant frequency (in MHz).
FREQC	Calculated	Frequency corrected for multiple stems in SFO.
FREQD	0	Design frequency (MHz), used in SFO if KMETHOD = 1.
HCORNER	Calculated	Average H along segments ICORNER1 to ICORNER2.
HMAX	Calculated	Peak magnetic field on specified boundary segments.
HPHI	5.0D3	Normalize to H_ϕ in A/m on segment end when NORM = 2.
IBETA	0	If>0, SFO makes transit-time table from BETA1 to BETA2.
ICCC	Calculated	1 for real, 2 for complex rf fields.
ICCP	1	>0 for stored energy, material power loss in SFO, SF7, WSFplot.
ICORNER1	0	First segment number of a corner arc for computing average H.
ICORNER2	0	Last segment number of a corner arc for computing average H.
ICYCLE	Calculated	Iteration number.
ICYLIN	1	Coordinate system (0: rectangular; 1: cylindrical).
IMAX	Calculated	KMAX+2.
INFODATA	Calculated	Number of tuning-code parameters written the in SFO output file.
IOBSEG	–1	Starting segment of the CCL outer boundary.
IPIVOT	1	Controls pivoting in matrix inversion routines.
IRESID	0	Calculate residuals for the potential if IRESID = 1.
IRMAX	25	Cycles between optimization of the mesh SOR parameter.
IRTYPE	0	R_s option: 0: normal; 1: superconductor; 2: user RS; 3: user RHO.
ISLOT	0	If 1, estimate coupling-slot power losses in SFO.
ITFILE	0	If 1 in SFO, write a plot file of transit-time data.
ITOT	Calculated	Number of mesh points, (KMAX+2)*(LMAX+2).
KDRI	Calculated	K coordinate of the drive point.
KMAX	Calculated	Number of horizontal logical mesh points.
KMETHOD	0	If 1, use BETA to compute wave number in SFO, else use DPHI.
KPROB	1	1 for Superfish problems, 0 for Poisson or Pandira.
LAST35	Calculated	Code number of last program to update the binary solution file.
LCYCLE	Calculated	Number of relaxation iterations in Automesh.
LDRI	Calculated	L coordinate of the drive point.
LINT	1	Logical-mesh coordinate for E_z dz integration.

Table III-3. Superfish variables. (continued)

Variable	Default value	Description
LMAX	Calculated	Number of vertical logical mesh points.
MAXCY	19	Maximum number of cycles to find resonance.
MAXPPR	Calculated	Maximum number of boundary points per region.
METHOD	Calculated	Method used to estimate frequency in Fish root finder.
NAIR	Calculated	Number of “air” points.
NBND	Calculated	Number of Dirichlet boundary points.
NBSLF	1	Left-side boundary condition.
NBSLO	0	Lower boundary condition.
NBSRT	1	Right-side boundary condition.
NBSUP	1	Upper boundary condition.
NCELL	0	Number of cells for a multicell problem.
NDRI	Calculated	Index of the drive-point coordinates.
NEGAT	Calculated	Zero-area triangles, indicates a problem generating the mesh.
NFE	Calculated	Number of “iron” points.
NHSTEM	1	Number of half stems on the symmetry plane.
NINTER	Calculated	Number of interface points.
NMATR	Calculated	Number of material-data records in the binary solution file.
NORM	0	0: EZERO; 1: EZEROT; 2: HPHI; 3: ASCALE; 4: point to point.
NPBOUND	Calculated	Number of boundary points in the mesh.
NPEG	–1	Number of boundary segments for field calculations.
NPINP	Calculated	Total mesh points (NAIR+NFE+NINTER+NBND+NSPL).
NPONTS	Calculated	Number of unknown relaxation points during mesh optimization.
NREG	Calculated	Number of regions.
NRESIST	0	Number of segments with different surface resistance.
NRMSEG	1	Segment number for normalization when NORM = 2.
NSEG	Calculated	Number of boundary segments.
NSPL	Calculated	Number of user-defined fixed-potential points.
NSTEM	0	Number of stems and post couplers in a multicell tank.
NSTEP	0	Number of steps for a resonance search.
OMEGAM	0.001	Used in optimizing the mesh SOR parameter.
PI	π	The number π.
PLCELL	360	Phase length (degrees) per cell for a multicell problem.
POWER	Calculated	Power on conducting boundaries.
Q2I	Calculated	1/2Q passed from CFish to SFO.
RESIDA	Calculated	Residual for the vector potential solution.
RESIDR	1.0D–8	Residual resistance for superconductors (Ω).
RESIK	Calculated	Value of $\lvert D(k^2) \rvert /k^2$ for an iteration.
RFMU	1.0	Relative permeability for rf surface resistivity.
RHO	1.7241D–6	User-supplied material bulk resistivity (Ω-cm).
RHOC	Calculated	Calculated resistivity (Ω-cm) for normal conductors.
RHOR	1.7241D–6	Reference resistivity (Ω-cm) at temperature TEMPR.

Table III-3. Superfish variables. (continued)

Variable	Default value	Description
RHOXY	1.6	Over-relaxation factor during mesh optimization.
RMASS	–2 (H^+)	Rest mass energy (MeV) or particle-type code number.
RS	0	RF surface resistance (Ω).
RSTEM	1	Stem radius for stems along boundary segments.
SAREA	Calculated	Total surface area used for power calculations.
SLOSS	0.03	Coupling-slot power increase per percent coupling.
T	Calculated	In SFO, transit-time factor integral.
TC	9.2	Critical temperature for superconductors (K).
TEMPC	20	Operating temperature for normal conductors (C).
TEMPK	2	Operating temperature for superconductors (K).
TEMPR	20	Reference temperature for normal conductors (C).
TRIAVG	Calculated	Average area of the mesh triangles.
TRIMAX	Calculated	Area of the largest mesh triangle.
TRIMIN	Calculated	Area of the smallest (positive-area) mesh triangle.
VOLUME	Calculated	Cavity volume for cylindrically symmetric problems.
XDRI	0	X coordinate of the drive point.
XK0	Calculated	The wave number $k = \omega/c$.
XKSQ	Calculated	Square of the wave number $k^2 = (\omega/c)^2$.
XMAXG	Calculated	Upper X bound of the problem geometry.
XMING	Calculated	Lower X bound of the problem geometry.
XNORM1	0	Starting X coordinate for NORM = 4 normalization option.
XNORM2	0	Ending X coordinate for NORM = 4 normalization option.
XYAREA	Calculated	Total cross sectional area in the problem.
YDRI	0	Y coordinate of the drive point.
YMAXG	Calculated	Upper Y bound of the problem geometry.
YMING	Calculated	Lower Y bound of the problem geometry.
YNORM1	0	Starting Y coordinate for NORM = 4 normalization option.
YNORM2	0	Ending X coordinate for NORM = 4 normalization option.
ZCTR	0	Electric center of a cell (usually where field peaks).
ZLONG	Calculated	E_0 integration length, and cavity length if CLENGTH = 0.

2. Problem variables for Poisson and Pandira

This section describes the problem variables used by static-field programs Poisson and Pandira. The default value is the initial setting in Automesh, unless it is a calculated value or a required input parameter. Automesh initializes calculated values to zero. A few variables have an initial value FINDEF = –1.0D+99, which the codes use to tell whether the user has supplied a value.

Table III-4. Poisson variables.

Variable	Default Value	Description
ANGLE	0.0	Extent of the arc for interpolating the potential.
ANGLZ	0.0	Initial point on arc for interpolating the potential.
BDES	FINDEF	If not default value, adjust XJFACT so $\lvert B \rvert$ = BDES.
CLIGHT	Calculated	Speed of light ($2.99792458 \times 10^{10}$ cm/s).
CONV	1.0	Length conversion (number of cm/unit).
DIAGDLL	0	If 1, DLL functions write diagnostics to file DiagDLL.txt.
DX1	Calculated	X mesh spacing in the first region.
DXMIN	Calculated	Minimum X mesh interval.
DYMIN	Calculated	Minimum Y mesh interval.
ENERGY	Calculated	Problem geometry total stored energy.
EPS0	ε_0	Permittivity of free space (8.85418×10^{-12} F/m).
EPSILA	5.0D–7	Convergence parameter for air and interface points.
EPSILI	5.0D–7	Convergence parameter for iron points.
EPSO	1.0D–5	Convergence parameter during mesh optimization.
ETAAIR	Calculated	Rate of convergence for air points at cycle ICYCLE.
ETAFE	Calculated	Rate of convergence for iron points at cycle ICYCLE.
FIXEPS	9.0	Permittivity if MODE = –1 (XJFACT = 0).
FIXGAM	0.004	Reluctivity if MODE = –1 (XJFACT $\neq$ 0).
FMU0	μ_0	Permeability of free space ($4\pi \times 10^{-7}$ T-m/A).
ICAL	1	Selects formula in Automesh for computing mesh-point currents.
ICCC	1	Always equal to 1 for Poisson problems.
ICYCLE	Calculated	Iteration (or cycle) number.
ICYLIN	0	Coordinate system (0: rectangular; 1: cylindrical).
ICYSEN	1	If zero, do not print boundary integrals.
IENERGY	–1	Calculate stored energy if 1.
IHDL	100000	Number of cycles in Poisson between H·dl integrals.
IMAX	Calculated	KMAX+2.
IPERM	0 or 1	0 = problem has real currents; 1 = permanent magnets only.
IPIVOT	1	Controls pivoting in matrix inversion routines.
IPRFQ	0	Print frequency during Poisson iterations.
IRMAX	25	Cycles between optimization of the mesh SOR parameter.
ISKIP	1	Cycles between calculation of the reluctance in Poisson.
ITOT	Calculated	Number of mesh points, (KMAX+2)*(LMAX+2).
IVERG	10	Number of cycles between Poisson convergence tests.
KBZERO	1	Logical K coordinate where $\lvert B \rvert$ = BDES.
KMAX	Calculated	Number of horizontal logical mesh points.
KMIN	1	Lower K bound for computing the field and gradient.
KPROB	0	1 for Superfish problems, 0 for Poisson or Pandira.
KTOP	KMAX	Upper K bound for computing the field and gradient.
KTYPE	2	Symmetry indicator for harmonic analysis.
LAST35	Calculated	Code number of the program to update the binary solution file.
LBZERO	1	Logical L coordinate where $\lvert B \rvert$ = BDES.

Table III-4. Poisson variables. (continued)

Variable	Default Value	Description
LCYCLE	Calculated	Number of relaxation iterations during mesh optimization.
LMAX	Calculated	Number of vertical logical mesh points.
LMIN	1	Lower L bound for computing the field and gradient.
LTOP	1	Upper L bound for computing the field and gradient.
MAP	1	Number of poles for conformal transformation.
MAXCY	100000	Maximum number of cycles (default 20 in Pandira).
MAXPPR	Calculated	Maximum number of points per region.
MODE	–2, –1	Material property indicator.
NAIR	Calculated	Number of “air” points.
NAMAX	Calculated	Number of points for recalculating couplings.
NBND	Calculated	Number of Dirichlet boundary points.
NBSLF	0	Left-side boundary condition.
NBSLO	1	Lower boundary condition.
NBSRT	0	Right-side boundary condition.
NBSUP	0	Upper boundary condition.
NEGAT	Calculated	Zero-area triangles, indicates a problem generating the mesh.
NFE	Calculated	Number of “iron” points.
NGMAX	Calculated	Number of points for recalculating γ values.
NGSAM	Calculated	Number of points for recalculating γ if NM6 = NM1.
NINTER	Calculated	Number of interface points.
NMATR	Calculated	Number of material-data records in the binary solution file.
NOTE	1	Determines relaxation order for the mesh points.
NPBOUND	Calculated	Number of boundary points in the mesh.
NPEG	–1	Number of boundary segments for field calculations.
NPINP	Calculated	Total mesh points (NAIR+NFE+NINTER+NBND+NSPL).
NPONTS	Calculated	Number unknown points during mesh optimization.
NPTC	0	Number of arc points for interpolating the potential.
NREG	Calculated	Number of regions.
NSEG	Calculated	Number of boundary segments.
NSPL	Calculated	Number user-defined fixed-potential points.
NTERM	5	Number of coefficients in the harmonic analysis.
NWMAX	Calculated	Dimension of coupling arrays in Poisson.
OMEGAM	0.001	Used in optimizing the mesh SOR parameter.
OMEGAP	0.001	Used in optimizing the Poisson SOR parameter.
PI	π	The number π.
RATIO	Calculated	BZERO/XJFACT for the “air” solution.
RESIDA	Calculated	Residual of the solution for air points.
RESIDI	Calculated	Residual of the solution for iron points.
RHOAIR	1.9	Over-relaxation factor for air and interface points.
RHOFE	1.0	Over-relaxation factor for iron points.
RHOGAM	0.08	Under-relaxation factor for the reluctance in Poisson.
RHOPT1	1.9	If equal to RHOAIR, causes optimization of RHOAIR.

Table III-4. Poisson variables. (continued)

Variable	Default Value	Description
RHOXY	1.6	Over-relaxation factor during mesh optimization.
RINT	0.0	Radius of the arc for interpolating the potential.
RNORM	RINT	Aperture radius used in the harmonic analysis.
RZERO	1.0	Magnet bore radius for conformal transformations.
SNEGA	Calculated	Total negative current for the solution.
SNEGG	Calculated	Total negative current at mesh generation.
SPOSA	Calculated	Total positive current for the solution.
SPOSG	Calculated	Total positive current at mesh generation.
STACK	1.0	Stacking or fill factor for iron regions.
STOTA	Calculated	Total current in the problem for the solution.
STOTG	Calculated	Total current at mesh generation.
TNEGC	0.0	Total negative current after conformal transformation.
TPOSC	0.0	Total positive current after conformal transformation.
TRIAVG	Calculated	Average area of the mesh triangles.
TRIMAX	Calculated	Area of the largest mesh triangle.
TRIMIN	Calculated	Area of the smallest (positive-area) mesh triangle.
VOLUME	Calculated	Mesh-region volume for cylindrical geometries.
XAZERO	FINDEF	Physical X coordinate where A = 0 for harmonic analysis.
XBZERO	FINDEF	Physical X coordinate where $\lvert B \rvert$ = BDES.
XJFACT	1.0	Factor multiplying currents and current densities.
XJFEND	1.0	Ending value of XJFACT for XJSTEPS additional calculations.
XJSTEPS	0	Number of steps between XJFACT and XJFEND.
XJTOL	1.0D–4	Tolerance on XJFACT when setting $\lvert B \rvert$ = BDES.
XMAXF	0.0	Upper X bound for computing the field and gradient.
XMAXG	Calculated	Upper X bound of the problem geometry.
XMINF	0.0	Lower X bound for computing the field and gradient.
XMING	Calculated	Lower X bound of the problem geometry.
XORG	0.0	X coordinate of arc center for harmonic analysis.
XYAREA	Calculated	Total cross sectional area in the problem.
YAZERO	FINDEF	Physical Y coordinate where A = 0 for harmonic analysis.
YBZERO	FINDEF	Physical Y coordinate where $\lvert B \rvert$ = BDES.
YMAXF	0.0	Upper Y bound for computing the field and gradient.
YMAXG	Calculated	Upper Y bound of the problem geometry.
YMINF	0.0	Lower Y bound for computing the field and gradient.
YMING	Calculated	Lower Y bound of the problem geometry.
YORG	0.0	Y coordinate of arc center for harmonic analysis.

F. Configuring Poisson Superfish features: SF.INI

Poisson Superfish codes have some configurable options that you can control with settings in the initialization file SF.INI. The Setup program put a copy of SF.INI containing suggested settings in your LANL directory. If Setup found an existing copy of SF.INI in the destination directory from a previous installation, it saved your settings and added any new or missing keywords. You can regenerate file SF.INI, restoring all the default settings, by deleting or renaming the current file and then running program MK_SFINI.

All codes read the [Global] section first, then a section named after the code (e.g. [Automesh], [Poisson], etc.). The order of the sections in SF.INI does not matter. Though we use mixed case for readability, neither the SF.INI keywords nor their assigned values are case sensitive. Table III-5 is a list of the SF.INI configuration variables. Some variables are used only in one code and others are used in several codes. For example, only WSFplot uses the BoundaryColor parameter. Autofish, Fish, CFish, Pandira, and all the tuning codes use the TAPE40 parameter.

Table III-5. Alphabetical list of SF.INI settings.

Keyword	Description
AlwaysBrowse	Always browse for a binary solution file.
ArcAndLineSteps	Default number of steps in SF7 for interpolation on lines and arcs.
ArcRadius	Default arc radius in SF7 for interpolation on arcs.
ArrowKeyStepSize	Cursor step size in WSFplot as a percentage of the screen width.
ArrowLineWidth	Width in pixels of field arrows on WSFplot display screen.
AssumedBeta	Use this Beta in SFO if calculated value is unphysical.
AxisLineWidth	Width in pixels of boundary axes on plotting code display screens.
BetaDigits	Utility code Beta, maximum significant digits (3 to 15) for beta.
BetaTolerance	Amount over 1.0 before SFO warns of unphysical Beta.
BoundaryAxes	Include numbered axes at all four edges of the WSFplot display.
BoundaryColor	Color of boundary segments in WSFplot.
BoundaryLineWidth	Width in pixels of region boundaries on WSFplot display screen.
BrowsingForFiles	Force use of the standard open dialog if no filename specified.
CFishCirclePlot	Create Quikplot file CircleFit.QKP showing fitted resonance circles.
CircleLineWidth	Width in pixels of field circles on WSFplot display screen.
CombineTitleLines	Sets whether to use one line or two lines on the WSFplot title.
ComplexFields	SF7 to write real and imaginary CFish fields.
ComputeStoredEnergy	Sets whether Poisson and Pandira calculate stored energy.
ContourLineWidth	Width in pixels of field contour lines in WSFplot display screen.
ContourMinimum	Lowest contour value as percent of range in WSFplot.
ContourMaximum	Highest contour value as percent of range in WSFplot.
CreateOutputTextFile	Sets whether WSFplot writes output file OUTWSF.TXT.
CurveLineWidth	Width in pixels of curves on plotting code display screens.
CutoffTerm	Limiting the number of terms in the field interpolator.
DecimalPlaces	Sets precision of field data in SF7 output file OUTSF7.TXT..
DeleteNoRingFiles	If true, CCLfish and CDTfish delete no-ring Superfish files.

Table III-5 SF.INI Settings (continued).

Keyword	Description
EGUNfileMeshSize	Default mesh size in SF7 for the grid when creating EGUN input file.
Electric_RZ_PowerLaw	Sets contour spacing for electric problems in cylindrical coordinates.
Electric_XY_PowerLaw	Sets contour spacing for electric problems in Cartesian coordinates.
EndingValue	Utility code Beta, ending value for a table.
EnergyDigits	Utility code Beta, maximum significant digits (3 to 15) for energy.
EquatorRingLimit	Equator tuning ring thickness limit in CCLfish and CDTfish.
FieldArrowColor	Color of field arrows in WSFplot.
FieldArrows	Include field arrows in WSFplot initial display.
FieldCircleColor	Color of field circles in WSFplot.
FieldCircles	Include field circles in WSFplot initial display.
FieldContourColor	Color of field contours in WSFplot.
FileIDFooter	Show Automesh input filename at bottom right of WSFplot display .
FishScan	Enable or disable writing file FishScan.TBL in Fish.
Force1MVperMeter	If true, SF7 uses E_0 = 1 MV/m when generating data for Parmela.
GammaDigits	Utility code Beta, maximum significant digits (3 to 15) for gamma.
GridSpacing	Arrow and circle spacing in WSFplot as a percentage of screen width.
GridXsteps	Default number of steps in X direction for grids in SF7.
GridYsteps	Default number of steps in Y direction for grids in SF7.
HardcopyDriver	Specifies the default hardcopy type at startup of plotting codes.
ImaginaryFieldColor	Color of imaginary field contours for CFish problems in WSFplot.
IncludeDirichletSegs	Sets whether SFO omits Dirichlet segments from power calculation.
IndependentVariable	Utility code Beta, choices: Energy, Beta, Gamma.
InternalArrays	Minimum array length needed if a code exhausts resources.
InterpolateNodeA	Tells field interpolator how to report vector potential A on nodes.
InterpolateNodeV	Tells field interpolator how to report voltage V on nodes.
InterpolateNodeH	Tells field interpolator how to report rf magnetic field H on nodes.
MakeParmelaFile	SF7 creates input file for Parmela (cylindrically symmetric problems).
MakePlotFile	SF7 creates Tablplot file when interpolating on lines, arcs, and curves.
Magnetic_RZ_PowerLaw	Sets contour spacing for magnet problems in cylindrical coordinates.
Magnetic_XY_PowerLaw	Sets contour spacing for magnet problems in Cartesian coordinates.
MassUnit	Utility code Beta, MeV (rest-mass energy) or amu (atomic mass units).
MaxColumns	Maximum number of Tablplot data columns.
MaxDataSets	Maximum number of data sets in Quikplot (up to 10).
MaxLines	Maximum length of data table in Tablplot.
MaxMemAvailable	Limit in KB of memory available to certain programs.
MaxPoints	Maximum number points per data set in Quikplot.
MeshLineWidth	Width in pixels of mesh triangles on WSFplot display screen.
MeshTriangles	Set whether to include the mesh triangles in initial WSFplot display.
MeshTriangleColor	Color of mesh triangles in WSFplot.
Normalization	Sets preferred normalization method in tuning codes.
NumberOfContours	Number of contours displays in WSFplot.
NumberOfValues	Utility code Beta, number of evenly spaced values for a table.
OtherMass	Utility code Beta, particle mass (in MeV unless MassUnit = amu).
ParmelaFields	Write Poisson or Pandira field data for program Parmela.
ParmelaWarning	Disable certain warning messages in Poisson or Pandira.
ParmilaData	Write transit-time data for Parmila in SFO.

Table III-5 SF.INI Settings (continued).

Keyword	Description
PlotOrigin	Location of X_{Min},Y_{Min} on the display window.
PlotTitle	Include the title at the top of the WSFplot initial display.
PrintBGammaTables	Include the internal permeability tables in output summaries.
RF_RZ_PowerLaw	Sets contour spacing for rf problems in cylindrical coordinates.
RF_XY_PowerLaw	Sets contour spacing for rf problems in Cartesian coordinates.
RootNameLength	Number of characters of original filename used for SF7 output files.
SaveTAPE35	Sets whether tuning programs save the binary solution files.
SaveTAPE36	Sets whether Automesh keeps file TAPE36.
ScanFormatComplex	Sets whether initial plot is complex plane for CFish frequency scans.
SF7_Table	Sets whether to include fitting data with SF7 results.
ShowWarning	Selects or deselects pop-up warning messages.
StartingValue	Utility code Beta, starting value for a table.
SlaterTerm	Sets whether WSFplot and SF7 computes Slater perturbation data.
StoreTempDataInRAM	Enable or disable storing scratch data in memory.
TAPE35Extension	Filename extension for binary solution files.
TAPE40	Specifies available drives for scratch files.
ThirdNN_Electric_RZ	Third nearest neighbor preference for RZ electrostatic problems.
ThirdNN_Electric_XY	Third nearest neighbor preference for XY electrostatic problems.
ThirdNN_Magnetic_RZ	Third nearest neighbor preference for RZ magnetostatic problems.
ThirdNN_Magnetic_XY	Third nearest neighbor preference for XY magnetostatic problems.
ThirdNN_RF_XY	Third nearest neighbor preference for XY rf field problems.
TitleAndAxisColor	Color of plot title and the boundary axes in WSFplot.
WallRingLimit	Wall tuning ring thickness limit in CCLfish and CDTfish.
WriteFieldContours	Sets whether WSFplot writes contour lines to OUTWSF.TXT.
XaxisDirection	X axis orientation in WSFplot. Choices are Horizontal or Vertical.
X1percent	Default X1 coordinate in SF7 as a percentage of the X range.
X2percent	Default X2 coordinate in SF7 as a percentage of the X range.
Y1percent	Default Y1 coordinate in SF7 as a percentage of the Y range.
Y2percent	Default Y2 coordinate in SF7 as a percentage of the Y range.
Zcenter	CDTfish transit-time factors at the geometric center.

One section of SF.INI has the heading [Global] and contains entries that all codes will use unless another entry appears in a code's own section. There can be only one [Global] section and one section for each program. If there are duplicate section headings, only the first one is used. If duplicate configuration lines appear within the same section, only the last one has any effect. You can have multiple copies of SF.INI with different configurations. A program first looks for SF.INI in the current directory. Next, it checks the environment variable SFINI for the directory containing SF.INI (or for the complete path and filename if you rename the file). Next, it looks in the directory containing the executable file, and finally it looks in the installation directory (as defined by environment variable SFDir). The first file found in this search is the one the program will use.

Each program writes the complete path of the SF.INI file it found in the output text file (OUTxxx.TXT, or the .LOG file for tuning programs). Codes that fail to find file SF.INI write a line saying so. If a code cannot open the SF.INI file it found after a few tries, then the code writes a warning in the output text file and alerts the user with a pop-up warning message. The codes make several attempts to read the file to avoid a sharing violation when two programs seek access to the file simultaneously.

1. InternalArrays setting

Automesh determines the number of regions and fixed boundary points needed from the input file. It also estimates the length of internal arrays necessary to store information about interpolated points along boundaries. The required length increases with the number of regions, the number of points per region, the number of line regions, and the fineness of the mesh. For some unusual geometries, a problem may exhaust these resources. In that case, the code will ask you to InternalArrays to a particular value. Otherwise, you should never need to change this settings.

2. Saving file TAPE36

Automesh uses file TAPE36 to store logical path data along boundary segments while processing the geometry in the input file. No other code uses file TAPE36 and much of the data also appears in file OUTAUT.TXT. Automesh erases file TAPE36 after it finishes generating the mesh unless the setting SaveTAPE36 = Yes appears in the appropriate section of file SF.INI. For the following settings, Autofish and the tuning codes will erase the TAPE36 file, but Automesh will save it:

```
[Global]
SaveTAPE36 = No

[Automesh]
SaveTAPE36 = Yes
```

3. Drives for scratch files and storing scratch data in memory

TAPE40 indicates which drives to use to store a scratch file if there is insufficient RAM available for this purpose. While solving the tridiagonal matrix problem, programs Autofish, CCLfish, CDTfish, DTLfish, ELLfish, MDTfish, RFQfish, SCCfish, Fish, CFish, and Pandira store a data matrix for each row, which they later read back in reverse order. These programs use available memory for these temporary data arrays if possible, thus reducing considerably the computation time for some problems. When a scratch file is needed, the code uses unique filenames, so you can run the same code simultaneously on more than one problem. The only restriction is that you run each problem from a separate directory.

If there is insufficient memory, the program creates a scratch file on the hard drive with the most free space. If it cannot find enough free disk space, the program stops with a message telling you how much extra disk space it needs. By default, Fish and Pandira search only drive C. You can change or expand the search with the TAPE40 configuration parameter in file SF.INI. For example, to force the codes to store the scratch file on drives D or F insert following line in the [Global] section of SF.INI:

```
TAPE40 = DF
```

If some of your drives are network drives, be sure that you have write privileges on a drive before adding it your list of available drives. The codes may crash if they try to write to a read-only network drive.

Setting StoreTempDataInRAM = No in SF.INI disables storing temporary data in memory and always uses the TAPE40 scratch file. One reason for doing this would be to prevent disk thrashing. The Poisson Superfish codes see the extra memory of Windows swap file or paging file as available RAM and so will store scratch data "in memory" instead of on disk. However, if the data actually goes to the swap file or paging file, then the codes might run very slowly. It is far more efficient for the codes to use their own scratch file. The following line in the [Global] section of SF.INI disables storing temporary data in memory and forces the code to create a TAPE40 file:

```
StoreTempDataInRAM = No
```

The default setting for this parameter is Yes, that is, to use free memory for scratch data if it is available. Another option for controlling how the codes store temporary data is the MaxMemAvailable setting. The StoreTempDataInRAM setting takes precedence over the MaxMemAvailable setting.

4. Limiting memory available to certain programs

The MaxMemAvailable setting in SF.INI limits the maximum amount of memory (in kilobytes) available to programs that allocate memory either for temporary data or for plotting arrays. This setting can make more memory available for other applications. Poisson Superfish codes that allocate memory for temporary data storage include Autofish, CCLfish, CDTfish, DTLfish, ELLfish, MDTfish, RFQfish, SCCfish, Fish, CFish, and Pandira. These codes ignore the MaxMemAvailable setting if you have also set StoreTempDataInRAM = No. The codes will store the temporary data on disk if the MaxMemAvailable setting is smaller than the memory needed for the temporary data.

The MaxMemAvailable setting can fix problems with the Windows swap file or paging file. See the discussion in the previous section about the StoreTempDataInRAM setting. If you set StoreTempDataInRAM = No, which always forces temporary data to the disk, then even small problems that could have used only RAM will also run slower. The solution is to set the MaxMemAvailable to a value that allows storage in memory for small problems, but forces the code to create its own scratch file for large problems. We recommend setting MaxMemAvailable to about 200M less than your machine's installed memory. For example, if your machine has 2G of RAM, use the following setting:

```
[Global]
MaxMemAvailable = 1800000
```

Using these entries in the [Global] section of SF.INI sets the default value for all programs. Any individual setting for a particular code would take precedence over the [Global] setting.

5. Forcing programs to always browse for an input file

Programs that reads or write the solution file also update a text file named TAPE35.INF in the directory containing the solution file. The INF file stores the names of recently used solution files. Later programs to run use this information to automatically open a solution file. If you wish a program to always browse for a solution file to open, set variable AlwaysBrowse = Yes in the appropriate section of SF.INI.

6. Setting the binary solution file extension

In the unlikely event that some other application on your computer uses the T35 extension, you can define a different extension for Poisson Superfish solution files. Variable TAPE35Extension in the [Global] section of SF.INI is the three-character extension for binary solution files. You can choose any string of up to three characters for this extension. Of course, you should avoid choosing one already used by Poisson Superfish or one that is commonly used by other applications.

7. Turning on and off selected warning messages

Poisson Superfish codes include many numbered error messages and warning messages. An error message describes a serious problem that prevents the code from continuing. A warning message is not serious enough to stop the program, but it alerts the user to conditions that may be unanticipated or that may cause a problem later. Table III-6 lists all the error messages and whether their pop-up messages are initially enabled or not. Warning messages always appear in the code's output text file, but the user can selectively choose whether a pop-up dialog box appears. To enable or disable pop-up warning messages use variable ShowWarning in file SF.INI. To make settings apply to all codes, insert ShowWarning lines in the [Global] section of SF.INI. To make the settings apply to a particular program insert the lines into the section for that program (for example, [Poisson]).

You can use multiple ShowWarning lines and each line can include multiple settings. To enable a pop-up warning include the warning number on a ShowWarning line. To disable a warning include the warning number with a minus sign on a ShowWarning line. You can enable or disable all warnings using keywords On or Off. You also can combine keyword On or Off with selected warning numbers either on the same line or on multiple lines. For example, to enable only warnings 143 and 144 in Automesh, use the following lines:

```
[Automesh]
ShowWarning = Off, 143, 144
```

To enable all warning messages in all programs <u>except</u> for warnings 143 and 144, use the following lines:

```
[Global]
ShowWarning = On, –143, –144
```

Warnings 201 through 209 will always produce a pop-up warning message if a code fails to open the SF.INI file that it found according to the usual search order. You cannot disable the warning because the setting to disable appears in the file the code could not read. Because failing to read SF.INI could affect code results, the warning message may be important. This type warning should be rare even when running multiple Poisson Superfish jobs simultaneously because the codes make several attempts to open the file to give other programs a chance to close it first.

Table III-6. Poisson Superfish warning messages.

Warning	Initially	Description
106	On	The Automesh input file contains no title lines.
107	On	The Automesh input file contains more than 10 title lines.
108	On	The supplied value of KTYPE is invalid.
109	On	The permeability code MODE is not valid.
110	Off	The setting MODE = −2 is not allowed for electrostatic problems.
111	On	NDRIVE is an obsolete REG namelist variable.
112	On	IREG is an obsolete REG namelist variable.
113	On	NREG is an obsolete REG namelist variable.
114	On	NPEG is an obsolete REG namelist variable.
115	On	NCELL is an obsolete REG namelist variable.
116	On	NSTEM is an obsolete REG namelist variable.
117	On	NRESIST is an obsolete REG namelist variable.
118	On	XMIN is an obsolete REG namelist variable.
119	On	XMAX is an obsolete REG namelist variable.
120	On	YMIN is an obsolete REG namelist variable.
121	On	YMAX is an obsolete REG namelist variable.
122	On	The region scale factor RX cannot be zero or negative.
123	On	The region scale factor RY cannot be zero or negative.
124	On	ITRI ≠ 2 for cylindrical electrostatic problems may reduce accuracy.
125	On	NEW is an obsolete PO namelist variable.
126	On	Ignoring invalid supplied value of AOVRB.
127	On	Ignoring a misplaced KREG entry that has no matching XREG.
128	On	Ignoring a misplaced XREG entry.
129	On	Ignoring a misplaced KREG entry beyond last defined XREG.
130	On	There are fewer nonzero KREG elements than XREG elements.
131	On	There are fewer nonzero LREG elements than YREG elements.
132	On	Ignoring a misplaced LREG entry that has no matching YREG.
133	On	Ignoring a misplaced YREG entry.
134	On	Ignoring a misplaced LREG entry beyond last defined YREG.
135	Off	An X line region crossing a hyperbolic segment may cause problems.
136	Off	A Y line region crossing a hyperbolic segment may cause problems.
137	Off	Automesh cannot add all of an X line region.
138	Off	Automesh cannot add all of a Y line region.
139	Off	Automesh found only one end point fo an X line region.
140	Off	Automesh found only one end point fo a Y line region.
141	Off	Extrapolated permeability at B = 0 is negative for an MT namelist.
142	Off	An intermediate point is being moved to a nearby mesh point.
143	On	Superfish REG variable supplied for a Poisson problem.
144	On	Poisson REG variable supplied for a Superfish problem.
145	On	An interpolated boundary point is being reassigned.
146	On	An intermediate point is being removed to avoid a bad-triangle error.
147	On	Modified coordinate file contains too many nodes.

Table III-6. Poisson Superfish warning messages.(continued)

Warning	Initially	Description
148	On	Modified coordinate file contains too many node displacements.
149	On	First and last points of a region have same K,L, but different X,Y.
150	Off	End points for a boundary segment could not be found.
151	On	Drive point is on a Dirichlet boundary where there is no H field.
152	On	Automesh reports mesh has overlapping and/or zero-area triangles.
153	On	Automesh reports mesh has near zero-area triangles.
154	On	Automesh reports the drive point is outside the cavity.
155	Off	Only one cycle allowed in Fish for multiple fixed-potentials.
156	Off	ParmelaFields option requires cylindrical symmetry in Pandira.
157	Off	ParmelaFields option requires cylindrical symmetry in Poisson.
158	Off	Parmela output needs physical coordinate grid in Pandira and Poisson.
159	Off	No Parmela output for logical coordinates in Pandira and Poisson.
160	Off	The interval between harmonic analysis terms is invalid.
161	Off	The median-plane symmetry indicator for harmonic analysis is invalid.
162	On	XMINF is outside the problem boundary in Poisson or Pandira.
163	On	XMAXF is outside the problem boundary in Poisson or Pandira.
164	On	YMINF is outside the problem boundary in Poisson or Pandira.
165	On	YMAXF is outside the problem boundary in Poisson or Pandira.
166	On	Not enough memory for storing Parmela fields with requested grid size.
167	On	KMIN is outside the logical coordinate range in Poisson or Pandira.
168	On	KTOP is outside the logical coordinate range in Poisson or Pandira.
169	On	LMIN is outside the logical coordinate range in Poisson or Pandira.
170	On	LTOP is outside the logical coordinate range in Poisson or Pandira.
171	On	The maximum number of terms for the harmonic analysis is 14.
172	Off	Not enough memory to calculate integrals in SF7.
174	Off	The H·dL integral in SF7 includes a material other than air.
175	Off	Parmela output file requires cylindrical symmetry in SF7.
176	Off	SFO cannot find segment selected for special surface resistance.
177	Off	Segment for coupling slot option in SFO not selected for rf power.
178	Off	Segment for coupling slot option in SFO does not exist.
179	Off	SFO is ignoring a segment with undefined logical coordinates.
180	Off	Stem location in SFO is outside the problem geometry.
181	On	Supplied particle velocity has unphysical value in SFO.
182	On	Calculated particle velocity has unphysical value in SFO.
183	On	Particle velocity has unphysical value for multicell problem in SFO.
184	On	Particle velocity for a cell has unphysical value in SFO.
185	On	Cannot calculate power along R = 0 in SFO.
186	On	WSFplot reports Superfish problem has no drive point.
187	On	WSFplot reports mesh has overlapping triangles or zero-area triangles.
188	On	Fish advises to run CFish to see affects of complex field initialization.
189	On	Fish or Pandira cannot use drive specified by TAPE40 in SF.INI.
190	On	RHOGAM may be too large for vanadium permendur.
191	Off	RHOGAM may need optimizing for user-defined permeability tables.
192	On	The Poisson SOR procedure is oscillating and not likely to converge.
193	On	Pandira does not compute stored energy in permanent magnet materials.
194	On	Matrix inversion failure in field interpolator.

Table III-6. Poisson Superfish warning messages.(continued)

Warning	Initially	Description
195	On	Matrix inversion failure in Cfish root finder.
201	On	Automesh, Autofish (or originating code) failed to open existing SF.INI.
202	On	Fish, CFish (or originating code) failed to open existing file SF.INI.
203	On	Poisson failed to open existing file SF.INI.
204	On	Pandira failed to open existing file SF.INI.
205	On	SFO (or originating code) failed to open existing file SF.INI.
206	On	SF7 failed to open existing file SF.INI.
207	On	Force failed to open existing file SF.INI.
208	On	WSFplot failed to open existing file SF.INI.
209	On	A tuning program failed to open existing file SF.INI.
210	On	Fish, CFish, or Pandira can use only fixed disks for temporary storage.

8. Setting the hardcopy graphics type

Program WSFplot, Quikplot, and Tablplot can produce a number of different hardcopy graphics types. The WSFplot chapter discusses the available graphic drivers. When the codes start, the Windows Print Manager is the default hardcopy driver. If you prefer one of the graphic file types, set HardcopyDriver to the filename extension listed in Table III-7 of the preferred output type. For example, for Windows bitmap graphic files in WSFplot, use the following setting:

```
[WSFplot]
HardcopyDriver = BMP
```

Whether you set a preference in SF.INI or not, you can always select a hardcopy driver on the program menu.

Table III-7. Filename extensions for hardcopy graphics types.

Extension	Description
(none)	Windows Print Manager (default)
BMP	Windows bitmap format
CGM	Computer Graphics Metafile
DXF	AutoCAD Drawing Exchange format
PS	Adobe PostScript
EPS	Encapsulated PostScript
HGL	Hewlett Packard Graphics Language original format
PLT	Hewlett Packard Graphics Language HP-GL/2 format
PIC	Lotus 1-2-3 Graphics format
PCX	ZSoft PC Paintbrush format
PNG	Portable Network Graphics
SVG	Scalable Vector Graphics
WMF	Windows Metafile
EMF	Enhanced Windows Metafile

9. Setting initial display preferences for WSFplot

The configuration variables in Table III-8. define the initial screen settings when first starting WSFplot. After starting WSFplot, many of the settings may be changed from the program menu. Run program WSFColor to see colors associated with the color numbers in the table.

Table III-8. WSFplot display parameters in SF.INI.

Keyword	Default	Description
PlotOrigin	LowerLeft	Location of X_{Min},Y_{Min} on the display window. Choices are LowerLeft, LowerRight, UpperLeft, and UpperRight
XaxisDirection	Horizontal	Choices are Horizontal or Vertical.
ArrowKeyStepSize	0.25	When arrow keys are pressed, cursor step size expressed as a percentage of the screen width.
GridSpacing	3.0	Field arrow and circle spacing expressed as a percentage of the screen width.
MeshTriangles	On	Include the mesh triangles when On. Affects only plots containing solution arrays.
ContourMinimum	0	Lowest contour value as percent of range.
ContourMaximum	100	Highest contour value as percent of range.
NumberOfContours	30	Zero value turns off contours.
FieldArrows	Off	Include field arrows with length proportional to field strength.
FieldCircles	Off	Include field circles with diameter proportional to field strength (for rf problems only).
PlotTitle	On	Include a title at the top of the display window.
FileIDFooter	On	Include line at bottom right containing the complete file specification and time stamp of the Automesh input file.
CombineTitleLines	No	The plot title uses either line 1 or lines 1 and 2 of the problem title.
BoundaryAxes	On	Include numbered axes at all four edges of the display window.
TitleAndAxisColor	150 (blue)	Color of the plot title and the boundary axes.
BoundaryColor	150 (blue)	Color of the boundary segments.
MeshTriangleColor	240 (lt. gray)	Color of the mesh triangles.
FieldContourColor	200 (magenta)	Color of the field contours (real part for CFish problems).
ImaginaryFieldColor	100 (green)	Color of the imaginary field contours for CFish problems.
FieldArrowColor	16 (red)	Color of the arrows showing field magnitude and direction.
FieldCircleColor	16 (red)	Color of the circles showing field perpendicular to the page.
ContourLineWidth	1	Width in pixels (1 to 10) of field contour lines.
ArrowLineWidth	1	Width in pixels (1 to 10) of field arrows.
CircleLineWidth	1	Width in pixels (1 to 10) of field circles.
BoundaryLineWidth	1	Width in pixels (1 to 10) of region boundaries.
MeshLineWidth	1	Width in pixels (1 to 10) of mesh triangles.
AxisLineWidth	1	Width in pixels (1 to 10) of boundary axes.

10. Setting the interval between contour lines in WSFplot

Program WSFplot displays up to a maximum of $2\times N_C + 1$ contour lines, where N_C is the number of contours of either algebraic sign. The actual number of contours plotted depends upon the values of G_{Max} and G_{Min}, the minimum and maximum values of the solution array, where G refers to either A_z, V, or H_z for problems in rectangular coordinates or to rA_ϕ, V, or rH_ϕ for problems in cylindrical coordinates. If either $G_{Max} = 0$ or $G_{Min} = 0$, or if they have the same sign, then the code will plot exactly N_C contours. The G = 0 contour appears only if G_{Min} is negative and G_{Max} is positive. The code plots the maximum number of contours ($2\times N_C + 1$) when $G_{Min} = -G_{Max}$. Other values for G_{Max} and G_{Min} will result in a varying numbers of contours. The configuration keywords listed in Table III-9 control the intervals between contour-line values $\pm G(n)$. If G_{Max} and G_{Min} have opposite signs or one of them is zero, then the contour values are given by the equations

$$\pm G(n) = \pm \Delta G n^p\,,\ \ \Delta G = \frac{\mathrm{MAX}(\mathrm{ABS}(G_{Max}),\mathrm{ABS}(G_{Min}))}{(N_C+1)^p}$$

where n is a contour counter that ranges from 0 to N_C. Exponent p is the power-law setting, and ΔG is the base contour interval. In this case, the first nonzero contour plotted has value $+\Delta G$ or $-\Delta G$. If G_{Max} and G_{Min} have the same sign and neither is zero, the code computes the contour values as follows:

$$\pm G(n) = \pm\left(G_0 + \Delta G n^p\right),\ \ \Delta G = \frac{G_{Max} - G_{Min}}{(N_C+1)^p}\ .$$

where G_0 is the minimum of $|G_{Max}|$ and $|G_{Min}|$. The allowed range for the power p is from 0.1 to 5.0. For most problem types, WSFplot uses p = 1, which results in equal intervals of G between adjacent contour lines.

For Superfish problems with cylindrical symmetry (KPROB = 1, ICYLIN = 1), the default is to use p = 2, which places more contour lines near $rH_\phi = 0$. This setting results in more contour lines in the vicinity of the beam axis where H_ϕ has an approximately linear variation with radius r. In multicell rf cavity problems, the concentration of contour lines near $rH_\phi = 0$ facilitates identification of the resonant mode. If equal intervals of rH_ϕ, were used instead, there might be no contour lines in the region where the field reverses. In this case one could not tell the difference between (for example) the zero mode and π mode of a two-cell structure.

Table III-9. Power-law settings for WSFplot contour lines.

Keyword	Problem type	Default
RF_XY_PowerLaw	Radio-frequency problems in Cartesian coordinates.	1
RF_RZ_PowerLaw	Radio-frequency problems in cylindrical coordinates.	2
Magnetic_XY_PowerLaw	Magnetic field problems in Cartesian coordinates.	1
Magnetic_RZ_PowerLaw	Magnetic field problems in cylindrical coordinates.	1
Electric_XY_PowerLaw	Electrostatic problems in Cartesian coordinates.	1
Electric_RZ_PowerLaw	Electrostatic problems in cylindrical coordinates.	1

You can set your preferences for contour-line intervals using the configuration variables in Table III-9. Before running WSFplot, edit the appropriate lines in the [WSFplot] section of SF.INI. The example file FTEST shows two interval settings of the contour lines for a ferrite-tuned rf cavity. Suppose you want equal intervals of A_z, V, and H_z for problems in rectangular coordinates, but quadratic intervals of rA_ϕ, V, and rH_ϕ for problems in cylindrical coordinates. For this case, use the following settings:

```
[WSFplot]
RF_XY_PowerLaw = 1
Magnetic_XY_PowerLaw = 1
Electric_XY_PowerLaw = 1
RF_RZ_PowerLaw = 2
Magnetic_RZ_PowerLaw = 2
Electric_RZ_PowerLaw = 2
```

11. Saving the WSFplot field contours

Program WSFplot creates output file OUTWSF.TXT if CreateOutputTextFile = Yes in the [WSFplot] section of file SF.INI. This file contains a list of the equipotential values plotted by WSFplot. The contours are lines of constant A_z for rectangular magnet problems, rA_ϕ for cylindrically symmetric magnet problems, V for electrostatic problems, H_z for rectangular rf problems, and rH_ϕ for cylindrically symmetric rf problems. If SF.INI configuration variable WriteFieldContours = Yes, then OUTWSF.TXT also will include the coordinates of the plotted contour lines.

12. Choosing the plotting code preference file

Programs WSFplot, Quikplot, and Tablplot write a *preference file* containing saved screen settings to be used the next time the program starts. The name and location of the file depends upon the SF.INI variable SaveSettingsTo. The choices are:

SaveSettingsTo = Local Directory
SaveSettingsTo = Individual File
SaveSettingsTo = *filename*
SaveSettingsTo = *directory*

where *filename* is the name of an actual file (including path) to be created and *directory* is the name of an existing directory. The default setting is "Local Directory," which means that the code will save file WSFPRF.TXT, QplotPRF.TXT, or TplotPRF.TXT in the current directory (usually where the input file resides). Another SF.INI variable, UseSavedSettings, determines whether the code attempts to use the settings in the preference file.

13. Configuring program Quikplot and Tablplot

Programs Quikplot and Tablplot use settings in Table III-10. The settings are limited only by available memory.

Keywords AxisLineWidth and CurveLineWidth set the initial line widths used for Quikplot and Tablplot displays. For details about the preference files refer to paragraph 11 and for details about the hardcopy driver refer to paragraph 8. in this section.

Table III-10. Quikplot and Tablplot parameters in SF.INI.

Keyword	Default	Description
MaxDataSets	10	Maximum number of data sets in Quikplot.
MaxPoints	10,000	Maximum number points per data set in Quikplot.
MaxColumns	16	Maximum number of Tablplot data columns.
MaxLines	10,000	Maximum length of data table in Tablplot.
AlwaysBrowse	Yes	Used only if no valid file is found on the command line. If Yes, open the File select dialog on startup. If No, open the last file if it exists.
StartNewLog	No	If Yes, create new a log file (Quikplot.log or Tablplot.log) on startup, otherwise append to existing log.
AxisLineWidth	1	Width in pixels (1 to 10) of boundary axes.
CurveLineWidth	1	Width in pixels (1 to 10) of plotted curves.
HardcopyDriver	(none)	Set default hardcopy type at startup (see paragraph 8 above).
SaveSettingsTo	Local Directory	See Choosing the plotting code preference file for details.
UseSavedSettings	Yes	Sets whether to read the preference file, if found.

14. Setting the filename length for SF7 output files

SF7 creates plot files for Tablplot and output files for some other codes by appending a sequence number to a portion of the name of the Poisson Superfish solution file. The default setting is 6 characters. For example, if the solution filename is MyDataFile.T35, the first few plot files would be MyData01.tbl MyData02.tbl and MyDatabut03.tbl. You can instruct SF7 to use more characters (up to the full length of the name) by setting SF.INI variable RootNameLength. Filenames, including the path, are limited to 256 characters. The minimum allowed for RootNameLength is 2 characters. The maximum is 246.

Suppose that you have in the same directory different solution files whose names are identical through the first 10 characters. To use up to 12 characters for the root name of the plot files, use the following setting:

```
[SF7]
RootNameLength = 12
```

15. Setting precision of field data printed in output text files

The precision of the printed fields in some output text files depends on DecimalPlaces. The allowed range for DecimalPlaces is 3 to 12. The default setting is 6. The affected files are the SF7 output file OUTSF7.TXT, and the Poisson and Pandira output files OUTPOI.TXT and OUTPAN.TXT. The setting affects all field components and their derivatives. The X,Y coordinates and the fitting data, if selected in SF7, are not affected. To select 8 decimal places of precision in the SF7 output, use the following setting:

```
[SF7]
DecimalPlaces = 8
```

16. Including field-interpolator fitting data with SF7 results

The ExpandedTable configuration variable sets whether SF7 writes fitting parameters from field interpolator to file OUTSF7.TXT in addition to the usual output. Using this feature you can check the fits obtained by the field interpolator by looking at the chi-squared goodness of fit parameter. To include additional fitting data in file OUTSF7.TXT use the following setting:

```
[SF7]
ExpandedTable = Yes
```

The expanded table includes the new column headings defined in Table III-11. The Chi^2 parameter includes no estimate of uncertainties. It is the sum of the absolute squares of differences between the data in the solution array and the fit at each included mesh point divided by the number of degrees of freedom. The number of degrees of freedom is the number of data points in the fit minus the number of fitted coefficients. The chi-squared is normalized to the square of the average of the absolute values of solution array (H, A, or V) for the points used in the fit. The reference point is the point in the mesh around which the code assembles first and second nearest neighbors for the fit. Near boundaries, the interpolator may add selected third nearest neighbors until the total number of neighboring points reaches 18.

Table III-11. SF7 expanded data-table columns.

Column	Description
D	T (for true) means the point is on or near a Dirichlet boundary.
N	T means the point is on or near a Neumann boundary.
Chi^2	Chi-squared per degree of freedom.
F#	The function number used to fit the data.
K,L	Logical coordinates of the reference point.
Num	Number of data points in the fit.
P_i	Fitted coefficients P_1 through P_8.

17. Configuring SF7 to write real and imaginary CFish fields

The ComplexFields configuration variable tells SF7 whether to write both real and imaginary components of H_z, E_x, and E_y (or H_ϕ, E_r, and E_z) for Superfish problems with complex fields. This setting affects only problems solved with CFish. The default setting for ComplexFields is No. If you need this detailed information, insert the following line in the [SF7] section of SF.INI before running SF7:

```
[SF7]
ComplexFields = Yes
```

18. Setting default values for the SF7 dialog window

Table III-12 lists keywords that define default values for most of the entries in the SF7 dialog window. The variables are in the [SF7] section of SF.INI. Variables X1percent, Y1percent, X2percent, and Y2percent set the initial values of the interpolation end points for lines and arcs or two corners of the grid. The range of 0 to 100 for these variables corresponds to the range of the problem boundaries X_{min} to X_{max} and Y_{min} to

Y_{max}. For example, if X_{min} = 10 and X_{max} =20, the setting X1percent = 20 would correspond to X1 = 12. If you usually run SF7 to interpolate fields along the axis of an rf cavity, the setting Y1 = 0 will be helpful.

Variable ArcAndLineSteps defines the number of steps between end points of the line or arc. Variables GridXsteps and GridYsteps define the number of steps between edges of the grid. Notice that there is one more interpolated point than steps.

Table III-12. SF7 parameters in SF.INI.

Keyword	Default	Description
X1percent	0	Default X1 coordinate as a percentage of the X range.
Y1percent	0	Default Y1 coordinate as a percentage of the Y range.
X2percent	100	Default X2 coordinate as a percentage of the X range.
Y2percent	100	Default Y2 coordinate as a percentage of the Y range.
ArcAndLineSteps	100	Default number of steps for interpolation on lines and arcs.
GridXsteps	100	Default number of steps in X direction for interpolation on grids.
GridYsteps	20	Default number of steps in Y direction for interpolation on grids.
ArcRadius	1.0	Default arc radius for interpolation on arcs.
MakePlotFile	No	Create Tablplot file when interpolating on lines, arcs, and curves.
MakeParmelaFile	No	Create input file for Parmela (cylindrically symmetric problems).
Force1MVperMeter	Yes	If No, SF7 uses SFO-calculated E0 for Parmela field map.
EGUNfileMeshSize	0.1	Default mesh size for the grid when creating EGUN input file.

19. Controlling field normalization in SF7 when generating Parmela data

Program Parmela assumes that field maps generated by SF7 have been normalized to 1.0 MV/m (at r = 0), averaged over the cell length. Parmela scales the fields according to the value of E_0 specified on the CELL, DTCELL, or TRWAVE lines in the Parmela input file. The SF7 dialog window includes a check box labeled “Force E0 = 1 MV/m” to force the 1.0-MV/m normalization. When supplying two field maps with different phases of the rf field for a traveling-wave accelerator, you may need a particular normalization. If you do not want the 1-MV/m normalization, uncheck the box. When running SF7 using .IN7 input files, insert the following line in the [SF7] section of SF.INI before running SF7:

```
[SF7]
Force1MVperMeter = No
```

Variable Force1MVperMeter also defines the initial setting of the check box in the SF7 dialog window.

20. Controlling some behavior of the field interpolator

The Poisson Superfish field interpolator generally performs very well on all types of static and rf field problems. However, users who find a case where performance is not optimal can try two types of adjustment to the field interpolator. One adjustment limits the number of terms in the harmonic fitting polynomial; the other adjustment prevents the interpolator from including third nearest neighbor points in the fit. Both of these features apply to all problem types except rf field problems with cylindrical symmetry. Note that these settings, if used, must appear in the [Global] section of SF.INI so they apply to all codes.

The SF.INI setting CutoffTerm allows user control of the magnitude of the cutoff term in the field interpolator. The field interpolator stops adding more terms to the harmonic fitting polynomial if the highest-order coefficient is already smaller than a certain fraction of the magnitude of the largest coefficient. The default fraction is 10^{-10} and it appears to work reasonably well for most of the problems we have seen. If you have reason to believe that the interpolated fields are noisy because too many terms are included in the fit, try larger settings (for example, 10^{-6}) for CutoffTerm in the [Global] section of SF.INI:

```
[Global]
CutoffTerm = 1.0E-6
```

In some cases, especially those with a coarse mesh, including third nearest neighbor points in the fit may result in a poorer fit than obtained with only first and second neighbor points. If you suspect this to be the case, try disabling third nearest neighbors for the particular type problem using one of the following settings in the [Global] section of SF.INI:

```
[Global]
ThirdNN_Electric_XY = No
ThirdNN_Electric_RZ = No
ThirdNN_Magnetic_XY = No
ThirdNN_Magnetic_RZ = No
ThirdNN_RF_XY = No
```

The default setting for all of these parameters is “Yes.” You also can use this feature to compare interpolated fields with an older version of Poisson Superfish. (We added the third nearest neighbor feature in version 6.04, released in April, 2001.)

When the field interpolator identifies the point of interest as a mesh node in the problem geometry, it checks an SF.INI variable for whether to report the fitted value of V (for electrostatic problems), A (for magnet problems), or H (for rf problems), or to report the actual value from the solution array. These values will generally be very nearly equal to one another. Differences depend upon the size of the mesh interval and details of the boundary shape. The values will be identical if the field interpolator uses one of the restricted harmonic polynomials near vertical or horizontal boundaries.

The place where the user is most likely to notice any differences is in the output of program SFO for electrostatic problems. If a boundary segment is not a horizontal or vertical line, the field interpolator uses a general polynomial that does not automatically satisfy a specified boundary condition on the segment (e.g., voltage = 0 along an electrode). To force the field interpolator to replace the fitted value of V, A, or H with the solution array value use one of the following settings in the [SFO] section of SF.INI:

```
[SFO]
InterpolateNodeV = No
InterpolateNodeA = No
InterpolateNodeH = No
```

The default setting for all three of these parameters is “Yes.” You also can set these preferences for other codes, but the situation where it would make any difference would rarely occur. To set the preference for all codes, put the keyword in the [Global] section of SF.INI. (We added this feature in version 6.19, released in April, 2002.)

21. Setting initial display preferences for Beta utility

The configuration variables in Table III-13 define the initial screen settings when first starting utility program Beta. After starting Beta, the settings may be changed from the program dialog window.

Table III-13. Beta display parameters in SF.INI.

Keyword	Default	Description
IndependentVariable	Energy	Choices: Energy, Beta, Gamma (code computes the other two).
StartingValue	1.0	First value of IndependentVariable for a table.
EndingValue	100.0	If equal to StartingValue, then result is single value rather than table.
NumberOfValues	101	Number of evenly spaced values from StartingValue to EndingValue.
MassUnit	On	Choices: MeV (rest-mass energy), amu (atomic mass units).
Particle	Proton	Choices: Proton, H-minus, Deuteron, D-minus, Electron, Other.
OtherMass	1.0	In MeV, unless MassUnit = amu.
EnergyDigits	8	Maximum significant digits (3 to 15) for computed kinetic energy.
BetaDigits	8	Maximum significant digits (3 to 15) for computed beta.
GammaDigits	8	Maximum significant digits (3 to 15) for computed gamma.

22. Writing the file FishScan.TBL in Fish or CFish

The FishScan configuration variable controls whether programs Fish and CFish, create and update a file called FishScan.TBL, which is a Tablplot input file containing the results of mode searches and frequency scans. By default, Fish (or CFish) creates or updates this file automatically when you use variable NSTEP to step the frequency. You can configure the codes to also write the FishScan.TBL file during resonance searches by setting FishScan to Always. Insert the following line in the [Global] section of SF.INI:

```
[Global]
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
```

Never and Always are the only values available. The codes ignore any other setting of FishScan.

CFish has another option that affects the initial display when starting Tablplot on file FishScan.TBL. If variable ScanFormatComplex = Yes (the default), then the initial display shows a plot of Im(δH1) versus Re(δH1) if the solution array is complex, where δH1 is the field error at the driving point. This display also labels every point by the value of the frequency for that data point. If ScanFormatComplex = No (or if the solution array is real), then the initial display shows $D(k^2)$ versus frequency as it does for the results written by program Fish.

23. Create file CircleFit.QKP during CFish resonance

If variable CFishCirclePlot = Yes in the [CFish] section of SF.INI, CFish writes an input file for program Quikplot called CircleFit.QKP, which contains points along the fitted resonance circle at each cycle of a mode search. Each curve is in a separate Data/EndData section labeled by the cycle number in the search. The file also contains a separate curve for each fitted circle showing the line between (0,0) and the center of the resonance circle. For more information see the discussion of the complex root finder in the Fish and CFish chapter.

24. Writing Poisson or Pandira field data for program Parmela

If you are a licensed user of Parmela, you can use settings in the [Poisson] and [Pandira] sections of SF.INI to generate Parmela input files of electrostatic or magnetostatic fields. (Note: Running postprocessor SF7 provides a more convenient method for producing this file.) Parmela is an electron linac design code, the name being derived from the phrase, “Phase and Radial Motion in Electron Linear Accelerators.” It is a multi-particle code that transforms a beam, represented by collection of particles, through a user-specified linac and/or transport system. To generate a Parmela input file, you must satisfy the following conditions:

1. The problem must have cylindrical symmetry (ICYLIN = 1),
2. Automesh input variables XMINF, XMAXF,YMINF, and YMAXF must specify a valid range of physical coordinates,
3. Automesh input variables KTOP and LTOP, the number of interpolation steps in the R and Z directions, must both be 2 or greater,
4. The ParmelaFields setting in SF.INI must be Yes.

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
```

When you use this option, the filename of the Parmela file 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 same file can be produced by program SF7.) The codes also write files EFLD.QKP (for electrostatic field problems) or BFLD.QKP (for magnetostatic field problems). These are Quikplot input files for viewing some of the data in the Parmela file. 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.

25. Writing permeability tables in Poisson and Pandira

Poisson and Pandira build long permeability tables with equal-spaced B intervals for fast interpolation during the solution. These tables consist of B,γ pairs interpolated from both the internal permeability tables and from user-supplied data. You can make the codes write these data tables to a Tablplot input file named BGammaTbl by setting the PrintBGammaTables option in either the [Global] section or the individual code sections of SF.INI. For example, to print the tables in Poisson, use the following lines:

```
[Poisson]
PrintBGammaTables = Yes
```

26. Calculating stored energy in Poisson and Pandira

The ComputeStoredEnergy variable in the [Poisson] and [Pandira] sections of SF.INI determines whether the codes will calculate the stored energy in the absence of an entry for IENERGY in the Automesh input file. Set ComputeStoredEnergy in either the [Global] section or the individual code sections of SF.INI. For example, to have only Pandira calculate stored energy by default, use the following lines:

```
[Global]
ComputeStoredEnergy = No

[Pandira]
ComputeStoredEnergy = Yes
```

The default in the codes is not to compute the stored energy. The reason for this setting is that the calculation can take considerable time, particularly for electrostatic problems. The ability to compute stored energy in permanent-magnet or anisotropic materials has not yet been added to Pandira. This feature may be available in a future release. The present code will write a warning message if you request a stored-energy calculation in problems with permanent-magnet or anisotropic materials.

27. Configuring particle velocity settings for SFO and tuning programs

For cylindrically symmetric problems, SFO calculates the transit-time factor and related integrals. The default parameters correspond to half of a 1βλ drift-tube linac cavity, where βλ is the distance traveled in one rf period by a particle of velocity βc. If variable KMETHOD = 1, and BETA has a value between 0 and 1, then SFO calculates CAPK, the phase change per unit length:

```
CAPK = XK0/BETA
```

where XK0 is the wave number k = ω/c and BETA is the particle velocity β. Otherwise (KMETHOD ≠ 1 or BETA ≤ 0 or BETA > 1), the calculation uses the equation CAPK =

DPHIR/DELZ, where DPHIR is the value of DPHI expressed in radians, and DELZ is the distance from the cell's electrical center to the end of the cell. The code then computes BETA = XK0/CAPK. If the calculation for BETA results in an unphysical value (see also next paragraph), then SFO uses the default value of BETA = 1.0. You can change this default value by setting variable AssumedBeta in [Global], [Autofish], or [SFO] sections of SF.INI.

In cases where SFO computes a value for BETA and the result exceeds 1.0, the code will reset it to exactly 1.0 if the value is within the range set by variable BetaTolerance. The default value of BetaTolerance is 0.00001. If BETA > 1.0 + BetaTolerance, then SFO will write a warning message and set BETA = AssumedBeta. The code does not consider the value unphysical if it lies within the allowed range.

When tuning programs CCLfish, CDTfish, DTLfish, ELLfish, and MDTfish calculate BETA from the supplied cavity length, they check to see if the values lies within the range defined by the BetaTolerance setting in SF.INI. If so, then the code sets BETA = 1.0 and does not issue a warning message.

28. Calculating fields on Dirichlet boundaries in SFO

For Superfish problems, postprocessor SFO omits Dirichlet boundaries from power and field calculations. There are several reasons for this default setting. Usually the cavity designer is interested in the power dissipated on the metal surfaces and the peak magnetic and electric fields on these surfaces. Segments that satisfy a Dirichlet boundary condition cannot contribute to the power losses since $H_\phi = 0$. Metal surfaces must satisfy the Neumann boundary condition, so the peak surface electric field will not occur on a Dirichlet boundary.

Occasionally, the need arises for listing the fields along the Dirichlet boundary segments in a Superfish problem. For example, one might use the complimentary fields to find the solution for a TE cavity mode. In this case, the roles of E and H are interchanged. The listed values for E in SFO output file actually correspond to magnetic field H_ϕ in the TM mode. You can prevent SFO from automatically skipping the Dirichlet boundary segments by setting IncludeDirichletSegs = Yes in either the [Global] section or the [SFO] section of SF.INI.

29. Writing transit-time data for Parmila in SFO

If you use Parmila, you can use settings in the [SFO], [Autofish], [CCLfish], [CDTfish], [DTLfish], [ELLfish], and [MDTfish] sections of SF.INI to generate a file containing transit-time data. Parmila is an ion linac design code, the name being derived from the phrase, "Phase and Radial Motion in Ion Linear Accelerators." It is a multi-particle code most often used to design drift-tube linacs. 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
```

When you use this option, the filename of the Parmila file is based upon the Automesh input file's root name. For example, if you start Automesh with input file PROB1, then the codes will write file PROB1.PMI. The PMI file includes properties of the entire cavity followed by lists of parameters for each cell containing an accelerating gap. The data written in this file also can be found in the SFO output file. However, writing the PMI file directly can save time searching for it in the longer SFO file. In addition, Parmila users can run the ReadPMI program to extract tables of data from a family of PMI files for use in Parmila.

30. Saving the binary solution files during tuning program sessions

The SaveTAPE35 variable in the [CCLfish], [CDTfish], [DTLfish], [ELLfish], [MDTfish], [RFQfish], and [SCCfish] sections of SF.INI determine whether the tuning program saves the binary solution file for each problem. Actual filenames depend on the preferences discussed in section 1 above. By default, tuning programs create files with extension T35 and retain them after completing the solution. Saving the solution files allows you view the mesh and field lines later using WSFplot, or (for example) to renormalize the fields in SFO to a different value of E_0 or E_0T.

One reason you might choose not to save the solution files would to make more disk space available for other files. However, you would then need to compute the solution again to view the fields in WSFplot or to run another postprocessor such as SF7. If variable SaveTAPE35 = No, then the tuning program deletes each solution file after completing the solution. The updated AM files and the SFO files contain the results of each solution, but there are no binary solution files in the directory after the tuning-code session. Each tuning program can have its own setting for SaveTAPE35. For example, the following settings configure CCLfish to discard solution files and CDTfish to save solution files:

```
[CCLfish]
SaveTAPE35 = No

[CDTfish]
SaveTAPE35 = Yes
```

31. Setting the preferred normalization method in tuning codes

The SF.INI Normalization setting declares a preference for the method of field normalization in the automated tuning programs CCLfish, CDTfish, DTLfish, ELLfish, and MDTfish. There are two possible values:

```
Normalization = E0
Normalization = E0T
```

These tuning codes have two input-file keywords related to this SF.INI setting. The keywords are E0_Normalization and E0T_Normalization After a problem has been completed the tuned-data file (a backup version of the control file) will contain both values. For example, suppose DTLfish completed a run for which it normalized fields to E_0 = 5.0 MV/m and the calculated transit-time factor was 0.8. The tuned-data file will include the following lines for that completed problem:

```
E0_Normalization          5
E0T_Normalization         4
```

Provided that you make no other changes to the geometrical parameters for this problem, it does not matter which method of normalization the code uses if you run this problem again by removing the minus sign on the START-line parameter. However, if you modify the geometry in some way, it may matter because the new geometry could result in a different transit-time factor. If SF.INI has a preference for E_0, then the code will again normalize to E_0 = 5.0 MV/m, and get a new value for the product E_0T.

The codes check the preference in SF.INI if the control file contains neither or both lines that set the field normalization. Using only one of these keywords in a problem's control file overrides the SF.INI setting. In the absence of any settings in either SF.INI or the control file, the codes normalize to E_0 = 1.0 MV/m. The SF.INI file distributed with the codes has preferences set to E_0 in the[DTLfish] and [MDTfish] sections and to E_0T in the [CCLfish], [ELLfish], and [CDTfish] sections.

32. Making CDTfish compute transit-time factors at the geometric center

The default setting in the CDTfish tuning program adjusts the drift tube noses to make the electrical center coincide with the geometric center of each gap. The code also can tune the cavities without adjusting the gap centers. For such fixed-gap runs, CDTfish runs the postprocessor SFO one additional time with the reference position Zc for transit-time-factor integrals located at the each cell's electrical center. This correction allows the beam-dynamics code Parmila to perform the most accurate simulation of the linac. CDTfish obtains the location of the electrical center with respect to the original value of Zc from the last completed Superfish run on the problem. You can force the code to use the cell's geometric center with the following setting in file SF.INI:

```
[CDTfish]
Zcenter = geometric
```

33. Setting a maximum tuning-ring thickness in CCLfish and CDTfish

Tuning codes CCLfish and CDTfish can include a tuning-ring feature either at the equator of the cavity or on both end walls of the cavity. The codes limit the maximum thickness of these tuning rings to a fraction of the full (or half) cavity length. For the equator ring, the default thickness is 15% of the full cavity length (not including the septum thickness); for the wall rings, the default thickness of each ring on the cavity end walls is 15% of the half cavity length (not including the septum thickness). You can change these settings to any percentage between 5% and 50% using SF.INI variables EquatorRingLimit and

WallRingLimit. You can set different limits for both type rings and for both codes as in the following example:

```
[CDTfish]
EquatorRingLimit = 20
WallRingLimit = 15

[CCLfish]
EquatorRingLimit = 10
WallRingLimit = 20
```

Use caution when setting the limits larger than 15 or 20%, especially for the wall tuning rings. A wall tuning ring raises the cavity frequency for a small thickness, but as the ring gets thicker it will eventually lower the cavity frequency.

34. Deleting or saving no-ring Superfish files in CCLfish and CDTfish

When tuning codes CCLfish and CDTfish design a cavity with a tuning-ring of a given thickness the codes perform an additional Superfish run to determine the frequency effect of the tuning ring. The files generated by this step are usually often not needed for anything else, so by default the codes delete the files. If you wish to save the no-ring files, you can set DeleteNoRingFiles as follows in the appropriate section of file SF.INI:

```
[CDTfish]
DeleteNoRingFiles = No

[CCLfish]
DeleteNoRingFiles = No
```

35. Controlling line regions through the drift tube in DTLfish and MDTfish

Tuning programs DTLfish and MDTfish insert a Y line region through the face angle segment on the drift tubes under certain circumstances. The code checks if there is sufficient room for several rows of the next larger mesh triangles between the outer nose radius and the corner radius. To prevent the code from inserting this Y line region, set AllowYregInGap as follows in the appropriate section of file SF.INI:

```
[DTLfish]
AllowYregInGap = No

[MDTfish]
AllowYregInGap = No
```

36. Computing Slater-Perturbation Frequency Shifts in WSFplot and SF7

For rf cavities, programs WSFplot and SF7 have options to compute the Slater-perturbation frequency shift for different shapes of a perturbing object. Table III-14 lists possible values of the keyword SlaterTerm. To turn on this feature in WSFplot or SF7 set SlaterTerm = Yes. WSFplot includes the computed frequency shift in the field display window, and SF7 adds three columns of data in the Tablplot input file.

Table III-14. SF.INI variable SlaterTerm for WSFplot and SF7.

SlaterTerm	Description	k_H	k_E
Yes	Turns feature on in WSFplot or SF7. WSFplot displays the Sphere term except near a metal boundary where it displays the Surface term.		
Surface	Perturbing volume does not distort the fields. In SF7, this keyword plots the Surface term when Tablplot starts.	1.0	1.0
Sphere	Perturbing volume is a sphere. In SF7, this keyword plots the Sphere term when Tablplot starts.	1.5	3.0
Cylinder	Perturbing volume is a circular cylinder.	2.0	2.0
(any other value)	Frequency shift data is not computed.	-	-

In SF7, if you use of the keywords listed in Table III-14, then the selected term will be one of the initially plotted columns when Tablplot is run on the resulting plot file. The three columns in SF7 correspond three possible shape factors. The Surface term corresponds to a displacement of a cavity surface in such a way that it does not distort the fields. The other shapes are a Sphere and a right Circular with axis perpendicular the rf fields. You can set different values in SF.INI for the two programs as shown in the following example:

```
[WSFplot]
SlaterTerm = Yes

[SF7]
SlaterTerm = Sphere
```

G. Precision of variables in Poisson Superfish

For real (floating-point) variables, Poisson Superfish programs use 8-byte (double precision) variables throughout, except in some parts of plotting routines. For the Lahey/Fujitsu Fortran compiler that we use, this corresponds to 53-bit precision or about 1 part in 9.0×10^{15}. Data stored in the binary solution file retain the full 8-byte precision. Complex variables consist of 16 bytes, each part being an 8-byte real variable. Plotting programs, including sections of WSFplot, use 4-byte real variables.

References to constants in the codes use the explicit representation of a double precision number to avoid round-off errors at 24 bits of precision. For example, the real number 2 would appear in the code as “2.0D0.” To obtain machine precision for the value of π, the code uses the statement:

```
        PI = 4.0D0*ATAN(1.0D0)
```

On Pentium processors running Lahey/Fujitsu Fortran 95 with double precision, this value of π corresponds to 3.1415926535897931. The codes use the exact value of the speed of light in cm/sec. The speed of light appears in variable CLIGHT:

```
        CLIGHT = 29979245800.0000D0
```

Codes that use ε_0 and μ_0 (the permeability and permittivity of free space) define these quantities in MKS units as follows.

```
        FMU0 = PI*4.0D–7
        EPS0 = 1.0D0/(FMU0*(0.01D0*CLIGHT**2))
```

The numerical value of FMU0 and EPS0 are:

$\mu_0 = 1.25663706143592 \times 10^{-06}$ T-m/A, and
$\varepsilon_0 = 8.85418781762039 \times 10^{-12}$ F/cm.

Poisson and Pandira use different units for μ_0 when converting input data for regions containing source terms. In these codes,

$\mu_0 = 0.4\pi$ Gauss-cm/A.

H. Field interpolation from the solution arrays

All codes that report field information use the Poisson Superfish field interpolator to compute the fields. For rf problems in Cartesian coordinates the field interpolator computes E_x, E_y, E, and H_z., and for rf problems in cylindrical coordinates it computes E_z, E_r, E, and H_ϕ. For magnet problems in Cartesian coordinates the field interpolator computes A_z, B_x, B_y, B, $\partial B_y/\partial y$, $\partial B_x/\partial y$, and $\partial B_y/\partial x$. It does not supply a value for $\partial B_x/\partial x$ because (in the absence of a magnetic monopole) $\partial B_x/\partial x = -\partial B_y/\partial y$. In regions of zero current density there are only two unique second derivatives since $\partial B_x/\partial y = \partial B_y/\partial x$. For magnet problems in cylindrical coordinates the field interpolator computes A_ϕ, B_r B_z, B, $\partial B_z/\partial r$, $\partial B_r/\partial z$, and the field index $n = (r/B_z)(\partial B_z/\partial r)$. For electrostatic problems in Cartesian coordinates the field interpolator computes V, E_x, E_y, and E, and for electrostatic problems in cylindrical coordinates it computes V, E_r E_z, and E.

The field interpolator fits one of several polynomial functions of X and Y (or R and Z) to the first and second nearest neighbors of a reference mesh point (see Figure III-2). The reference point will usually be the nearest mesh point except near boundaries. Near boundaries, the code may choose a first nearest neighbor of the closest point in order to maximize the number of points used in the polynomial fit. Choosing the reference point in this way ensures that the fit includes all of the first nearest neighbors of the reference point.

Near boundaries, some of the second nearest neighbors are unavailable. For certain problems (usually those that can potentially use high order polynomial fitting functions), this lack of available mesh point can seriously degrade the fit. For these cases, the interpolator may add several of the third nearest neighbors until the total number of neighboring points reaches 18. The code chooses available third nearest neighbors that are physically closest to point P in Figure III-2.

The interpolator does not use third nearest neighbors for rf problems with cylindrical symmetry. For this type problem, including more points can actually degrade the fit because of the relatively low-order polynomials in these fitting functions. For cylindrical rf problems, the quality of the fits using only a few terms and a restricted number of mesh points near boundaries is generally very good.

For each type problem that may use third nearest neighbors (any problem with Cartesian coordinates, and static magnetic field or electric field problems) you can disable the use of third nearest neighbors using settings in the [Global] section of file SF.INI.

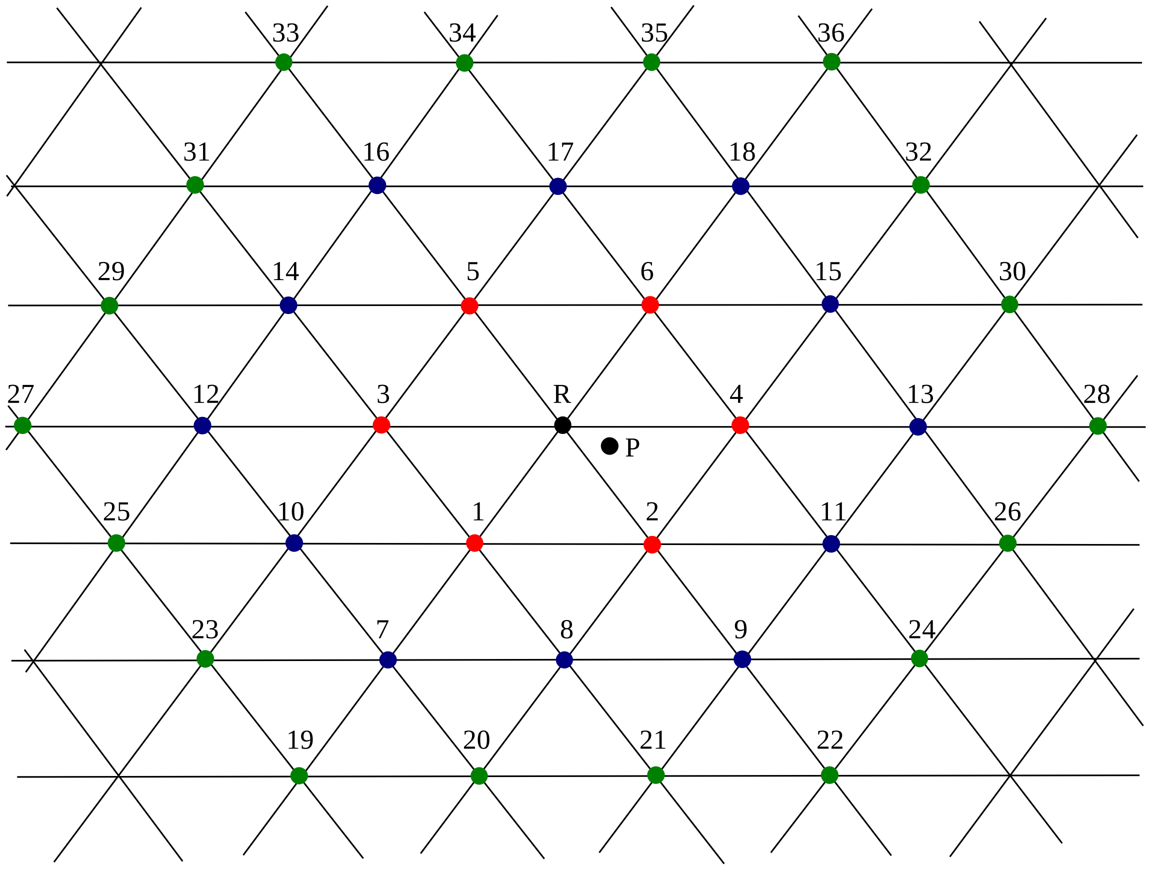

Figure III-2. Triangular mesh showing reference point and neighboring points. Reference point R is the mesh point closest to point P at which the fields are interpolated. Points 1 through 6 are first nearest neighbors, points 7 through 18 are second nearest neighbors and points 19 through 36 are third nearest neighbors.

Different polynomial functions satisfy either Neumann or Dirichlet boundary conditions on or near vertical or horizontal boundaries. Formulas used by the field interpolator appear in the chapters listed in Table III-15. These chapters have been reproduced from John L. Warren’s treatment in the 1987 Reference Manual (LA-UR-87-126).

If you are a code developer you can incorporate the field interpolator into your own programs. Users with Lahey/Fujitsu Fortran, version 5.7 can call subroutines RDTape35 and Interpolate directory from their own programs. For other language systems, you may link your program with one the appropriate dynamic link library (DLL). These tools are found in subdirectories of the DeveloperFiles\PoissonSuperfish directory.

Table III-15. Electromagnetic theory sections.

Topic	Manual section	1987 Reference Manual section
X,Y static fields	Chapter XXI	Section B.13.2, page 26.
R,Z static fields	Chapter XXI	Section B.13.2, page 36.
X,Y rf fields	Chapter XXIV	Section C.13.1, page 10.
R,Z rf fields	Chapter XXIV	Section C.13.1, page 3.

Because the field interpolator uses first and second (and possibly third) nearest neighbors, you may see erroneous values if the mesh is too coarse in the vicinity of thin metal objects. For example, near a metal plate that is only one row of mesh triangles thick, the code may include points in the fit from the other side of the plate. These interpolated fields will be incorrect. The only solution is to make this part of the mesh finer. You can check the fits by looking at the chi-squared goodness of fit parameter printed by program SF7 using the extended table option. Poisson and Pandira also include the chi-squared in the table of interpolated fields in file OUTPOI.TXT or OUTPAN.TXT. The chi-squared reported by the codes is normalized to the square of the average of the absolute values of solution array (H, A, or V) for the points used in the fit.

The 1987 Reference Manual described a method for interpolating static electric and magnetic fields using harmonic polynomials (see Chapter XXI). Harmonic polynomials are functions of X and Y that are solutions to Laplace's equation. The present codes use harmonic polynomials for static electric and magnetic field problems. RF problems with cylindrical symmetry do not use harmonic polynomials. Furthermore, the present interpolator uses only the first few (through n = 2) of the Cartesian harmonic polynomials. High powers of X and Y cannot be fit with the limited amount of data available from just first and second nearest neighbors. This is the main reason that Billen and Young replaced the old interpolator found in routines known as WORK and CWORK. Fields interpolated by the XY routine WORK were extremely unreliable especially for electrostatic problems.

The functions used by the new interpolator for XY geometries still satisfy Laplace's equation, but only locally, that is near the point of interest X_0,Y_0. Terms of higher order than quadratic that contain $(X-X_0)$ or $(Y-Y_0)$ do not themselves contribute to the partial derivatives of A. It is unnecessary for the interpolation polynomial to satisfy Laplace's equation at all X and Y since we are only using it to get an approximate result at X_0,Y_0. For another location, we use another polynomial.

The new interpolator for cylindrical magnetic and electrostatic fields does not differ very much from the old interpolator. The main differences are in the application of boundary conditions on horizontal Dirichlet and Neumann boundaries. The old code did not attempt to satisfy boundary conditions. The interpolator can use cylindrical harmonic polynomials through order 8, if necessary. It chooses a lower order if the Chi-squared increases with the addition of another term, if the determinant computed by the matrix inversion routine drops by more than a factor of 10^{34}, or if the highest-order coefficient is already 10^{10} times smaller than the magnitude of the largest coefficient.

The function numbers below appear in the WSFplot field-display window. Interior points in the mesh use the most general fitting function available for the problem at hand. Near certain boundaries the code uses a more restricted fitting function to satisfy the appropriate boundary conditions. The interpolation point is X_0,Y_0. The horizontal coordinate of a Neumann or Dirichlet vertical boundary is X_b, and the vertical coordinate of a Neumann or Dirichlet horizontal boundary is Y_b. You can see the fitted coefficients P_1, P_2, etc. if you use the expanded table option in SF7.

1. Problems with Cartesian (XY) coordinates use functions 17 to 25

All Poisson Superfish problems that have ICYLIN = 0 use functions 17 to 25. The setting ICYLIN = 0 specifies a geometry with rectangular coordinates.

a. *Function 17: general function for interior non-boundary points*

$$\begin{aligned}\mathrm{Fn17} &= P_1\\ &+ P_2\,(X-X_0)\\ &+ P_3\,(Y-Y_0)\\ &+ P_4\,[(X-X_0)^2-(Y-Y_0)^2]\\ &+ P_5\,(X-X_0)\,(Y-Y_0)\\ &+ P_6\,(X-X_0)^2\,(Y-Y_0)\\ &+ P_7\,(X-X_0)\,(Y-Y_0)^2\end{aligned}$$

b. *Function 18: vertical Neumann boundary*

$$\begin{aligned}\mathrm{Fn18} &= P_1\\ &+ P_2\,(Y-Y_0)\\ &+ P_3\,[(X-X_b)^2-(Y-Y_0)^2]\\ &+ P_4\,(X-X_b)^2\,(Y-Y_0)\\ &+ P_5\,(X-X_b)^2\,(Y-Y_0)^2\end{aligned}$$

c. *Function 19: horizontal Neumann boundary*

$$\begin{aligned}\mathrm{Fn19} &= P_1\\ &+ P_2\,(X-X_0)\\ &+ P_3\,[(X-X_0)^2-(Y-Y_b)^2]\\ &+ P_4\,(Y-Y_b)^2\,(X-X_0)\\ &+ P_5\,(Y-Y_b)^2\,(X-X_0)^2\end{aligned}$$

d. *Function 20: vertical Neumann and horizontal Neumann boundaries*

$$\begin{aligned}\mathrm{Fn20} &= P_1\\ &+ P_2\,[(X-X_b)^2-(Y-Y_b)^2]\\ &+ P_3\,(X-X_b)^2\,(Y-Y_b)^2\end{aligned}$$

e. *Function 21: vertical Dirichlet and horizontal Dirichlet boundaries.*

$$\begin{aligned}\mathrm{Fn21} &= P_1\,(X-X_b)\,(Y-Y_b)\\ &+ P_2\,(X-X_b)\,(Y-Y_b)^3\\ &+ P_3\,(X-X_b)^3\,(Y-Y_b)\end{aligned}$$

f. *Function 22: vertical Dirichlet boundary*

$$\begin{aligned} \mathrm{Fn22} &= P_1 (X-X_b) \\ &+ P_2 (X-X_b) (Y-Y_0) \\ &+ P_3 (X-X_b) (Y-Y_0)^2 \end{aligned}$$

g. *Function 23: horizontal Dirichlet boundary*

$$\begin{aligned} \mathrm{Fn23} &= P_1 (Y-Y_b) \\ &+ P_2 (Y-Y_b) (X-X_0) \\ &+ P_3 (Y-Y_b) (X-X_0)^2 \end{aligned}$$

h. *Function 24: vertical Neumann and horizontal Dirichlet boundaries*

$$\begin{aligned} \mathrm{Fn24} &= P_1 (Y-Y_b) \\ &+ P_2 (X-X_b)^2 (Y-Y_b) \\ &+ P_3 (X-X_b)^2 (Y-Y_b)^3 \end{aligned}$$

i. *Function 25: vertical Dirichlet and horizontal Neumann boundaries*

$$\begin{aligned} \mathrm{Fn25} &= P_1 (X-X_b) \\ &+ P_2 (Y-Y_b)^2 (X-X_b) \\ &+ P_3 (Y-Y_b)^2 (X-X_b)^3 \end{aligned}$$

2. Electrostatic problems with cylindrical symmetry use functions 27 to 29

Poisson and Pandira problems with ICYLIN = 1 and XJFACT = 0 use functions 27, 28 and 29. The setting ICYLIN = 1 specifies a geometry using cylindrical coordinates and XJFACT = 0 specifies an electrostatic problem. Radial coordinate R corresponds to the input file’s horizontal coordinate X, and longitudinal coordinate Z corresponds to the vertical coordinate Y. Functions 27 to 29 use harmonic polynomials. No attempt is made to exactly satisfy the boundary conditions on vertical boundaries other than R = 0. When using function 27, the interpolator starts with the first three terms and then adds terms in pairs until the quality of the fit no longer improves.

a. *Function 27: general function for interior non-boundary points*

$$\begin{aligned}
\mathrm{Fn27} = {} & P_1 \\
& + P_2\,(Z-Z_0) \\
& + P_3\,[(Z-Z_0)^2 - 0.5\,R^2] \\
& + P_4\,[(Z-Z_0)^3 - 1.5\,R^2\,(Z-Z_0)] \\
& + P_5\,[(Z-Z_0)^4 - 3.0\,R^2(Z-Z_0)^2 + 0.375\,R^4)] \\
& + P_6\,[(Z-Z_0)^5 - 5.0\,R^2\,(Z-Z_0)^3 + 1.875\,R^4\,(Z-Z_0)] \\
& + P_7\,[(Z-Z_0)^6 - 7.5\,R^2\,(Z-Z_0)^4 + 5.625\,R^4\,(Z-Z_0)^2 - 0.3125\,R^6] \\
& + P_8\,[(Z-Z_0)^7 - 10.5\,R^2\,(Z-Z_0)^5 + 13.125\,R^4\,(Z-Z_0)^3 - 2.1875\,R^6\,(Z-Z_0)] \\
& + P_9\,[(Z-Z_0)^8 - 14.0\,R^2\,(Z-Z_0)^6 + 26.25\,R^4\,(Z-Z_0)^4 - 8.75\,R^6\,(Z-Z_0)^2 \\
& \quad - 0.2734375\,R^8] \\
& + P_{10}\,[(Z-Z_0)^9 - 18.0\,R^2\,(Z-Z_0)^7 + 47.25\,R^4\,(Z-Z_0)^5 - 26.25\,R^6\,(Z-Z_0)^3 \\
& \quad + 2.4609375\,R^8\,(Z-Z_0)] \\
& + P_{11}\,[(Z-Z_0)^{10} - 22.5\,R^2\,(Z-Z_0)^8 + 78.75\,R^4\,(Z-Z_0)^6 - 65.625\,R^6\,(Z-Z_0)^4 \\
& \quad + 12.3046875\,R^8\,(Z-Z_0)^2 - 0.24609375\,R^{10}] \\
& + P_{12}\,[(Z-Z_0)^{11} - 27.5\,R^2\,(Z-Z_0)^9 + 123.75\,R^4\,(Z-Z_0)^7 - 144.325\,R^6\,(Z-Z_0)^5 \\
& \quad + 45.1171875\,R^8\,(Z-Z_0)^3 - 2.70703125\,R^{10}\,(Z-Z_0)]
\end{aligned}$$

b. *Function 28: horizontal Dirichlet boundary*

$$\begin{aligned}
\mathrm{Fn28} = {} & P_1\,(Z-Z_b) \\
& + P_2\,[(Z-Z_b)^3 - 1.5\,R^2\,(Z-Z_b)] \\
& + P_3\,[(Z-Z_b)^5 - 5.0\,R^2\,(Z-Z_b)^3 + 1.875\,R^4\,(Z-Z_b)] \\
& + P_4\,[(Z-Z_b)^7 - 10.5\,R^2\,(Z-Z_b)^5 + 13.125\,R^4\,(Z-Z_b)^3 - 2.1875\,R^6\,(Z-Z_b)] \\
& + P_5\,[(Z-Z_b)^9 - 18.0\,R^2\,(Z-Z_b)^7 + 47.25\,R^4\,(Z-Z_b)^5 - 26.25\,R^6\,(Z-Z_b)^3 \\
& \quad + 2.4609375\,R^8\,(Z-Z_b)] \\
& + P_6\,[(Z-Z_b)^{11} - 27.5\,R^2\,(Z-Z_b)^9 + 123.75\,R^4\,(Z-Z_b)^7 - 144.325\,R^6\,(Z-Z_b)^5 \\
& \quad + 45.1171875\,R^8\,(Z-Z_b)^3 - 2.70703125\,R^{10}\,(Z-Z_b)]
\end{aligned}$$

c. *Function 29: horizontal Neumann boundary*

$$\begin{aligned}
\mathrm{Fn29} = {} & P_1 \\
& + P_2\,[(Z-Z_b)^2 - 0.5\,R^2] \\
& + P_3\,[(Z-Z_b)^4 - 3.0\,R^2(Z-Z_b)^2 + 0.375\,R^4)] \\
& + P_4\,[(Z-Z_b)^6 - 7.5\,R^2\,(Z-Z_b)^4 + 5.625\,R^4\,(Z-Z_b)^2 - 0.3125\,R^6] \\
& + P_5\,[(Z-Z_b)^8 - 14.0\,R^2\,(Z-Z_b)^6 + 26.25\,R^4\,(Z-Z_b)^4 - 8.75\,R^6\,(Z-Z_b)^2 \\
& \quad + 0.2734375\,R^8] \\
& + P_{11}\,[(Z-Z_b)^{10} - 22.5\,R^2\,(Z-Z_b)^8 + 78.75\,R^4\,(Z-Z_b)^6 - 65.625\,R^6\,(Z-Z_b)^4 \\
& \quad + 12.3046875\,R^8\,(Z-Z_b)^2 - 0.24609375\,R^{10}]
\end{aligned}$$

3. Magnetostatic problems with cylindrical coordinates use functions 37 to 39

Poisson and Pandira problems with ICYLIN = 1 and a nonzero value of XJFACT use functions 37 to 39. The setting ICYLIN = 1 specifies a geometry using cylindrical coordinates and a nonzero XJFACT specifies a magnetostatic problem. Radial coordinate

R corresponds to the input file's horizontal coordinate X, and longitudinal coordinate Z corresponds to the vertical coordinate Y. Functions 27 to 29 use harmonic polynomials so that the interpolator can calculate second derivatives of the vector potential (i.e. field gradients). No attempt is made to exactly satisfy the boundary conditions on vertical boundaries other than R = 0.

a. Function 37: interior non-boundary points

$$\begin{aligned}\text{Fn37} = {} & P_1 R^2 \\ & + P_2 R^2 (Z\text{–}Z_0) \\ & + P_3 [R^2 (Z\text{–}Z_0)^2 - 0.25 R^4] \\ & + P_4 [R^2 (Z\text{–}Z_0)^3 - 0.75 R^4 (Z\text{–}Z_0)] \\ & + P_5 [R^2 (Z\text{–}Z_0)^4 - 1.5 R^4 (Z\text{–}Z_0)^2 + 0.125 R^6)] \\ & + P_6 [R^2 (Z\text{–}Z_0)^5 - 2.5 R^4 (Z\text{–}Z_0)^3 + 0.625 R^6 (Z\text{–}Z_0)] \\ & + P_7 [R^2 (Z\text{–}Z_0)^6 - 3.75 R^4 (Z\text{–}Z_0)^4 + 1.875 R^6 (Z\text{–}Z_0)^2 - 0.078125 R^8]\end{aligned}$$

b. Function 38: horizontal Dirichlet boundary

$$\begin{aligned}\text{Fn38} = {} & P_1 R^2 (Z\text{–}Z_b) \\ & + P_2 [R^2 (Z\text{–}Z_b)^3 - 0.75 R^4 (Z\text{–}Z_b)] \\ & + P_3 [R^2 (Z\text{–}Z_b)^5 - 2.5 R^4 (Z\text{–}Z_b)^3 + 0.625 R^6 (Z\text{–}Z_b)] \\ & + P_4 [R^2 (Z\text{–}Z_b)^7 - 5.25 R^4 (Z\text{–}Z_b)^5 + 4.375 R^6 (Z\text{–}Z_b)^3 - 0.546875 R^8 (Z\text{–}Z_b)]\end{aligned}$$

c. Function 39: horizontal Neumann boundary

$$\begin{aligned}\text{Fn39} = {} & P_1 R^2 \\ & + P_2 [R^2 (Z\text{–}Z_b)^2 - 0.25 R^4] \\ & + P_3 [R^2 (Z\text{–}Z_b)^4 - 1.5 R^4 (Z\text{–}Z_b)^2 + 0.125 R^6)] \\ & + P_4 [R^2 (Z\text{–}Z_b)^6 - 3.75 R^4 (Z\text{–}Z_b)^4 + 1.875 R^6 (Z\text{–}Z_b)^2 - 0.078125 R^8]\end{aligned}$$

4. RF problems with cylindrical coordinates use functions 57 to 68

Superfish problems with cylindrical symmetry have ICYLIN = 1 and use functions 57 to 68. Radial coordinate R corresponds to the input file's vertical coordinate Y, and longitudinal coordinate Z corresponds to the horizontal coordinate X.

a. Function 57: general function for interior non-boundary points

$$\begin{aligned}\text{Fn57} = {} & P_1 R \\ & + P_2 R^2 \\ & + P_3 R (Z\text{–}Z_0) \\ & + P_4 R^2 (Z\text{–}Z_0) \\ & + P_5 [R^2 (Z\text{–}Z_0)^2 - R^4/4] \\ & + P_6 R^3\end{aligned}$$

b. Function 58: horizontal Neumann boundary

$$\begin{aligned} \mathrm{Fn58} = {} & P_1 \\ & + P_2\,(Z-Z_0) \\ & + P_3\,(Z-Z_0)^2 \\ & + P_4\,(R-R_b)^2 \\ & + P_5\,(R-R_b)^2\,(Z-Z_0) \\ & + P_6\,(R-R_b)^2\,(Z-Z_0)^2 \end{aligned}$$

c. Function 59: vertical Neumann boundary

$$\begin{aligned} \mathrm{Fn59} = {} & P_1\,R \\ & + P_2\,R^2 \\ & + P_3\,R^3 \\ & + P_4\,R\,(Z-Z_b)^2 \\ & + P_5\,R^2\,(Z-Z_b)^2 \end{aligned}$$

d. Function 60: horizontal Neumann and vertical Neumann boundaries

$$\begin{aligned} \mathrm{Fn60} = {} & P_1 \\ & + P_2\,(Z-Z_b)^2 \\ & + P_3\,(R-R_b)^2 \\ & + P_4\,(R-R_b)^2\,(Z-Z_b)^2 \end{aligned}$$

e. Function 61: horizontal Dirichlet and vertical Dirichlet boundaries

$$\begin{aligned} \mathrm{Fn61} = {} & P_1\,(R-R_b)\,(Z-Z_b) \\ & + P_2\,(R-R_b)^2\,(Z-Z_b) \end{aligned}$$

f. Function 62: horizontal Dirichlet boundary at nonzero R

$$\begin{aligned} \mathrm{Fn62} = {} & P_1\,(R-R_b) \\ & + P_2\,(R-R_b)^2 \\ & + P_3\,(R-R_b)\,(Z-Z_0) \\ & + P_4\,(R-R_b)^2\,(Z-Z_0) \\ & + P_5\,(R-R_b)^3 \\ & + P_6\,(R-R_b)\,(Z-Z_0)^2 \end{aligned}$$

g. Function 63: vertical Dirichlet boundary

$$\begin{aligned} \mathrm{Fn63} = {} & P_1\,(Z-Z_b) \\ & + P_2\,R\,(Z-Z_b) \\ & + P_3\,R^2\,(Z-Z_b) \end{aligned}$$

h. Function 64: horizontal Neumann and vertical Dirichlet boundaries

$$\begin{aligned} \mathrm{Fn64} = {} & P_1\,(Z-Z_b) \\ & + P_2\,(Z-Z_b)\,(R-R_b)^2 \\ & + P_3\,(Z-Z_b)\,(R-R_b)^3 \end{aligned}$$

i. Function 65: horizontal Dirichlet and vertical Neumann boundaries

$$\begin{aligned} \mathrm{Fn65} = {} & P_1 (R-R_b) \\ & + P_2 (R-R_b)^2 \\ & + P_3 (R-R_b)^3 \\ & + P_4 (R-R_b) (Z-Z_b)^2 \\ & + P_5 (R-R_b)^2 (Z-Z_b)^2 \end{aligned}$$

j. Function 66: near R = 0 and no other boundaries

$$\begin{aligned} \mathrm{Fn66} = {} & P_1 R^2 \\ & + P_2 (Z-Z_0) R^2 \\ & + P_3 R^3 \\ & + P_4 (Z-Z_0)^2 R^2 \end{aligned}$$

k. Function 67: near R = 0 and a vertical Neumann boundary

$$\begin{aligned} \mathrm{Fn67} = {} & P_1 R^2 \\ & + P_2 R^2 (Z-Z_b)^2 \\ & + P_3 R^3 \end{aligned}$$

l. Function 68: near R = 0 and a vertical Dirichlet boundary

$$\begin{aligned} \mathrm{Fn68} = {} & P_1 (Z-Z_b) R^2 \\ & + P_2 (Z-Z_b) R^3 \end{aligned}$$

I. Poisson Superfish program limits

Older versions of Poisson Superfish limited the number of mesh points at compile time by a parameter statement. The codes now allocate memory as needed for arrays that store information about each mesh point. Thus, the number of mesh points is limited only by the computer's memory.

The codes will run most efficiently if the entire problem fits into available RAM. Table III-16 gives the minimum amount of memory used by each program in terms of variables listed in Table III-17. Automesh computes many of the values in Table III-17 from information in the input file. Programs Pandira, Fish, CFish, and the tuning codes allocate arrays of dimension (NROW,NROW) when inverting the tridiagonal matrix. IPIVOT sets the pivoting strategy. The default value is zero or no pivoting. The code does partial pivoting for IPIVOT = 1, and full pivoting for IPIVOT = 2.

Some codes allocate additional arrays that are typically much smaller than those represented in the table. The required arrays lengths increase with the number of regions, the number of points per region, the number of line regions, and the fineness of the mesh. You can list in file OUTAUT.TXT some of the array lengths and their percentage usage for a problem by setting ArrayUsage = Yes in the [Global] section of SF.INI.

Table III-16. Memory requirements of the codes.

Program	Approximate memory in bytes used by program
Automesh	3,973,000 + 64*ITOT + 48*(MXLR+MYLR) + 8*MXLR*MYLR + 128*MBP + 48*MAXREG + 88*MXLOG+ 56*NREG + 72*MAXPPR + 16*MAXMAT + 3312*MAXTABLE
Fish	3,960,000 + 92*ITOT + 16*NROW^2 + 52*NROW + 4*IPIVOT*NROW + 12*NREG + 80*MAXTABLE
CFish	3,960,000 + 156*ITOT + 32*NROW^2 + 100*NROW + 4*IPIVOT*NROW + 12*NREG + 96*MAXTABLE
Poisson	3,956,000 + 92*ITOT + 104*NAMAX + 12*NREG + 176*MAXMAT + 4928*MAXTABLE
Pandira	3,969,000 + 164*ITOT + 16*NROW^2 + 52*NROW +4*IPIVOT*NROW + 8*NINTER + 12*NREG + 176*MAXMAT + 4928*MAXTABLE
Force	3,956,000 + 69*ITOT + 8*NAMAX + 52*MAXPPR + 12*NREG + 176*MAXMAT + 4928*MAXTABLE
SFO	3,952,000 + 52*ITOT + 32*KMAX + 12*NREG + (96*MAXTABLE, Superfish) (176*MAXMAT + 4928*MAXTABLE, Poisson)
SF7	3,862,000 + 52*ITOT + 12*NREG + (96*MAXTABLE, Superfish) (176*MAXMAT + 4928*MAXTABLE, Poisson)
WSFplot	4,076,000 + 68*ITOT + 12*NREG + (96*MAXTABLE, Superfish) (176*MAXMAT + 4928*MAXTABLE, Poisson)
Quikplot	2,990,000 + 28*(MaxDataSets*MaxPoints) + 8*MaxPoints
Tablplot	3,045,000 + 12*(MaxColumns*MaxLines) +248*MaxColumns + 28*MaxLines
Autofish	3,973,000 + allocated arrays*
CCLfish	3,994,000 + allocated arrays*
CDTfish	3,985,000 + allocated arrays*
DTLfish	3,998,000 + allocated arrays*
ELLfish	3,969,000 + allocated arrays*
MDTfish	3,977,000 + allocated arrays*
RFQfish	4,072,000 + allocated arrays*
SCCfish	3,969,000 + allocated arrays*

* The size of allocated arrays for the last several codes is: 102*ITOT +16*NROW^2 + 52*NROW + 48*(MXLR+MYLR) + 8*MXLR*MYLR + 128*MBP + 48*MAXREG + 88*MXLOG + 12*NREG.

J. Temporary data arrays used by Fish, CFish, and Pandira

Depending upon your computer's memory configuration and settings in file SF.INI, programs Fish, CFish, Pandira, and the tuning codes may write a temporary file on one of your hard drives. While solving the tridiagonal matrix problem, these codes store a data matrix for each row, which they later read back in reverse order. If enough memory is available, the codes will allocate memory for the arrays. The codes that allocate memory for temporary data storage are Autofish, CCLfish, ELLfish, CDTfish, DTLfish, MDTfish, RFQfish, SCCfish, Fish, CFish, and Pandira. If there is insufficient memory, these programs create a temporary disk file to store the temporary data arrays. Using actual RAM (and not disk space accessed through a paging or swap file) for these temporary data arrays reduces considerably the computation time for some problems. If a file is necessary, the code tries to make unique scratch-file names, so you can run different problems simultaneously. The only restriction is that you run each problem from a separate directory.

Table III-17. Lengths of some allocated arrays.

Variable	Description
ITOT	Number of mesh points in the problem.
NREG	Total number of regions in the problem.
MAXPPR	Largest number of points along a single boundary region.
MBP	Maximum number of fixed boundary points in the problem.
MAXREG	Maximum number of regions.
MXLOG	Length of internal arrays for interpolated points along boundaries.
NROW	Minimum of LMAX and KMAX.
IPIVOT	Pivoting strategy used to solve the tridiagonal matrix.
NINTER	Number of interface points in Poisson and Pandira problems.
NAMAX	Number of points for which Poisson recalculates couplings.
MAXMAT	Maximum number of materials.
MAXTABLES	Maximum number of material tables.
MaxDataSets	Maximum number of Quikplot data sets in SF.INI.
MaxPoints	Maximum number of points per Quikplot data set in SF.INI.
MaxColumns	Maximum number of Tablplot columns in SF.INI.
MaxLines	Maximum number of lines in the Tablplot Data section in SF.INI.

1. Using a scratch file

The programs create the scratch file on the hard drive with the most free space. By default, Superfish and Pandira solvers search only drive C as specified by the TAPE40 configuration parameter in file SF.INI. You can change or expand the search by editing the SF.INI file. If a code cannot find enough free disk space, it stops with a message telling you how much extra disk space it needs.

There is no practical limit on the size of a scratch file except for available disk space. The file-size limit in the Lahey LF95 Fortran compiler is 18 EBytes (1.8×10^{19} bytes).

2. Forcing the codes to use a scratch file instead of memory

Two settings in file SF.INI affect whether the codes use scratch file or memory for the temporary data. Setting StoreTempDataInRAM = No in SF.INI disables storing temporary data in memory and always uses the TAPE40 scratch file. Another option for controlling how the codes store temporary data is the MaxMemAvailable setting. See the discussion under the SF.INI configuration file for details.

3. Deleting unerased scratch files

The code creates filenames for scratch files from the letter of the current drive and the current directory name. The name should be unique, so you can codes run simultaneously on more than one problem provided you run from different directories. If you start Fish from directory D:\Fish\Project1, the scratch file name would be FIOJECT1.D40. The name is the first 2 and last 6 characters from the directory path (in this case FishProject1), and the extension is the drive letter followed by “40”. (In the older UNIX version of Poisson Superfish, the scratch file is called TAPE40.) Under normal circumstances, each code deletes its scratch file. Even if the code stops abnormally and the scratch file remains on the disk, the next time the code runs from the same directory it will first delete old copies of the file. This scheme is not fool proof, however. You will eventually run out of disk space if you run Fish, CFish, or Pandira from different directories and stop the code abnormally. Examples of an abnormal termination include Fortran run-time errors and operator entries of Ctrl-C or Ctrl-Break. To avoid an abnormal termination, use the Esc key to stop Fish or Pandira in the middle of a calculation.

If you suspect that your disk has unerased scratch files, you should try to get rid of them. Look for files with extension “?40” where the ? character is the DOS wildcard symbol for a single character. The codes create their scratch files in the root directory of the drive with the largest free space.

Temporary data arrays used by Fish, CFish, and Pandira