File Parmela3.DOC contains the following bookmarks. Select the topic you are trying to find from the list and double click the highlighted text.

ADJUSTline
BendsInOtherDirections
CATHODEelement
CFieldLine
CHARGEline
COILdistributedElement
CoupledCavityFields
DCbeams
DocumentationFileList
DRIFTelement
DynamicsCalculation
EGUNprogram
EmittanceOutput
ERRORSline
FieldsAsTextFile
FindingElementNumbers
Format3dTextFiles
FourierCoefficientsForCELLlines
InjectionMethod
INPUTline
ISISprogram
LANLINIsettings
MK_LAINI
NavigationTips
OUTPUTselections
Parmela
ParticleColorsInPargraf
PhaseSpacePlots
PrintingHighlightedMaterial
PrintingRangeOfPages
PrintingTheEntireDocument
QUADelement
ROTATEelement
SampleInputFiles
SAVEcommandFilenames
SCHEFFline
SEXTUPOLEelement
SOLENOIDelement
SpaceChargeNeutralization
SPECIALline
STRIPPERelement
TAPE7contents
TechnicalSupport
BENDelement
BUNCHERelement
CELLelement
CHANGEline
CHOPPERelement
CONTINUEline
CustomDictionaryForWord
DistributedBackgroundFields
DownloadingFilesByFTP
DTCELLelement
EM3DfieldDistributedElement
ElementDisplacement
ENDline
ESQUADelement
FilesForCodeDevelopers
Format2dTextFiles
FourierCoefficientsForTRWAVElines
HardCopyDrivers
INPUTbeamParameters
INPUTfromProgramISIS
ListOfSymbols
MarginRecommendation
MultipleIonSpecies
OUTPUTbuffers
Pargraf
ParmelaHistory
PhaseConventions
PoissonDistributedElement
PrintingOnA4Paper
PrintingTheCurrentPage
ProgramLimits
RFQOUTline
RUNline
SAVEandRESTARTlines
SBLOADline
SegmentedBend
SignConvention
SpaceChargeCalculation
SPCH3Dline
STARTline
TANKelement
TBCIprogram
TemplateFileForWord

ThreeDin
TITLEline
TravelingWaveFields
TRWAVEelement
UpdatesOnInternet
UsingTheEditGoToMenu
UsingTheTableOfContents
VideoMode
WordSettings
ZLimitline
TimeStepEmittance
TravelingWaveExample
TRWCFieldline
TRWAVEentryCell
UpdatingTheTableOfContents
UsingTheHypertextLinks
UtilityPrograms
WIGGLERelement
WordViewerSettings
ZOUTline

LA-UR-96-1835
Revised May 23, 2005

Parmela

Lloyd M. Young

Documentation by

James H. Billen

Los Alamos
National Laboratory

Parmela

LA-UR-96-1835

Copyright and disclaimer

Copyright, 1985-2005, The Regents of the University of California.

This software was produced under U. S. Government contract W-7405-ENG-36 by Los Alamos National Laboratory, which is operated by the University of California for the U. S. Department of Energy. Neither the Government nor the University makes any warranty, express or implied, or assumes any liability or responsibility for the use of this software.

Acknowledgments

The authors of Parmela version 3 wishes to acknowledge the extremely important contribution of Kenneth R. Crandall, who wrote the original version of Parmela in 1980. He introduced the use of time (or phase angle) as the independent variable, which is retained in the present version. Even after retiring from Los Alamos National Laboratory in the mid 1980s, Ken Crandall contributed to the further development of Parmela. Working as a consultant, he wrote new sections of the code for a traveling-wave accelerator tank and a wiggler element.

The author is also grateful to the many users of these codes who have pointed out bugs and made suggestions for additional features in this version.

This documentation was produced by the Los Alamos Accelerator Code Group (LAACG) at Los Alamos National Laboratory. The activities of the LAACG are supported by the US Department of Energy, Office of Energy Research, through the Division of High Energy Physics.

Original release: February 10, 1996
Revision date (version 3.39): May 23, 2005

Table of Contents

I. How To Use This Document 1

A. Documentation files 1

B. Suggested settings for viewing and printing 1

1. Microsoft Word template and dictionary files 2
2. Settings in Microsoft Word 2
3. Settings in Microsoft Word Viewer 2

C. Using the table of contents 3

D. Using the hypertext links 3

E. Using the Edit, Go To… menu 3

F. Printing sections of this documentation 4

1. Printing the entire document 4
2. Printing the current page 4
3. Printing a range of pages 4
4. Printing highlighted material 4

G. Updating the table of contents 5

II. The Parmela Software Installation 6

A. The Lahey/Fujitsu Fortran compiler and *Winteracter* library 6

B. Software installation 7

1. Installation disk drive and directory structure 7
2. LANL, Root directory for Los Alamos computer codes 7
3. Documentation files 8
4. Example files for Parmela 8
5. Tablplot program 8
6. Files provided for program developers 8

C. Technical support and code updates 8

1. Electronic mail support 9
2. Email discussion group 9
3. Obtaining updated installation files 9
4. Email notification of new features and bug fixes 10

D. Configuring Parmela using settings in LANL.INI 10

1. Filename for saved coordinates 15
2. EgunDataFile, particle distribution file from program EGUN 15
3. DioutFile and DpoutFile, files related to program ISIS data 16
4. Black or white background color in plotting codes 16

III. Parmela, an electron linac design code 16

A. History of the development of Parmela 16

B. Other Los Alamos versions of Parmela 17

C. Description of the Parmela program 18
1. Input data 18
2. Background static fields from distributed elements 18
3. User-supplied field maps 19
a. Text-file format for 3-D field maps 19
b. Text-file format for 2-D field maps 21
4. Sign convention for charged particles 22
5. Particle coordinates, good particles and lost particles 23
6. Starting the dynamics calculation and the master clock 24
7. Phase conventions and phase settings in Parmela 25
8. Particle dynamics routine in Parmela 25
9. Parmela output buffers 26
10. Program limits in Parmela 26
11. Simulating DC beams in Parmela 27

IV. Running Parmela 28

A. Starting program Parmela 28
1. Running Parmela from batch files 28

B. Input-file keywords for Parmela 30
1. RUN, the first line in the Parmela input file 31
2. TITLE, defines a title line 32
3. CATHODE, a spherical cathode 32
4. DRIFT, a drift space 32
5. SOLENOID, a solenoid lens 33
6. QUAD, a magnetic quadrupole lens 33
7. ESQUAD, an electrostatic quadrupole lens 33
8. STEERER, a magnetic steering element 34
9. BEND, a dipole magnet with bend in the x direction 34
a. Simulating bends in other directions 37
b. Making a segmented bend 37
10. BUNCHER, for a buncher cavity 38
11. CHOPPER, a beam chopper 39
12. STRIPPER, a beam stripper 40
a. Simulating a charge-state distribution 40
13. CELL, an rf cavity or cell 40
14. CELL2, an rf cavity excited at two frequencies simultaneously 42
15. DTCELL, an rf gap with quadrupole magnets in the drift tubes. 43
16. CField, reads rf field data 45
17. TANK, an accelerator tank 47
18. ROTATE, rotates coordinates about the beam axis 48
19. SBLOAD, specifies single-bunch beam loading 48
20. TRWAVE, a traveling-wave accelerator 49
a. Generating output files for program Tablplot 51
b. Simulating the fringe field in the first TRWAVE cell 51
c. Simulating the fringe field in the last TRWAVE cell 51
21. TRWCField, reads Superfish field data for traveling wave cells 51
22. WIGGLER, a free-electron-laser wiggler 53
23. SEXTUPOLE, a magnetic sextupole lens 55
24. RFQOUT, writes particle coordinates to files 55

25. ZOUT, writes element data 56
26. COIL, a background solenoidal magnetic field 57
27. Poisson, background fields from program Poisson 57
28. EM3DField, 3D background fields. 59
29. CHARGE, specifies the charge and particle type 60
30. INPUT specifies the beam parameters 60
a. Type 0, data from program ISIS 61
b. Type 1 or 10, random distribution in three phase-plane ellipses 62
c. Type 2 or 20, random distribution in six-dimensional ellipse 62
d. Type 3, generated from program EGUN data 63
e. Type 4 and 40, locates individual particles at six specified coordinates 63
f. Type 5, 6, 50, and 60, random in 4-D transverse hyperspace 64
g. Type 7, Kapchinskiy-Vladimirskiy distribution 64
h. Type 8, uniform spatial distribution 64
i. Type 9, Gaussian or uniform distribution with no energy spread 64
j. Type 30, generated from program PARMILA or PARMTEQ data 66
31. OUTPUT, selects the type of output 66
a. OutType 1, write all particle coordinates 67
b. OutType 2 and 3, old text-based plots 68
c. OutType 4, write phase and energy data 68
d. OutType 5, Pargraf plot file 68
32. ERRORS, specifies alignment errors 68
a. ErrorType 1, specific displacements at each end of the element 69
b. ErrorType 2, random displacements in x and y of the entire element 69
c. ErrorType 3, random displacements in x and y at the beginning of an element 69
d. ErrorType 4, random displacements in x and y at both ends 69
e. ErrorType 5, random radial displacement of the entire element. 69
f. ErrorType 6, random radial displacement at the end of the element 70
g. ErrorType 7, random radial displacements at both ends 70
h. ErrorType 8, specific displacement and angle errors at start of elements 70
i. ErrorType 9, random radial and angular displacements at start of elements 70
33. SCHEFF, defines parameters for space-charge calculations 70
34. SPCH3D, 3-D space-charge calculation 72
35. CHANGE, changes one parameter of an element 74
36. ADJUST, scales parameters of several elements 74
37. START, starts the dynamics calculation 75
38. SAVE and RESTART, starting with saved particle coordinates 75
a. Filenames for saved particle coordinates 76
39. CONTINUE, starts again after changing parameters 76
40. ZLimit, drop particles based upon position 76
41. SPECIAL, defines special-purpose parameters 76
a. Defining the rest frame and position of the space-charge mesh 76
b. Restricting particles in the output buffers 77
c. Control of space-charge neutralization in the beam 77
d. Time adjustment for different mass particles in a DC beam 78
42. END, stops the Parmela run 79

C. Symbols for physical and logical variables 79

V. Pargraf, the Plotting Program for Parmela 85

A. Starting program Pargraf 85
1. Hardcopy of graphics screens 86

a. BMP (Windows Bitmap Format) 86
b. CGM (Computer Graphics Metafile) 86
c. DXF (AutoCAD Drawing Exchange Format) 87
d. PS (PostScript) and EPS (Encapsulated PostScript) 87
e. HP-GL and HP-GL/2 (Hewlett Packard Graphics Language) 87
f. PCX (ZSoft PC Paintbrush Format) 88
g. PNG (Portable Network Graphics) 88
h. SVG (Scalable Vector Graphics) 88
i. WMF (Windows Metafile) and EMF (Enhanced Windows Metafile) 88

B. Files used by Pargraf 89

C. Parmela graphics output 89
1. SUBNUM 3, four phase-space plots 90
2. SUBNUM 5, x and r longitudinal profiles 91
3. SUBNUM 6, input and output phase-space projections 92
4. SUBNUM 7, variety of phase-space and spectrum plots 93
5. SUBNUM 8, variety of longitudinal profiles 95
6. SUBNUM 9, three longitudinal profiles of x, $\Delta\phi$, and ΔW 96

D. Emittance definitions in the Pargraf output text file 96

VI. Utility Programs for Use with Parmela 98

A. EFLD, Fourier coefficients for CELL and DTCELL lines 98

B. EFLDTR, Fourier coefficients for TRWAVE lines 100

C. ThreeDin, writes 3-D field data readable by CField line 102

D. MK_LAINI: Create or update file LANL.INI. 102

VII. Sample Input Files 104

A. Parmela and Pargraf example files 104
1. The DEMO directory 104
2. The ATF directory 108
3. The EPMIX directory 109
4. The TRWAVE directory 110

B. Creating field input for Parmela 112

C. Generating Fourier coefficients for Parmela 113

VIII. Information about other codes 114

A. ISIS, particle-in-cell code 114

B. EGUN, electron-gun code 114

C. TBCI, wake-field interactions with beam bunches 115

List of Tables

Table II-1. Disk directories. 7
Table II-2. LANL.INI settings for Parmela. 11
Table II-3. LANL.INI settings for Pargraf. 14
Table II-4. Filenames for different SAVE commands. 15
Table IV-1. Quadrupole magnet options for DTCELL elements. 45
Table IV-2. Contents of the SF7 output file for Parmela. 47
Table IV-3. Contents of file named by ElementOutName. 67
Table IV-4. Contents of file named by TimeStepOutName. 67
Table IV-5. Definition of physical and logical variables. 80
Table V-1. The Pargraf program menu. 85
Table V-2. HP-GL Color Translation. 88
Table V-3. LANL.INI settings for input and output files. 89
Table V-4. Particle colors in Pargraf displays. 90
Table V-5. Column heading in the time-step emittance table. 92
Table V-6. Choices for *Graphtype* 7 plots. 93
Table V-7. Description of headings for emittance tables. 95
Table V-8. Choices for *Graphtype* 8 plots. 96
Table VI-1. Program prompts in EFLDTR. 101
Table VI-2. Combinations of N_L and N_U recommended for Parmela. 102
Table VII-1. Contents of the LANL\Examples\Parmela directories. 104
Table VII-2. Files produced by running RUN_INP.BAT. 105
Table VII-3. Files in the ATF directory. 109
Table VII-4. Files in the EPMIX directory. 110
Table VII-5. Files in the TRWAVE directory. 110
Table VII-6. Files in the Solenoid directory 112
Table VII-7. Files in directory Fourier.C. 113
Table VII-8. Files in directory Fourier.TR. 113

List of Figures

Figure IV-1. Example file Job1.INI _____ 29
Figure IV-2. Batch file Run1.Bat using multiple LANL.INI files. _____ 29
Figure IV-3. Batch file Run2.Bat using multiple directories and one LANL.INI file. _____ 30
Figure IV-4. Geometrical parameters for the BEND element._____ 36
Figure IV-5. Multiple BEND lines for producing output through a bend._____ 38
Figure IV-6. DTL cell layouts between two drift tubes. _____ 43
Figure IV-7. Example input file lines for the first few cells of a DTL. _____ 45
Figure V-1. Start of the time-step emittance table. _____ 92
Figure V-2. Table heading for transverse emittance data. _____ 94
Figure V-3 Table heading for longitudinal emittance data. _____ 94
Figure VI-1. Coupled-cavity linac cell. _____ 99
Figure VI-2. Traveling-wave accelerator. _____ 101
Figure VII-1a. First part of the Parmela input file INPUT.ACC. _____ 106
Figure VII-2a. First part of Pargraf control file SIMPLE.PGF._____ 107
Figure VII-3. The Parmela input file ATFNM.ACC._____ 109
Figure VII-4. The Parmela input file TRAVELWV.ACC. _____ 112

I. How To Use This Document

This section provides some tips on navigating this documentation when viewing it on-line and on printing either the entire document or various sections of it.

This document was produced using Microsoft Word, Version 7 and the Internet Assistant for Microsoft Word. You can read and print it with Word 6, Word 7, and Microsoft Word Viewer. There are several ways to move from place to place without scrolling through entire files. Some navigation features will be unavailable unless you use Microsoft Word and you have installed the Internet Assistant, which is available free from Microsoft. Here are some recommended settings and procedures.

A. Documentation files

This file, LANL\Docs\Parmela3.DOC, contains all the documentation for the electron-linac design code Parmela, its plotting program Pargraf, and several other programs. We recommend that you also install Poisson Superfish, which generates some input files for Parmela. The Poisson Superfish documentation (LA-UR-96-1834) also will be found in the LANL\Docs directory.

B. Suggested settings for viewing and printing

Here are some suggested settings for viewing this document on line. There are some differences in applying these settings in Word and Word Viewer. Refer to the separate sections below for the application you are using.

We recommend using Normal view with the window maximized to fill the screen. Using Normal view (rather than Page Layout view) speeds up most operations in a large document. This is not a recommendation to use the Full Screen setting in the View menu since you will most likely want the use of the scroll bars. To switch to Normal View, check Normal in the View menu or click the leftmost button next to the bottom scroll bar.

Use the Zoom command in the View menu to make the text comfortable to read on your monitor. In Word, you also can use the Percent box on the Standard toolbar. Type a number between 10 and 200 in the Percent box to choose a size other than the standard selections. Using the Wrap to Window setting described below in combination with a size a little larger than 100% makes fairly readable text.

For printing, we recommend using the default margin settings of 1.00 inch top and bottom and 1.25 inches left and right. To change or check the margin settings, click File, Page Setup… and go to the Margins tab. We also recommend, if available, using a Hewlett Packard or compatible laser-jet printer. If this is not possible, then before printing anything, you may want to update the table of contents.

If you will be printing on A4 paper, you may want to use the margin settings 1.135 inches (28.8 mm) left and right, 1.00 inch (25.4 mm) top, and 1.69 inches (42.9 mm) bottom. These settings use the same “footprint” as the default settings for US letter paper. The pagination in the document should agree with the table of contents as distributed for most

printers with 600-dot-per-inch resolution. Furthermore, tables that span more than one page should print correctly with new headings and titles at the top of continuation pages. Tables and figures that use nearly all of the 6-inch width between left and right margins should not generate error messages about margins from the printer.

1. Microsoft Word template and dictionary files

When displaying Parmela documentation and other Los Alamos software documentation, Microsoft Word uses two files in the LANL directory. The file LANLHELP.DOT is a Word template file containing styles, macros, and other settings. We use macros in LANLHELP.DOT when editing the documentation, but users reading the documentation do not need them.

The file LANLHELP.DIC is a custom dictionary file containing a list of terms that Word would ordinarily flag as misspelled words. This file contains proper names, accelerator jargon, program keywords, LANL.INI settings, and mathematical variable names used in the Los Alamos documentation. If in Word 7 you see a large number of words underlined in red, then Word is probably not using the custom dictionary LANLHELP.DIC. You can add this file to the list of custom dictionary files in the Tools, Options, Spelling tab. We also recommend checking the boxes to ignore words in upper case and words with numbers.

2. Settings in Microsoft Word

You need only the Standard toolbar when viewing this document. To remove the other toolbars, leaving more room for text, click View, Toolbars… and remove the check marks on all but Standard, then click OK.

Click Tools, Options… and check the following settings. There are several items to inspect on the View tab. Under Nonprinting Characters, deselect all of the options (that is, remove the check marks). Under Show, select Wrap to Window, but deselect Draft Font, Picture Placeholders, Field Codes, and Bookmarks. Choose Never in the Field Shading box. On the Print tab, under Include with Document, be sure there is a check mark next to Drawing Objects, otherwise figures will not print. On the Save tab, you may wish to deselect Allow Fast Saves and Automatic Save Every xx Minutes.

You can toggle the table grid lines on and off by checking or unchecking Gridlines in the Table menu. This setting has no effect on printing. Tables in this document do not contain cell borders.

3. Settings in Microsoft Word Viewer

Click View, Options… and check the following settings. There are several items to inspect on the View tab. Under Nonprinting Characters, deselect all of the options (that is, remove the check marks). Under Show, select Wrap to Window, but deselect Draft Font, Picture Placeholders, Field Codes, and Bookmarks. Choose Never in the Field Shading box. Checking Table Gridlines displays a grid in tables, but has no effect on printing. Tables in this document do not contain cell borders.

On the Print tab, under Include with Document, be sure there is a check mark next to Drawing Objects, otherwise figures will not print.

C. Using the table of contents

In this document, you can double click a page number in the table of contents to jump to that page in the file. In Word Viewer, use a single click. This feature is also available in each individual document files for Poisson Superfish. However, it is not available in the Poisson Superfish master table of contents in file SFTOC.DOC because of the method used to create the master table of contents.

D. Using the hypertext links

As you read this document on line, you will notice that some text is highlighted in blue. This highlighted text is a hypertext link (or hyperlink) to a location in this documentation that contains more detailed information of the subject. There are two type of links. The links within the same file use a GOTOBUTTON field and are not underlined. A link to another file uses a hyperlink inserted with the Internet Assistant for Microsoft Word. The links that refer to another file are underlined. Links to other files may not work in some older version of Word Viewer. They work in Microsoft Word if you have installed the Internet Assistant, which is available free from Microsoft.

Double-click in Word, or single-click in Word Viewer, in the blue text area to jump to that location. If the section you are jumping to is later in the same document, then the cursor will usually appear at the bottom of the screen. You will need to scroll up a few lines to see the material you are interested in. When jumping to a location earlier in the document, the cursor appears at the top of the screen.

We have included at the start of each file a list of all the bookmarks within the file. You can get to this list by pressing Ctrl-Home to move to the top of the file. Select the topic you are trying to find and click in the highlighted text.

E. Using the Edit, Go To… menu

You can use Go To… in the Edit menu to jump to a specific page number or move forward or backward by some number of pages. Use this method when viewing the table of contents. Bring up the Go To… menu and select Page in the Go To What list on the left. Enter the page number in the text window, then press Go To button on the right.

Another way to use the Go To… dialog box is to jump to a bookmark in the text. The bookmarks correspond to the hypertext links. For convenience, the bookmarks have long, descriptive names. Scroll through the list, which is sorted alphabetically and when the desired topic appears in the Enter Bookmark Name window, press the Go To button. You can also type in the name of a bookmark if you remember it from past use. Here are some shortcut methods for bringing up the Go To… dialog box:

- Hold down the Ctrl key and then press G, or
- Press the F5 key.

F. Printing sections of this documentation

You can print this entire document and then rely on the printed copy for help. Some people may prefer this method, but others might find it more efficient to view the document on-line. The navigation tools (the table of contents, the hypertext links, and the Edit, Go To dialog box) will help you find what you need very quickly compared to leafing through a printed copy.

Please note that the documentation changes from time to time as we add new features or fix bugs. Even in the absence of code changes, we will probably continue to update this document for some time after introducing this new format in February, 1996.

There are several ways to print material from this document. You can print the entire document, the current page, a range of pages, or material that you highlight by holding down the mouse button and dragging over the text. First, decide on the method you will use, then follow the procedure outlined in the following subsections.

1. Printing the entire document

If three-hole paper is available, we recommend using it to save time punching holes. To print the entire document, Select Print in the File menu, then in the Page range section check the All setting, and click OK. If you use the recommended settings for the margin in Word or Word Viewer, then page numbering should be consistent with the master table of contents. (If not, you may need to update the table of contents.)

2. Printing the current page

To print just the page you are viewing, click anywhere in the text to be sure the cursor is in the page you are viewing. Select Print in the File menu, then in the Page range section check the Current setting, and click OK.

3. Printing a range of pages

To print a range of pages, use the status bar at the bottom of the screen to identify the page numbers of the starting and ending page you want to print. Also note the section that the pages are in. For example, this paragraph should be in Sec 3. Select Print in the File menu, then in the Page range section click in the data entry window next to Pages. Type “p” and the beginning page number followed immediately by “s” and the section number, then a dash and the ending page and section numbers (for example, p26s4-29s4). Click OK. The Pages selection will be chosen as soon as you start typing the page numbers.

4. Printing highlighted material

In the Print dialog box, printing a Selection refers to printing highlighted material. First, while holding down the left mouse button, drag the mouse over the material you wish to print. Release the mouse button. Select Print in the File menu, then in the Page range section check the Selection setting, and click OK.

G. Updating the table of contents

Your printer driver and default printer setup affects the pagination in a document. We recommend using the page layout settings discussed above and a Hewlett Packard or compatible laser-jet printer. If you notice that page numbers in the table of contents do not agree with page numbers you have printed, you can update the table of contents in this file. However, this process requires Microsoft Word. You cannot edit the document using Word Viewer.

Make sure your default printer is set to the one you will use to print the documentation files. When the default printer is set properly, place the cursor anywhere in the table contents in this file, press the right mouse button and select Update Field. In the dialog window that appears select Update Page Numbers Only. Repeat the Update Field operation for the lists of tables and figures that follow the table of contents. If the dialog window appears for these two updates, select Update Page Numbers Only.

II. The Parmela Software Installation

Parmela is an electron-linac particle-dynamics code. The name comes from the phrase, "Phase and Radial Motion in Electron Linear Accelerators." It is a versatile multi-particle code that transforms the beam, represented by a collection of particles, through a user-specified linac and/or transport system. It includes options for 2-D and 3-D space-charge calculations. Parmela integrates the particle trajectories through the fields. This approach is especially important for electrons where some of the approximations used by other codes (e.g. the "drift-kick" method commonly used for low-energy protons) would not hold. Parmela works equally well for electrons, positrons, ions, or mixtures of different species. Parmela can read field distributions generated by either Fish for rf problems or Poisson for magnet problems and electrostatic problems. This version of Parmela was written and is supported by Lloyd M. Young, Los Alamos National Laboratory (retired). James H. Billen maintains the documentation.

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

Parmela 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 plotting code Pargraf. The resulting programs run under Windows 95/98, 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.

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/

B. Software installation

The software installation for Parmela is similar to the installation of the Poisson Superfish codes, which you can download by FTP and install separately. Setup installs the executable programs Parmela.EXE, Pargraf.EXE, EFLD.EXE, EFLDTR.EXE, and Tablplot.EXE in the LANL directory. The directory LANL\Examples\Parmela contains several subdirectories with sample files for Parmela. If you have installed Poisson Superfish, which we recommend, please refer to Poisson Superfish software installation in file SFINTRO.DOC for more information.

1. Installation disk drive and directory structure

You can install the software on any other hard drive with enough free space. The Setup program recommends that you install all Los Alamos software in the same directory. In this documentation, we will assume that the installation directory is C:\LANL. The installation procedure creates the directories listed in Table II-1 on the hard drive of your choice.

Table II-1. Disk directories.

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

2. LANL, Root directory for Los Alamos computer codes

LANL is an abbreviation of Los Alamos National Laboratory. The installation program creates this directory and some subdirectories. The LANL directory contains the READ.ME file, CHANGES files and executable files. Other directories contain sample input files for running the codes. To run the Los Alamos codes, the installation directory must be in the PATH.

3. Documentation files

LANL\Docs is one of several subdirectories created by the installation procedure. This directory includes documentation files for all Los Alamos codes, including this file (Parmela3.Doc).

4. Example files for Parmela

LANL\Examples is one of several subdirectories created by the installation procedure. This directory includes sample files for all Los Alamos codes. The example files for Parmela are found in subdirectories of LANL\Examples\Parmela. For more information about the Parmela examples refer to Chapter VII.

5. Tablplot program

The ZOUT line in Parmela writes input files for program Tablplot, a general purpose plotting code from the Poisson Superfish distribution. The Parmela installation includes the executable code Tablplot.EXE, but not its documentation. For documentation of Tablplot, you need please refer to file SFCODES3.DOC after installing Poisson Superfish.

6. Files provided for program developers

Subdirectory LANL\DeveloperFiles\Parmela contains Fortran source files for reading two binary output files written by Parmela. These source files are for program developers who wish to write their own postprocessors. They are provided as is without additional documentation except that provided in the files themselves. Los Alamos National Laboratory cannot provide consulting for source-code developers. File RDTAPE2.FOR is the Fortran source code for a subroutine that reads files TAPE2.T2 or TAPE3.T3. EXAMPLE.FOR is a short program that demonstrates the calling procedure for RDTAPE2. In order to be compatible with binary files written by Parmela, the source files must be compiled with Lahey/Fujitsu Fortran LF95, version 5.7 or higher.

Files TAPE2.T2 and TAPE3.T3 are binary files when the Parmela input file contains the line “OUTPUT 5.” Both of these files are text files if the input file contains “OUTPUT 1.” The program RDTAPE2 is intended for use with the binary files.

Program ThreeDin reads a file of 3-D rf fields generated by MatLab and writes a binary file readable by the CField line in Parmela. File ThreeDin.FOR in directory LANL\DeveloperFiles\Parmela is the self-documented source code for this utility program. The user can modify the code to read 3-D fields in other formats as needed.

C. Technical support and code updates

The code’s author, Lloyd M. Young, maintains the Parmela and Pargraf 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 Parmela users and 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 on Internet.

1. Electronic mail support

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. For technical assistance with Parmela or Pargraf, please send email to <laacg@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 Parmela with other code users can subscribe to the mailing list Parmela_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 Parmela may subscribe. The character string "[PM]" appears in the subject line of messages posted to Parmela_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 Parmela_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)

You should connect using the account name and password assigned to you at the time you obtained your original code distribution and update the entire distribution. The passwords are case sensitive. 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://UserName:PassWord@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). The _READ_ME.1ST file on the FTP server has for more information about downloading files.

We recommend that you also install the Poisson Superfish codes. These codes can create input files for Parmela. Your FTP account will have access to directory PoissonSuperfish\Version7, which contains the latest version of Poisson Superfish.

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. Do not mix the installation files with those from a previous distribution.

4. Email notification of new features and bug fixes

We maintain an electronic mailing list of registered users of Parmela. Registered users receive notification of new features and bug fixes by email. If you obtained your copy of the codes from another Parmela user at your institution, please register with us as a new user be sending an email message to <laacg@lanl.gov>.

A separate mailing list notifies users of Poisson Superfish of changes and bug fixes in these codes. To register as a Poisson Superfish user, please include your postal mailing address, phone number, and fax number in an email message to Superfish@lanl.gov.

D. Configuring Parmela using settings in LANL.INI

Los Alamos programs, including Parmela and Pargraf, use some configuration options in file LANL.INI. The installation program put a copy of LANL.INI containing suggested settings in your LANL directory. If Setup found an existing copy of LANL.INI in the destination directory from a previous installation, it saved your settings and added any new or missing keywords. You can regenerate file LANL.INI, restoring all the default settings, by deleting or renaming the current file and then running program MK_LAINI.

All codes read the [Global] section first, then a section named after the code (e.g. [Parmela], [Pargraf], etc.). 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. The order of the sections in LANL.INI does not matter. Table II-2 lists LANL.INI settings for Parmela and Table II-3 lists settings for Pargraf.

Table II-2. LANL.INI settings for Parmela.

LANL.INI setting	Description	Default value
ParmelaIn	Parmela input file	INPUT.ACC
DosErrorLevel	If Yes (or True), suppress the pop-up stop message when Parmela ends. For normal termination Parmela sets the DOS error level to zero. If an error has occurred the error level is nonzero.	No
CellDataFile	File containing accelerating fields in each cell.	none
ParmelaOutFile	Parmela output file [PRN for direct to printer].	OUTPAR.TXT
SaveCoreRoot	Root filename for files of all coordinates, produced by SAVE commands for later RESTART commands.	SAVECOR

Table II-2. LANL.INI settings for Parmela. (continued)

LANL.INI setting	Description	Default value
RFfldFile	Base filename for Tablplot files containing rf fields at specified locations, produced by the ZOUT line. The code uses up to the first 5 characters supplied.	RFFLD
BfieldOutFile	Data file containing the magnetic field versus longitudinal position, produced by ZOUT commands.	none
DioutFile	Output file from the program ISIS.	DIOUT
ParmelaIn	Parmela input file	INPUT.ACC
DosErrorLevel	If Yes (or True), suppress the pop-up stop message when Parmela ends. For normal termination Parmela sets the DOS error level to zero. If an error has occurred the error level is nonzero.	No
CellDataFile	File containing accelerating fields in each cell.	none
ParmelaOutFile	Parmela output file [PRN for direct to printer].	OUTPAR.TXT
SaveCoreRoot	Root filename for files of all coordinates, produced by SAVE commands for later RESTART commands.	SAVECOR
RFfldFile	Base filename for Tablplot files containing rf fields at specified locations, produced by the ZOUT line. The code uses up to the first 5 characters supplied.	RFFLD
BfieldOutFile	Data file containing the magnetic field versus longitudinal position, produced by ZOUT commands.	none
DioutFile	Output file from the program ISIS.	DIOUT
DpoutFile	Parmela-generated distribution derived from weighted data in the DioutFile.	DPOUT
BeamLoss	Filename for beam loss data.	none
BeamLoad	Filename for beam loading data.	none
EgunDataFile	Particle distribution file from program EGUN.	EGUN.DAT
CfieldPath	Path to files containing fields on a rectangular grid for CField commands and to Poisson data files.	blank
ScGridFile	File for saving space-charge grids.	SCGRID
ElementOutName	Binary or text output file containing coordinates of particles at the end of beam-line elements. (Name the file with extension .TXT when using OUTPUT 1 instead of OUTPUT 5.)	TAPE2.T2
TimeStepOutName	Binary or text output file containing coordinates of particles at the end of specified time steps. (Name the file with extension .TXT when using OUTPUT 1 instead of OUTPUT 5.)	TAPE3.T3
TimeStepEmittance	Filename for the Tablplot file containing time-step-emittance and beam-size data.	none
Part_In_Dst	Name of a particle distribution file generated by programs Parmila or Parmteq. In the form Part_In_Dst =@filename, filename is the name of a file that lists multiple particle-distribution files. If the extension is .TXT the file is a text file, otherwise it is an unformatted binary file.	PART_RFQ.DST
Part_Out_Dst	Name of a particle distribution file generated by the RFQOUT line. If the extension is .TXT the file is a text file, otherwise it is an unformatted binary file, of the type determined by LANL.INI keyword LaheyLF90DstFile.	RFQ_IN.DST

Table II-2. LANL.INI settings for Parmela. (continued)

LANL.INI setting	Description	Default value
LaheyLF90DstFile	If LaheyLF90DstFile = Yes, then the RFQOUT line in the Parmela input file causes the code to write a distribution file (see Part_Out_Dst) compatible with program Parmteqm and Parmila, which are Lahey Fortran 90 (LF90) programs. If this keyword is No, then the Part_Out_Dst file uses the Lahey/Fujitsu Fortran 95 (LF95) format for binary files.	Yes
RotatingFrame	A charged particle gains angular momentum when it enters a solenoidal magnetic field. When RotatingFrame = Yes, Parmela writes particle coordinates to the output file in a rotating frame of reference that removes this angular momentum.	Yes
3dTextFile	If Yes (or True), Parmela opens all files that appear after EM3DField or CField lines as text files regardless of the filename extension. Otherwise, the code only opens files with extension TXT as text files, and assumes any other file is a binary file.	No
ScreenOutput	If No (or False), Parmela does not write anything to the console screen. Using this feature may speed up execution, especially if there are a large number of lost particles. The danger is that error and warning messages that would only appear on the screen will not be seen.	Yes
Spch3dArraySize	Limits the size of the 3-D space-charge arrays, if possible. For example, the default setting means the mesh size is limited to 16x16x2048, or 8x8x8192, etc. When the number of mesh intervals increases in the z direction, and the array would exceed the size specified by Spch3dArraySize, then the code reduces the number of mesh intervals in the x and y directions, but not less than 8 in each direction.	524288
GaussianSize	For the INPUT 6 Gaussian distribution, if GaussianSize = 2, the RMS beam size is σ. If GaussianSize = 2, the RMS beam size is $\sigma/2^{0.5}$. For compatibility with version 3.38 and earlier, set this value to 2. (The only permitted value is 1 or 2.)	1

You can have multiple copies of LANL.INI with different configurations. Each code first looks for LANL.INI in the current directory. Next, it checks the environment variable LANLINI for the directory containing LANL.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 ParmelaDir). The first file found in this search is the one the program will use. If the code cannot find a copy of LANL.INI, then it uses default settings listed in Table II-2 for Parmela and Table II-3 for Pargraf. The LANL.INI file can include comments, which start with semicolons or exclamations marks. The code ignores everything to the right of comment indicator.

Most of the LANL.INI settings for Parmela control the names used for input and output files. In some cases you can specify a filename and path of your choice. In other cases you cannot change the filename, but you can change the path to the file. For file settings, if the

default is “none,” then Parmela does not create the file unless a filename appears in the INI file.

Table II-3. LANL.INI settings for Pargraf.

LANL.INI setting	Description	Default value
ElementOutName	Binary output file written by Parmela (selected by OUTPUT 5) containing coordinates of particles at the end of beam-line elements.	TAPE2.T2
TimeStepOutName	Binary output file written by Parmela (selected by OUTPUT 5) containing coordinates of particles at the end of specified time steps.	TAPE3.T3
PargrafControl	Input control file specifying plot options.	SIMPLE.PGF
PargrafOut	Pargraf output text file [PRN for direct to printer]	OUTGRAF.TXT
PromptForOptions	If yes, Pargraf will wait for menu commands or keyboard input. If No, Pargraf will start in movie mode. The Esc key will put Pargraf into normal wait for keyboard input between screens.	Yes
Graphics	If Off, skip displays, but generate output text file. Turning Graphics off allows batch files to run in the background.	On
Slice	If Yes, Pargraf displays slice emittance plots.	No
BG_Color	Background color (either White or Black).	Black
MaxColors	Number of colors showing particle densities on scatter plots (maximum available is 14).	5
NumPartThreshold	Number of macroparticles in the calculation above which particle densities are displayed in colors.	5000
EmittPer	Controls the percentage of particles included in one of the emittance ellipses printed in the Pargraf output text file for the SUBNUM 7 option. The allowed range is from 10.0% to 99.9%.	90.0

```
; Start of sample LANL.INI file.
;
[Global]
BG_COLOR=black              ! Sets background screen color
;HPGLLineWidth=.15
;PostScriptLineWidth=.15

[Parmela]
; File names including the path are limited to 64 characters.
ParmelaIn=Input.Acc         ! Default input filename
CellDataFile=None           ! None means don't output the cell data.
ParmelaOutFile=OUTPAR.TXT
SaveCoreRoot=SAVECOR        ! Root name for files storing coordinates arrays
RFfieldFile=RFfld           ! First 5 characters of a series of Tablplot files.
BfieldOutFile=None          ! None means don't output the Bfield data.
DosErrorLevel=Yes           ! If yes, allows running from a batch file, suppresses Stop message
DioutFile=DIOUT             ! Output file from the program ISIS
DpoutFile=DPOUT             ! Distribution generated from weighted data in the Diout file
;BeamLoss=None              ! Default is None for no beam-loss file
;BeamLoad=None              ! Default is None for no beam-loading file
```

```
EgunDataFile=EGUN.DAT
3dTextFile=No              ! If yes, files after CField and EM3DField lines are text files.
; CfieldPath is also the path to the TRWCField and Poisson data files.
; The path must include the trailing blackslash (\).
CfieldPath=C:\LANL\Examples\Parmela\
ScGridFile=SCGRID
; When using OUTPUT 5 in Parmela, use the same settings in Parmela
; and Pargraf for variable ElementOutName and for TimeStepOutName.
; When using OUTPUT 1, name the files with extension .TXT.
ElementOutName=TAPE2.T2
TimeStepOutName=TAPE3.T3
; In the rotating frame of reference (the default),
; the angular velocity depends on the axial B field.
GaussianSize=1             ! For INPUT 6, 1: RMS = sigma; 2: RMS = 0.707*sigma (thru v3.38).
RotatingFrame=On           ! Off means do not view output beam in rotating frame

[Pargraf]
BG_COLOR=white             ! Sets background screen color in PARGRAF
ElementOutName=TAPE2.T2
TimeStepOutName=TAPE3.T3
Slice=No                   ! If Yes, Pargraf displays slice emittance plots.
MaxColors=5                ! Number of particle-density colors on scatter plots (max = 14)
NumPartThreshold=5000      ! Minimum number of particles for particle density  colors.
EmittPer=99                ! Sets contour percentage level for emittance output.
PargrafControl=SIMPLE.PGF
PargrafOut=OUTGRAF.TXT
PromptForOptions=Yes       ! Puts PARGRAF in movie mode if No.
; End of sample LANL.INI file.
```

1. Filename for saved coordinates

SaveCoreRoot in LANL.INI specifies the root filename for files of all coordinates produced by SAVE commands for later use with the RESTART command. The default root name is SAVECOR. By root name we mean that it may be only part of the filename. When Parmela executes a SAVE command with the optional *SaveNumber* parameter, it appends a letter to the root name. *SaveNumber* is a number from 1 to 26 that labels the root name with an additional letter from A to Z. Table II-4 shows the files written for several SAVE commands using the default root name for the files. If you plan to use the *SaveNumber* option, be sure to limit the root name to a maximum of 7 characters. The code does not use an extension for these filenames.

Table II-4. Filenames for different SAVE commands.

Command	File
SAVE	SAVECOR
SAVE 1	SAVECORA
SAVE 10	SAVECORJ

2. EgunDataFile, particle distribution file from program EGUN

EgunDataFile in LANL.INI sets the name that Parmela uses for output files from program EGUN. The default name is EGUN.DAT. This file contains a particle distribution

calculated by EGUN. There is no practical limit to the number of lines of data in EGUN.DAT because Parmela allocates arrays at runtime to store the data. Parmela uses data from EGUN.DAT when the INPUT keyword has Type = 3. The format from EGUN must be properly edited for Parmela.

3. DioutFile and DpoutFile, files related to program ISIS data

DioutFile in LANL.INI sets the name that Parmela uses for output files from the program ISIS, which is a 2-D particle-in-cell trajectory tracing code. The default name is DIOUT. This file contains a 3-D particle distribution that was derived from the 2-D ISIS distribution by randomizing the azimuthal position of the particles. Parmela reads the DioutFile when the INPUT keyword has *Type* = 0. For more information see the discussion *Type* 0, data from program ISIS under INPUT.

DpoutFile sets the name that Parmela uses for another file that contains the distribution derived from the DioutFile. The default name is DPOUT. This is the particle distribution actually used in Parmela.

4. Black or white background color in plotting codes

Configuration variable BG_Color determines the background screen color for the plotting program Pargraf. The default color is black. BG_Color has no effect on hardcopies. Hardcopy generates files with a white background which simplifies importing plot files into word processors such as Microsoft Word. For white backgrounds insert the following line in the [Global] section of LANL.INI:

```
BG_COLOR = WHITE
```

You can also set an environment variable called BG_Color. However, a setting in LANL.INI will override the environment setting. Some users run small batch files to change the BG_Color setting and the screen resolution. If you use this technique, you should avoid putting any BG_Color lines in LANL.INI.

III. Parmela, an electron linac design code

A. History of the development of Parmela

Kenneth R. Crandall wrote the original Parmela program in 1980 at the suggestion of Donald Swenson. The first users were Los Alamos staff members John Frasier, Bruce Carlsten, Lloyd Young, and others working on a free-electron laser (FEL) design. Lloyd Young added the element that simulates the photocathode. Lloyd Young also used Parmela to design and simulate the performance of an electron accelerator for a race-track microtron (RTM) project, adding new features to the code for the RTM project. He also modified the code for another application of interest to private industry. In about 1982, Lloyd Young wrote up a working manual for his version and assumed responsibility for maintaining what eventually became this official Los Alamos Parmela code.

Ken Crandall's original code used time (or phase angle) as the independent variable, and this is retained in the present version. Even after retiring from Los Alamos National

Laboratory in the mid 1980s, Ken Crandall contributed to the further development of Parmela. Working as a consultant to Los Alamos, he wrote new sections of the code for a traveling-wave accelerator tank and a FEL wiggler element.

In 1992, James H. Billen began collaborating with Lloyd Young on development of the PC version of the Poisson Superfish codes. Part of this work involved standardizing the features in Poisson Superfish that create data files for input to Parmela. While Lloyd Young continued to support the Parmela program, Jim Billen incorporated its documentation into the format that was being used for Poisson Superfish at the time. Jim Billen eventually rewrote the Parmela documentation and released the first Microsoft Word version in February, 1996.

Several early versions of Parmela were distributed to a number of laboratories around the world that were working on electron accelerators for various applications. Ken Crandall continued modifying his own version of Parmela while working with the FEL designers. Many other users modified the code for their own application and distributed copies to institutions working on similar projects.

With the early proliferation of the source code, some versions have become corrupted as multiple programmers have modified them. We get requests to consult on bugs and features we've never seen. Sometimes people request "fresh" copies because they don't trust a version they have inherited. To avoid these problems, our current policy is to distribute only executable code. By distributing only executable versions of our accelerator design codes we get good feedback from the users on bugs and on features that they would like to see incorporated in the codes. All users benefit by getting these updated versions. The codes remain almost constantly under development as we add new features for our own work and at the request of other users. This approach to distributing codes has resulted in many improvements, which has greatly benefited the entire community. A collateral advantage is that the users spend their time testing the codes and applying them to solve their particular problems rather than adding personal features. Many reported bugs have been fixed and we have added enhancements, many at the specific request of users.

B. Other Los Alamos versions of Parmela

Lloyd Young's Parmela, which is distributed by the Los Alamos Accelerator Code Group, is only one of at least two versions of the code still in common use at Los Alamos. Another version derives from two earlier versions, both of which can be traced back to one of Lloyd Young's earlier codes. Brian McVey optimized the code for fast execution on the Cray computer's vector processor. At about the same time, Bruce Carlsten had developed a version for an advanced FEL project known as APEX. This code included the capability of reading three-dimensional electromagnetic fields calculated by the MAFIA program, a feature that has since been added to the present code. In 1993, Harunori Takeda combined features from these two versions into a single code that runs on the Cray Unicos operating system.

C. Description of the Parmela program

Parmela was designed to be very flexible. Two portions of the code serve as the skeleton. The main program Parmela responds to the input data and controls the logical flow of the program, and the subroutine PARDYN controls the logic of the particle dynamics calculations. We continue to add new capabilities to Parmela, some at the request of several dozen registered users of the code outside of Los Alamos National Laboratory.

1. Input data

The input data define the transport system, specify the control logic, and specify the input and output options. All linear dimensions and coordinate values are in centimeters. Sections that describe the input lines will define the units for other parameters. Each line recognized by Parmela consists of a keyword followed by various parameters. The keyword determines the action taken by Parmela. The order of the keywords is unimportant except as noted. For example, the elements in the transport system must appear in the proper sequential order. Some keywords define beam-line elements, described by three or more parameters. The first three parameters have the same meaning for all elements:

1. The length L of the element,
2. The radial aperture size R_a at the downstream end of the element, and
3. An *OutputFlag*, indicating whether or not to output data at the downstream end of the element.

For each beam-line element, Parmela stores its sequential *Element* number, its *ElementType*, plus all of its parameters including any internally generated data. Each type of beam-line element has a unique *ElementType* identifying number. The code maintains a table of the longitudinal (z) locations of the downstream ends of each element. If you insert a ZOUT line in the input file after the last element, Parmela will generate a list of the elements in the Parmela output file. Using the ZOUT line is the easiest way to find the element numbers to use on (for example) an ERRORS line. The first element starts at z = 0.

If the number of parameters on the input line exceeds 11 or is less than 2, then Parmela writes an error message and skips data until it finds another RUN line or an END line. If the number of elements has reached the maximum number, the job aborts with an error message. The maximum number of elements is 1000 unless *RunNumber* has a larger setting on the RUN line.

2. Background static fields from distributed elements

You can include distributed elements, or background fields, in Parmela. An example of a distributed element would be the field produced by a set of Helmholtz coils. This background field would be superimposed on the field from the discrete elements. A COIL, Poisson, or EM3DField line in Parmela can define a background magnetic field that extends over other elements. The COIL line generates a background solenoidal magnetic field from supplied geometrical information about the coil and the total current of the coil. The Poisson line supplies a 2-D static field map, and the EM3DField line

supplies a 3-D static field map as discussed in the next section. The Poisson line and EM3DField line also can define a background electrostatic field. The code includes effects of the fields on the impulse applied to each particle at each integration step.

The background static fields are only active in the DRIFT, CELL, TRWAVE, DTCELL, and TANK elements. In other elements such as QUAD, BEND, SEXTUPOLE, and SOLENOID, the background fields are ignored.

Although there is no limit to the number of background fields, they may not overlap one another. To overlay multiple maps, the user must combine the fields into one map, for example in a spreadsheet. The following properties and restrictions apply to distributed elements:

- There is no limit on the number of field maps in a problem.
- A region may contain one static magnetic map and one electrostatic map defined by Poisson or EM3DField lines. In the case of static magnetic fields, the COIL field map is included.
- If two field maps of the same type (both static magnetic or both electrostatic) overlap, only one is used.
- A 3-D map takes precedence over a 2-D map, which for static magnetic maps, takes precedence over the map defined by COIL lines.
- If two 2-D maps or two 3-D maps overlay one another, the map with the smaller *MapNumber* takes precedence.
- The first COIL line must include values for Z_{min} and Z_{max}.
- Subsequent COIL lines use the values of Z_{min} and Z_{max} from the first COIL line.
- Fields from COIL lines may overlap background fields defined by Poisson or EM3DField lines, but the COIL fields are not used where the overlap occurs. For example, suppose a series of COIL lines define a background field over the whole beam line. A Poisson or a EM3DField line might apply a local magnetic field over a small region. The user must take care to avoid large discontinuities in the field in this case.

3. User-supplied field maps

Parmela input lines Poisson, CField, and TRWCField read 2-D field maps in the format written by Poisson Superfish codes Poisson, Pandira, and SF7. Parmela input lines CField and EM3DField read 3-D field maps. The following sections describe the formats of 2-D and 3-D field-map files. The 3-D field maps use Cartesian coordinates and the 2-D field maps use cylindrical coordinates.

a. Text-file format for 3-D field maps

Parmela input lines CField and EM3DField read 3-D field maps as either unformatted binary files (see program ThreeDin) or as text files. The CField line requires the digit "3" after the *CellType* parameter for a 3-D field map. Otherwise, the CField line reads a 2-D

field map as discussed in the following section. The Fortran code fragments shown below write the data in the proper order for the various maps. Parmela assumes the files are text files if the filename has extension .TXT or if variable 3dTextFile = Yes in the LANL.INI file.

Prepare a 3-D electrostatic field map for the EM3DField line as shown in the following code fragment. All length dimensions are in cm and the minimum X and Y coordinates are assumed to be zero. The electric field components are in V/m.

```
open(33,file='3D_ES.TXT')
write(33,*)izl,nmx,nmy
write(33,*)zmin,zmax,xmax,ymax
do i=-nmx,nmx
   do j=-nmy,nmy
      do k=1,izl
         write(33,*)Ez(i,j,k),Ex(i,j,k),Ey(i,j,k)
      enddo
   enddo
enddo
close(33)
```

Prepare a 3-D static magnetic field map for the EM3DField line as shown in the following code fragment. All length dimensions are in cm and the minimum X and Y coordinates are assumed to be zero. The magnetic field components are in Gauss.

```
open(33,file='3D_MAG.TXT')
write(33,*)izl,nmx,nmy
write(33,*)zmin,zmax,xmax,ymax
do i=-nmx,nmx
   do j=-nmy,nmy
      do k=1,izl
         write(33,*)Bz(i,j,k),Bx(i,j,k),By(i,j,k)
      enddo
   enddo
enddo
close(33)
```

Prepare a 3-D rf field map for the CField line as shown in the following code fragment. All length dimensions are in cm and the minimum X and Y coordinates are assumed to be zero. The electric field components are in MV/m and the magnetic field components are in A/m.

```
open(33,file='3D_RF.TXT')
write(33,*)nmx,nmy,izl
write(33,*)xmax,ymax,(zmax-zmin)
do k=0,izl-1
   do j=-nmy,nmy
      do i=-nmx,nmx
         write(33,*)Ex(i,j,k),Ey(i,j,k),Ez(i,j,k),Hx(i,j,k),
*        Hy(i,j,k),Hz(i,j,k)
      enddo
   enddo
enddo
```

```
close(33)
```

b. Text-file format for 2-D field maps

Parmela input lines Poisson, CField, and TRWCField read 2-D field maps as text files. (As discussed in the previous section, the CField line reads a 3-D field map if the line contains the digit "3" after the *CellType* parameter.) The Fortran code fragments shown below write the data in the proper order for the various maps. For 2-D maps, note that the last number on the lines containing the limits of coordinates R and Z is the number of increments, not the total number of entries for each coordinate. Also, notice that the order of the radial and longitudinal components differs between static field maps and rf field maps. This difference has its origin in the conventions used in Poisson Superfish codes for problems with cylindrical coordinates. For Poisson and Pandira (static) problems, the codes interpret input coordinates (X,Y) as (R,Z) cylindrical coordinates. For Superfish (rf) problems, input coordinates (X,Y) become (Z,R) in cylindrical coordinates.

Prepare a 2-D electrostatic field map for the Poisson line as shown in the following code fragment. All length dimensions are in cm. The electric field components are in V/m.

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

Prepare a 2-D static magnetic field map for the Poisson line as shown in the following code fragment. All length dimensions are in cm. The magnetic field components are in Gauss.

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

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

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

```
        write(33,*)Ez(i,j),Er(i,j),E(i,j),H(i,j)
      enddo
    enddo
    close(33)
```

4. Sign convention for charged particles

Because Parmela was originally written for only a single particle species, namely electrons, the negative sign of the electron's charge was not an important consideration. In practice, particles specified by combinations of the CHARGE line and subsequent INPUT lines can have either sign of the charge. The only important considerations are that particles of opposite charge are bent in opposite directions in a magnetic field and accelerated in opposite directions in an electric field. Thus, for positively charged particles, we can make the following statements about elements affected by the sign of the charge. In items 1 through 3 the phase Φ is the phase of the master clock.

1. For CELL and DTCELL elements, particles are accelerated in the +z direction for positive E_z field components when $\Phi + \phi_0$ is in the range 0 to +180 degrees, where ϕ_0 is the phase of the cell, which appears on the CELL line.
2. For TANK elements, particles are accelerated in the +z direction for positive E_z field components when $\Phi + \phi_s + (N_{cell} - 1)\Delta\phi_{tank}$ is in the range 0 to +180 degrees, where ϕ_s is the synchronous phase of the tank, $\Delta\phi_{tank}$ (= 180 degrees) is the cell-to-cell phase shift for the tank, and N_{cell} is the cell number. The first cell has $N_{cell} = 1$. Phase shift ϕ_0 appears on the TANK line.
3. For TRWAVE elements, particles are accelerated in the +z direction for positive E_z field components when $\Phi + \phi_0 + (N_{cell} - 1)\Delta\phi_{TRW}$ is in the range –90 to +90 degrees, where ϕ_0 is the phase of the first cell in the traveling-wave tank, $\Delta\phi_{cell}$ is the cell-to-cell phase shift for the tank, and N_{cell} is the cell number. The first cell has $N_{cell} = 1$. Phase shift ϕ_0 appears on the TRWAVE line.
4. For QUAD elements, particles are focused in x and defocused in y for positive values of the magnetic field gradient B′, which appears on the QUAD line.
5. For ESQUAD elements, particles are focused in x and defocused in y for positive values of the voltage V′, which appears on the ESQUAD line.
6. For SEXTUPOLE elements, particles on axis are not deflected, and particles at positions $x \neq 0$ are deflected toward –x for positive values of the focusing strength B/R_a^2, which appears on the SEXTUPOLE line.
7. For BEND elements, the beam line or +z direction follows the reference trajectory of the bend, which is the path of a fictitious particle (meaning not necessarily a particle in the beam) that enters the bend on axis and with zero divergence and has energy W_r on the BEND line. As this design particle traverses the bend, the coordinate system rotates with the particle. The BEND element deflects particles of opposite charge in opposite directions. (Versions of Parmela before 3.38 did not take account of the sign of the charge.) Particles with the same charge as the reference particle and with less energy than W_r move on an arc of smaller radius than the reference trajectory. As a result, their x coordinate increases. Higher energy charged particles with the same charge move on a larger radius arc and their x coordinate decreases.

You must keep these definitions in mind when setting up a problem file. If the beam line includes BEND elements, you must choose a positive value for *Charge* on the CHARGE line in order to transport the beam through the BEND elements regardless of the actual sign of the charge. The value *Charge* = +1 is the default setting in Parmela, and it is the correct setting for all problems that consist of only a single ion or electron species. Consider a simulation of a negative hydrogen beam (H^-) with two other ion species present: negatively charged electrons (e^-) and positively charged protons (H^+). For this calculation, in which H^- is the primary beam of interest, one would assign *Charge* = +1 to the H^- and e^- particles and *Charge* = –1 to the H^+ particles.

5. Particle coordinates, good particles and lost particles

The maximum number of particles is limited only by the computer's available memory. Each particle has six coordinates: the three position coordinates x, y, and z and the dimensionless momentum components $(\beta\gamma)_x$, $(\beta\gamma)_y$, and $(\beta\gamma)_z$. Parmela uses the same right-handed curvilinear coordinate system that TRANSPORT uses, which appears in many accelerator treatments. The beam moves in the positive z direction. When looking in the direction from which the beam is coming, +x is to the right and +y is up.

It is convenient to define a reference particle whose initial z coordinate and kinetic energy appear on the RUN line. The remaining four coordinates of the reference particle are initially zero. The coordinates of the reference particle occupy the first location in the internal particle arrays. You can run Parmela with only the single reference particle, if desired. If there are more particles, their initial coordinates derive from one or more INPUT lines. You can define a variety of different distributions. You can generate the longitudinal coordinates by specifying differences from the location of the reference particle. The code can transport more than one type of particle simultaneously. The default particle has the mass of the electron, but the RUN line can specify up to three different mass particles. The CHARGE line selects one of these defined masses and the magnitude and sign of the particle charge for use by subsequent INPUT lines.

When the dynamics calculations start (with a START or RESTART line), the code transfers all the particle coordinates to a new array. For bookkeeping, this array includes the sequential *Element* number of the beam-line element currently containing each particle. Particles upstream of the first element ($z < 0$) have *Element* number zero. Parmela allows these particles to drift until they reach the first element. Particles downstream of the last specified element are also in a drift space. Drift spaces with $z > 0$ may be immersed in a background field that will influence the particle trajectories. Background fields do not affect particles with $z < 0$.

The total number of particles at the beginning of the calculation is N_T. Another variable, N_G, keeps track of the number particles still in the calculation. All particles start as "good" particles, and remain so until they are lost for one reason or another. When a particle is lost, the code swaps its coordinates with the coordinates in location N_G of the internal arrays and decrements N_G by one. This procedure keeps the good particles in the lower part of the array so the code need not test for lost particles. If the reference particle gets lost, the code picks a new one from the remaining good particles. A particle is lost if:

1. its transverse coordinates are outside the specified radial aperture R_a in any element;

2. it arrives at a CHOPPER element at a phase outside the chopper's acceptance window;
3. it falls behind the reference particle by more than the distance Z_{Limit}, or
4. it was injected as a space-charge neutralizing particle, but has since passed the longitudinal coordinate Z_{LB}.

The value of Z_{Limit} is set by a ZLimit line. Without a ZLimit line, the default Z_{Limit} is a large value. The value of Z_{LB} appears on the SPECIAL line. Injected particles may also be lost based upon the *InjectionLoss* probability that appears on the SPECIAL line.

6. Starting the dynamics calculation and the master clock

A START line starts the beam-dynamics calculation. The START line includes an initial phase angle Φ_0, which refers to a time through the basic frequency f_0 specified on the RUN line. It is convenient to use Φ_0 to shift the phase of the particles with respect to the elements. Otherwise, the phase of each element would have to be shifted. The other parameters on the START line are: $\Delta\Phi$, the integration step size in degrees; *NumberOfSteps*, the number of steps to take at this step size; *SpaceChargeSteps*, the number of steps between space-charge impulses; and *OutputSteps*, the number of steps between calling for output.

The independent variable Φ is the phase angle of the master clock. Some elements in the system may have time-varying fields which may or may not oscillate at the reference frequency f_0, and may have a phase shift with respect to the reference clock. One can think of these elements as having their own internal clocks which are set with respect to the reference clock. When a particle is located in one of these elements, Parmela converts the reference phase to a local phase variable for computing the proper field values.

At the beginning of each integration step, the code checks to see if it should apply a space-charge impulse. If so, it calculates the space-charge field and applies the appropriate impulses to all of the particles. Particles with $z < 0$ are included in calculating the space-charge forces (unless there is a cathode at $z = 0$), but they do not receive a space-charge impulse. Then each particle is transformed separately through the integration step $\Delta\Phi$. If a background field is present, its impulse is applied at the beginning of the step. Next, the code applies the transformation for the element in which the particle is located. If a particle crosses the boundary between elements during the step, then the step is divided into segments. When a particle arrives at the end of an element, the code checks whether it should write output information at that point. If so, it stores the particle coordinates in an output buffer until all the particles have passed this location.

The dynamics calculation stops after all particles have exited the last element in the system, or after *NumberOfSteps* time steps have been taken. More input lines are then processed. After a CONTINUE line, the calculations resume using the values of $\Delta\Phi$, *NumberOfSteps*, *SpaceChargeSteps*, and *OutputSteps* specified on the CONTINUE line. You also can save the coordinates, using a SAVE line, for a possible restart (with a RESTART line) at these same conditions.

7. Phase conventions and phase settings in Parmela

RF linac designers customarily choose one of two common conventions to describe the time variation of the rf accelerating field. The "cosine convention" is common in books and articles about rf linacs, where phase $\phi = 0$ refers to the crest of the rf wave. For example, the axial electric field in a standing-wave rf cavity imparts energy to a particle according to the equation

$$\Delta W = q\int_{-L/2}^{L/2} E_z(z,t) = q\int_{-L/2}^{L/2} E_z(z)\cos(\omega t + \phi).$$

where L is the cavity length. The electric field is typically maximum at z = 0, and in this case the maximum energy gain occurs for $\phi = 0$. Some computer codes (including Parmela) use a sine convention, where the phase $\phi = 0$ refers to the point where the rising field crosses zero. (The circular accelerator community commonly uses the sine convention.) In this case, the energy gain for the same example cavity is written

$$\Delta W = q\int_{-L/2}^{L/2} E_z(z,t) = q\int_{-L/2}^{L/2} E_z(z)\sin(\omega t + \phi),$$

and for a field that peaks at z = 0, the maximum energy gain occurs for $\phi = 90$ degrees. In Parmela, users must determine the phase of rf structures relative to the phase Φ of the master clock. A suggested method for setting the phase for CELL, CELL2, DTCELL, TANK, TRWAVE, BUNCHER, and CHOPPER elements is to run Parmela up to the start of the element, request output at the end of the preceding element, and find the phase at that point in file OUTPAR.TXT. Note that if you later change preceding elements (for example, a drift length) the required phase setting for the rf element will change.

8. Particle dynamics routine in Parmela

Subroutine PARDYN controls the logic involved in following the particle coordinates through the specified number (*NumberOfSteps*) of integration steps of size $\Delta\Phi$, or until all particles are either lost or pass the last element in the system. If space-charge impulses are included, it applies them before the first step and after each step specified by the parameter *SpaceChargeSteps*. Using information from the SCHEFF line (and for 3-D calculations the SPCH3D line), the code calculates and applies the impulses for a phase interval equal to *SpaceChargeSteps*×$\Delta\Phi$. The code keeps track of the particle number N_p, and another parameter N_{LE}, which counts the number of particles that have passed the last element in the system. Particles with negative z position do not receive space-charge impulses.

Like many other beam-dynamics codes that use a version of the SCHEFF routine, Parmela first transforms particle coordinates and momenta to the rest frame of the space-charge mesh, calculates and applies the space-charge forces, then transforms the coordinates and momenta back to the lab frame. The same technique applies to the 3-D space-charge routine added in version 3 of Parmela. By default, the space-charge mesh moves at the relativistic γ of the reference particle and is centered on the reference particle. For certain special applications, the SPECIAL line allows you to specify a different relativistic γ for the mesh and the longitudinal center of the mesh. When the

SCHEFF routine transforms a relativistic beam into the rest frame of the mesh the Lorentz-contracted particles become stretched out longitudinally. This effect and your choice for the *MeshFactor* both influence the required initial size of the space-charge mesh on the SCHEFF line. Parmela regenerates the mesh when the reference particle γ reaches *MeshFactor* times its value at the last mesh generation. The number of mesh nodes remains fixed, and the mesh frame length increases to account for the Lorentz contraction making the distance between nodes larger.

For each particle number N_p, we know its current *Element* number N_E, its spatial coordinates x, y, and z, its dimensionless momentum components $(\beta\gamma)_x$, $(\beta\gamma)_y$, and $(\beta\gamma)_z$, its dimensionless energy γ, and its mass and charge. In addition, we know the particle phase Φ_p and $\Delta\Phi_p$, its remaining phase step to the end of the current master-clock step. At the start of a master-clock time step, all the particles start with $\Delta\Phi_p = \Delta\Phi$. The code transports particles through some elements using phase steps smaller than $\Delta\Phi$. For example, if the *ElementType* indicates the particle is in a linac cell, then the code checks $\Delta\Phi_{max}$ for the cell, which has a default value of 10 degrees. If $\Delta\Phi_p > \Delta\Phi_{max}$, then the code resets $\Delta\Phi_p$ to $\Delta\Phi_{max}$ for the element in which the particle resides.

At each step, the code calculates and applies the impulse resulting from all the fields in the element, including the fields from distributed elements. Using the new longitudinal velocity it finds the distance Δz that the drifting particle would travel in phase interval $\Delta\Phi_p$. If the particle would drift past the end of the element (or past z = 0, if it started upstream of the first element), then the code reduces $\Delta\Phi_p$ and Δz so that the particle would just arrive at the end of the element (or z = 0). The code checks if the particle is within the element's specified aperture. If not, the particle is lost, its coordinates are swapped with the coordinates of particle N_G, and N_G is reduced by one. Otherwise, Φ_p increments by $\Delta\Phi_p$ and we find a new value for the remaining phase interval $\Delta\Phi_p$.

9. Parmela output buffers

When a particle arrives at the end of an element, PARDYN checks several things. If this is the reference particle (N_P =1), PARDYN saves its phase and energy for later describing the longitudinal phase space of the beam. If the *OutputFlag* is nonzero for the element, then PARDYN tries to store the particle number N_p and its phase-space coordinates in the appropriate output buffer. Parmela maintains a buffer for output after specified time steps and several separate output buffers for beam-line elements. Because of a limit of 75 output buffers for elements, it may be impossible to save all particle coordinates at every specified element. After a START or RESTART line, the program clears all the output buffers. Each time the first particle arrives at the end of a new element, the code assigns and starts filling a new output buffer for that element. After all of the particles have passed an element with an assigned output buffer, the code writes the accumulated coordinates and clears that buffer for use by another element.

10. Program limits in Parmela

The maximum number of particles is limited only by the computer's available memory. The code also can use as large a space-charge mesh as you have memory for.

The code requires at least 2 MB of memory. It can use more depending on the number of particles in the calculation. The error message "no buffer available for element # particle

#" means that you need more memory for this problem. If memory is not available, try reducing the number of particles or using fewer intervals in the space-charge mesh. Another way to reduce the memory requirements is to set *OutputFlag* =0 on some beam-line elements for which you do not need to see the output.

11. Simulating DC beams in Parmela

Parmela determines the time step size from the rf frequency f_0 (in MHz) supplied on the RUN line and the integration step size $\Delta\Phi$ (in degrees) on the START line. In order to simulate a DC beam (i.e., a continuous stream of particles), choose convenient values for f_0 and $\Delta\Phi$ and generate a beam pulse that is longer than the longitudinal slice of beam you wish to study. For example, consider a DC electron gun. If you choose a frequency of 2998 MHz, then when an electron reaches nearly the velocity c ($\beta = 1$), it will travel 10 cm in 360 degrees. In this case, you might choose $\Delta\Phi = 3.6$ degrees so a $\beta = 1$ electron would move 1 mm in each time step. The beam pulse might be several wavelengths of the frequency f_0 so that the effects of space charge on the ends of the beam do not effect the slice of beam under study. Make the space-charge mesh long enough to contain the whole beam pulse. Use the SPECIAL line to select a subset of the beam centered in the long pulse. In particular, see sections a and b under the SPECIAL line.

IV. Running Parmela

This section discusses the logistics of running the Parmela code. The code reads and processes keywords from the input file. Some keywords define specific beam-line elements and others define certain properties that the code will use while processing beam-line elements. Some input keywords define the type of output to produce.

A. Starting program Parmela

To start Parmela, double click on an input file that has extension .ACC, or use the following command line:

Parmela *filename*

where *filename* is the name of the input file. Parmela reads the file LANL.INI, which contains setting for the filenames used in the code. If you do not include *filename* on the command line, then Parmela opens the input file specified in LANL.INI. It then opens the input file and begins to process the keyword lines found in the input file.

1. Running Parmela from batch files

To run several jobs sequentially from a batch file, add keyword DosErrorLevel = Yes in the [Parmela] section of file LANL.INI to suppress the pop-up stop message when Parmela ends. (When testing the procedure, check the first few lines of the Parmela output text file to be sure you are using the LANL.INI file that you intended.) For normal termination Parmela sets the DOS error level to zero. If an error has occurred the error level will be nonzero.

When starting batch jobs from Windows tools, you can use "Start /W" to instruct the operating system to wait until completion of the task before proceeding to the next line in the batch file.

Unless your machine has multiple processors, there is nothing to be gained by running several batch jobs simultaneously. Because Parmela does not launch any threads, it does pay to run the same number of batch jobs as processors. Be sure that multiple batch procedures never operate in the same directory at the same time.

There are several strategies one might adopt. If you run the code multiple times in the same directory, the batch file would need to rename output files that you wish to save to avoid overwriting files of the same name on subsequent runs. A more robust method would use a different LANL.INI file for each Parmela run. Each LANL.INI file would declare unique filenames for a particular run's input and output files. As an example, consider the file Job1.INI shown in Figure IV-1. You might have a large number of such files numbered sequentially. The prepared LANL.INI files for higher-numbered jobs are all like Figure IV-1, with "job1" replaced by "job2," "job3," etc.

This example has unique filenames for the Parmela input file, Parmela output text file, output particle distribution, and the Pargraf output text file. Successive Parmela runs in batch file Run1.Bat (see Figure IV-2) will use the same input particle distribution and will

overwrite the Pargraf input files Tape2.T2 and Tape3.T3. (These files can be large. If you choose to save them by supplying unique names be sure to save them to a disk with sufficient space.) Each Pargraf run, which occurs immediately after Parmela ends, uses the same input file, but creates a unique output file containing emittance data. Pargraf runs to completion unattended because the line PromptForOptions = No appears in the LANL.INI file.

```
[Global]
ElementOutName=Tape2.T2    ! These files are not saved in this example.
TimeStepOutName=Tape3.T3   ! To get emittance data, run Pargraf before starting the next Parmela run.

[Parmela]
ParmelaIn=job1.acc         ! Unique Parmela input file for this run.
Part_Out_Dst=job1_out.dst  ! Output particle distribution named after input file job1.acc.
LaheyLF90DstFile=Yes       ! Part_Out_Dst file will be in LF90 format for Parmteqm and Parmila.
CellDataFile=none          ! Don't output the cell data.
ParmelaOutFile=job1_out.txt ! Parmela output named after input file job1.acc.
SaveCoreRoot=SAVECOR
BfieldOutFile=none         ! Don't write the Bfield data.
part_in_dst=alljobs_in.dst ! Parmela runs use the same input particle distribution.
DosErrorLevel=yes          ! Required for batch procedures.

[Pargraf]
PargrafControl=alljobs.pgf ! Pargraf runs use same input file in this example.
PargrafOut=job1graf.txt    ! Pargraf output named after input file job1.acc.
Slice=No                   ! Turns on or off slice emittance calculations.
PromptForOptions=No        ! Puts Pargraf in movie mode, ends automatically.
```

Figure IV-1. Example file Job1.INI

The batch file shown in Figure IV-2 copies this file to LANL.INI for the first Parmela run.

```
SET LANLDIR=C:\LANL\          Your installation directory may differ.

rem     Goto        Restart
copy    Job1.ini    LANL.INI
Start /W    %LANLDIR%Parmela
Start /W    %LANLDIR%Pargraf
copy    Job2.ini    LANL.INI
Start /W    %LANLDIR%Parmela
Start /W    %LANLDIR%Pargraf
:Restart
copy    Job2.ini    LANL.INI
Start /W    %LANLDIR%Parmela
Start /W    %LANLDIR%Pargraf
```

Figure IV-2. Batch file Run1.Bat using multiple LANL.INI files.

Move the :Restart line and activate the Goto line as necessary to start the batch file on the first uncompleted task.

Another simple method sets up a series of directories and runs each job in its own directory using the same LANL.INI file for all the Parmela runs. Suppose you have three subdirectories under C:\Projects named Dir1, Dir2, and Dir3, each containing a Parmela input file named Input.Acc. If the LANL.INI file contains the line ParmelaIn = Input.Acc, then the Parmela command line does not require an input filename. Figure IV-3 shows batch file Run2.Bat, which runs Parmela sequentially in each directory. This procedure leaves behind all of the output files generated by Parmela. Thus, you can return to the directory later to run Pargraf, if desired.

```
SET LANLDIR=C:\LANL\          Your installation directory may differ.

C:
CD          C:\Projects\Dir1
Start /W    %LANLDIR%Parmela
CD          C:\Projects\Dir2
Start /W    %LANLDIR%Parmela
CD          C:\Projects\Dir3
Start /W    %LANLDIR%Parmela
```

Figure IV-3. Batch file Run2.Bat using multiple directories and one LANL.INI file.

B. Input-file keywords for Parmela

Parmela reads keywords and parameters from a text file. Keywords start in column 1, and are followed by a blank, comma, or an equal sign. After the keyword, up to 300 parameters may follow on the same or additional lines. Parameters are free format separated by spaces or commas. The following sections define the proper order for the parameters associated with each keyword. The code expects only numbers after a keyword. You can express numerical values using exponential notation (for example, 1.0e−7). The code ignores any letters appearing after a keyword, except for the "e" in a number expressed in exponential notation.

Comments may appear on any line. The code ignores all text after a semicolon, colon, or exclamation point. A comment character must either be the first character on the line or it must appear after a valid keyword or after a number on continuation lines.

Many of the keywords define a beam-line element characterized by a length L, a radial aperture R_a at the exit of the element and a value for the *OutputFlag*. These are the first three parameters for beam-line elements. The length L and radius R_a are in cm. If *OutputFlag* is nonzero, then Parmela generates output at end of the element. Here are a few general rules regarding the order of keywords in the input file:

1. The RUN line should be the first line in the Parmela input file.
2. The following keywords can appear anywhere in the input file before the START line: COIL, Poisson, INPUT, OUTPUT, and ERRORS.
3. If the CATHODE element appears, it must be the first element in the file. The only exception is that a zero-length DRIFT element can precede the CATHODE element in order to generate output of the initial beam coordinates.

4. The CHARGE line must precede the INPUT lines that will add particles to the distribution with the properties defined by the CHARGE line. A new CHARGE line changes properties for subsequent INPUT lines.
5. The CField line must appear after the CELL or DTCELL line that will use the fields specified by the CField line.
6. TITLE, SCHEFF, SPCH3D, and ADJUST lines may appear anywhere in the file, but they must be before a START, RESTART, or CONTINUE line for their effect to be used in that part of the simulation.

1. RUN, the first line in the Parmela input file

The RUN line defines certain global parameters for the rest of the calculation. The format is:

RUN *RunNumber*, *PrintFlag*, f_0, Z_0, W_0, *LinacType*, m_1c^2, m_2c^2, m_3c^2

If the value of *RunNumber* is 1000 or less, then *RunNumber* serves as an identifying run number. If *RunNumber* is greater than 1000, then it defines the maximum number of elements in the beam line. Otherwise, the maximum number of elements is 1000.

If the value of *PrintFlag* is nonzero, then Parmela echoes subsequent input lines to the output file. The parameter f_0 is the rf frequency of the linac in MHz. Z_0 is the initial longitudinal position of the reference particle in cm. The beginning of the first element is always z = 0. When the first element in the linac is a cathode, Z_0 should be a negative distance large enough to place the entire beam behind the cathode surface. Parmela places the reference particle is in the middle of the beam's phase spread, as determined by the INPUT line(s). Compute the distance traveled by the reference particle at its initial energy in a time corresponding to half the phase spread. Set Z_0 to the negative of this distance.

W_0 is the initial kinetic energy of the reference particle in MeV. The *LinacType* parameter specifies certain types of common linac cavities. Its value on the RUN line sets the default value to use if *LinacType* = 0 on a CELL line. If you use CField lines to supply actual field values, or if you supply Fourier coefficients on CELL lines (rather than use stored Fourier coefficients), the value of *LinacType* is unimportant. When using stored sets of Fourier coefficients, set *LinacType* = 1 for the disk-and-washer structure; 2 for the side-coupled cavity; 6 for the race-track microtron side-coupled cavity.

The optional mc^2 parameters are the rest-mass energies in MeV of up to three particles. The parameter *ParticleType* on a CHARGE line selects one of the particle masses for use by subsequent INPUT lines. A new CHARGE line changes properties for subsequent INPUT lines. Thus, *ParticleType* = 1 corresponds to rest-mass energy m_1c^2; *ParticleType* = 2 corresponds to m_2c^2; *ParticleType* = 3 corresponds to m_3c^2. The default rest mass for all three particles is 0.5110034 MeV, which corresponds to electrons and positrons.

After a RUN line, Parmela writes the *RunNumber*, date and time, frequency f_0, and the starting location Z_0 and energy W_0 of the reference particle. The code zeroes the coordinate arrays for the particles and calculates the normalized momentum (p/mc) from W_0. It also sets particle counter $N_T = 1$, unless it is already greater than zero, and it sets the number of "good" particles $N_G = N_T$.

2. TITLE, defines a title line

The line immediately after the TITLE keyword is an 80-character title for the problem.

3. CATHODE, a spherical cathode

The following line simulates a photocathode by defining a spherical surface as the source of particles:

CATHODE 0, R_a, *OutputFlag*, $R_{cathode}$, kT

Note the zero placeholder for the length L, which is not used for this element. $R_{cathode}$ is the radius of curvature in cm of the spherical cathode. In Parmela, a cathode is a zero-length element located at z = 0. A zero-length DRIFT element may be placed before the CATHODE element, but no other element is allowed before the CATHODE element. Positive values of $R_{cathode}$ mean that the center of curvature of the cathode is at a positive z location. Negative values of $R_{cathode}$ means the center of curvature lies at a negative value of z. A zero value for $R_{cathode}$ corresponds to a flat cathode.

The last parameter kT is an optional cathode temperature in units of electron volts (eV). If kT is omitted or equal to zero, then all particles leave the cathode perpendicular to the cathode surface and with the energy W_0 specified on the RUN line. The temperature kT can be used to give the beam a nonzero emittance. The value of kT should not exceed the energy W_0. Remember that W_0 has units of MeV and kT has units of eV. Parmela adds to each particle's initial momentum vector a momentum vector corresponding to the energy kT. The code generates randomly the magnitude and direction (in three dimensions) of this additional momentum vector. The value of kT sets the maximum magnitude of this vector.

You can optionally enter the value kT as a negative number to make all particles in the resulting distribution have the same energy, equal to the absolute value of the kT, isotropically distributed in the forward direction. In this case, the energy on the run line has no effect after the particles pass the Cathode element.

After a CATHODE line, Parmela takes particles arriving at z = 0 (usually from an INPUT line, *Type*-9 distribution) and translates them forward to the surface of the spherical cathode. The code first makes each particle direction perpendicular to the cathode surface and then, for nonzero values of kT, the code applies random momentum changes corresponding to temperature kT.

The existence of the CATHODE element does not effect the space-charge calculation. In order to ensure that the space charge and image-charge effects are computed properly, use the *RingOption* parameter on the SCHEFF line. If *RingOption* is 4 or larger, the code assumes a cathode at z = 0.0. Refer to the SCHEFF section for more details about the *RingOption* settings.

4. DRIFT, a drift space

The format of the DRIFT line is:

DRIFT L, R_a, *OutputFlag*

where L is the length of the drift, and R_a is the radial aperture, both in cm. At this point the code performs some checking of the input data. Similar checks are done after the following beam-line elements: SOLENOID, QUAD, ESQUAD, SEXTUPOLE, BEND, BUNCHER, CHOPPER, CELL, DTCELL, and TANK.

If the maximum number of elements has not been exceeded, the code increments the element counter by one, and stores the relevant data for this element for later use in the dynamics calculation. The code computes the longitudinal position of the downstream boundary of the element by adding L to the downstream location of the previous element.

5. SOLENOID, a solenoid lens

The following line defines a hard-edged magnetic solenoid having uniform longitudinal magnetic field:

SOLENOID L, R_a, *OutputFlag*, B

where B is the magnetic field in Gauss. The treatment is the same as in the code TRANSPORT [K. L. Brown, F. Rothacker, D. C. Carey, and Ch. Iselin, "TRANSPORT, a Computer Program for Designing Charged Particle Beam Transport Systems," Stanford Linear Accelerator Report SLAC-91, Revised UC-28 (May, 1977); also published as CERN-73-16 and NAL-91]. After a SOLENOID line, Parmela performs the same checks and computations done after a DRIFT line.

If the solenoidal magnetic field extends over several elements, use a COIL, Poisson, or EM3DField line to define a background magnetic field.

6. QUAD, a magnetic quadrupole lens

The following line defines a magnetic quadrupole lens:

QUAD L, R_a, *OutputFlag*, B′

where B′ is the magnetic field gradient in Gauss/cm. The QUAD element is a simple hard-edge quadrupole. The code does not assume a doublet. A doublet would consist of two QUAD elements separated by a DRIFT element. One QUAD element of the doublet has a positive magnetic gradient and the other a negative gradient. The length L is the effective length. The magnetic field gradient B′ is the pole-tip field divided by the bore radius. For positive values of B′, positively charged particles (*Charge* > 0 on a CHARGE line preceding one or more INPUT lines) are focused in x and defocused in y. Just the opposite focusing occurs for particles with *Charge* < 0. See the discussion in section C.b about sign conventions for charged particles.

After a QUAD line, Parmela performs the same checks and computations done after a DRIFT line.

7. ESQUAD, an electrostatic quadrupole lens

The following line defines an electrostatic quadrupole lens:

ESQUAD L, R_a, *OutputFlag*, V, $\Delta\Phi_{max}$

where V is the voltage in kV on the pole tips, which are a distance R_a from the beam axis. If the maximum integration step size $\Delta\Phi_{max}$ is smaller than $\Delta\Phi$, the code uses it for the

step size though the cell. The ESQUAD element conserves energy. Particles have the same energy before and after the fringe-field region of the lens on either end. The code does not assume a doublet. A doublet would consist of two ESQUAD elements separated by a DRIFT element. One ESQUAD element of the doublet has a positive voltage and the other a negative voltage. The length L is the effective length. For positive values of the voltage V, positively charged particles (*Charge* > 0 on a CHARGE line preceding one or more INPUT lines) are focused in x and defocused in y. Just the opposite focusing occurs for particles with *Charge* < 0. See the discussion in section C.b about sign conventions for charged particles.

The effective length of the electrostatic quad is the length of the equivalent hard edge electrostatic quad. The ESQUAD element assumes a fringe field of length equal to the radial aperture on both ends. If the aperture is greater than half the effective length, then length of the fringe field half the effective length. The total extent of each fringe-field region is twice the radial aperture with half of the fringe field outside the effective length and half inside the effective length. The effective length is measured between the points where the radial field strength is half the strength inside the quad. David Swenson of Eaton Semiconductor Equipment Operations compared simulations of a single quadrupole modeled in Parmela to and KOBRA, a 3-D code. The focal lengths agreed to within 3% using length L on the ESQUAD line as pole length.

After an ESQUAD line, Parmela performs the same checks and computations done after a DRIFT line.

8. STEERER, a magnetic steering element

The following line defines a magnetic quadrupole lens:

STEERER 0, R_a, *OutputFlag*, B_xL, B_yL

Note the zero placeholder for the length L, which is not used for this element. Variables B_xL and B_yL are the products of the magnetic field components and the effective length of the steering element for the x and y directions. The dimensions are Gauss-cm. This element has zero length and does not include a space-charge calculation.

9. BEND, a dipole magnet with bend in the x direction

The BEND line defines a dipole magnetic field that bends particles with charge of the same sign as the reference particle in the +x direction. Parameters on the BEND line define the reference trajectory. A particle that enters the bend on axis and with zero divergence and has energy W_r (in MeV on the BEND line) follows the reference trajectory. Notice that no beam particle including the reference particle mentioned above and elsewhere in this documentation will follow the reference trajectory unless it happens to meet all these conditions. The format of the BEND line is:

BEND L, R_a, *OutputFlag*, W_r, α_r, β_1, β_2, ψ_1, ψ_2, R_{Edge1}, R_{Edge2}, K_1, g/2, K_2

where L is the length of the reference trajectory, and R_a is the horizontal aperture of the magnet (in the plane of the bend). Note that R_a for the BEND element differs from its usual meaning for other elements. The aperture is not necessarily circular since you can specify the vertical half gap height g/2. Variable α_r is the angle of bend of the reference trajectory. The bend angle α_r must be positive and always refers to a bend in the +x

direction. If you require a bend in another direction use ROTATE elements before and after the bend.

The reference trajectory has a radius of curvature $\rho = L/\alpha_r$. (In this equation for ρ, note that α_r is in radians, through the code entry is in degrees.) As a fictitious particle traverses the reference trajectory, the coordinate system rotates with that particle. The BEND element deflects particles of opposite charge in opposite directions. (Versions of Parmela before 3.38 did not take account of the sign of the charge.) Particles with the same charge as the reference particle and with less energy than W_r move on an arc of smaller radius than the reference trajectory. As a result, their x coordinate increases. Higher energy charged particles with the same charge move on a larger radius arc and their x coordinate decreases.

Angles on the BEND line are in degrees. The leading and trailing edge angles on the bending magnet are β_1 and β_2. Fringe-field correction angles for the leading and trailing edges are ψ_1 and ψ_2. R_{Edge1} and R_{Edge2} are the radii of curvature in cm of the leading and trailing pole face edges. The code assumes straight edges if these parameters are zero. The constants K_1 and K_2 are integrals of the field related to the extent of the fringing field; they are used to calculate ψ_1 and ψ_2 (see the discussion below). The term g/2 is the optional half gap height of the magnet in cm. Its default value is the horizontal aperture R_a. The vertical aperture is g/2. Figure IV-4 shows some of the geometrical parameters.

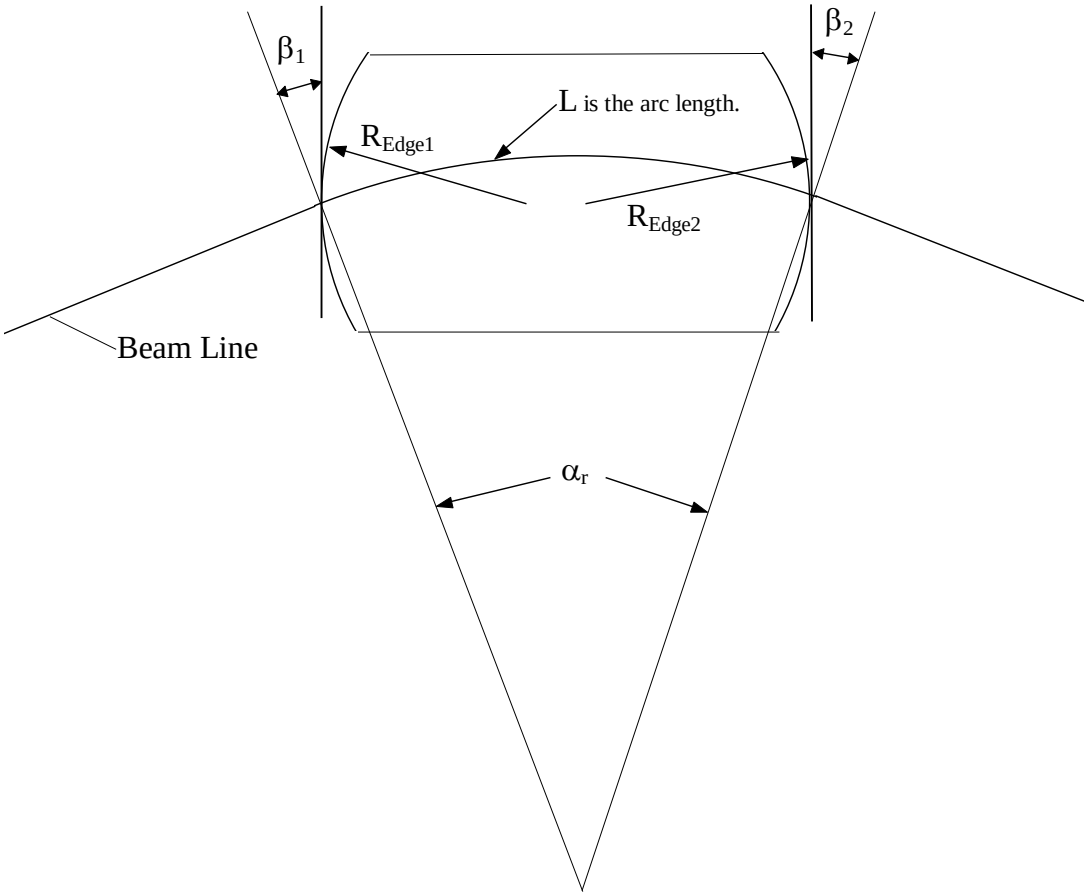

Figure IV-4. Geometrical parameters for the BEND element.
The entry and exit angles β_1 and β_2 shown in the diagram both have a positive sign.

Parmela includes terms of first and second order in the BEND calculation. For the first-order terms, the code uses entry and exit angles β_1 and β_2 and the pole-face radii R_{Edge1} and R_{Edge2}. The sign conventions are the same as in the code TRANSPORT. Zero values for β_1 and β_2 means that the beam enters and exits the magnet perpendicular to the poles, which produces focusing in the plane of the bend. Positive angles result in less focusing in the bend plane. See the treatment by Karl L. Brown in "A First- and Second-Order Matrix Theory for the Design of Beam Transport Systems and Charged Particle Spectrometers," Stanford Linear Accelerator Report SLAC-75, Revised UC-28 (May, 1969). The cover of the report says it is available for a few dollars from the Clearinghouse for Federal Scientific and Technical Information in Springfield, VA 22151. This may be what is now called the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161. The SLAC-75 report also appears in Advances in Particle Physics 1, pages 71-134 (1967).

For the second-order corrections to β_1 and β_2, the code uses ψ_1 and ψ_2 to correct the transverse focal length for the finite extent of the fringing fields. If K_1 is nonzero, then the code uses K_1 and K_2 to calculate values for ψ_1 and ψ_2 according to the equation:

$$\psi_i = K_1 \frac{g}{\rho} \sec\beta_i \left(1+\sin^2\beta_i\right)\left(1-K_1 K_2 \frac{g}{\rho}\tan\beta_i\right)$$

To supply your own values for ψ_1 and ψ_2, enter them as indicated on the BEND line and set $K_1 = 0$. If K_1 is nonzero, the calculated ψ_1 and ψ_2 override values found on the BEND line. You must still include placeholders. A reasonable value for K_1 is 0.4. The default value of K_2 is 2.8 and corresponds to a square-edged magnet. The functional form of the K_2 integral appears in the Ph. D. thesis of Leonid Sagalovsky ["Third-Order Charged Particle Beam Optics," PACS index: 41.80.-y, University of Illinois (1989)]. The notation used by Sagalovsky differs from other treatments such as the SLAC-75 report. We recommend using the square-edge magnet value of $K_2 = 2.8$, which is quoted in the manual for the code TRANSPORT [K. L. Brown, F. Rothacker, D. C. Carey, and Ch. Iselin, "TRANSPORT, a Computer Program for Designing Charged Particle Beam Transport Systems," Stanford Linear Accelerator Report SLAC-91, Revised UC-28 (May, 1977); also published as CERN-73-16 and NAL-91].

Note that integral K_1 refers to both the entrance and exit gaps in Parmela. Although, in principle, the fringe fields can be different at the two edges, Parmela assumes that they are the same at both edges and that only differences in the angles β_1 and β_2 lead to differences in ψ_1 and ψ_2. Karl L. Brown's report gives the following equation for computing a value of K_1 from the measured fields of the magnet:

$$K_1 = \int_{-\infty}^{+\infty} \frac{B_y(z)\left[B_0 - B_y(z)\right]}{gB_0^2} dz,$$

where $B_y(z)$ is the magnitude of the magnetic field on the midplane at position z, g is the distance between the magnet poles at the central orbit, and B_0 is the value of $B_y(z)$ well inside the magnet. The position z is the perpendicular distance measured from the entrance face of the magnet.

After a BEND line, Parmela performs the same checks and computations done after a DRIFT line. In addition, it converts angles α_r, β_1, β_2, ψ_1, and ψ_2 to radians. It stores the normalized momentum of the particle with energy W_r and computes the radius of curvature of the bend.

a. Simulating bends in other directions

The BEND element always deflects the beam particles in the +x direction. You can simulate bends in other directions by appropriate coordinate rotations before and after the BEND. For example, to bend the beam in the –x direction, insert ROTATE elements before and after the bend, each with $\theta_R = 180$ degrees. For a BEND in the y plane, use ±90-degree rotations.

b. Making a segmented bend

Suppose you want to examine the beam at several places within a bending magnet. By proper choices of the edge angles β_1 and β_2 and the fringe-field correction parameters, you can break the bend up into several pieces. For example, the following BEND line specifies a 120-degree bending magnet with 30-degree leading and trailing edge angles.

BEND 18.85 2.54 1 16.1 120. 30. 30. 0. 0. 0. 0. 0.45 1.3 2.8

The total length of the bend is L = 18.85 cm and the radius of curvature is $\rho = L/\alpha_r$ = 18.85/2.094 = 9.000 cm. The original BEND line specifies values of K_1 = 0.45 and K_2 = 2.8 and a half gap of 1.3 cm. Figure IV-4 shows 10 bends each of length 1.885 cm that will produce the same result with output every 12 degrees. The values ψ_1 = 8.4915 on the first line and ψ_2 = 8.4915 on the last line are the result of a manual calculation using the values K_1 = 0.45, K_2 = 2.8, g = 2.6 cm, ρ = 9.000 cm, and β_1 = 30 degrees. The very small values of ψ_1 and ψ_2 everywhere else effectively turns off the fringe-field correction on the internal edges. (Fringe effects are still present, but the fringe field effect of the end of one bend element is canceled by the fringe field effect at the entrance of the next element.) Omitting the value for K_1 on each BEND line forces Parmela to use the values of ψ_1 and ψ_2 that appear on the line. Regardless of any values of R_{Edge1} on the first line and R_{Edge2} on the last line that you might supply for the external edges, all of the internal edges should have zero edge angle.

```
BEND 1.885 2.54 1 16.1 12. 30. 0.    8.4915      0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 0.     0.0000001   0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 0.     0.0000001   0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 0.     0.0000001   0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 0.     0.0000001   0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 0.     0.0000001   0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 0.     0.0000001   0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 0.     0.0000001   0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 0.     0.0000001   0.0000001
BEND 1.885 2.54 1 16.1 12. 0. 30.    0.0000001   8.4915
```

Figure IV-5. Multiple BEND lines for producing output through a bend.

10. BUNCHER, for a buncher cavity

The following line defines a single rf cavity usually used to bunch the beam:

BUNCHER 0, R_a, *OutputFlag*, ΔW_{max}, f_B, ϕ_B, W_B

Note the zero placeholder for the length L, which is not used for this element. Variable ΔW_{max} is the maximum energy gain in MeV that a particle at the crest of the rf field would receive. The buncher frequency in MHz is f_B, and ϕ_B is the phase in degrees of the buncher rf field with respect to the master clock. The phase of the buncher rf fields is given by the expression:

$$\phi_{rf} = \phi_B + \Phi \frac{f_B}{f_0},$$

where Φ is the phase of the master clock and f_0 is the rf frequency of the linac defined on the RUN line. The section on phase conventions suggests a procedure for finding the correct phase setting.

After a BUNCHER line, Parmela performs the same checks and computations done after a DRIFT line. The reference energy W_B is an optional parameter for the buncher. The

default value for W_B is the current value of the reference-particle energy. Parmela uses this particle energy to calculate a normalized momentum, from which it computes the wave number k. The wave number appears in the argument kr of the modified Bessel functions used in computing the radial component of the electric field.

11. CHOPPER, a beam chopper

The CHOPPER line defines a beam chopper. Parmela can chop the beam both longitudinally and transversely depending upon the settings on the CHOPPER line. The format is:

CHOPPER 0, R_a, *OutputFlag*, f_C, ϕ_C, $\Delta\phi_C$, Δx, Δy, W_{min}, W_{max}

Note the zero placeholder for the length L, which is not used for this element. Variable ϕ_C is the phase in degrees of the chopper with respect to the master clock. The frequency f_C is in MHz. The phase of the center of the chopper acceptance window is given by the expression:

$$\phi_{aw} = \phi_C + \Phi \frac{f_C}{f_0},$$

where Φ is the phase of the master clock and f_0 is the rf frequency of the linac defined on the RUN line. The section on phase conventions suggests a procedure for finding the correct phase setting. The half width (in degrees) of the acceptable phase spread for the chopper is $\Delta\phi_C$. Particles outside the width of the acceptable phase spread are dropped from the calculation. If $\Delta\phi_C$ is zero, Parmela does not perform any longitudinal chopping.

Parameters Δx and Δy define a rectangular aperture with half widths Δx and Δy in cm. Particles outside the rectangular aperture are dropped from the calculation. If Δx is zero, Parmela does not perform any transverse chopping.

Parameters W_{min} and W_{max} define the minimum and maximum kinetic energy of particles passing the chopper. Particles outside this energy range are dropped from the calculation. If both W_{min} and W_{max} are zero or not present, then Parmela does not perform any chopping based upon kinetic energy.

Notice that you can simulate offset or rotated rectangular apertures by using the ERRORS line or the ROTATE line in conjunction with a CHOPPER line. To displace the aperture transversely, find the element number assigned to the CHOPPER element and include an ERRORS line with *ErrorType* = 1. For example, suppose that you wish to chop from the beam all particles below y = 0. Suppose further that the actual beam-pipe radius is 3 cm and the CHOPPER is element 30. The following lines will achieve the desired result:

```
CHOPPER   0, 50, 1, 0, 0, 0, 3, 3
ERRORS    1, 30, 30, 0, 3
```

Specifying a very large value for R_a on the CHOPPER line ensures that no particles will be dropped because they are outside the displaced circular aperture associated with the element.

After a CHOPPER line, Parmela performs the same checks and computations done after a DRIFT line.

12. STRIPPER, a beam stripper

The STRIPPER line defines a zero-length element that changes the charge state of particles in the beam. The format is:

STRIPPER 0, R_a, *OutputFlag*, *FinalCharge*, P, *Charge*, *ParticleType*

The first four parameters must appear, but the parameters following *FinalCharge* are optional. Note the zero placeholder for the length L, which is not used for this element. Parameter *FinalCharge* specifies the final charge state of particles leaving this STRIPPER element. The possible values of *FinalCharge* are the same as for *Charge*, on the CHARGE line, namely from −127 to +127, including zero. Thus the STRIPPER element can be used to neutralize beam particles.

Variable P is the probability that a particle will be changed to the charge *FinalCharge*. Parameters *Charge* and *ParticleType* are the same ones used on the CHARGE line. If they appear on the STRIPPER line, these parameters limit the operation to only those particles of the specified charge state and *ParticleType*.

a. Simulating a charge-state distribution

You can use more than one STRIPPER line in succession to simulate a particular charge-state distribution. For example, you may wish to generate the equilibrium charge-state distribution for a beam passing through a thin foil. Suppose that a beam of H^- ions passes through a foil and that you know from published data or measurement that the emerging beam is 70% H^+, 20% H^0, and 10% H^-. The following two lines would produce this distribution:

STRIPPER 0, R_a, *OutputFlag*, +1, 0.90

STRIPPER 0, R_a, *OutputFlag*, 0, 0.2857, +1

where R_a is the radial aperture of the stripper and *OutputFlag* is the usual entry for generating output after this beam-line element. The first STRIPPER element leaves 10% of the beam in the −1 charge state. The second line converts 2/7 of the +1 charge-state ions to neutrals.

13. CELL, an rf cavity or cell

The CELL line defines properties of an entire rf cavity or a portion of an accelerator containing a single accelerating gap. A CELL line is also used in combination with TRWAVE lines to accurately simulate the fringe field at either end of a traveling-wave structure. The following line defines a single cell of an rf linac:

CELL L, R_a, *OutputFlag*, ϕ_0, E_0, *CellType*, $\Delta\Phi_{max}$, *Config*, f_{Cell}, *LinacType*, B, C(1), …, C(14)

where, as always, L is the length of the element. If a symmetric full cell is specified as discussed below, then *Config* is zero and L is the full length of the cell. If only the left or right side of the cell is specified, then *Config* is nonzero and L is the length of the half cell.

The CELL has phase ϕ_0 in degrees with respect to the master clock. For CELL elements, the electric field is zero when the phase is zero. The phase of the cell rf fields is given by the expression:

$$\phi_{rf} = \phi_0 + \Phi \frac{f_{Cell}}{f_0},$$

where Φ is the phase of the master clock and f_0 is the rf frequency of the linac defined on the RUN line. Positively charged particles (*Charge* > 0 on a CHARGE line preceding one or more INPUT lines) are accelerated in the +z direction when $\Phi + \phi_0$ is in the range 0 to +180 degrees. Negatively charged particles with *Charge* < 0 are accelerated in the opposite direction. See the discussion in section C.b about sign conventions for charged particles. (We recommend that you use the ZOUT line and view the fields with Tablplot to be sure the fields are what you expect.) The section on phase conventions suggests a procedure for finding the correct phase setting.

Variable E_0 is the average axial electric field in the cell in MV/m. The parameter *CellType* is an index that identifies the rf field data for both CELL and DTCELL elements. Be sure to use a unique *CellType* index for a CELL line (unless the fields are identical to the field of a DTCELL element with the same *CellType*).

The basic time step or phase step used for all elements is $\Delta\Phi$ (in degrees) as defined on the START line. If the maximum integration step size $\Delta\Phi_{max}$ on the CELL line is smaller than $\Delta\Phi$, the code uses it for the step size though the cell. The default value for $\Delta\Phi_{max}$ is 10 degrees of the cell frequency f_{Cell}. Note that f_{Cell} corresponds only to the current CELL line. It does not affect other CELL lines of the same *CellType*.

The *Config* parameter specifies the configuration of the cell field. Normally, *Config* = 0 for symmetric cells. Parmela stores fields only for the second half of the cell and uses the cell symmetry to calculate fields in the first half. If you supply the Fourier coefficients C(1) through C(14), they must correspond to a right half cell (beam direction is left to right) with its center at z = 0. See Figure VI-1 under the discussion of program EFLD, which computes these Fourier coefficients. The z component of electric field should fall to zero at ±L/2, where L is the cell's full length. Given a set of coefficients C(j), the field components are

$$E_z(r,z) = \sum_{j=1}^{14} C(j) I_0(\kappa_j r) \cos\frac{(2j-1)\pi}{L} z,$$

$$E_r(r,z) = \sum_{j=1}^{14} C(j) \frac{(2j-1)\pi r}{L} \frac{I_1(\kappa_j r)}{\kappa_j r} \sin\frac{(2j-1)\pi}{L} z,$$

and

$$B_\theta(r,z) = \sum_{j=1}^{14} C(j) \frac{kr}{c} \frac{I_1(\kappa_j r)}{\kappa_j r} \cos\frac{(2j-1)\pi}{L} z,$$

where $\kappa_j = \sqrt{\left(\frac{(2j-1)\pi}{L}\right)^2 - \left(\frac{2\pi}{\lambda}\right)^2}$,

$k = 2\pi/\lambda$, and λ is the free-space wavelength of the resonant mode. Set *Config* = +1 or *Config* = –1 to specify fields for one side of an asymmetric full cavity. For asymmetric cells, fields derived from Fourier coefficients are not as accurate as they would be for symmetric cells. If *Config* is positive, the field distribution is reversed longitudinally with respect to the original Superfish (or 3-D code) calculation. If *Config* is negative, the distribution is not reversed! Take special care to make sure the field direction is what you want. To check the fields, include a ZOUT line after the CELL line with Z_1 and Z_2 on the ZOUT set to include the fields. Use Tablplot to view the fields.

The parameter *LinacType* specifies the type of linac cavities associated with this *CellType*. Use *LinacType* = 1 for the disk-and-washer structure ; 2 for side-coupled cavities; 6 for race-track microtron side-coupled cavity. Parmela already has stored sets of Fourier coefficients at several values of the particle velocity β for each type of linac cell. It uses these default Fourier coefficients to generate fields for a cell unless you supply your own coefficients in C(14) on the CELL line for that *CellType*, or unless you use a CField line to read actual field values for that *CellType*. If *LinacType* = 0 on a CELL line, then Parmela uses the value of *LinacType* entered on the RUN line.

The optional magnetic field B specifies a hard-edged, constant-field solenoid field for this cell. This is not a recommended option. Usually, set B to zero and define a solenoidal field with a COIL, Poisson, or EM3DField line. These elements can extend over multiple elements, if needed.

If Fourier coefficients appear on a CELL line, they must be parameters 12 through 25. These coefficients override the default set of Fourier coefficients associated with the *LinacType* for this *CellType*. Suppose several CELL lines refer to the same *CellType.* Parmela will use the Fourier coefficients from the last CELL line having the same *CellType*. These coefficients may be the ones associated with a particular *LinacType* or they may be a set C(1) to C(14) entered explicitly on a CELL line. Finally, please note that actual field data from a CField file for a particular *CellType* will override the Fourier coefficients for that *CellType*.

After a CELL line, Parmela performs the same checks and computations done after a DRIFT line. The code stores other information about the cell that it will later use when applying impulses to particles in the cell. Included in this information is the average axial electric field strength E_0, and values for $\sin\phi_0$ and $\cos\phi_0$.

14. CELL2, an rf cavity excited at two frequencies simultaneously

The CELL2 line is similar to the CELL line, except that the rf cavity contains fields at two rf frequencies. The following line defines a single cell of an rf linac:

CELL2 L, R_a, *OutputFlag*, $\phi_{0,1}$, $E_{0,1}$, *CellType1*, $f_{Cell,1}$, $\phi_{0,2}$, $E_{0,2}$, *CellType2*, $f_{Cell,2}$, *Config*

The default value of *Config* is negative and refers to full cell maps.

15. DTCELL, an rf gap with quadrupole magnets in the drift tubes.

The DTCELL line defines properties of a drift-tube linac (DTL) cell that includes an optional hard-edge quadrupole magnet (same as a QUAD element). The following line defines either a full DTL cell or a right or left half cell as shown in Figure IV-6:

DTCELL L, R_a, *OutputFlag*, ϕ_0, E_0, *CellType*, $\Delta\Phi_{max}$, *Config*, *QuadOption*, L_{Quad}, B′, C(1), …, C(14)

where, as always, L is the length of the element. If a symmetric full cell is specified as discussed below, then *Config* is zero, L is the full length of the cell, and L_{Quad} is the full length of the quadrupole magnet. If only the left or right side of the cell is specified, then *Config* is nonzero, L is the length of the half cell, and L_{Quad} is the length of the half quadrupole magnet in that cell.

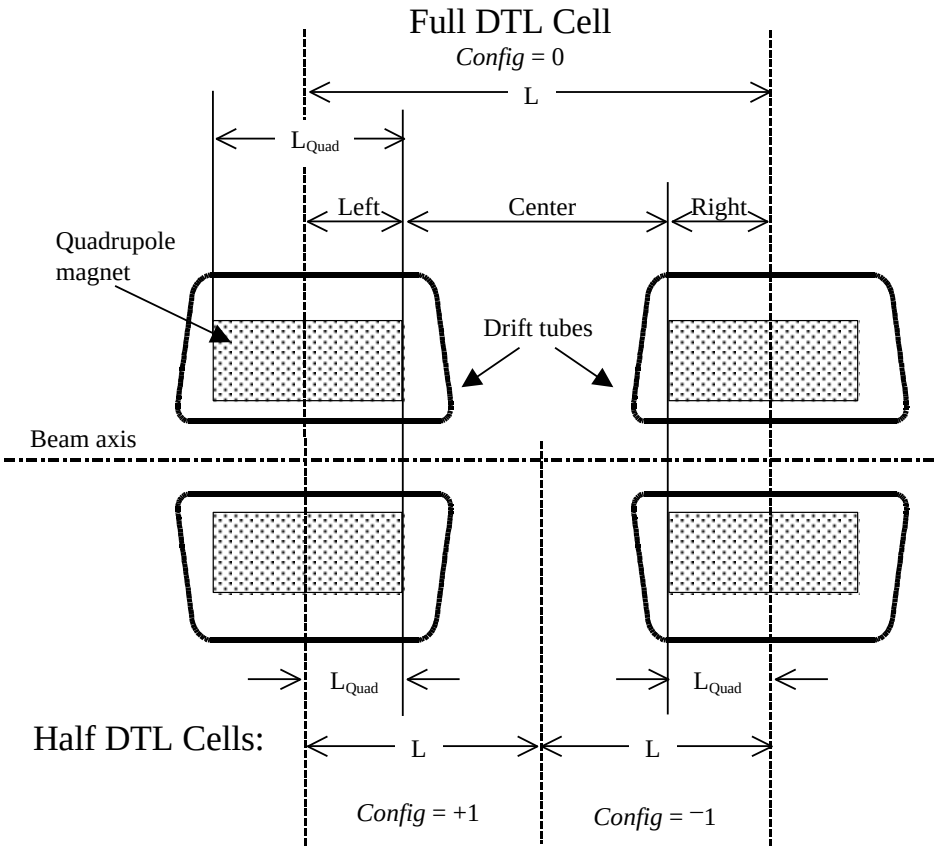

Figure IV-6. DTL cell layouts between two drift tubes.
Full cell settings appear above the drift tubes and half cell settings are below the drift tubes.

The DTCELL has phase ϕ_0 in degrees with respect to the master clock. For DTCELL elements, the electric field is zero when the phase is zero. The phase of the cell rf fields is given by the expression:

$$\phi_{rf} = \phi_0 + \Phi ,$$

where Φ is the phase of the master clock. The rf frequency of the DTCELL is defined on the RUN line. Positively charged particles (*Charge* > 0 on a CHARGE line preceding one

or more INPUT lines) are accelerated in the +z direction when $\Phi + \phi_0$ is in the range 0 to +180 degrees. Negatively charged particles with *Charge* < 0 are accelerated in the opposite direction. See the discussion in section C.b about sign conventions for charged particles. The section on phase conventions suggests a procedure for finding the correct phase setting.

Variable E_0 is the average axial electric field in the DTCELL in MV/m. The parameter *CellType* is an index that identifies the rf field data for both CELL and DTCELL elements. Be sure to use a unique *CellType* index for a DTCELL line (unless the fields are identical to the field of a CELL element with the same *CellType*).

The basic time step or phase step used for all elements is $\Delta\Phi$ (in degrees) as defined on the START line. If the maximum integration step size $\Delta\Phi_{max}$ on the DTCELL line is smaller than $\Delta\Phi$, the code uses it for the step size though the cell. The default value for $\Delta\Phi_{max}$ is 10 degrees of the linac frequency.

The *Config* parameter specifies the configuration of the field in the DTL cell. Beam direction is left to right. Normally, *Config* = 0 for symmetric cells. Parmela stores fields only for the second half of the cell and assumes symmetry to calculate fields in the first half. It is usually best to read field data with a CField line because the field E_z may not fall to zero in a DTL cell (particularly if the bore is large). You can supply Fourier coefficients C(1) through C(14), exactly as described for a CELL element, but if you do, then $E_z = 0$ at $\pm L/2$, where L is the cell's full length.

Set *Config* = +1 or *Config* = –1 to specify fields for one side of an asymmetric full cavity. For DTL cells, fields derived from Fourier coefficients are not as accurate as they would be for rf cavities specified by a CELL line. If *Config* is positive, the field distribution is reversed longitudinally with respect to the original Superfish calculation. If *Config* is negative, the distribution is not reversed! Take special care to make sure the field direction is what you want. To check the fields, include a ZOUT line after the DTCELL line with Z_1 and Z_2 on the ZOUT set to include the fields. Use Tablplot to view the fields.

Acceleration by the rf electric field occurs over the entire length of the DTL cell, but the magnetic field region is determined by the L_{Quad} in cm, the magnetic field gradient B′ in Gauss/cm and the *QuadOption* setting. lists the four *QuadOption* settings that allow you to tailor the magnet settings needed for any quadrupole focusing lattice. As an example, Figure IV-7 shows the first few lines in the Parmela input file for a DTL with a FFODDO transverse focusing lattice. In this structure, every third drift tube does not contain a quadrupole magnet. This second line in this input-file fragment is a comment line to help identify the DTCELL parameters. The first DTCELL line sets *CellType* = 61 so that the code allocates arrays large enough for 61 different field distributions. (To preserve a convenient numbering scheme for cells 2 though 60, *CellType* = 1 is not used.) For clarity in specifying half of a DTL cell (*Config* nonzero), we recommend setting *QuadOption* = 0 and entering explicitly the value of B′ required for that half cell. The example in Figure IV-7 follows this recommendation on the first two DTCELL lines.

Table IV-1. Quadrupole magnet options for DTCELL elements.

QuadOption	Left section		Center section		Right section	
	Gradient	Length	Gradient	Length	Gradient	Length
2	0	$L_{Quad}/2$	0	$L - L_{Quad}$	B′	$L_{Quad}/2$
1	B′	$L_{Quad}/2$	0	$L - L_{Quad}$	0	$L_{Quad}/2$
0	B′	$L_{Quad}/2$	0	$L - L_{Quad}$	B′	$L_{Quad}/2$
−1	B′	$L_{Quad}/2$	0	$L - L_{Quad}$	−B′	$L_{Quad}/2$

Quad 1.75 1.25 1 -3700

;	L	R_b	Of	ϕ_0	E_0	*CTyp*	$\Delta\Phi_{max}$	*Cfig*	*QOp*	L_{Quad}	B′	comment
DTCell	2.7163	1.25	1	0	1.1300	61	5	1	0	1.75	-3700	! quad in first half
DTCell	2.7163	1.25	1	0	1.1300	61	5	-1	0	1.75	0	! no quad in this half
DTCell	5.4601	1.25	1	0	1.1550	2	5	0	2	3.5	3700	! quad in second half
DTCell	5.4894	1.25	1	0	1.1802	3	5	0	0	3.5	3700	! quads in both halves
DTCell	5.5210	1.25	1	0	1.2055	4	5	0	1	3.5	3700	! quad in first half
DTCell	5.5510	1.25	1	0	1.2310	5	5	0	2	3.5	-3700	
DTCell	5.5820	1.25	1	0	1.2566	6	5	0	0	3.5	-3700	
DTCell	5.6139	1.25	1	0	1.2823	7	5	0	1	3.5	-3700	
DTCell	5.6467	1.25	1	0	1.3082	8	5	0	2	3.5	3700	
DTCell	5.6804	1.25	1	0	1.3343	9	5	0	0	3.5	3700	

Figure IV-7. Example input file lines for the first few cells of a DTL. This particular DTL uses a FFODDO transverse focusing lattice in which every third drift tube does not contain a quadrupole magnet.

After a DTCELL line, Parmela performs the same checks and computations done after a DRIFT line. The code stores other information about the DTCELL that it will later use when applying impulses to particles in the DTCELL element. Included in this information is the average axial electric field strength E_0, and values for $\sin\phi_0$ and $\cos\phi_0$.

16. CField, reads rf field data

The CField line reads rf field data from a file specified on the next line of the input file. The code can read either a 2-D field map generated by Superfish postprocessor SF7 or a 3-D field map. A 3-D field map is either a binary file of the form written by utility program ThreeDin or a text file.

The following two lines read rf field data from a file produced by the Superfish postprocessor SF7. (The Poisson Superfish codes must be installed separately.)

CField *CellType*
filename

where *filename* is the name of the SF7 output file containing the 2-D field map. SF7 writes the file with extension ".T7."

The following two lines read a 3-D map.

CField *CellType, 3*
filename

where, in this case, *filename* is the name of the binary file containing the 3-D rf field data. For 3-D fields, if the file has extension TXT, then Parmela will open the file as a text file. Otherwise, the type of file depends on the setting of LANL.INI variable 3dTextFile.

The line immediately after a CField line is the name of a file containing the fields on a rectangular grid. The following restrictions apply to the CField lines.

- The CField line and the *filename* line must appear together. (Do not use leading spaces on the *filename* line unless they are part of the filename.)
- The lines must appear after the first CELL or DTCELL line with the same *CellType.*
- The lines must appear before the START or RESTART line.
- The lines must precede any ZOUT line used to generate a Tablplot file of the fields read by the CField line.
- If you use a mix of 3-D field maps and 2-D field maps, they cannot have the same *CellType*. It is preferable, but not required, for the 2-D field maps to have *CellType* larger than the largest 3-D *CellType.*
- Each 3-D field map is completely independent of other 3-D maps. That is, maps can have different numbers of increments in all 3 directions and the physical size of the increments can be different for each map.
- All 2-D field maps must have the same number of increments in both directions (e.g., all 60 x 10, or all 100 x 20, etc.). The physical size of the increments can be different. For example, if all the maps are 60 x 10, the first 2-D map may range from z = 0 to 6 cm and r = 0 to 1 cm with increments $\delta z = 0.1$ cm and $\delta r = 0.1$ cm, and another map may range from z = 0 to 12 cm and r = 0 to 0.5 cm with increments $\delta z = 0.2$ cm and $\delta r = 0.05$ cm.

You need only one CField line for each *CellType*. If several CField lines refer to the same *CellType*, then Parmela uses the fields from the last CField line of the same *CellType*. Parmela uses these actual fields instead of generating the fields from Fourier coefficients (see CELL or DTCELL for more information).

For both type field maps, the units for the electric fields are MV/m and the units for magnetic fields are A/m.

Parmela expects that the fields for either 2-D or 3-D maps have been normalized to 1.0 MV/m (at r = 0), averaged over the cell length. For each CELL, DTCELL, or TRWAVE line that uses the field distribution that you enter with the CField line, Parmela scales the fields according to the value of E_0 specified on the CELL, DTCELL, or TRWAVE line, assuming that the fields supplied have been normalized to 1.0 MV/m. By default, both programs SF7 and ThreeDin normalize fields to $E_0 = 1.0$ MV/m. The SF7 dialog window includes a check box to force the 1.0-MV/m normalization. If you do not want this normalization, uncheck the box, or when running SF7 using .IN7 input files, set variable Force1MVperMeter = No in file SF.INI.

For 2-D maps in SF7, select the Grid option and before running the job, check the box marked “Create field map for another program” and select “Parmela file”. Also, specify

the corners of a rectangle (Z_{min}, R_{min}) and (Z_{max}, R_{max}) on which to interpolate the fields. For Parmela, R_{min} must be zero, and R_{max} should be the bore radius. Enter the number of increments in the z and r directions, using the same settings for all the cell types in the Parmela input file. All T7 files used by Parmela must use exactly the same number of increments. The SF7 code computes the increments $\delta z = (Z_{max} - Z_{min})/N_z$ and $\delta r = (R_{max} - R_{min})/N_r$ and writes the T7 file, which contains the information listed in Table IV-2. The file will contain $N_z + 1$ lines containing the field components for $r = R_{min}$, followed by another $N_z + 1$ lines with the field components for $r = R_{min} + \delta r$, etc. The last $N_z + 1$ lines correspond to $r = R_{max}$.

Table IV-2. Contents of the SF7 output file for Parmela.

Line	Data	Description
1	Z_{min}, Z_{max}, N_z	Limits of z in cm and the number of z increments.
2	f	Resonant frequency in MHz from the Superfish solution.
3	R_{min}, R_{max}, N_r	Limits of r in cm and the number of r increments.
4	E_z, E_r, E, H_ϕ	Electric and magnetic fields at (z, r) = (Z_{min}, R_{min}).
5	E_z, E_r, E, H_ϕ	Electric and magnetic fields at (z, r) = ($Z_{min} + \delta z$, R_{min}).
6	E_z, E_r, E, H_ϕ	Electric and magnetic fields at (z, r) = ($Z_{min} + 2\delta z$, R_{min}).
:	:	:
$N_z + 4$	E_z, E_r, E, H_ϕ	Electric and magnetic fields at (z, r) = (Z_{max}, R_{min}).
$N_z + 5$	E_z, E_r, E, H_ϕ	Electric and magnetic fields at (z, r) = (Z_{min}, $R_{min} + \delta r$).
$N_z + 6$	E_z, E_r, E, H_ϕ	Electric and magnetic fields at (z, r) = ($Z_{min} + \delta z$, $R_{min} + \delta r$).
:	:	:
$(N_z + 1)(N_r + 1) + 2$	E_z, E_r, E, H_ϕ	Electric and magnetic fields at (z, r) = (Z_{max}, $R_{max} - \delta r$).
$(N_z + 1)(N_r + 1) + 3$	E_z, E_r, E, H_ϕ	Electric and magnetic fields at (z, r) = (Z_{max}, R_{max}).

17. TANK, an accelerator tank

The following line defines a multiple-cell rf linac. The TANK line is not recommended for electrons with energy below 10 MeV.

TANK L, R_a, *OutputFlag*, E_0T, *NumberOfCells*, *PhaseLength*, ϕ_s

where E_0T is the average axial electric field (in MV/m) times the transit-time factor, *NumberOfCells* is the number of identical cells in the tank, *PhaseLength* is the number of π radians of phase shift between adjacent cells in the tank. For common accelerating structures, such as the coupled-cavity linac, the correct value of *PhaseLength* is 1. The last parameter ϕ_s is the synchronous phase in degrees for the tank. That is, ϕ_s gives the phase of the rf fields when the synchronous particle (sometimes called the design particle) is at the center of each cell. Positively charged particles (*Charge* > 0 on a CHARGE line preceding one or more INPUT lines) are accelerated in the +z direction when $\Phi + \phi_s$ is in the range 0 to +180 degrees. Negatively charged particles with *Charge* < 0 are accelerated in the opposite direction. See the discussion in section C.b about sign conventions for charged particles. Note that the tank frequency is assumed to be f_0, which is defined on the RUN line. The section on phase conventions suggests a procedure for finding the correct phase setting.

After a TANK line, Parmela performs the same checks and computations done after a DRIFT line. The code divides the tank length L by the *NumberOfCells* to get the length of each cell.

18. ROTATE, rotates coordinates about the beam axis

The following line rotates the coordinates about the beam axis:

ROTATE 0, R_a, *OutputFlag*, θ_R

Note the zero placeholder for the length L, which is not used for this element. The angle of rotation in degrees is θ_R. Positive angles are counterclockwise when looking along the z axis in the direction the beam propagates.

ROTATE elements are often used in conjunction with a BEND element. In Parmela, the bend angle is always a positive number and refers to a bend in the +x direction. You can simulate a bend in another direction by using ROTATE elements before and after a BEND element. For example, a bend in the −x direction would use a 180-degree rotation before and after the bend.

Two ROTATE elements on either side of a CHOPPER element can simulate a rectangular aperture whose sides are not aligned with the coordinate axes.

19. SBLOAD, specifies single-bunch beam loading

The SBLOAD line activates a single-bunch beam loading calculation during the particle dynamics. This calculation applies a correction to a particle's energy resulting from cavity fields excited by every particle ahead of it in the bunch. It does not include any changes in the transverse momentum. The format is:

SBLOAD 0, R_a, 0, L_Z, N_{SB}, A_1, A_2, A_3, A_4, A_5

Note the zero placeholder for the length L and the *OutputFlag*, which are not used for this element. Variable L_Z is the axial length in cm of an active region for the beam loading calculation. Particles within $\pm L_Z$ of the reference particle position are subject to the beam loading. The next variable N_{SB} is the number of similar cavities whose effect is applied to the beam. This is simply a multiplicative factor.

The single bunch beam loading simulation is based on a TBCI calculation of the effect for a beam with a longitudinal Gaussian distribution. The TBCI code calculates the change in energy of the particles as a function of their z position in the bunch. From a Green's function expansion, the following empirical equation describes the beam-loading effect on the energy of a single particle a distance Δz behind another particle:

$$\Delta\gamma = -\frac{q N_{SB}}{10^6 mc^2}\left[A_1 e^{-A_2\Delta z} + A_3\Delta z\, e^{-A_4\sqrt{\Delta z}}\cos\left(A_5\Delta z\right)\right],$$

where q is the charge of each macroparticle, and mc^2 is the particle rest-mass energy in MeV. In Parmela, the calculation is based on cavities with *LinacType*=2 at 1300 MHz. A scaling algorithm generates parameters for other frequencies. If you do not supply the coefficients A_1, through, A_5 on the SBLOAD line, then Parmela uses the following set of default coefficients:

$$A_1 = 1.5316 \times 10^{13},$$

$$A_2 = 1.40056 \frac{f_0}{1300},$$

$$A_3 = -5.024 \times 10^{14} \frac{f_0}{1300},$$

$$A_4 = 5.41 \sqrt{\frac{f_0}{1300}},$$

and $A_5 = 0$.

The numerical values derive from a fit to the calculated TCBI data for the very commonly used *LinacType* = 2 side-coupled cavities operating at 1300 MHz. The dimensions of the coefficient A_1 are eV/Coulomb. This term does not change with frequency. Coefficient A_3 has dimensions of eV/Coulomb·cm.

The TBCI program is not distributed by the Los Alamos Accelerator Code Group. For information about availability see the section under Information about other codes.

20. TRWAVE, a traveling-wave accelerator

The TRWAVE line defines parameters for a traveling-wave linac. The format is:

TRWAVE L, R_a, *OutputFlag*, ϕ_0, E_0T, *TRWtankNumber*, $\Delta\Phi_{max}$, f_{TR}, *TRWcellType*, N_L, N_U, *PhaseLength*, *TRWcells*, *TRWprint*, Z_1, Z_2, R_1, R_2, $\Delta\phi$, C(N_L:N_U)

Parameter *TRWcells* is the total number of cells in the traveling-wave accelerator. The only limit on the number of cells is the computer's memory resources. There also is no practical limit on the number of traveling-wave tanks. The code allocates arrays necessary to hold all the data. The first TRWAVE line for a new tank must include all the terms through *TRWcells*. The parameters after *TRWcells* are optional.

After the first TRWAVE line, the input file must contain a series of TRWAVE lines, one line for each cell in the tank. For example, if *TRWcells* = 30, there must be 30 TRWAVE lines in sequence before any other keywords. The TRWAVE lines for each cavity must include the first 9 parameters, through *TRWcellType*. All of these lines should have the same value of the traveling-wave accelerator frequency f_{TR}.

The first TRWAVE line gives the phase ϕ_0 for the entire tank. For TRWAVE elements, the electric field is a maximum when the phase is zero. Positively charged particles (*Charge* > 0 on a CHARGE line preceding one or more INPUT lines) are accelerated in the +z direction when $\Phi + \phi_0$ is in the range –90 to +90 degrees. Negatively charged particles with *Charge* < 0 are accelerated in the opposite direction. See the discussion in section C.b about sign conventions for charged particles. The section on phase conventions suggests a procedure for finding the correct phase setting.

On later TRWAVE lines, if ϕ_0 differs from the value used on first line, the difference represents a phase error for that cell. The traveling-wave linac frequency in MHz is f_{TR},

and E_0T is the product of the average axial electric field E_0 in MV/m and the transit-time factor T, which are defined as follows:

$$E_0 = \frac{1}{L}\int_{Z_s}^{Z_e} \left|E_z\right| dz ,$$

$$T = \frac{1}{E_0 L}\int_{Z_s}^{Z_e} E_z(z)(\cos kz + \sin kz)dz ,$$

where k is the wave number ($k = 2\pi/\beta\lambda$), and the length L is the difference between the starting and ending points over which E_0 is defined: $L = Z_e - Z_s$. These integrals correspond to the instant in time when the electric field is maximum in one cell. When Parmela uses the Fourier coefficients discussed below, the code normalizes them to the value 1.0 for the coefficient that determines the amplitude of the fundamental space harmonic. This term is responsible for the acceleration of the particles. Parmela multiplies the fields by the value of E_0T supplied on the TRWAVE line without any explicit reference to the transit-time factor. Normalizing the fundamental space harmonic term to unity is equivalent to multiplying the fields by the transit-time factor, when the electrons are traveling at the design velocity for the accelerating cell.

Parameter *PhaseLength* is the phase shift per cell in units of π radians. For example, a structure with a phase shift of $3\pi/4$ per cell should have the setting *PhaseLength* = 0.75, and one with $2\pi/3$ per cell should have *PhaseLength* = 0.666667.

The basic time step or phase step used for all elements is ΔΦ (in degrees) as defined on the START line. If the maximum integration step size $\Delta\Phi_{max}$ on the TRWAVE line is smaller than ΔΦ, the code uses it for the step size through the cell. The default value for $\Delta\Phi_{max}$ is 10 degrees of the traveling-wave accelerator frequency f_{TR}. Note that you can change the step size through individual cells in the same traveling-wave tank.

Parmela generates electromagnetic fields for traveling-wave accelerators using the expansion of the field in Fourier-Bessel series described by G. A. Loew and R. B. Neal in "Linear Accelerators" edited by Lapostolle and Septier, pages 44 to 95. Program EFLDTR reads the fields calculated by Superfish and generates a set of Fourier coefficients that you can supply on the TRWAVE line. See Figure VI-2 under the discussion of program EFLDTR. The variables N_L and N_U are the lower and upper dimensions of the Fourier coefficient array $C(N_L:N_U)$. For example, if C has terms in the range (–5, –4, …, 0, 1, …, 5), then $N_L = -5$ and $N_U = +5$. For this setting, all 11 Fourier coefficients must appear on the TRWAVE line. The Fourier coefficients can appear on any TRWAVE line as parameters 20 through 30.

Parameter *TRWtankNumber* is a unique identifying number for each traveling-wave accelerator tank. It must be a number from 1 to 7. Each TRWAVE line has the same *TRWtankNumber* in a tank.

Parameter *TRWcellType* is a unique number for each type of traveling-wave cell. For positive values of *TRWcellType*, Parmela calculates fields from two standing waves stored in an array with index *TRWcellType*. Fourier coefficients from the first occurrence of a positive *TRWcellType* define the fields for all subsequent TRWAVE lines of the

same *TRWcellType* regardless of the *TRWtankNumber*. If *TRWcellType* is negative, the code uses the same Fourier coefficients on all subsequent TRWAVE lines in that tank unless changed by another set of coefficients. For negative values of *TRWcellType*, it finds the fields from the Fourier coefficients each time they are needed and the calculation will be much slower.

a. Generating output files for program Tablplot

If *TRWprint* is nonzero, then, upon encountering a ZOUT line, the code writes a Tablplot input file containing the fields at *TRWprint* locations from Z_1 to Z_2. (The recommended method sets these parameters on the ZOUT line itself, but this older method will still work.) The output field components are the same as those described for the ZOUT line.

b. Simulating the fringe field in the first TRWAVE cell

The first cell of a TRWAVE structure actually starts in the center of the cell. The reason for starting in the center of the cell is to simulate accurately the fringe field into the traveling-wave structure. The fringe field and the first half of the cell can be simulated with a leading CELL line as discussed in section VII.0.0 “

The TRWAVE directory contain examples that show how the field of the leading CELL line can be adjusted to provide a smooth transition from the cell to the traveling-wave field. Because of the different definitions used for the phase of CELL and TRWAVE elements, the phase on the CELL line is $\phi_0 = 90$ degrees. Thus, the leading CELL-line phase should be 90 degrees plus the phase ϕ_0 specified on the TRWAVE line.

c. Simulating the fringe field in the last TRWAVE cell

You can optionally simulate the fringe field in the last TRWAVE cell by setting the length of this cell to half the normal length and adding a trailing CELL line similar to leading CELL line described above in section b. The trailing CELL line can be identical leading CELL line, except that the *Config* parameter must have the opposite sign. The phase ϕ_0 on the trailing CELL line depends on the phase of the last cell in the TRWAVE section. For example, consider a TRWAVE section with a phase advance of $2\pi/3$ radians per cell, which corresponds to a change in phase of 120 degrees between cells. If the section has 85 cells, then the last cell is in phase with the first cell (84 times 120 degrees is a whole number multiple of 360 degrees). For this case, the phase ϕ_0 on trailing fringe-field CELL line will be the same as ϕ_0 on the leading CELL line. If the section had 86 cells, then the last TRWAVE cell will be later by 120 degrees and ϕ_0 on trailing fringe-field CELL line would be reduced by 120 degrees.

21. TRWCField, reads Superfish field data for traveling wave cells

The following three lines read rf field data from files produced by the Superfish postprocessor SF7. (The Poisson Superfish codes must be installed separately.)

TRWCField *CellType*
filename1
filename2

The two lines immediately after a TRWCField line are the names of two files, each containing electromagnetic fields on a rectangular grid for a traveling-wave structure half-cell, but for different boundary conditions. (Whereas a CField line can read either 2-D or 3-D field maps, the TRWCField line reads only 2-D field maps.) We refer to the fields in *filename1* as the cosine solution and to the fields in *filename2* as the sine solution. We discuss these solutions in more detail below. The following restrictions apply to the TRWCField lines.

- The TRWCField line and the two *filename* lines must appear together. (Do not use leading spaces on the *filename1* or *filename2* lines unless the spaces are part of the filename.)
- The lines must appear after the lines with the TRWAVE data with the same *CellType*.
- The lines must appear before the START or RESTART line.
- The lines must precede any ZOUT line used to generate a Tablplot file of the fields read by the CField line.

You need only one TRWCField section for each *CellType*. If several TRWCField lines refer to the same *CellType*, then Parmela uses the fields from the last TRWCField line of the same *CellType*. Parmela uses these actual fields instead of generating the fields from Fourier coefficients (see CELL for more information).

The fields are in the format written by program SF7. In SF7, select the Grid option and before running the job, check the box marked "Create field map for another program" and select "Parmela file". For the TRWCField line you must run Superfish and SF7 twice to generate two data files. The two files correspond to different boundary conditions resulting in field patterns that differ in phase by 90 degrees for use in simulating a traveling wave solution. For each TRWAVE line that uses the field distribution from a TRWCField line, Parmela scales the fields according to the value of E_0 specified on the TRWAVE line. Use the following procedure to get the proper field normalization in your two input files. Example directory TRWAVE2 illustrates this procedure.

For *filename1* (the cosine file) use a Neumann boundary condition at the left edge of the first half cell as shown in Figure VI-2. Notice that the Superfish calculation will generally use a longer piece of the traveling-wave structure in order to impose the desired boundary conditions for the operating mode, but the interpolated fields supplied to Parmela correspond only to the half cell on the right-hand side of the cell. In SFO, use the NORM = 4 option with ENORM = 1.0 MV/m and supply the axial coordinates at the edges of the half cell (i.e., from z = 0 to z = +L/2, where L is the cell length). Then, when running SF7, uncheck the box labeled "Force E0 = 1 MV/m" so that SF7 does not renormalize the fields to 1.0 MV/m over the entire problem length. (Or, if using a .IN7 input file, specify "NoNormalization" on the Parmela command line.) Rename the .T7 file, if necessary, so it has a unique name and enter the name on the first line after the TRWCField line. Note the total stored energy U_{cos} (in the SFO output file) for this normalization.

For *filename2* (the sine file) use a Dirichlet boundary condition at the left edge of Figure VI-2 and also reverse the boundary condition at the right edge of the Superfish problem geometry. Solve for the fields by running Autofish (or Automesh, Fish, and SFO). The

frequency of the sine solution should be the same as that of the cosine solution within the accuracy of Superfish. For the first run of this problem use the same normalization method described above for the cosine solution. In the SFO output file, note the stored energy U_{sin}. Change ENORM in the input file from 1.0 MV/m to $(U_{cos}/U_{sin})^{0.5}$. Rerun the sine problem, including SF7, again unchecking the box labeled "Force E0 = 1 MV/m." Enter the name of the resulting file on the second line after the TRWCField line.

22. WIGGLER, a free-electron-laser wiggler

The following line defines a magnetic wiggler following the linac in a free-electron laser:

WIGGLER L, R_a, *OutputFlag*, *WigglerPeriods*, *StepsPerPeriod*, B, K_y/K_w, 0, 0, *PrintFlag*

Parameter *WigglerPeriods* is the number of wiggler periods of length λ_W over the element length L, and *StepsPerPeriod* is the number of integration steps through one wiggler period. The magnetic field B has units of Gauss. Note the use of two zero-valued placeholders for unused parameters. If *PrintFlag* is nonzero, then Parmela writes the coordinates x, y, and z, and the momentum components $(\beta\gamma)_x$, $(\beta\gamma)_y$, and $(\beta\gamma)_z$, after each integration step. For an ideal wiggler, the magnetic field components are defined as follows:

$$B_x = B\frac{K_x}{K_y}\sinh K_x x \sinh K_y y \cos K_W z\,,$$

$$B_y = B\frac{K_y}{K_y}\cosh K_x x \cosh K_y y \cos K_W z\,,$$

and

$$B_z = B\frac{K_W}{K_y}\cosh K_x x \sinh K_y y \sin K_W z\,,$$

where K_W, the wave number for the wiggler is

$$K_W = \frac{2\pi}{\lambda_W},$$

where λW = L/WigglerPeriods,

and

$$K_x^2 + K_y^2 = K_W^2\,.$$

A wiggler with vertical focusing only and no horizontal focusing has K_y/K_W=1. For equal focusing in both horizontal and vertical directions $K_y/K_W = 1/\sqrt{2}$. The equations of motion for the wiggler element are:

$$\frac{d\vec{p}}{dt} = \vec{F} = q\vec{v}\times\vec{B},$$

where q is the electric charge. Writing this in terms of the vector quantity $\vec{\beta} = \vec{v}/c$,

$$mc\frac{d\left(\vec{\beta}\gamma\right)}{dt} = qc\vec{\beta}\times\vec{B}.$$

Changing the infinitesimal time step to a discrete step Δt, we can write three equations for the change in the momentum components:

$$\Delta\left(\beta\gamma\right)_x = \frac{q}{m}\left(\beta_y B_z - \beta_z B_y\right)\Delta t\,,$$

$$\Delta\left(\beta\gamma\right)_y = \frac{q}{m}\left(\beta_z B_x - \beta_x B_z\right)\Delta t\,,$$

$$\Delta\left(\beta\gamma\right)_z = \frac{q}{m}\left(\beta_x B_y - \beta_y B_x\right)\Delta t\,.$$

Conservation of momentum requires that

$$\left(\beta\gamma\right)_x^2 + \left(\beta\gamma\right)_y^2 + \left(\beta\gamma\right)_z^2 = p^2 = \text{constant}.$$

The dynamics uses the drift-impulse-drift method (also known as the leap-frog method), but forcing the momentum to be constant. The basic time step or phase step used for all elements is $\Delta\Phi$ (in degrees) as defined on the START line. The phase step $\Delta\Phi_W$ used in the wiggler is the smaller of $\Delta\Phi$ and the quantity $360\Delta z/\beta_z\lambda$, where λ is the free-space wavelength corresponding to the linac frequency f_0, and $\Delta z = \lambda_W/$*StepsPerPeriod*. At each integration step the code first calculates the drift in each coordinate direction to get the new particle position halfway through the interval:

$$x \to x + \frac{1}{2}\left(\frac{\Delta\Phi_W}{360}\right)\beta_x\lambda\,,$$

$$y \to y + \frac{1}{2}\left(\frac{\Delta\Phi_W}{360}\right)\beta_y\lambda\,,$$

and

$$z \to z + \frac{1}{2}\left(\frac{\Delta\Phi_W}{360}\right)\beta_z\lambda\,.$$

At the halfway point in time, the code applies the impulse using the equations above with $\Delta t = \Delta\Phi_W/360\cdot f_0$ to get new values for the components of the momentum $\beta\gamma$. The x and y components are incremented:

$$\left(\beta\gamma\right)_x \to \left(\beta\gamma\right)_x + \Delta\left(\beta\gamma\right)_x,$$

and

$$\left(\beta\gamma\right)_y \to \left(\beta\gamma\right)_y + \Delta\left(\beta\gamma\right)_y.$$

But, to keep the total momentum constant, the z component is calculated as follows:

$$(\beta\gamma)_z = \sqrt{p^2 - (\beta\gamma)_x^2 - (\beta\gamma)_y^2}\ .$$

After the impulse, the code applies another drift using the new velocities as described above. This completes one integration step.

23. SEXTUPOLE, a magnetic sextupole lens

The following line defines a magnetic sextupole lens:

SEXTUPOLE L, R_a, *OutputFlag*, B/R_a^2

where B/R_a^2 is the strength of the sextupole in units of Gauss/cm^2. The strength can be positive or negative. Particles at x = 0 are not deflected. For B > 0, positively charged particles (*Charge* > 0 on a CHARGE line preceding one or more INPUT lines) at positions $x \neq 0$ are deflected toward the –x direction. Negatively charged particles with *Charge* < 0 are deflected in the opposite direction. See the discussion in section C.b about sign conventions for charged particles.

24. RFQOUT, writes particle coordinates to files

The RFQOUT line writes particle coordinates to files readable by programs Parmteq, ParmteqM, and Parmila. These files contain only particles with *ParticleType* = 1 specified on a CHARGE line, which corresponds to the rest-mass energy m_1c^2 on the RUN line. The setting *ParticleType* = 1 is the default in Parmela, so that all particles will appear in these output files if no CHARGE line appears in the input file. The format of this zero-length element is:

RFQOUT 0, R_a, *OutputFlag*, Φ_{delay}, *NumberOfFiles*, Φ_{offset}.

As particles pass the RFQOUT element, Parmela stores their coordinates in a buffer for later sorting into separate distributions. The code waits to write the RFQOUT files until after the master clock has advanced Φ_{delay} degrees past the time the reference particle arrived at the RFQOUT element. One reason for this delay is that the code does not know the reference-particle phase at the RFQOUT element until it arrives there. You should make this delay long enough so that all the particles of interest will be accumulated in the buffer. After the delay, Parmela sorts the particles into 360-degree time slices and writes the particle distributions within each time slice to a separate file. The files are named RFQ_INxx.DST, where xx is a two-digit integer from 00 to 99. A separate file named RFQ_IN.DST (without the 2-digit suffix) contains all the particles from the numbered files in a single 360-degree particle distribution. You can change the name of the particle distribution file with LANL.INI setting Part_Dst_Out. If the extension is .TXT the file is a text file, otherwise it is an unformatted binary file, of the type determined by LANL.INI keyword LaheyLF90DstFile. If LaheyLF90DstFile = Yes, then Parmela writes a distribution file compatible with program ParmteqM and Parmila, which are Lahey Fortran 90 (LF90) programs. If this keyword is No, then the Part_Out_Dst file uses the Lahey/Fujitsu Fortran 95 (LF95) format for binary files.

The parameter *NumberOfFiles* specifies how many 360-degree time slices to write. The first file, RFQ_IN00.DST, contains particles within 180 degrees fore and aft of the

reference particle phase plus Φ_{offset}. Later files alternate between fore and aft. The next file in the sequence, RFQ_IN01.DST, corresponds to particles with phase 180 to 540 degrees larger than the phase of the reference particle. (These particles are behind the reference particle in space.) File RFQ_IN02.DST contains particles with phase 180 to 540 degrees smaller than the reference particle. Other odd- and even-numbered files are additional 360-degree increments farther behind or ahead.

25. ZOUT, writes element data

The optional ZOUT line prints a list of elements and their locations and also can create a Tablplot input file containing rf fields used by DTCELL, TRWAVE, or DTCELL. element. The format is:

ZOUT N_{Print}, $\Delta\phi$, Z_1, Z_2, X_1, Y_1, X_2, Y_2

where all the parameters are optional and specify output of rf fields in CELL, DTCELL, and TRWAVE elements. The first 5 parameters must be present to write rf field data to a Tablplot input file. N_{Print} is the number of z locations at which to write rf fields between starting and ending longitudinal positions Z_1 and Z_2, $\Delta\phi$ is the phase increment between output files. You can define two sets transverse coordinates for off-axis fields by including values for X_1, Y_1 and X_2, Y_2. If X_2, Y_2 are both zero or not included, the code writes the fields at X, Y = 0, 0 and at X, Y = X_1, Y_1.

The rf field output consists of components E_z, E_x, E_y, H_x, H_y, H_z if a 3-D field map appears anywhere in the Parmela input file before the ZOUT line. A 2-D field map has r-z symmetry where, for r = 0, E_z is only nonzero component, and for r ≠0, E_z, E_r, and B_θ are the only nonzero components.

If $\Delta\phi$ is nonzero, ZOUT produces several Tablplot files, each one for a different phase of the rf. Parmela increments the rf phase by $\Delta\phi$ and generates another file until an rf cycle has been completed. That is, the files correspond to values of the system rf phase from 0 degrees up to (but not including) 360 degrees. The phase of the rf fields is the sum of the CELL phase and the system rf phase. The code builds a filename using first five characters of the RFfldFile setting in the [Parmela] section of LANL.INI and three digits numerically equal to the rf phase in degrees. For example, file RFfld045.TBL contains fields at an rf phase of 45 degrees You can plot these data using program Tablplot by double-clicking on the .TBL file. Viewing these plotted fields can serve as useful check on the parameters you have supplied on a CELL, DTCELL, or TRWAVE line.

The ZOUT line must follow all the keywords that define the beam-line elements. In particular, if you use ZOUT to view rf fields used by a CELL line, then ZOUT must follow the CELL line. This command writes to the Parmela output file (default name OUTPAR.TXT) the beam-line configuration after each beam-line keyword. This information includes the location and type of each element in the system.

For elements containing magnetic fields (e.g. from COIL lines or from Poisson lines), Parmela writes the magnetic field versus longitudinal position to a separate data file if one was specified with the BfieldOutFile keyword in file LANL.INI. These fields

correspond to the end of each element. If the element is a cell, then Parmela also writes the amplitude and phase of the accelerating field.

26. COIL, a background solenoidal magnetic field

The following line, which can appear anywhere before the START line, defines a background solenoidal magnetic field from a single current loop:

COIL Z_C, R_C, I_C, Z_{min}, Z_{max}

where Z_C is the longitudinal position in cm of the loop of radius R_C in cm and total current I_C in amps. Variables Z_{min} and Z_{max} are the lowest and highest z coordinates over which to apply the background field. There is no limit on the number of COIL lines in a problem, and they all use Z_{min} and Z_{max} from the first COIL line, which must include values for Z_{min} and Z_{max}. Fields from a series of COIL lines may overlap fields defined by Poisson and EM3DField lines. However, the COIL-defined fields are not used where it overlaps a field map over a smaller region. The section on background field maps describes how Parmela deals with maps whose z range overlaps. For each current loop, Parmela calculates the fields to sixth order using the following expansions:

$$B_z(z,r) = B_z(z,0) - \frac{r^2}{4}\left(\frac{d^2B}{dz^2} - \frac{d^4B}{dz^4}\frac{r^2}{16} + \frac{d^6B}{dz^6}\frac{r^4}{576}\right),$$

and

$$B_r(z,r) = -\frac{r}{2}\left(\frac{dB}{dz} - \frac{d^3B}{dz^3}\frac{r^2}{8} + \frac{d^5B}{dz^5}\frac{r^4}{192}\right).$$

Parmela calculates the derivatives at the same time it calculates the field components. Because the derivatives are calculated directly from the coil positions and current, the results do not suffer any inaccuracies from numerical derivatives derived from the variation of the B field itself.

After ZOUT lines, Parmela writes the magnetic field versus longitudinal position associated with COIL lines to a separate data file if one was specified with the BfieldOutFile keyword in file LANL.INI. The default is not to write this file.

27. Poisson, background fields from program Poisson

The Poisson line followed by a *filename* supplies a 2-D static background field map to Parmela. For 3-D fields use the EM3DField line. The two lines can appear together anywhere before the START line. Parmela reads either background solenoidal magnetic field or cylindrically symmetric electrostatic field data from the specified file, which you have produced using the Poisson Superfish postprocessor SF7. (Programs Poisson and Pandira can also produce these files.) The format is:

Poisson Z_{off}, *Multiplier*, X_{off}, Y_{off}, $\delta\theta_x$, $\delta\theta_y$, *FileType*, *MapNumber*
filename

where *FileType* = 1 specifies an electrostatic field map, and *FileType* = 0 specifies a magnetic field map, and *MapNumber* is a unique identifier for each independent field map of the same type *FileType*. For example, you may use electrostatic maps (1, 2, …, 5)

and multiple magnetic maps (1, 2, …, N). There is no limit on the number of maps. However, field maps may not overlap one another. The section on background field maps describes how Parmela deals with maps whose z range overlaps.

The line immediately after the Poisson line is the name of the file to read from the directory specified by the CfieldPath setting in file LANL.INI. (Do not use leading spaces on the *filename* line unless the spaces are part of the filename.) The default setting for CfieldPath is blank, which means that the files are in the default directory. If you supply a path to a directory, include the final backslash character (\).

Variable Z_{off} is the relative longitudinal position of the field calculated in Poisson to the position on the Parmela beam line. For example, suppose that in the Poisson geometry the solenoidal field map spans a range of z from –20 cm to +20 cm. If you want the center of this range (Poisson coordinate z = 0) to lie at z = 50 cm in the Parmela calculation, then set Zoff = 50.

The field *Multiplier*, which is simply a scaling factor, should not be too different from 1. If you need a much stronger or weaker field, run the Poisson problem again so that the *Multiplier* is near unity. For electrostatic fields, the sign of the *Multiplier* is important. Remember that without a CHARGE line in the input file, Parmela uses the default settings of CHARGE = +1 and the rest-mass energy of the electron. If your problem simulates electrons and you have not included the line "CHARGE –1" then you should set *Multiplier* = –1 on the Poisson line.

Variable X_{off} is used as displacement of the magnetic axis and also, optionally, as a flag to specify a file containing electrostatic fields instead of magnetic fields. If X_{off} is equal to 1 (and the line contains no more parameters), then the Poisson file contains electrostatic fields. For example,

Poisson Z_{off}, *Multiplier*, 1
filename

Be sure to include no more parameters after the integer 1 on the Poisson line for this optional electrostatic case. For any other value of X_{off} (or if more than three entries appear on the Poisson line), the code assumes magnetic field data unless *FileType* is present. *FileType* = 1 specifies an electrostatic field map, and *FileType* = 0 specifies a magnetic field map. The Poisson files for magnetic fields and electrostatic fields have exactly the same format, so care must be taken especially when using both types of Poisson files in the same Parmela problem. Use the *MapNumber* setting on the Poisson line if your problem has multiple field maps of the same type. There are a separate counters for magnetic field maps and electrostatic field maps.

For Poisson fields, X_{off} and Y_{off} are displacements in cm of the magnetic axis from the beam axis. The displacements X_{off} and Y_{off} are defined at the starting z position of the magnetic field. You can also include angular misalignments $\delta\theta_x$ and $\delta\theta_y$ in milliradians. The transverse displacements and angular misalignments apply to both types of fields. However, if a magnetic field map overlaps an electrostatic field map, the code uses the misalignments from the magnetic field map.

Parmela assumes that fields supplied with the Poisson line are cylindrically symmetric about the z axis. In Poisson and Pandira, this corresponds to the vertical or y axis of the

problem geometry. The Poisson data file contains magnetic-field or electric-field data written by SF7, Poisson, or Pandira. In SF7, select the Grid option and before running the job, check the box marked "Create field map for another program" and select "Parmela file". For more information about creating this file, refer to Data file for Parmela in the Poisson Superfish documentation file SFFILES.DOC.

28. EM3DField, 3D background fields.

The EM3DField line followed by a *filename* supplies a 3-D static background field map to Parmela. For 2-D fields use the Poisson line. The two lines can appear together anywhere before the START line. Parmela reads either background magnetic field or electrostatic field data from the specified file. The format is the same for magnetic and electrostatic fields:

EM3DField Z_{off}, *Multiplier*, *FileType*, *MapNumber* X_{off}, Y_{off}, $\delta\theta_x$, $\delta\theta_y$,
filename

where *FileType* = 1 specifies an electrostatic field map, and *FileType* = 0 specifies a magnetic field map, and *MapNumber* is a unique identifier for each independent field map of the same type *FileType*. For example, you may use electrostatic maps (1, 2, …, 5) and multiple magnetic maps (1, 2, …, N). There is no limit on the number of maps, and the size and physical dimensions of each map is independent of all other maps. However, field maps may not overlap one another. The section on background field maps describes how Parmela deals with maps whose z range overlaps.

The line immediately after the EM3DField line is the name of the file to read from the directory specified by the CfieldPath setting in file LANL.INI. (Do not use leading spaces on the *filename* line unless the spaces are part of the filename.) The default setting for CfieldPath is blank, which means that the files must be in the default directory. If you supply a path to a directory, include the final backslash character (\). If the file has extension TXT, then Parmela will open the file as a text file. Otherwise, the type of file depends on the setting of LANL.INI variable 3dTextFile.

Variable Z_{off} is the relative longitudinal position of the field map. For example, if the field map spans a range of z from –20 cm to +20 cm. If you want the center of this range (0 cm) to lie at z = 50 cm in the Parmela calculation, then set Zoff = 50.

The field *Multiplier*, which is simply a scaling factor, should not be too different from 1. If you need a much stronger or weaker field, calculate the fields again so that the *Multiplier* is near unity. For electrostatic fields, the sign of the *Multiplier* is important. Remember that without a CHARGE line in the input file, Parmela uses the default settings of CHARGE = +1 and the rest-mass energy of the electron. If your problem simulates electrons and you have not included the line "CHARGE –1" then you should set *Multiplier* = –1 on the EM3DField line.

Variable X_{off} is used as displacement of the magnetic axis. The code assumes magnetic field data unless *FileType* is present. The files for magnetic fields and electrostatic fields have exactly the same format, so care must be taken especially when using both types of files in the same Parmela problem.

For EM3DField fields, X_{off} and Y_{off} are displacements in cm of the magnetic axis from the beam axis. The displacements X_{off} and Y_{off} are defined at the starting z position of the magnetic field. You can also include angular misalignments $\delta\theta_x$ and $\delta\theta_y$ in milliradians. The transverse displacements and angular misalignments apply to both types of fields. If a magnetic field map and an electrostatic map overlap, the code uses the misalignments from the magnetic field map. If a EM3DField field overlaps a Poisson field of the same type, the code uses the EM3DField field.

29. CHARGE, specifies the charge and particle type

Parmela can track up to three different mass particles simultaneously. In addition, macroparticles of each mass can have any charge state from –127 to +127 (including zero). A CHARGE line defines the properties of particles for subsequent INPUT lines. Another CHARGE line changes the properties for INPUT lines following the new CHARGE line. The format of the CHARGE line is:

CHARGE *Charge*, *ParticleType*

where *Charge* is an integer in the range –127 to +127 that specifies the electronic charge on the particle, and *ParticleType* is a number (1, 2, or 3) that refers to one of the rest-mass energies included on the RUN line. For example, *ParticleType* = 1 refers to the rest-mass energy m_1c^2.

The CHARGE line is optional. Without a CHARGE line in the input file, Parmela uses the default settings of CHARGE = +1 and the rest-mass energy of the electron. The charge state of ions can be changed later by using a STRIPPER element in the beam line.

The input file can include several CHARGE lines and corresponding sets of INPUT lines. If you simulate space-charge neutralization effects using the injection method, you must not use 3 for the *ParticleType*, because the code appropriates the use of the internal data arrays for *ParticleType* = 3.

30. INPUT specifies the beam parameters

One or more INPUT lines before the START line specifies the input particle distribution for Parmela. Each INPUT line adds N_A particles to the total number of particles already in the beam. The particles added by an INPUT line will have the mass and charge specified by the nearest preceding CHARGE line. Parmela assigns particles a sequence number as it adds them to the distribution. The SPECIAL line uses these sequence numbers when you select groups of particles for special treatment. Program Pargraf plots the three particles (and their different charge states) in different colors. Most distribution *Types* use a line similar to one of the following format:

INPUT *Type*, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, α_z, β_z, ε_z, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT *Type*, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

The different distribution *Types* are discussed in the subsections below. Variable N_A is the number of particles to add to the distribution using properties defined on this INPUT line and preceding CHARGE line. The next six parameters after N_A are the Twiss or Courant-Snyder transverse ellipse parameters α, β, and ε for each phase plane x-x′ and y-

y′. Parameters α_x and α_y are dimensionless, β_x and β_y have units of cm/radian, and the unnormalized emittance terms ε_x and ε_y have units of cm radian.

Parmela allows two different methods for entering the longitudinal ellipse parameters. On the first INPUT line above, the method is the same as for the transverse parameters, where β has units of cm/radian, and ε refers to the unnormalized total emittance in units of cm radian. As suggested by the second INPUT line, if you set $\varepsilon_z = 0$, then the other two longitudinal ellipse parameters are the phase half width Δϕ in degrees and the energy half width ΔW in MeV for an upright ellipse. You can put all the particles at the reference particle's longitudinal coordinates by entering zero values for all three longitudinal parameters.

Parmela does not actually store a phase and energy for each particle. Instead, the code stores coordinates z and $(\beta\gamma)_z$. For each particle, Parmela chooses a phase within the range $\pm\Delta\phi$ and then computes the corresponding displacement z_i. The resulting longitudinal position for that particle is $z = Z_0 + z_i$, where Z_0 is the initial longitudinal position of the reference particle on the RUN line. In like fashion, Parmela chooses an energy difference within the range ±ΔW, adds it to the reference particle energy W_0 (on the RUN line), and then computes the quantity $(\beta\gamma)_z$ for that particle.

The last six parameters are displacements of the particle distribution for the present INPUT line. The spatial displacements δx and δy are in cm; the divergence-coordinate displacements δx′ and δy′ are in milliradians; the phase displacement δϕ is in degrees; and the energy shift δW is in MeV.

Some of the distribution *Types* use a different format for the INPUT line. These exceptions will be discussed in the individual subsections that follow. For all the distributions, a negative value for *Type* has a special meaning. A negative *Type* resets the random number generator to the same seed on every INPUT line. This feature can be helpful if you want identical distributions for a series of runs.

a. Type 0, data from program ISIS

A *Type* 0 distribution consists of data read from a file generated by program ISIS. The INPUT line has the format:

INPUT 0, N_A

where N_A is the number of particles to generate from the ISIS output file specified by the DioutFile setting in LANL.INI. The default filename is DIOUT. (You can use other codes to generate ISIS-like data.) Each line of the file corresponds to a single particle from the ISIS run, and contains the following data in fields that are each 13 characters long.

x, $(\beta\gamma)_x$, y, $(\beta\gamma)_y$, z, $(\beta\gamma)_z$, Q

where x, y, and z are the particle's spatial coordinates, $(\beta\gamma)_x$, $(\beta\gamma)_y$, and $(\beta\gamma)_z$, are the particle's dimensionless momentum components, and Q is the charge in arbitrary units (but consistent throughout the file). In ISIS, individual particles can have different amounts of charge. However, Parmela uses only equal-charge particles. Therefore, Parmela must convert the ISIS data to a distribution with equal-charge particles before it can use the data. In Parmela, the probability for creating particles using coordinates from

a particular line in the DioutFile depends upon the particle charge on that line. The procedure is as follows:

1. Read the entire file and sum up the total charge Q_{Total}.
2. Compute the charge Q_P associated with each Parmela particle: $Q_P = Q_{Total}/N_A$.
3. Define Q_A, the accumulating total unassigned charge, set initially to zero.
4. Read a first line from the ISIS file and add the charge Q_L on that line to Q_A.
5. If Q_A exceeds Q_P, generate $n = Q_L/Q_P$ particles all at the z coordinate on the line, but evenly distributed in angle around the z axis. Deduct the assigned charge nQ_P from Q_A. If Q_A is less than Q_P, do not create any Parmela particles with the coordinates of the line.
6. Repeat steps 4 and 5 until the entire file has been read, at which point all N_A particles will have been generated.

In this scheme, particles with zero charge are ignored. Parmela generates a reference particle with the average energy and average z of the input distribution. It does not move the particles to the location z_0 specified on the RUN line.

Parmela writes the distribution it generates from the DioutFile to another file specified by variable DpoutFile in LANL.INI. The default name for this file is DPOUT.

The ISIS program is not distributed by the Los Alamos Accelerator Code Group. For information about availability see the section under Information about other codes.

b. Type 1 or 10, random distribution in three phase-plane ellipses

For these distributions, Parmela adds N_A particles to the beam generated randomly in three phase plane ellipses. For *Type* 10 distributions, Parmela adjusts the transverse divergence coordinates x′ and y′ to force the angular momentum to be zero. The INPUT line has one of the following formats:

INPUT 1, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, α_z, β_z, ε_z, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT 10, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, α_z, β_z, ε_z, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

INPUT 1, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT 10, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

c. Type 2 or 20, random distribution in six-dimensional ellipse

For these distributions, Parmela adds N_A particles generated randomly in a six-dimensional ellipse. For *Type* 20 distributions, Parmela adjusts the transverse divergence coordinates x′ and y′ to force the angular momentum to be zero. The INPUT line has one of the following formats:

INPUT 2, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, α_z, β_z, ε_z, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT 20, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, α_z, β_z, ε_z, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

INPUT 2, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT 20, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

d. Type 3, generated from program EGUN data

A *Type* 3 distribution consists of data read from a file generated by program EGUN. The format of the INPUT line is

INPUT 3, N_A, *NumberOfRays*, $\Delta\phi$, *MeshSize*

where N_A is the number of particles to generate from each line of EGUN data, *NumberOfRays* is the number of lines to read from the EGUN file, $\Delta\phi$ is the phase spread in degrees, and *MeshSize* is the EGUN mesh size in cm. The EgunDataFile setting in file LANL.INI is the filename for EGUN data files in Parmela. The default filename is EGUN.DAT. This file contains one line for each particle in the following format:

RayNumber, R, RDOT, ZDOT, TDOT, R(CATHODE), I/R(CATHODE)/2π.

where R is in units of *MeshSize*. If the EGUN.DAT file has R in units of cm, then set *MeshSize* = 1.

There is no practical limit to the number of lines of data in the EGUN file because Parmela allocates arrays at runtime to store the data. Please consult your EGUN documentation for more information.

The EGUN program is not distributed by the Los Alamos Accelerator Code Group. For information about availability see the section under Information about other codes.

e. Type 4 and 40, locates individual particles at six specified coordinates

You can use multiple INPUT 4 lines of the following form to specify the six phase-space coordinates of each particle individually, or use the INPUT 40 line to read a specified number of data lines, one line for each particle from a text file. The format for a single particle is

INPUT 4, x, x′, y, y′, $\Delta\phi$, ΔW

Programs Parmila and ParmteqM use a format for this input *Type* that includes a particle number on the INPUT line. Parmela also allows the following format provided that the particle number $N_P = 1$ on the first INPUT line and also that the INPUT 4 line appears before any other INPUT line.

INPUT 4, N_P, x, x′, y, y′, $\Delta\phi$, ΔW

The transverse spatial coordinates x and y are in cm, the phase $\Delta\phi$ is in degrees; and the energy ΔW is in MeV. The quantities $\Delta\phi$ and ΔW are measured with respect to the phase and energy of the reference particle. The phase of the reference particle is zero at the position defined on the RUN line. The energy of the reference particle is defined on the RUN line. The divergence coordinates x′ and y′ are fractional momenta defined as:

$$x' = \frac{p_x}{p}, \quad y' = \frac{p_y}{p},$$

where p is the total momentum and p_x and p_y are components in the two transverse directions. For small values of p_x and p_y relative to p, x′ and y′ are approximately equal to divergence angles in radians.

The format for the INPUT 40 line is

INPUT 40, *NumberOfLines*

where *NumberOfLines* is the number of lines of particle coordinate data to read from the file specified by LANL.INI variable Part_In_Dst. The file should have extension TXT indicating a text file. Each line contains the same data as a Type 4 INPUT line: x, x′, y, y′, $\Delta\phi$, ΔW.

f. Type 5, 6, 50, and 60, random in 4-D transverse hyperspace

For these distributions, Parmela adds N_A particles to the beam generated randomly in a four-dimensional transverse hyperspace. *Type* 5 and *Type* 50 distributions have random phase and energy spread within an ellipse. *Type* 6 and *Type* 60 distributions have a uniform phase and random energy spread. For *Type* 50 and *Type* 60 distributions, Parmela adjusts the transverse divergence coordinates x′ and y′ to force the angular momentum to be zero. The INPUT line has one of the following formats:

INPUT 5, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, α_z, β_z, ε_z, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT 50, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, α_z, β_z, ε_z, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

INPUT 6, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT 60, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

g. Type 7, Kapchinskiy-Vladimirskiy distribution

Type 7 refers to a Kapchinskiy-Vladimirskiy transverse distribution. The longitudinal distribution has random phase and energy spread within an ellipse. The INPUT line has one of the following formats:

INPUT 7, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT 70, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

h. Type 8, uniform spatial distribution

For this distribution, Parmela adds N_A particles to the beam generated randomly with a uniform spatial distribution in all three directions. The divergence coordinates x′, y′, and z′ are chosen randomly within each phase-plane ellipse. Parmela converts the longitudinal coordinates z and z′ to phase and energy, ϕ and W. The INPUT line has one of the following formats:

INPUT 8, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW
INPUT 80, N_A, α_x, β_x, ε_x, α_y, β_y, ε_y, $\Delta\phi$, ΔW, 0, δx, $\delta x'$, δy, $\delta y'$, $\delta\phi$, δW

i. Type 9, Gaussian or uniform distribution with no energy spread

Distribution *Type* 9 refers to a Gaussian or uniform distribution with the same width in both transverse directions, and a different Gaussian or uniform distribution in phase. The energy spread is equal to zero, and the transverse divergence coordinates x′ and y′ are all zero. The format of the INPUT line is:

INPUT 9, N_A, σ_r, r_{max}, σ_ϕ, ϕ_{max}, $\delta\phi$, N_{b1}, N_{b2}, δx, δy, x_{max}/y_{max}

Parameters σ_r and r_{max} are in cm and σ_ϕ , ϕ_{max}, and the phase displacement $\delta\phi$ are in degrees. The transverse displacements δx and δy are in cm. The radial distribution is a Gaussian with characteristic width σ_r but with a cutoff at radius r_{max} (see the discussion in

the next paragraph for the case of an elliptical transverse shape). Starting with version 3.39 (March, 2005), Parmela generates the distribution so that the RMS beam size is σ_r and σ_ϕ in their respective directions. In version 3.38 and earlier the RMS beam size is smaller by the factor $1/2^{0.5}$ (0.7071…). For compatibility with older versions, you can set GaussianSize = 2 in the [Parmela] section of LANL.INI. (The only permitted value for GaussianSize is 1 or 2.)

No particles originate with $r > r_{max}$ (or, for elliptical cross sections, $x > x_{max}$ and $y > y_{max}$). Parmela sets a maximum of $4\sigma_r$ for the cutoff radius. If you supply a value of $r_{max} > 4\sigma_r$, then the code sets $r_{max} = 4\sigma_r$. To produce a uniform radial distribution with a cutoff at radius r_{max}, make the value of $\sigma_r > 10r_{max}$. Similarly, $2\sigma_\phi$ is the full width in the phase coordinate with cutoff at $\pm\phi_{max}$. To produce a uniform temporal distribution with a cutoff at $\pm\phi_{max}$, make the value of $\sigma_\phi > 10\phi_{max}$.

The parameter x_{max}/y_{max} is the ratio of an elliptical beam distribution's minor axes. If omitted, the code assumes $x_{max}/y_{max} = 1.0$ distribution is circular transversely. If x_{max}/y_{max} is not unity, then Parmela generates a distribution that has an elliptical cross section, such that $x_{max} \times y_{max} = (r_{max})^2$.

This input option is specifically intended to simulate electrons emitted from a photo cathode. Light from the drive laser usually has a Gaussian distribution in the radial direction. Two features of a system may cause a radial cutoff in the emission. The laser light may pass through an aperture, or the cathode itself may not be active beyond a certain radius. By using a series of *Type* 9 distributions with appropriate phase displacements $\delta\phi$, you can construct a flat-top bunch with Gaussian edges.

Variables N_{b1} and N_{b2} are the base numbers for random and quasi-random loading of the particles, a method called Quiet start by Jerome Gonichon, MIT. Generation of the radial position depends on N_{b1} and the azimuth depends on N_{b2}. In the following discussion, N_b stands for either one of the base numbers. If N_b is zero, then Parmela uses the random number generator supplied with Lahey/Fujitsu Fortran, version 5.7. This was the method in use for INPUT *Type* 9 distributions in all Parmela versions before January, 2000.

If $N_b = 1$, the code generates a list of N "equally spaced random numbers" between 0 to 1 but then assigns them randomly to the N-particle array. By assigning these numbers randomly the list does not appear to be correlated. All of the equally spaced numbers are represented, but randomly mixed.

If N_b is greater than 1, it must be a prime number. Then the routine generates a series of numbers called the Hammersley's sequence. The numbers generated depend on a seed number. The results can vary significantly depending on the seed. The seed or base numbers in the Hammersley's sequence must be prime numbers. Hence, the requirement that N_b be prime. (It is the user's responsibility to select prime numbers.) The pseudo-random number generators do have repeating patterns, the higher the base number the lower the frequency of oscillation. A suggested pair of starting numbers are $N_{b1} = 3$ and $N_{b2} = 5$. A Los Alamos internal memorandum by Richard Sheffield ["Modified Parmela particle input routine," RLS:05-96, October 22, 1996.] describes the effect of the Quiet start routine *Type* 9 INPUT lines.

j. Type 30, generated from program PARMILA or PARMTEQ data

A *Type* 30 distribution consists of data read from a file generated by programs Parmela, Parmila, or Parmteq. The format of the INPUT line is

INPUT 30, $\Delta\phi_{30}$, W_{30}, ΔW_{30}, *UseParticle1*, *WriteDistribution*, Z_{off}

The Part_In_Dst setting in the [Parmela] section of file LANL.INI defines the filename for this binary file. The default filename is PART_RFQ.DST. If you have more than one particle-distribution file, then use the following statement in LANL.INI:

```
[Parmela]
Part_In_Dst =@filename
```

where filename is the name of a file that lists the particle-distribution files, one file to a line. The code will open each file and add the particles to the beam.

For INPUT 30 cases, Parmela uses the beam current found in the particle distribution file(s) for space-charge calculations and ignores the current on the SCHEFF line. The code scales the current as necessary if the bunch frequency found in the input file differs from the frequency on the Parmela RUN line. For example, if the RUN-line frequency is 700 MHz, but the input file generated by Parmteq has a bunch frequency of 350 MHz, then Parmela doubles the current from the input file for the space-charge calculations. Parmela also corrects the phase spread of the beam to correspond to the RUN-line frequency. These current and phase adjustments ensure that charge per particle is the same in both the original calculation by Parmteq (or Parmila) and in the Parmela calculation.

All of the parameters after "INPUT 30" are optional parameters. The three parameters $\Delta\phi_{30}$, W_{30}, and ΔW_{30} define a range in longitudinal phase space for computing the average phase and energy. Parmela assigns these average coordinates to the reference particle. When computing the average phase and energy, Parmela excludes particles outside the defined range. An example of where this option is useful would be an input particle distribution with some very low-energy particles compared to the energy of bunch. If *UseParticle1* is nonzero, then Parmela will use particle 1 of the input distribution as the reference particle. This setting overrides the entries for $\Delta\phi_{30}$, W_{30}, and ΔW_{30} and the code does no averaging to find the reference particle coordinates. Include zero values for $\Delta\phi_{30}$, W_{30}, and ΔW_{30} as placeholders.

If parameter *WriteDistribution* is nonzero, then Parmela writes a text file of distribution coordinates. If parameter Z_{off} is present, then all particle coordinates are displaced by distance Z_{off} in cm. Positive values correspond to moving the particles forward.

31. OUTPUT, selects the type of output

Parmela generates output selected by *OutType* on the OUTPUT line. There can be only one OUTPUT line in the input file. The third parameter for each beam-line element is an *OutputFlag* for the element. A nonzero *OutputFlag* generates output after the element. In addition, Parmela generates output every *OutputSteps* of the master clock throughout the calculation as defined on the START, RESTART, or CONTINUE line.

The type of output depends upon the value of *OutType*, the first parameter on the OUTPUT line. This keyword can be anywhere in the input file before the START line. The OUTPUT line has the format:

OUTPUT *OutType*, …

Some older *OutType* options required additional parameters. The recommended choice is *OutType* = 5 to produce a Pargraf file for later graphical inspection of the results. Values of *OutType* from 1 to 4 generate data in text files. Using any value of *OutType* other than 5 precludes using Pargraf to view the results. The following subsections describe the various options.

a. OutType 1, write all particle coordinates

Use *OutType* 1 with no other parameters to write all the particle coordinates as text in two Parmela output files (default names TAPE2.T2 and TAPE3.T2):

OUTPUT 1

When using *OutType* 1, we recommend changing the settings for ElementOutName and TimeStepOutName in file LANL.INI so that the filenames have extension TXT (e.g. TAPE2.TXT and TAPE3.TXT). The file named by ElementOutName will contain the data columns listed in Table IV-3 after every beam-line element. The file named by TimeStepOutName will contain the data columns listed in Table II-4 at the end of every *OutputSteps* of the master clock. Using the TXT extension allows you to open the files automatically with Notepad or another text editor.

Table IV-3. Contents of file named by ElementOutName.

Column	Parameter	Description
1	x	x coordinate in cm.
2	x′	x divergence coordinate in milliradians.
3	y	x coordinate in cm.
4	y′	x divergence coordinate in milliradians.
5	ϕ	Phase coordinate in degrees
6	W	Energy in MeV
7	n	Particle number.

Table IV-4. Contents of file named by TimeStepOutName.

Column	Parameter	Description
1	x	x coordinate in cm.
2	$(\beta\gamma)_x$	Dimensionless momentum in x direction.
3	y	x coordinate in cm.
4	$(\beta\gamma)_y$	Dimensionless momentum in y direction.
5	z	Phase coordinate in degrees
6	$(\beta\gamma)_z$	Dimensionless momentum in z direction.
7	*ParticleType*	Particle species.
8	*Charge*	Electric charge of the particle.

b. OutType 2 and 3, old text-based plots

OutType 2 or 3 are old options for making text plots of particle profiles and phase and energy text graphs in output file TAPE2.T2. These options are no longer supported. Use *OutType* 5 instead.

c. OutType 4, write phase and energy data

Use *OutType* 4 with no other parameters to write the values of ϕ_{in}, ϕ_{out}, and W_{out} as text in the Parmela output file TAPE2.T2.

OUTPUT 4

d. OutType 5, Pargraf plot file

Use *OutType* 5 with no other parameters to write two unformatted binary files for program Pargraf. One file contains coordinates of particles at the end of beam-line elements (default name TAPE2.T2) and the other contains similar data at the end of specified time steps (default name TAPE3.T3).

OUTPUT 5

32. ERRORS, specifies alignment errors

You can use the ERRORS line to introduce transverse alignment errors starting on selected elements. The type of error depends upon the value of *ErrorType*, the first parameter on the ERRORS line. All ERRORS lines should follow all definitions of the beam-line elements and precede the START line. The ERRORS line has the format:

ERRORS *ErrorType*, *FirstElement*, *LastElement*, …

Some values of *ErrorType* require additional parameters. The parameters *FirstElement* and *LastElement* are the element numbers on which to apply the misalignment errors. To find values of *FirstElement* and *LastElement*, count the number of beam-line elements in the input file up to first and last elements on which you want to apply the misalignment errors. All of the transverse linear displacements are entered in cm. The following subsections describe the various options.

Please note the following with regard to how Parmela treats ERRORS lines:

- The code displaces the beam-particle coordinates to achieve the type of error specified. Because the code uses this method, the ERRORS line may not be the best way to simulate a displacement error in cases where the beam has significant space charge.
- Each ERRORS line displaces all of the remaining beam-line elements. A subsequent ERRORS line adds to any previous displacements.
- Errors accumulate if *FirstElement* and *LastElement* are not equal. For example, if you refer to elements 5 through 7, and apply a 0.1-cm displacement in x, then element 5 moves 0.1 cm, element 6 moves a total of 0.2 cm and element 7 and all following elements move a total of 0.3 cm with respect to their original positions.

a. ErrorType 1, specific displacements at each end of the element

Use *ErrorType* 1 to specify a particular set of transverse displacements at both ends of each element in the range. The format is:

ERRORS 1, *FirstElement*, *LastElement*, δx_1, δy_1, δx_2, δy_2

where δx_1 and δy_1 are the displacements at the beginning of the element, and δx_2 and δy_2 are the displacements at the end of the element. For zero-length elements, the code ignores δx_2 and δy_2. For elements with nonzero length, the code uses displacements δx_2 and δy_2 to compute the angle of the element. As an example, suppose you want to displace only element 5 by 0.1 cm in the x direction. You would need the following two ERRORS lines:

ERRORS 1, 5, 5, 0.1, 0, 0.1, 0
ERRORS 1, 6, 6, –0.1, 0, –0.1, 0

Use *ErrorType* 1 in combination with the CHOPPER line to simulate a rectangular aperture that is not centered on the beam axis.

b. ErrorType 2, random displacements in x and y of the entire element

Use *ErrorType* 2 to specify random displacements independently in x and y. The entire element is displaced with no tilt. The format is:

ERRORS 2, *FirstElement*, *LastElement*, δx_{max}, δy_{max}

where δx_{max} and δy_{max} are the tolerances on the displacement errors.

c. ErrorType 3, random displacements in x and y at the beginning of an element

Use *ErrorType* 3 to specify random displacements independently in x and y at the beginning of the element. This type of error will result in a tilt of the element since the end of the element is assumed to be aligned. The format is:

ERRORS 3, *FirstElement*, *LastElement*, δx_{max}, δy_{max}

where δx_{max} and δy_{max} are the tolerances on the displacement errors.

d. ErrorType 4, random displacements in x and y at both ends

Use *ErrorType* 4 to specify random displacements independently in x and y at both ends of the elements. The format is:

ERRORS 4, *FirstElement*, *LastElement*, δx_{1max}, δy_{1max}, δx_{2max}, δy_{2max}

where δx_{1max} and δy_{1max} are the tolerances on the displacement errors at the beginning of the element, and δx_{2max} and δy_{2max} are the tolerances on the displacement errors at the end of the element.

e. ErrorType 5, random radial displacement of the entire element.

Use *ErrorType* 5 to specify random radial displacements of the elements in the range. The azimuthal direction of the displacement is also chosen at random and the entire element is displaced with no tilt. The format is:

ERRORS 5, *FirstElement*, *LastElement*, δr_{max}

where δr_{max} is the tolerance on the displacement errors.

f. ErrorType 6, random radial displacement at the end of the element

Use *ErrorType* 6 to specify random radial displacements of the ends of the elements in the range. The azimuthal direction of the displacement is also chosen at random. This type of error will result in a tilt of the element since the beginning of the element is assumed to be aligned. The format is:

ERRORS 6, *FirstElement*, *LastElement*, δr_{max}

where δr_{max} is the tolerance on the displacement errors.

g. ErrorType 7, random radial displacements at both ends

Use *ErrorType* 7 to specify random radial displacements independently of both ends of the elements in the range. The azimuthal direction of the displacement on each end is also chosen at random. The format is:

ERRORS 7, *FirstElement*, *LastElement*, δr_{1max}, δr_{2max}

where δr_{1max} is the tolerance on the displacement errors at the beginning of the element, and δr_{2max} is the tolerance on the displacement errors at the end of the element.

h. ErrorType 8, specific displacement and angle errors at start of elements

Use *ErrorType* 8 to specify particular displacements in x and y and particular angular errors in both the x-z and y-z planes at the beginning of all elements in the range. The format is:

ERRORS 8, *FirstElement*, *LastElement*, δx, δy, $\delta\theta_x$, $\delta\theta_y$

where δx and δy are the displacement errors at the beginning of the element, and $\delta\theta_x$ and $\delta\theta_y$ are the angular displacement errors. The angular displacements are in degrees.

i. ErrorType 9, random radial and angular displacements at start of elements

Use *ErrorType* 9 to specify random radial displacements at the beginning of all elements in the range as well as random angular displacements. The format is:

ERRORS 9, *FirstElement*, *LastElement*, δr_{max}, $\delta\theta_{max}$, δr_{limit}

where δr_{max} is the tolerance on the displacement errors at the beginning of the element, and $\delta\theta_{max}$ is the tolerance on the angular displacement errors. The angular displacement is with respect to the original axis in radians. The last parameter δr_{limit} places a limit on the maximum total radial displacement that can result from the combination of both the radial and angular displacement errors. This limit applies only at the end of elements having a nonzero length.

33. SCHEFF, defines parameters for space-charge calculations

The following line defines parameters that Parmela needs to calculate space-charge impulses. The format is:

SCHEFF I_T, ΔR_{SC}, ΔZ_{SC}, N_R, N_Z, N_{Bunch}, L_{Bunch}, *RingOption*, *MeshFactor*, R_{Wall}, b

where I_T is the particle beam current in amperes. By particle beam current we mean the current that would correspond to charge +1 particles. This distinction is only important if the beam consists of multiply charged ions. For example, for a 10-mA beam of He^{2+} ions, the particle current would be $I_T = 0.005$ A. The current I_T also refers to the average current over one rf period. Thus, for example, an electron current of 1 ampere would correspond to 1 nC of charge per bunch for a frequency of 1000 MHz.

Variables ΔR_{SC} and ΔZ_{SC} are the total lengths in the lab frame in cm of the mesh region in the radial and longitudinal directions, and N_R and N_Z are the number of mesh intervals in these directions. There are no maximum values for N_R or N_Z because the code allocates the memory it needs for these arrays. We recommend using 10 to 20 mesh intervals in the radial direction and no more than about 30. The longitudinal interval depends on the accuracy desired. You may need a finer z mesh near a cathode, for example. Memory requirements and calculation time are proportional to $(N_R)^2 N_Z$.

Variable N_{Bunch} is the number of adjacent longitudinal bunches to include in the calculation. The code includes beam bunches both ahead and behind the bunch of interest. Variable L_{Bunch} is the distance between bunches. If L_{Bunch} is zero, then Parmela sets it equal to the default value ΔZ_{SC}.

The *RingOption* parameter controls the number of infinitesimally thin rings of charge used to approximate the elliptic integrals involved in calculating the space-charge field at a given particle's position. Viewed in the r-z plane, these rings are points properly positioned within the rectangular area of each mesh box. If *RingOption* is negative, then Parmela uses only one ring of charge. If *RingOption* = 0, it uses a 2×2 array of charge rings. If *RingOption* > 0, then the number of rings depends on the aspect ratio of the box and how far the point of interest is from the box. If *RingOption* is even, then radial mesh intervals give smaller intervals at smaller values of r. This behavior reduces the number of radial intervals needed when the beam is small in some places and large in other places. If *RingOption* is odd, the mesh intervals have equal dr.

For *RingOption* = 4, 5, 7, and 9, the code assumes a metal wall at z = 0.0 and calculates the image charge. Also, for these options, the mesh remains centered on the cathode until the reference particle has advanced to one-fourth the length of the mesh from the cathode. *RingOption* = 9 lengthens the mesh automatically when particles are outside the longitudinal mesh. Use this option only with a well-defined bunch in the z-direction. The recommended setting is *RingOption* = 3 unless the calculation is for electrons emitted from a cathode. Then use *RingOption* = 5. For photocathode accelerators use *RingOption* = 7, which has equal-volume rings. Because most of these rings are near the maximum radius of the mesh, the code will automatically adjust the radius of the mesh to closely match the beam size.

The *MeshFactor* parameter determines how often the space-charge routine SCHEFF regenerates the mesh. The code regenerates the mesh when γ for the reference particle reaches *MeshFactor* times its value at the last mesh generation, where $\gamma = (W_r + mc^2)/mc^2$. The number of mesh nodes remains fixed, and the mesh frame length increases to account for the Lorentz contraction making the distance between nodes larger. The default value for the *MeshFactor* is 1.5. The initial longitudinal mesh length ΔZ_{SC} must be large enough to accommodate the additional growth in the length of the beam bunch in the rest frame of the mesh. (The mesh length ΔZ_{SC} always refers to the lab frame.) For

example, for the default *MeshFactor* setting of 1.5, ΔZ_{SC} should be at least 50% larger than the length of the bunch at the time the mesh is regenerated. Otherwise, some particles will end up outside the bounds of the mesh just before the next mesh regeneration (e.g. when $\gamma \approx 1.49$ times its value when the code last computed the mesh dimensions).

The method of the space-charge field interpolation changes when γ exceeds 5. For $\gamma \leq 5$, Parmela calculates fields at the mesh corners and interpolates at intermediate locations using these corner fields. For $\gamma > 5$, the code calculates the fields at the longitudinal centers of the mesh rings and interpolates at intermediate locations using these points.

The term R_{Wall} is the radius of the conducting wall. When R_{Wall} is nonzero, the SCHEFF calculation includes the effects of image charges in the wall. If R_{Wall} is zero, there is no wall or it is assumed to be so far away as to be negligible. Using this option significantly increases the running time.

A nonzero value for the impact parameter b in cm signals a point-to-point space-charge calculation. If b is zero, the code uses the mesh space-charge calculation. The point-to-point calculation can take a lot of computer time. It is fully 3-D, but it is noisy. It may underestimate the radial space charge force at high energy if the number of particles is too low. If you use this 3-D space-charge calculation check its accuracy by comparing the radial expansion of the beam with either theory or the 2-D SCHEFF calculation. The 3-D space-charge calculation interprets *RingOption* = 4 or *RingOption* = 5 the same way as the 2-D calculation (i.e. a metal wall at z = 0). It is not obvious how to choose the size of the impact parameter. It should probably scale with the step size: use a smaller impact parameter with smaller step size.

If the Parmela input file contains a SPCH3D line for 3-D space-charge calculations, the parameters ΔZ_{SC}, N_{Bunch}, L_{Bunch}, and *RingOption* on the SCHEFF line have different meaning from those described above. See the next section for details.

34. SPCH3D, 3-D space-charge calculation

Parmela can include a 3-D space-charge calculation instead of the 2-D ring space-charge calculation. The codes uses an adaptation of the fast 3-D particle-in-cell (PIC) routine written by Robert Ryne. The format is:

SPCH3D N_x, N_y, N_z, X_{max}, Y_{max}, Z_{max}

All parameters on the SPCH3D line are optional. Variables N_x, N_y, N_z. specify the number of mesh points along the three grid axes. Each of N_x, N_y, N_z must be a multiple of 2. If any other value appears in the input file, the code will choose the nearest multiple of 2 below the specified value. Each of these parameters has a default value of 32. The values N_x, N_y, N_z. serve as starting values for an optimization algorithm that tries to maintain approximately the same mesh spacing (within a factor of 2) in each of the three coordinate directions. The minimum number of mesh points in any direction is 8.

Variables X_{max}, Y_{max}, Z_{max} set the physical dimensions of the grid axes to $\pm X_{max}$, $\pm Y_{max}$, $\pm Z_{max}$, respectively. Any or all of these variables may be nonzero. Use zero as a placeholder, if necessary. For each one that is zero or unspecified, Parmela adjusts the mesh in that direction to encompass all the particles in the simulation. A nonzero entry

for X_{max}, Y_{max}, or Z_{max} fixes the dimensions along the respective axis of the space-charge grid. For example, if $X_{max} = 2.0$, then the mesh in the x direction will contain (nominally) N_x nodes evenly spaced between –2.0 cm and +2.0 cm.

The SPCH3D calculation is quite accurate provided the aspect ratio of the mesh is within the range of about 0.25 to 4. This is not a hard limit. The aspect ratio can be outside this range, but if it very much outside, the error in the magnitude of the space-charge force may be significant. For example, if the aspect ratio is greater than 4 (z mesh intervals >4 times the average of the x and y intervals), the radial force may be too large. Parmela writes a warning in file OUTPAR.TXT if the aspect ratio exceeds 4, but it also will automatically try to increase the number of mesh intervals in the z direction. Variable Spch3dArraySize in file LANL.INI limits the size of the 3-D space-charge arrays. For example, the following entry in LANL.INI would mean the mesh size is limited to 16x16x2048, or 8x8x8192, etc.

```
[Parmela]
Spch3dArraySize=524288
```

If Parmela attempts to increase the number of mesh intervals increases in the z direction, and the array would exceed the size specified by Spch3dArraySize, then the code reduces the number of mesh intervals in the x and y directions, but it will not reduce the number of x and y intervals below 8 in each direction. If this situation occurs, the user can stop the calculation, and add appropriate SPCH3D and CONTINUE lines in the input file to control the number of intervals in the x, y, and z directions. The status bar at the bottom of the window lists the ratio and overall size of the mesh.

As the beam propagates, if any particles appear outside the mesh, the OUTPAR.TXT file will contain a warning message. The message will indicate the current fraction of the beam within the mesh. For the beam particles outside the mesh, Parmela applies a space-charge kick computed for a point charge at the beam-bunch centroid.

Like the SCHEFF calculation, the 3-D space-charge calculation takes place in the rest frame of the bunch. Thus, the Z_{max} dimension scales with the value of γ for the beam bunch.

When a SPCH3D line appears in the Parmela input file, the code uses the following parameters from the SCHEFF line: the beam current I_T, and parameters ΔZ_{SC}, N_{Bunch}, L_{Bunch}, and *RingOption*. The beam current parameter has the same meaning as on the SCHEFF line, but the other parameters have different uses with the SPCH3D line. If *RingOption* is 4 or greater, the code computes an image charge for particles leaving a flat cathode until the center of the bunch is ΔZ_{SC} from the cathode. If N_{Bunch} is nonzero, the 3-D mesh has cyclic boundary conditions in the z direction, and the mesh is L_{Bunch} long if L_{Bunch} is nonzero. If L_{Bunch} is zero, the mesh length is $\beta\lambda$, where β is the beam velocity, and λ is the wavelength corresponding to the frequency on the RUN line.

It is possible to start a Parmela calculation with the 3-D space-charge calculation and, at a certain point along the beam line, switch back to the 2-D SCHEFF calculation. To turn off the 3-D space-charge and use SCHEFF again use the line:

```
SPCH3D      –1
```

If you use this feature, be sure to include a new SCHEFF line to redefine parameters ΔZ_{SC}, N_{Bunch}, L_{Bunch}, and *RingOption* for the 2-D calculation.

For good statistics, a 3-D space-charge calculation requires more particles than a 2-D calculation does. You should consider using 100,000 particles or more.

35. CHANGE, changes one parameter of an element

You can use the CHANGE line (together with a RESTART line) to modify a single parameter on a beam-line element. The format is:

CHANGE *Parameter, Element, NewValue*

where *Parameter* is a number identifying the position on the keyword line of the parameter you want to change, *Element* is a number that identifies the beam-line element, and *NewValue* is the new numerical value of the parameter. To find the value of *Element*, the best method is to use the ZOUT line and refer to a previous PAROUT file, which will include a list of the beam-line elements. Otherwise, count the number of beam-line elements in the input file up to the one you want to modify. You should not change the length L of an element, but all the other parameters are legal to change. The code does no checking of your entries on a CHANGE line.

Suppose you want to change the magnetic field gradient of a quadrupole magnet to 20 Gauss/cm, and that the quadrupole magnet in question happens to be the seventh element in the Parmela beam-transport line. The field gradient is the fourth parameter on the QUAD line. For this case, use the following CHANGE line before the RESTART line:

CHANGE 4, 7, 20

36. ADJUST, scales parameters of several elements

You can use the ADJUST line to modify common parameters on a range of beam-line elements. The format is:

ADJUST *Parameter, FirstElement, LastElement, Multiplier, ElementType, Error*

where *Parameter* is a number identifying the position on all the keyword lines of the parameter you want to scale, *FirstElement* and *LastElement* are numbers that identify the first and last beam line elements, and *Multiplier* is a scale factor. To find values of *FirstElement* and *LastElement*, the best method is to use the ZOUT line and refer to a previous PAROUT file, which will include a list of the beam-line elements. Otherwise, count the number of beam-line elements in the input file up to the first and last elements you want to modify. The ADJUST line multiplies the selected *Parameter* on all the selected elements by *Multiplier*. The last two items *ElementType* and *Error* are optional. If *ElementType* is specified it will only adjust the *Parameter* on elements identified by *ElementType*. If *Error* is specified, then the scale factor is *Multiplier* plus a random number between *–Error* and *+Error* in units of percent of *Multiplier*. If *Multiplier* is zero, then the parameter will be multiplied by a random number between *–Error* to *+Error*.

Use this feature with caution. The ADJUST line does not apply to all parameters and no checking is done by the program to determine if you are modifying a valid parameter. For example, the phase of a tank is not a valid parameter, but its field strength is. If *ElementType* is 7 (indicating a cell), then the phase is a valid parameter. In this case the

phase will be increased (not multiplied) by the value of *Multiplier* plus a random number between –*Error* and +*Error*. The units are degrees in this case only.

37. START, starts the dynamics calculation

The START line starts the beam-dynamics calculation. The format is:

START Φ_0, $\Delta\Phi$, *NumberOfSteps*, *SpaceChargeSteps*, *OutputSteps*

where Φ_0 is the starting phase angle in degrees for the dynamics calculation and $\Delta\Phi$ is the integration step size in degrees. The next item *NumberOfSteps* is the number of integration steps and *SpaceChargeSteps* is the number of steps between space-charge impulses. Parmela generates output selected by the OUTPUT line every *OutputSteps* throughout the calculation. If *SpaceChargeSteps* = 0, the code does not perform space-charge calculations. Immediately after reading a START, RESTART, or CONTINUE line, Parmela performs *NumberOfSteps* time steps (unless all the particles are either lost or exit the last element before that occurs) and then the code reads the next line from the input file.

After a START line the code sets the master-clock phase Φ equal to the initial value Φ_0. It initializes a number of arrays and performs other bookkeeping tasks before passing control to the PARDYN subroutine, which controls the dynamics calculation.

38. SAVE and RESTART, starting with saved particle coordinates

The SAVE and RESTART lines provide a way to change the beam-line layout and start the calculation over just before the changed section without running the whole problem over. This method can save a lot of time. The format is:

SAVE *SaveNumber*

RESTART $\Delta\Phi$, *NumberOfSteps*, *SpaceChargeSteps*, *OutputSteps*, ΔZ_{Ref}, *SaveNumber*

The SAVE line saves all particle coordinates in a file for later use by a RESTART line. A SAVE command can appear after a START, CONTINUE, or RESTART line.

The RESTART line includes the same parameters found on the START line plus the ΔZ_{Ref} and *SaveNumber* parameters. Variable ΔZ_{Ref} is a change in the z position of the reference particle. Since the reference particle is used to center the space-charge grid, you might choose to use this parameter to move the reference particle closer to the center of the beam bunch before proceeding with the calculation. *SaveNumber* corresponds to the same number used on a SAVE line.

After a SAVE line, the code stores the present value of N_G as N_{GS}, copies the present value of the phase Φ to Φ_0, and writes this information plus the particle coordinates to the disk file. After the RESTART line, Parmela resets parameters $\Delta\Phi$, *NumberOfSteps*, *SpaceChargeSteps*, and *OutputSteps* to the values specified, resets N_G to N_{GS} and reads the particle coordinates from the file. It then follows the procedure described after START.

a. Filenames for saved particle coordinates

The keyword SaveCoreRoot in LANL.INI specifies the root name for the saved-coordinate files. The default filename is SAVECOR. This is its complete name if no *SaveNumber* appears on the SAVE or RESTART line. For SAVE and RESTART lines with the optional *SaveNumber* parameter, Parmela appends a letter to the root name. *SaveNumber* is a number from 1 to 26 that labels the root name with an additional letter from A to Z. Table shows the files written for several SAVE commands using the default root name for the files. If you use the *SaveNumber* option, be sure to limit the root name to 7 characters. The code does not use an extension for these filenames.

39. CONTINUE, starts again after changing parameters

The CONTINUE line has the same effect as consecutive SAVE and RESTART lines. The advantage of using CONTINUE instead of SAVE and RESTART is that it avoids the unnecessary steps of writing and then reading the disk file of particle coordinates. The CONTINUE line can only appear after a START or RESTART line in the input file. The format is:

CONTINUE $\Delta\Phi$, *NumberOfSteps*, *SpaceChargeSteps*, *OutputSteps*, ΔZ_{Ref}

where the parameters are the same as those found on the RESTART line.

40. ZLimit, drop particles based upon position

Use the ZLimit line to control when to stop transporting particles that are too far from the reference particle to matter anymore. The format is:

ZLimit Z_{Limit}

If a particle falls more than Z_{Limit} cm behind the reference particle, the code drops it from the calculation. A negative value of Z_{Limit} has the special meaning that particles beyond the magnitude of Z_{Limit} are not lost but are kept in the calculation. However, they are not necessarily included in the output. The reason is that the code writes the output buffer for a particular element when the reference particle is Z_{Limit} beyond the end of the element.

41. SPECIAL, defines special-purpose parameters

The SPECIAL line provides a way to customize Parmela for a variety of special applications. The format of the line, which can include up to 21 parameters, is:

SPECIAL γ_{frame}, Z_{center}, *FirstOutputSet1*, *LastOutputSet1*, 0, *FirstOutputSet2*, *LastOutputSet2*, $\Delta\Phi_{inject}$, $m_{inject}c^2$, q_{inject}, *FirstInjectSet1*, *LastInjectSet1*, *InjectionProbablity*, *FirstInjectSet2*, *LastInjectSet2*, *ParticleTypeSet2*, N_{max}, *SCfactor*, Z_{LB}, *TimeMultiplier*, *InjectionLoss*

Note the zero placeholder for the unused fifth parameter. You can omit trailing parameters that you do not need to define, but you should include zero placeholders for parameters that occur earlier in the list than one you are trying to define. The code uses only the last SPECIAL line that occurs before a START, RESTART, or CONTINUE line.

a. Defining the rest frame and position of the space-charge mesh

By default in Parmela, the space-charge mesh is centered on the reference particle and moves with the relativistic γ of the reference particle. With the SPECIAL line, you can

specify different values for either or both of these parameters. The relativistic γ for the mesh is γ_{frame}, and Z_{center} is the longitudinal center position of the space-charge mesh. Entering a value for Z_{center} prevents the space-charge mesh from moving with the reference particle. The mesh then needs to be large enough to encompass the whole area of interest.

b. Restricting particles in the output buffers

By default, all particles are included in the output buffers. You can use the SPECIAL line to restrict which particles the code includes in phase-space plots and other formats produced from the output buffers. This feature can help eliminate end effects from the results of a simulation. For example, simulation of a DC beam in the low-energy transport line to an rf accelerator will require additional beam particles before and after the region of interest. The DC beam might span many rf periods of the accelerator. The RFQOUT line selects 360-degree slices of the beam for input into an rf linac following the beam transport line.

Parameters *FirstOutputSet1* and *LastOutputSet1* are the first and last particle numbers of a contiguous set of particles. Parameters *FirstOutputSet2* and *LastOutputSet2* define another set of particles. Thus, one set could select the central particles added by a series of INPUT lines for one type of particle and charge, and for the other set one could select a portion of another ion species. New ion species are entered by redefining the *ParticleType* and *Charge* on a CHARGE line before a series of INPUT lines. The output particle sets are independent of each other. You can use either one or both of them in a Parmela run.

c. Control of space-charge neutralization in the beam

Parmela uses two different ways to simulate space-charge neutralization of a propagating ion beam. Both methods can be used during the same simulation, but not simultaneously in the same physical region. In the injection method, the code injects new particles into the beam at the current locations of certain subsets of the primary beam particles. In the assumed-neutralization method, you simply define the extent of the space-charge neutralization that has already occurred by some unspecified mechanism. To use the space-charge neutralization options in Parmela effectively requires close attention to available experimental beam measurements.

The idea of the injection method is to approximate a steady-state condition in which electrons created by collisions of positively-charged beam particles with residual gas molecules remain trapped in the beam. Of course, these collisions also produce positive ions. However, in the steady state, most positive ions, which are repelled radially outward by the beam's space charge, will have already left the beam. This assumption reduces the computation time that is necessary to allow the slow moving positive ions to achieve the steady-state condition. As an example, suppose an ion source produces a beam consisting of mostly protons and some molecular hydrogen ions. By injecting only electrons at locations of the two ion species in the beam, Parmela attains the steady-state condition with less computation time. When the assumption that positive ions have already left the beam is inadequate, Parmela also can inject a positive ion at the same time and place that it injects the electron. To use this feature, one must specify values for the parameters *FirstInjectSet2*, *LastInjectSet2*, and *ParticleTypeSet2*, which are defined below.

Parmela creates injected particles with initially zero velocity, but then subjects them to all the forces produced by the net space charge and the externally applied fields. $\Delta\Phi_{inject}$ is the interval in degrees between injection of particles. If you do not supply a value for $\Delta\Phi_{inject}$, the default injection interval is equal to $\Delta\Phi$, the integration step size. The code only checks whether to insert particles at each integration step, so $\Delta\Phi$ is the smallest injection interval. If you want to inject particles at regular intervals, then make $\Delta\Phi_{inject}$ an integer multiple of $\Delta\Phi$. The space-charge neutralizing particles have rest-mass energy $m_{inject}c^2$ and charge q_{inject}. There is no default value for $m_{inject}c^2$. You must supply a value for the rest-mass energy to use the injection method. If $q_{inject} = +1$, these particles have the same sign as particles entered with *Charge* = +1 on the CHARGE line preceding a series of INPUT lines. If $q_{inject} = -1$, they have the opposite sign. The injected particles use internal arrays reserved for particles with *ParticleType=3*. Do not use *ParticleType=3* on any CHARGE lines if you intend to use the injection method.

Parmela inserts neutralizing particles at the (x, y, z) coordinates of one or two contiguous sets of particle numbers. Parameters *FirstInjectSet1* and *LastInjectSet1* are the first and last particle numbers for one set, and *FirstInjectSet2* and *LastInjectSet2* are the first and last particle numbers for the second set. Parameter *InjectionProbablity* is the probability that a space charge neutralizing particle will be injected at a given site for the first set. There are no default values for these particle numbers. You must supply particle numbers for one of these sets, but the second set is optional. If the second set is specified, then a second neutralizing particle with opposite charge to the first will be injected. The type of particle injected for the second set is specified by *ParticleTypeSet2*, which can assume the same values (1, 2, or 3) as the parameter *ParticleType* on the CHARGE line.

When using the injection method, you can enter a value for N_{max} to restrict the maximum number of particles in the calculation. Otherwise, the default value for N_{max} is the current dimensioned size of the particle arrays in the code. The maximum number of space-charge neutralizing particles injected into the beam is N_{max} minus the number of particles added by all the INPUT lines. Parmela will drop some of the neutralizing particles from the calculation at intervals of $\Delta\Phi_{inject}$ if you supply a value for *InjectionLoss*, the last parameter on the SPECIAL line. The *InjectionLoss* parameter specifies the probability that a given space-charge neutralizing particle will be lost. The default value is zero, which means that particles are never lost.

For the assumed-neutralization method, *SCfactor* is the neutralization factor for particles with z values greater than the lower bound Z_{LB}. No neutralization corresponds to *SCfactor* = 1.0. Thus, a value of 0.02 would correspond to a beam whose space charge has been 98% neutralized. Parameter Z_{LB} (in cm) serves another purpose related to the particles added by the injection method. Parmela eliminates any space-charge neutralizing particles with z greater than Z_{LB}. A positive value on the ZLimit line can also serve the same purpose by eliminating space-charge neutralizing particles that fall too far behind the reference particle.

d. Time adjustment for different mass particles in a DC beam

Simulating a DC beam that consists of multiple ion species can be difficult because the lighter-mass particles will tend to outrun the heavier particles. *TimeMultiplier* is a correction applied only to particles added with *ParticleType*=2 on the CHARGE line and subsequent INPUT lines. The correction allows these particles to keep up with the

primary beam particles. Without this adjustment, a simulation of multiple ion species of different masses would not be feasible with limited computer resources.

An example best illustrates how to use the *TimeMultiplier* factor. Suppose an ion source produces a primary beam of H^+ ions. The H^+ beam accounts for 90% of the source's total output current. The other 10% of the ion-source current is in an accompanying H_2^+ beam. At the very low energies characteristic of ion sources, both particles will remain nonrelativistic even after substantial acceleration in a low-energy beam-transport line. Subjected to the same DC electrostatic fields, the protons and molecular ions will both gain energy at the same rate, but the proton velocity will increase $\sim\sqrt{2}$ times faster than the molecular-ion velocity. By setting *TimeMultiplier* ≈ 1.414 for the molecular ions, Parmela makes time advance faster for this beam keeping it in approximately the same physical space as the primary proton beam.

You must take care when using the *TimeMultiplier* parameter. This feature is not yet fully implemented for all the beam-line elements. At present, the *TimeMultiplier* affects only zero-length elements and the DRIFT element. It is not implemented in BEND, CELL, DTCELL, and SOLENOID elements.

42. END, stops the Parmela run

The END line indicates the end of the input file. The program stops after reading the END line.

C. Symbols for physical and logical variables

On the next several pages, Table IV-5 lists alphabetically all the physical and logical variables used with the input-file keywords. Greek symbols appear in the list where its spelled-out name would be. For example, "Φ_0" would be listed with words starting with letter p for "phi". The table order does not distinguish between upper and lower case. Numbers appear before letters, if applicable. Logical parameters and some integers use italics. In some cases, parameters that are always used together appear in the same entry. The spelling of the first parameter determines the entry's location in the table.

Physical parameters throughout the document use mathematical symbols. The individual sections on each keyword include the most complete description of these variables.

Table IV-5. Definition of physical and logical variables.

Variable	Description
A_1, A_2, A_3, A_4, A_5	Empirical coefficients for single-bunch beam loading.
α_x, β_x, ε_x,	Twiss or Courant-Snyder ellipse parameters for the x-x′ phase plane.
α_y, β_y, ε_y,	Twiss or Courant-Snyder ellipse parameters for the y-y′ phase plane.
α_z, β_z, ε_z,	Twiss or Courant-Snyder ellipse parameters for the z-z′ phase plane.
α_r	Angle of bend of the reference trajectory (degrees).
b	Impact parameter for 3-D point-to-point space-charge calculations (cm).
B	Magnetic field (Gauss).
B′	Magnetic field gradient in a QUAD element (Gauss/cm).
β_1, β_2	Leading and trailing edge angles on a BEND element (degrees).
$(\beta\gamma)_x$, $(\beta\gamma)_y$, $(\beta\gamma)_z$	Dimensionless momentum components in the x, y, and z directions.
B_x, B_y	Magnetic-field components for a STEERER element (Gauss).
CellType	Identifying number for cells.
Charge	CHARGE-line parameter that sets the particle charge for subsequent INPUT lines.
C(I)	Fourier coefficients used on CELL, DTCELL, and TRWAVE elements.
Config	Specifies the configuration of the fields for a CELL or DTCELL element.
$\delta\phi$	Phase displacement on INPUT lines (degrees).
$\Delta\phi$	Phase spread on INPUT lines, except INPUT 4 where it is phase with respect to reference particle; phase interval on ZOUT lines (degrees).
$\Delta\phi_{30}$	Optional phase range for finding average on INPUT 30 line (degrees).
$\Delta\Phi$	Integration step size (degrees).
$\Delta\phi_C$	Half width of the acceptable phase spread for a CHOPPER element (degrees).
$\Delta\Phi_{inject}$	Interval between injection of space-charge neutralizing particles (degrees).
$\Delta\Phi_{max}$	Maximum integration step size for some elements (degrees).
$\Delta\Phi_p$	A particle's remaining phase step in the dynamics calculations (degrees).
δr_{limit}	Limit on the maximum total radial displacement for ErrorType.
ΔR_{SC}	Total length of the radial space-charge mesh (cm).
$\delta\theta_{max}$	Tolerances on the angular displacement errors on ERRORS lines (degrees).
$\delta\theta_x$, $\delta\theta_y$	Angular displacement errors on ERRORS lines (degrees); or angular misalignments of the Poisson field magnetic axis from the beam axis (milliradians).
δW	Shift in the energy coordinate on INPUT lines (MeV).
ΔW	Energy spread on INPUT lines, except INPUT 4 where it is energy with respect to reference particle (MeV).
ΔW_{30}	Optional energy range for finding average on INPUT 30 line (MeV).
ΔW_{max}	Maximum energy gain in a BUNCHER element for a particle at the rf crest (MeV).
δx, δy	Spatial displacements on INPUT lines (cm).
$\delta x'$, $\delta y'$	Divergence-coordinate displacements on INPUT lines (milliradians).
δx, δy, δr	Displacements of an element on ERRORS lines (cm); may include subscripts 1 and 2 denoting the for beginning and end of the element.
δx_{max}, δy_{max}, δr_{max}	Tolerances on displacement errors on ERRORS lines (cm); may include subscripts 1 and 2 denoting the beginning and end of the element.
Δx, Δy	Spatial limits for transverse chopping on CHOPPER lines (cm).
X_{max}, Y_{max}, Z_{max}	Physical dimensions of 3-D space-charge grid.

Table IV-5. Definition of physical and logical variables. (continued)

Variable	Description
ΔZ_{SC}	For 2-D space-charge calculations, total length of the longitudinal space-charge mesh (cm). For 3-D, maximum distance of the bunch center from a flat cathode over which cathode image charges are computed.
E_0	Average axial electric field for a CELL, DTCELL, or TANK element (MV/m).
E_0T	Average axial electric field times transit-time factor used on TANK and TRWAVE elements (MV/m).
Element	An element number used on CHANGE lines
ErrorType	First parameter on an ERRORS line, indicating the type of errors to apply.
E_z, E_x, E_y	Electric field components written by a ZOUT line.
f_0	Basic frequency of the linac (MHz).
f_B	Frequency of a BUNCHER element (MHz).
f_C	Frequency of a CHOPPER element (MHz).
f_{Cell}	Frequency of a CELL element, default is f_0 on the RUN line (MHz).
FileType	Type of field map on Poisson and EM3DField lines (1 = electrostatic fields, 0 = magnetic fields).
f_{res}	A resonant frequency computed by the SUPERfish code (MHz).
f_{TR}	Frequency of a traveling-wave accelerator (MHz).
FirstElement	First element of a range for ADJUST and ERRORS lines.
FirstInjectSet1	First particle of contiguous set 1 where neutralizing particles are injected.
FirstInjectSet2	First particle of contiguous set 2 where neutralizing particles are injected.
FirstOutputSet1	First particle of contiguous set used in beam-loading calculations.
FirstOutputSet2	First particle of contiguous set added to output buffers.
g/2	Half gap height of the magnet in a BEND element (cm).
γ_{frame}	Relativistic γ for the rest frame of the space-charge mesh.
H_x, H_y, H_z	Magnetic field components written by a ZOUT line.
I_C	Total current of a COIL element (amps).
InjectionLoss	Probability that a given space-charge neutralizing particle will be lost.
InjectionProbability	Probability that neutralizing particles are injected at particle set 1 coordinates.
I_T	Total beam current on the SCHEFF line for space-charge calculations (amperes). Refers to particle current (as if all particles have charge state +1)
K_1, K_2	Constants used to calculate correction angles ψ_1 and ψ_2.
kT	Cathode temperature in units of electron volts (eV).
K_x, K_y, K_W	Wave numbers for a WIGGLER element (cm^{-1}).
L	Length of an element (cm).
λ_W	Wavelength of the WIGGLER period (cm).
LastElement	Last element of a range for ADJUST and ERRORS lines.
LastInjectSet1	Last particle of contiguous set 1 where neutralizing particles are injected.
LastInjectSet2	Last particle of contiguous set 2 where neutralizing particles are injected.
LastOutputSet1	Last particle of contiguous set used in beam-loading calculations.
LastOutputSet2	Last particle of contiguous set added to output buffers.
L_{Bunch}	For 2-D space-charge calculations, the repetition length or distance between adjacent bunches. For 3-D, longitudinal mesh length if both N_{Bunch} and L_{Bunch} are nonzero.
LinacType	Type of rf structure: 1 for disk-and-washer; 2 for side-coupled cavity; 6 for race-track microtron side-coupled cavity.

Table IV-5. Definition of physical and logical variables. (continued)

Variable	Description
L_{Quad}	Length of the quadrupole magnet for DTCELL elements (cm).
L_Z	Distance on either side of the reference particle for single-bunch beam-loading calculations (cm).
m_1c^2, m_2c^2, m_3c^2	Rest-mass energies of particles (MeV).
$m_{inject}c^2$	Rest-mass energy of injected space-charge neutralizing particles (MeV).
MapNumber	Unique identifier for field maps supplied by Poisson and EM3DField lines.
MeshFactor	Determines how often the space-charge routine SCHEFF regenerates the mesh.
MeshSize	The EGUN mesh size for Type 3 INPUT distributions (cm).
Multiplier	Scaling factor used for Poisson fields or ADJUST lines.
N_A	Number of particles to add to the distribution for an INPUT line.
N_b N_{b1}, N_{b2}	Base numbers for random and quasi-random loading of the particles for INPUT Type 9 distributions.
N_{Bunch}	For 2-D space-charge calculations, number of identical beam bunches upstream and downstream of the particle bunch. For 3-D, use cyclic boundary conditions in the z direction if N_{Bunch} is nonzero.
N_G	The number of "good" particles remaining in the calculation.
N_L	Lower dimension of the Fourier coefficient array $C(N_L:N_U)$ for TRWAVE elements.
N_{max}	Maximum number of particles allowed in the beam when using injection of space-charge neutralizing particles.
N_p	Particle number for Type 4 INPUT distributions.
N_{Print}	Number of print locations on ZOUT line.
N_R, N_Z	Number of radial and longitudinal intervals in the space-charge mesh.
N_{SB}	Number of cavities for a single-bunch beam-loading calculation.
N_T	The total number of macroparticles in the calculation.
N_U	Upper dimension of the Fourier coefficient array $C(N_L:N_U)$ for TRWAVE elements.
NumberOfCells	Number of identical cells in a TANK.
NumberOfFiles	Number of 360-degree time slices written by the RFQOUT element.
NumberOfLines	Number of lines of particle coordinate data for Type 40 INPUT distributions.
NumberOfRays	Number of lines to read from the EGUN file for Type 3 INPUT distributions.
NumberOfSteps	Number of integration steps to take.
N_x, N_y, N_z	Number of mesh points along the three grid axes for 3-D space-charge calculations.
OutputFlag	If nonzero, output is at end of an element.
OutputSteps	Number of master-clock phase steps between requested output.
ParticleType	CHARGE-line parameter that sets the rest-mass energy for subsequent INPUT lines.
ParticleTypeSet2	ParticleType for second space-charge neutralizing particle on the SPECIAL line.
PhaseLength	Number of π radians between adjacent cells in TANK and TRWAVE elements.
p_x, p_y	Transverse momentum components.
q_{inject}	Sign (±1) of injected space-charge neutralizing particles.
QuadOption	Quadrupole magnet configuration for DTCELL elements.
ϕ	A particle phase coordinate (usually in degrees).
ϕ_0	Phase of CELL, DTCELL, TANK, or TRWAVE element (degrees).
Φ_0	Starting phase angle for dynamics calculations (degrees).
ϕ_B	Phase of a BUNCHER element (degrees).

Table IV-5. Definition of physical and logical variables. (continued)

Variable	Description
ϕ_C	Phase of a CHOPPER element (degrees).
Φ_{delay}	Time since reference particle passed RFQOUT element (degrees).
ϕ_{max}	Cutoff phase for Type 9 INPUT distributions (degrees).
Φ_p	A particle's phase angle in the dynamics calculations (degrees).
ϕ_{rf}	A calculated phase of the rf fields for some elements (degrees).
ϕ_s	Synchronous phase for a TANK element (degrees).
PrintFlag	If PrintFlag is nonzero, Parmela also echoes each input line to the output file.
ψ_1, ψ_2	Fringe-field correction angles for edges of BEND elements (degrees).
R_1	Radius for evaluation of the E_z component in the rf field plot files.
R_2	Radius for evaluation of the E_r and B_ϕ components in the rf field plot files.
R_a	Size of the aperture radius at the exit of an element (cm).
R_C	Radius of a COIL element (cm).
$R_{cathode}$	Radius of curvature of the spherical cathode (cm).
R_{Edge1}, R_{Edge2}	Radii of curvature of the leading and trailing pole face edges (cm).
ρ	Radius of curvature of the reference trajectory in a BEND: $\rho = L/\alpha_r$.
RingOption	For 2-D space-charge calculations, controls the number charge rings, For 3-D, RingOption = 4 signals an image charge calculation within ΔZ_{SC} of a flat cathode.
r_{max}	Cutoff radius for Type 9 INPUT distributions (cm).
RunNumber	Identifying run number on the RUN line. (RunNumber >1000 sets the maximum number of beam-line elements in the calculation.)
R_{Wall}	Radius of the beam pipe for image charge calculations (cm). If zero, there is no wall.
σ_ϕ	Gaussian width parameter for the phase coordinate for Type 9 INPUT distributions (degrees).
σ_r	Radial Gaussian width for Type 9 INPUT distributions (cm).
SCfactor	Space-charge factor when using the assumed-neutralization method.
SpaceChargeSteps	Number of steps between space-charge impulses.
StepsPerPeriod	Number of integration steps through one WIGGLER period.
θ_R	Angle of rotation about the z axis for a ROTATE element (degrees).
TimeMultiplier	Time speed-up factor applied to particles with ParticleType = 2.
TRWcells	Number of cells in a traveling-wave linac tank.
TRWcellType	Index identifying the cell type in traveling-wave tanks.
TRWprint	Number of lines of data written to the rf field plot files.
TRWtankNumber	Index identifying the TRWAVE tank number.
Type	Identifying number for the type of INPUT particle distribution.
UseParticle1	If nonzero, Parmela uses particle 1 of an INPUT 30 distribution as the reference particle.
V	Voltage on the pole tips of an electrostatic quadrupole (kV).
W	A particle energy (usually in MeV).
W_0	Initial kinetic energy of the reference particle (MeV).
W_{30}	Optional central energy for finding average on INPUT 30 line (MeV).
W_B	Reference energy for a BUNCHER element (MeV).
WriteDistribution	If nonzero, Parmela writes a text file of an INPUT 30 particle distribution.
WigglerPeriods	Number of WIGGLER periods of length λ_W in the element length L.

Table IV-5. Definition of physical and logical variables. (continued)

Variable	Description
W_{min}, W_{max}	Optional kinetic energy limits for a CHOPPER element.
W_r	Reference energy for a BEND element (MeV).
x, y, z	Spatial coordinates (usually in cm).
x′, y′	Transverse divergence coordinates, actually ratios of momenta p_x/p and p_y/p.
x_{max}/y_{max}	Ratio of ellipse minor axes for INPUT 9 distributions.
X_{off}, Y_{off}	Displacements of the Poisson field magnetic axis from the beam axis (cm).
Z_0	Initial longitudinal position of reference particle (cm). The beginning of the first element is defined to be at z = 0.
Z_1, Z_2	Beginning and ending z coordinates for the ZOUT data.
Z_C	Longitudinal position of the center of a COIL element (cm).
Z_{center}	Longitudinal center position of a fixed space-charge mesh.
Z_{LB}	Lower bound for the assumed-neutralization method and above which injected neutralizing particles are dropped from the calculation (cm).
Z_{center}	Longitudinal center position of a fixed space-charge mesh.
Z_{LB}	Lower bound for the assumed-neutralization method and above which injected neutralizing particles are dropped from the calculation (cm).
Z_{Limit}	Distance from the reference particle at which particles are dropped from the calculation (cm).
Z_{min}, Z_{max}	Lowest and highest z coordinates over which to apply the COIL background fields (cm).
Z_{off}	Relative longitudinal position of the Poisson field to the position on the Parmela beam line (cm).

V. Pargraf, the Plotting Program for Parmela

After a Parmela calculation that included the OUTPUT 5 line in the input file, use Pargraf to display results of the calculation on the screen. Pargraf reads a control file (default name SIMPLE.PGF) and two files generated by Parmela. One file contains coordinates of particles at the end of beam-line elements (default name TAPE2.T2) and another contains similar data at the end of specified time steps (default name TAPE3.T3). Pargraf reads data from these files depending upon the type of output requested in the control file. (Pargraf plots particles of different mass or charge in different colors)

A. Starting program Pargraf

To start Pargraf, double-click on the control file with extension .PGF or use the following command line:

Pargraf *filename*

where *filename* is the name of the control file. Pargraf reads the file LANL.INI, which contains setting for other filenames used in the code. If you do not include *filename* on the command line, then Pargraf opens the control file specified in LANL.INI. It then opens the control file and begins to process the keyword lines.

Table V-1 lists the available Pargraf commands, which appear on the program menu. You also can enter keystroke commands in the usual manner for Windows program: by simultaneously pressing the Alt key and the active letter for the command. The active letter for each command is underlined on the menu.

Table V-1. The Pargraf program menu.

Menu item	Description
Graphics toggle	For future development. This command is not active at this time.
Exit	Exit Pargraf.
Start Hard Copy	Prints the screen to selected driver.
Driver	Popup menu to select an output driver.
Device	Popup menu to select non-Print Manager printers.
Options	Popup menu to configure the selected driver.
Units	Used to select units for some output drivers.
Next	Advance to next screen.
Previous	Go back to previous screen in same SUBNUM section.
Skip	Skip to next SUBNUM selection in the Pargraf control file.
Movie	Turn move mode on or off. Initial mode can be set in the file LANL.INI
View	Selects white or black background color, whether to display space charge neutralizing particles (if any), whether to display slice emittance plots. Initial mode can be set in the file LANL.INI

In the movie mode, all the screens are plotted one after another as fast as the computer can do them. To make a hardcopy, click on Start Hard Copy. To end the program, click on Exit. Environment variable BG_Color determines the initial background screen color.

The default color is black. Set BG_Color equal to WHITE for an initial white background. The background screen color has no effect on hardcopies.

1. Hardcopy of graphics screens

The Driver menu lists several software drivers for producing hardcopy graphics files or printing directly to a printer using the Windows Print Manager. Select Start Hard Copy to create the hardcopy output. Pargraf re-plots the present screen using the active driver. A check mark (✓) appears next to the currently selected driver. The software drivers have configurable options that you can edit from the Options menu.

The Options menu also includes the full path and name of the graphics output file (if any). You can choose a different name, if desired, but we recommend that you retain the suggested filename extension. In the case of BMP/PCX/PNG bit-image driver, the filename extension will correspond to one of these three graphic types. If you select a different graphic type in the Options menu, and if you do not alter the default filename in any way, then the code will generate the a new default filename with the proper extension. However, the new name will not immediately appear in the File field. If you close the Options dialog box and reopen it, the new filename will appear. If you modify the filename, be sure that the extension corresponds to the correct graphic type. The code does not check the extension if you override the default filename.

If the active driver is the BMP/PCX/PNG bit-image driver, then Pargraf sets the default image size to match the current screen size. If you resize the window, the image size also changes. If you want a different size image, you can override the settings Options menu and start a hardcopy before re-sizing the image.

The following sections describe the graphics export formats supported by Pargraf. Much of the information has been supplied with ***Winteracter*** development package of Interactive Software Services Ltd.

a. BMP (Windows Bitmap Format)

Windows Bitmap Format (BMP) is the Windows bitmap format, so many Windows packages support it. Commonly, BMP files are uncompressed, but 16 and 256 color BMP files can also be compressed (but not 16/24/32 bit color files). A few Windows packages do not support the compressed format. Most mainstream packages support compressed files, but the uncompressed format is slightly more universal. Uncompressed files can be very large, so use of the compressed format (or one of the other bitmap formats, PCX or PNG) is usually recommended. The Options dialog allows selection of both compressed and uncompressed BMP formats

b. CGM (Computer Graphics Metafile)

The Computer Graphics Metafile (CGM) standard was created to help standardize graphics output. In practice, it has proved sufficiently ambiguous that each graphics importer or exporter interprets the standard in a different way, thereby adding to the problem instead of solving it. CGM is potentially a good graphics export option since files are compact, it is widely supported as an import format, and its image description capabilities are relatively good. CGM is a vector based format. Only five line types are

supported, which is only a limitation with non-Windows packages (because Windows itself only supports 5 line types).

Font support among CGM importers is somewhat variable. The ***Winteracter*** CGM driver used in Pargraf does embed full font information in CGM files. The code writes Windows-style font names in the CGM files (e.g. "Courier New," "Arial," etc.) because they are understood by the widest range of importers among Windows based applications.

c. DXF (AutoCAD Drawing Exchange Format)

The AutoCAD Drawing Exchange Format (DXF) is a vector oriented graphics file format, mainly intended for use with CAD and other drawing packages such as AutoCAD and AutoSketch. It is also supported as an import format by a number of other word processors. The DXF format is extremely verbose, so output files tend to be very large. Font support is erratic amongst different importers, so only one standard "hardware" font is used. DXF provides some basic fill support (3 or 4 sided areas only). The ***Winteracter*** DXF driver supports the full AutoCAD set of 255 colors. Some packages support a more limited palette of 14 distinct colors, of which only the first 7 colors are compatible with the full AutoCAD set. The DXF driver Options dialog can select either 255, 14, or 7 colors depending on the target importer software capabilities.

d. PS (PostScript) and EPS (Encapsulated PostScript)

The PostScript (PS) driver can generate either standard PostScript files or Encapsulated PostScript (EPS) files. The EPS format is most useful when the finished output is destined for a PostScript printer. Most EPS importers make no attempt to actually decode the contents of an Encapsulated PostScript file. Instead, the EPS file is read "as-is", on the assumption that the PostScript output device will make sense of the file. Usually, an EPS file appears as an empty frame in most word processing programs. Two notable exceptions are Corel Draw 7 and GhostScript/GSView, both of which can interpret and display PS and EPS files.

e. HP-GL and HP-GL/2 (Hewlett Packard Graphics Language)

HP-GL is the graphics language used by Hewlett Packard pen plotters. Many packages claim to be able to read HP-GL, but each has its own interpreter and some are better than others. Some even vary between releases (e.g. WordPerfect for Windows v6.0 is completely unusable with virtually any HP-GL file). Despite this, HP-GL is a good format to try for simple line graphics of the type generated by Pargraf. Other formats are better if solid fills are required. The most common problem encountered when importing HP-GL is color selection. Pargraf translates color numbers 0 through 255 as indicted in Table V-2. There is no standard for what HP-GL pen numbers mean. They are simply numbered from 1 upwards. On most devices the pen numbers will produce the color shown in the table.

Table V-2. HP-GL Color Translation.

Color number	HP-GL pen	Color
16 to 47	2	red
48 to 79	4	yellow
80 to 111	3	green
112 to 143	7	cyan
144 to 175	5	blue
176 to 207	6	magenta
all other	1	black

HP-GL/2 is a later variant of HP-GL widely implemented on modern HP laser printers and high end inkjet printers. We do not recommend the use of HP-GL/2 as an export format, since most applications are very poor at importing this newer implementation of HP-GL. HP-GL/2 is best used as a hardcopy (rather than export) format, where it can produce reasonably high quality output.

f. PCX (ZSoft PC Paintbrush Format)

The Paintbrush PCX format is probably the one of the most reliable formats. The color depth of the saved file affects its size. Monochrome images are smallest. A few software packages do not recognize the 24-bit PCX format.

g. PNG (Portable Network Graphics)

Portable Network Graphics (PNG)is a bit image format developed as a replacement for the GIF file format. It supports much better compression than either PCX or BMP, but is currently (late 2001) less widely supported. PNG is a supported standard in Internet Explorer and Netscape. Thus, PNG is a good choice when publishing images on the web or on an intranet.

h. SVG (Scalable Vector Graphics)

The Scalable Vector Graphics (SVG) format has been defined by the World Wide Web Consortium (W3C) to allow web browser display of re-scaleable vector based images. As such, SVG is effectively a web equivalent of the PostScript or CGM formats. For viewing SVG files in your web browser download the free plug-in from the Adobe web site. SVG is an ASCII format that appears to be fully featured, offering a comprehensive and reliable set of 2D graphics capabilities.

i. WMF (Windows Metafile) and EMF (Enhanced Windows Metafile)

Windows Metafile (WMF) is the “native” Windows metafile format. Logically, WMF files are similar to CGM files, but their internal structure is very closely linked to that of the Windows application programming interface (API). As with most areas of Windows programming WMF exists in several variants. The basic 16-bit “Standard WMF” format is the most widely supported, with the notable exception of some versions of Word and Excel, which require a slightly different format known as an Aldus Placeable Metafile. Since most importers also accept Aldus format WMF files, the ***Winteracter*** WMF driver generates Aldus format files by default. In the HardCopy Option menu, you can select the

Standard WMF format or a third EMF ("Enhanced WMF") format. Microsoft introduced EMF for the Win32 API. Some popular applications (e.g. Word 7.0) cannot read EMF files. Most Windows packages support at least one variant of WMF as an import format, though their ability to read them correctly varies. The least reliable importers are those that need to understand the precise contents of the WMF file. Such applications (typically drawing packages) provide their own WMF interpreter. The more reliable importers are those that simply include the picture as an object, relying upon the appropriate Windows API function to replay the metafile.

In common with the rest of the Windows Graphics Device Interface (GDI), WMF files have just 5 line types, rather than the 7 nominally supported in ***Winteracter***. Certain line types will be indistinguishable from one another. Clipping of circles, arcs, ellipses and non-hatched polygons only works in Enhanced WMF files. Clipping is not handled correctly in 16-bit WMF files, so it is disabled in the ***Winteracter*** WMF driver when generating Standard or Aldus format files. A Windows API restriction limits 16-bit Standard WMF files to short (8.3 format) filenames. The Aldus variant of this format is not subject to this limitation, but the Windows API limits the Aldus file size to 16 MB.

B. Files used by Pargraf

Pargraf uses several input and output files. You can specify a filename and path of your choice. The names and paths listed in Table V-3 are in the [Pargraf] section of file LANL.INI.

Table V-3. LANL.INI settings for input and output files.

LANL.INI setting	Description	Default value
PargrafControl	Pargraf control file specifying plot options	SIMPLE.PGF
PargrafOut	Pargraf output text file [PRN for direct to printer]	OUTPAR.TXT
ElementOutName	Input file generated by Parmela containing coordinates of particles at the end of beam-line elements	TAPE2.T2
TimeStepOutName	Input file generated by Parmela containing coordinates of particles at the end of specified time steps	TAPE3.T3

C. Parmela graphics output

Program Pargraf uses the colors listed in Table V-4 for each of the three particle masses and the two signs of the electric charge. The parameters *ParticleType* and *Charge* appear on a CHARGE line preceding one or more INPUT lines in the Parmela input file. The masses themselves are defined on the RUN line.

Table V-4. Particle colors in Pargraf displays.

Color	*ParticleType*	*Charge*
Green	1	+
Red	1	–
Violet	2	+
Light white	2	–
Gray	3	+
Light green	3	–

Several graphics formats are available for use with Parmela. Two keywords in file LANL.INI specify the names for the particle-coordinate output files. ElementOutName sets the name of the file containing coordinates of particles at the end of beam-line elements (default name is TAPE2.T2). TimeStepOutName sets the name of a similar file for the end of specified time steps (default name is TAPE3.T3).

After Parmela has finished running, these files may be processed by Pargraf, which reads commands from its own control file (default name SIMPLE.PGF). To use Pargraf to plot results, you must run Parmela with the following OUTPUT line in the Parmela input file:

```
OUTPUT    5
```

Here is the form of the command lines in Pargraf control file:

```
SUBNUM    Graphtype
OUTPUT    Form, Writeflag, Element1, Element2, [Element3,… -or- Elementstep]
OPTCON    p1, p2, p3, …
BEGIN
END
```

Graphtype on the SUBNUM line is the type of graphics output. Possible values for *Graphtype* are 3, 5, 6, 7, 8, or 9. We use some of these integers for compatibility with graphics processors in other codes. Each *Graphtype* will be explained in detail later.

The *Form* parameter on the OUTPUT line must be either 1 or 2. For *Form* 1, each element number *Element1*, *Element2*, etc., refers to a specific single beam-line element. For *Form* 2, *Element1* and *Element2* are the first and last elements in a Fortran-like DO loop with step size of *Elementstep*. The default value for *Elementstep* is 1. Set *WriteFlag* = 0 to suppress output, or *WriteFlag* = 1 to generate output at the specified elements. To find element numbers, the best method is to use the ZOUT line in Parmela and refer to the PAROUT file, which will include a list of the beam-line elements.

OPTCON is an acronym for OutPuT CONstants. Parameters p_1, p_2, p_3, etc. on the OPTCON line control the graphics boundaries and other setup information for the plot. These parameters are explained in the subsections below, because they depend on the type of output selected.

1. SUBNUM 3, four phase-space plots

Use *Graphtype* 3 on the SUBNUM line to produce the following four plots:

1. The phase of each particle at end of the current element versus the phase of the particle at z=0.
2. The energy of each particle at end of the current element versus the phase of the particle at z=0.
3. y′ versus y for "good" particles at the end of the first element selected for display by the present OUTPUT line.
4. x′ versus x for "good" particles at the end of the first element selected for display by the present OUTPUT line.

The two plots of y′ versus y and x′ versus x generated at each element specified by a given OUTPUT line always correspond to the same longitudinal position. These plots display the phase space distribution of the particles that have survived to the current element, but the display corresponds to the coordinates that the particles had at the location of the first element selected for the *Graphtype* 3 displays. This type of plot is useful for determining the transverse acceptance of an accelerator or beam line.

The OPTCON line has the format:

OPTCON $\phi_{In,max}$, $\phi_{Out,max}$, $\Delta W_{Out,max}$, x_{max}, *Decimals*, x'_{max}, *Decimals*

where $\phi_{In,max}$ is the maximum phase in degrees plotted on the horizontal axis for the input phase, $\phi_{Out,max}$ is the maximum phase in degrees plotted on the vertical axis for the output phase, $\Delta W_{Out,max}$ is the maximum relative energy in keV plotted on the vertical axis, x_{max}, is the maximum position coordinate in cm, and x'_{max} is the maximum divergence coordinate in milliradians. *Decimals* sets the number of decimal digits to use for labels on the axis defined immediately before it on the OPTCON line. Using keV for the energy makes a convenient scale without requiring a choice for the number of decimals.

2. SUBNUM 5, x and r longitudinal profiles

Use *Graphtype* 5 on the SUBNUM line to produce the two profile plots: one of x versus z, and another of r versus z.

The OPTCON line has the format:

OPTCON z_{start}, z_{end}, *Zticks*, x_{max}, *Decimals*, r_{max}, *Decimals*, *Lines*, z_1, r_1, …

where z_{start} and z_{end} define the starting and ending z coordinates for the profiles. *Zticks* is number of intervals for tick marks and labels along the z axis. Parameters x_{max} and r_{max} set the vertical scales on the respective profiles, and *Decimals* sets the number of decimal digits to use for labels on the axis defined immediately before it on the OPTCON line. If $z_{start} = -z_{end}$, they refer to starting and ending z coordinates with respect to the reference particle position.

The rest of the parameters starting with *Lines* are optional. They define endpoints of lines drawn on the plots to show the location of the beam tube or other features along the accelerator. *Lines* is the number of z and r coordinate pairs to follow.

SUBNUM 5 now writes time-step emittance and beam-size information in the Pargraf output text file. The time-step emittance is calculated in the rotating frame of reference in the presence of a background solenoidal magnetic field (same as is done for SUBNUM 7 output). Small differences in α (especially for small α) may be observed between the

value calculated by SUBNUM 7 and that printed by SUBNUM 5. These differences are caused by space charge and external fields. For the time-step emittance, Parmela spreads out the beam in z, but all at same time. The normal emittance calculation has all the particles at one z position, but with different phases (times).

The values of the ellipse parameters α and β and the emittance can be completely erroneous inside elements whose fields have a large effect on the particle velocities (e.g. quadrupole magnets). Take special care with the interpretation of the time-step information.

Figure V-1 shows the heading that appears in the Pargraf output file before the tabulated time-step data. The column headings all appear on one line in the file, but for readability we show three lines here. Table V-5 lists the meaning of these column headings.

```
TIME-STEP position, emittance, and beam size data
Units are cm for <X>, rms X … rms Z; alpha is unitless;
beta has units of meter/radian; The emittance is unnormalized and has units of meter×1.0e-6.
Ngood   <Zpos(cm)>   <X>   exrms   alphax   betax   rms X
                     <Y>   eyrms   alphay   betay   rms Y
                     <E>   ezrms   alphaz   betaz   rms Z
```

Figure V-1. Start of the time-step emittance table.
The last three lines in this figure actually appear on one line in the Pargraf output file.

Table V-5. Column heading in the time-step emittance table.

Column heading(s)	Description
Ngood	Number of particles remaining in the calculation.
<Zpos(cm)>	Average value of the z coordinate of all the particles.
<X>, <Y>	Average value of the x and y coordinates of all the particles.
exrms, eyrms, and ezrms	Un-normalized rms emittance in x, y, and z directions.
alphax, alphay, and alphaz	Courant-Snyder α_{rms} in the x, y, and z directions.
betax, betay, and betaz	Courant-Snyder β_{rms} in the x, y, and z directions.
rms X, rms Y, and rms Z	The rms beam radius in the x, y, and z directions.
<E>	Average value of the energy of all the particles.

3. SUBNUM 6, input and output phase-space projections

Use *Graphtype* 6 on the SUBNUM line to produce input and output phase-space projections on the x-x′, y-y′, and Δϕ-ΔW planes. There are 16 parameters on the OPTCON line in the following order:

OPTCON x_{max}, *Decimals*, x'_{max}, *Decimals*, $\Delta\phi_{max}$, *Decimals*, ΔW_{max}, *Decimals*,
x_{max}, *Decimals*, x'_{max}, *Decimals*, $\Delta\phi_{max}$, *Decimals*, ΔW_{max}, *Decimals*

The first 8 parameters concern the input plots, and the last 8 are for the output plots, otherwise they are identical. Each pair of parameters defines a graph boundary and the number of decimal digits to include on the labels for that graph. The settings for x-x′ plots are also used for the y-y′ plots. Each scale has a total of 5 tick marks. For example,

if x_{max} = 0.02 and its corresponding *Decimals* setting is 2, the x scale will include tick marks with labels –0.02, –0.01, 0.00, 0.01, and 0.02.

4. SUBNUM 7, variety of phase-space and spectrum plots

Use *Graphtype* 7 on the SUBNUM line to produce up to four plots, one in each corner of the screen. There are several different *Choices* for the four plots as described in Table V-6. The format of the OPTCON line is:

OPTCON *Choice*, x_{max}, *Decimals*, x'_{max}, *Decimals*, $\Delta\phi_{max}$, *Decimals*, ΔW_{max}, *Decimals*, $P_{\Delta W}$

For each SUBNUM 7 line, the Pargraf output text file includes a table of either transverse emittance data (see Figure V-2, which shows x,y data) or longitudinal emittance data (see Figure V-3). If the OPTCON *Choice* is 4 or 6, then the y coordinate in Figure V-2 is replaced with the radial coordinate r. Table V-7 describes the meaning of the column headings for both transverse and longitudinal tables. As mentioned in the table, the user can select the percentage of beam particles to include in one of the emittance ellipses by setting the LANL.INI parameter EmittPer.

Table V-6. Choices for *Graphtype* 7 plots.

Choice	Emittance table	Plot 1	Plot 2	Plot 3	Plot 4
1	Longitudinal	Δϕ-ΔW	phase spectrum	energy spectrum	x-y
2	Transverse x,y	x-x′	y-y′	Δϕ-ΔW	x-y
3	Transverse x,y	x-x′	y-y′	Δϕ-ΔW	x and y profiles
4	Transverse x,r	x-x′	r-r′	r-r×ϕ′ *	x and r profiles
5	Longitudinal	x versus Δϕ	y versus Δϕ	x versus ΔW	Δϕ-ΔW
6	Longitudinal	x versus Δϕ	r versus Δϕ	r versus ΔW	Δϕ-ΔW
7	Transverse x,r	x-x′	r-r′	x-y	Δϕ-ΔW
8	Transverse x,y	x-x′	y-y′	Δϕ-ΔW	log(x) and log(y) profiles

* r×ϕ′ is the angular velocity.

Each pair of parameters defines a graph boundary and the number of decimal digits to include on the labels for that graph. The settings for the x axis are also used for y and r axes, and the settings for x′ axis are also used for y′ and r′ axes.. Each scale has a total of 5 tick marks. For example, suppose x_{max} = 0.02 and its corresponding *Decimals* setting is 2. The x scale will include tick marks with labels –0.02, –0.01, 0.00, 0.01, and 0.02.

If one or more of the parameters x_{max}, x'_{max}, $\Delta\phi_{max}$, and ΔW_{max} is zero, Pargraf will automatically find the corresponding maximum value from the particle distribution and set that parameter to a value consistent with the maximum value and the *Decimals* setting. A negative value of the *Decimals* setting is meaningful in this automatic mode.

The optional parameter $P_{\Delta W}$ affects the energy spread plot for *Choice* 1. $P_{\Delta W}$ is the full width of an energy band expressed as a percentage of the average beam energy. Pargraf displays near the bottom of the plot the largest percentage beam current that fits within the specified band width. For example, if $P_{\Delta W}$ = 5.5, the text line might read "energy spectrum, % in 5.5%=97.1," which is interpreted to mean that 97.1% of the particles lie

within an energy band of width equal to 5.5% of the mean energy. This energy band is not necessarily centered on the mean energy. The code adjusts the location of the energy band to enclose the maximum number of particles provided the band width is smaller than the full range of the energy spectrum graph. If $P_{\Delta W}$ exceeds the energy range of the graph, then the code substitutes the percentage that corresponds to the plotted range. If the beam energy is extremely small the code may have trouble with this calculation and not actually do the calculation.

The energy spectrum plot includes another text line of the form "rms(dKE)/KE=0.73%," which refers to the root-mean-square energy spread of the beam expressed as a percentage of the mean energy.

If the SLICE option is checked in the VIEW menu or if SLICE=YES in file LANL.INI, Pargraf will compute and plot the slice emittances. The code displays color coded plots of x-x′, y-y′, or r-r′ for a total of nine equal-length longitudinal slices.

			normalized emittance units are (cm-mrad)						|
nel	part	part	rms,n	emax,n	emax,n	rms,n	emax,n	emax,n	|
no.	in	out	x	x,99.9%	x,100%	y	y,99.9%	y,100%	|
	|		unnormalized (lab) alpha (unitless)			beta (cm/rad)			
	|alpha	beta	alpha	beta	position	unnormalized emittance (cm-rad)			
	| x	x	y	y	Z	rms,x	tot,x	rms,y	tot,y

Figure V-2. Table heading for transverse emittance data.
Though shown as two 3-line headings in this figure, these headings appear side by side in the Pargraf output file (separated by the vertical bars). The table has a total of 18 columns. Pargraf writes a transverse table for SUBNUM 7 with OPTCON *Choice* = 2, 3, 4, 7, or 8.

nel	part	part	z-emittance (deg-kev)			dpb	dwb	alpha	beta	position	z-emittance (cm) unnormalized	
	in	out	rms	99.9%	100%					Z	rms	100%

Figure V-3 Table heading for longitudinal emittance data.
Pargraf writes the longitudinal table for SUBNUM 7 with OPTCON *Choice* = 1, 5, or 6.

Table V-7. Description of headings for emittance tables.

Heading(s)	Description
nel	The element number.
part in	The total number of particles entering this element.
part out	The number of particles leaving this element.
rms,n p	Normalized rms emittance in cm-mrad for the phase plane indicated: p = x, y, or r for the x-x′, y-y′, or r-r′ phase plane.
emax,n p, 99.9%	Normalized emittance in cm-mrad containing 99.9% of the particles in the phase plane indicated: p = x, y, or r for the x-x′, y-y′, or r-r′ phase plane. The user has control of this percentage though LANL.INI parameter EmittPer, with allowed range 10.0% to 99.9%; default is 90.0%.
emax,n p, 100%	Normalized emittance in cm-mrad containing 100% of the particles in the phase plane indicated: p = x, y, or r for the x-x′, y-y′, or r-r′ phase plane.
alpha, beta p	Courant-Snyder ellipse parameters $\alpha_{p,rms}$ (dimensionless) and $\beta_{p,rms}$ (in cm/radian) for the rms emittance ellipse in the phase plane indicated: p = x, y, or r for the x-x′, y-y′, or r-r′ transverse phase plane. For the longitudinal emittance data, the alpha and beta refer to $\alpha_{z,rms}$ and $\beta_{z,rms}$. These are unnormalized quantities as measured in the lab frame.
position Z	Longitudinal position in cm at the end of the element.
unnormalized emittance rms,p tot,p	Unnormalized emittance in cm-radian for the rms ellipse and the ellipse containing 100% of the particles, each for the phase plane indicated: p = x, y, or r for the x-x′, y-y′, or r-r′ phase plane. Note: units of cm-radians are used in the Parmela input file.
z-emittance (deg-keV) rms 99.9% 100%	Longitudinal emittance in units of degrees-keV for the rms ellipse, and ellipses containing 99.9% and 100% of the particles. The user has control of the first percentage though LANL.INI parameter EmittPer, with allowed range 10.0% to 99.9%; default is 90.0%.
z-emittance (cm) rms 100%	Longitudinal emittance in units of cm-radian for the rms ellipse, and the ellipse containing 100% of the particles. These are unnormalized quantities as measured in the lab frame.
dpb	An acronym for “delta phase bar.” This is the difference in degrees between the average value of the phase and the phase of the reference particle.
dwb	An acronym for “delta W bar” where W is the symbol for particle energy. This is the difference in keV between the average energy and the energy of the reference particle.

5. SUBNUM 8, variety of longitudinal profiles

Use *Graphtype* 8 on the SUBNUM line to plot various *Choices* for two coordinates versus distance along transport system. Select the quantities to plot using *Choice1* and *Choice2* from Table V-8. The bar over a phase-space coordinate refers to the average value for all the particles. The format of the OPTCON line is:

OPTCON *Choice1*, *Choice2*, z_{start}, z_{end}, *Zticks*, x_{1max}, *Decimals*, x_{2max}, *Decimals*

where z_{start} and z_{end} define the starting and ending z coordinates for the profiles. *Zticks* is number of intervals for tick marks and labels along the z axis. Parameters x_{1max} and x_{2max} set the vertical scales on the *Choice1* and *Choice2* profiles, respectively. *Decimals* sets the number of decimal digits to use for labels on the axis defined immediately before it on the OPTCON line.

When plotting the x or y coordinates of all the particles (*Choice* 1 or 2 in Table V-8), Pargraf plots a pair of white points (black points on a white background) that correspond to the aperture of the element at each longitudinal position.

Table V-8. Choices for *Graphtype* 8 plots.

Choice	Quantity to plot
1	x coordinates of all particles
2	y coordinates of all particles
3	$\Delta\phi$ coordinates of all particles
4	ΔW coordinates of all particles
5	$\bar{x}$ and x_{max}
6	$\bar{y}$ and y_{max}
7	$\overline{\Delta\phi}$ and $\Delta\phi_{max}$
8	$\overline{\Delta W}$ and ΔW_{max}
9	x′ coordinates of all particles
10	y′ coordinates of all particles
11	x and y rms emittance for all particles
12	z rms emittance for all particles

6. SUBNUM 9, three longitudinal profiles of x, $\Delta\phi$, and ΔW

Use *Graphtype* 9 on the SUBNUM line to plot the three profiles of x, $\Delta\phi$, and ΔW versus distance along transport system. The format of the OPTCON line is:

OPTCON z_{start}, z_{end}, *Zticks*, x_{max}, *Decimals*, $\Delta\phi_{max}$, *Decimals*, ΔW_{max}, *Decimals*

where z_{start} and z_{end} define the starting and ending z coordinates for the profiles. *Zticks* is number of intervals for tick marks and labels along the z axis. Parameters x_{max} , $\Delta\phi_{max}$, and ΔW_{max} set the vertical scales on the three vertical axes. *Decimals* sets the number of decimal digits to use for labels on the axis defined immediately before it on the OPTCON line.

D. Emittance definitions in the Pargraf output text file

The reference particle is in the center of the plot of energy versus phase. In the x-x′ and y-y′ phase planes, the rms emittance is defined as:

$$\varepsilon_{x,rms} = \frac{1}{N}\sqrt{\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} x_i'^2 - \left(\sum_{i=1}^{N} x_i^2 x_i'^2\right)^2} \ ,$$

$$\varepsilon_{y,rms} = \frac{1}{N}\sqrt{\sum_{i=1}^{N} y_i^2 \sum_{i=1}^{N} y_i'^2 - \left(\sum_{i=1}^{N} y_i^2 y_i'^2\right)^2} \ ,$$

where N is the total number of particles, x_i and x'_i are the ith particle's space and divergence coordinates, and similarly for the y-y′ coordinates. For the longitudinal coordinates, the definition is:

$$\varepsilon_{z,rms} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\phi_i^2\sum_{i=1}^{N}W_i^2 - \left(\sum_{i=1}^{N}\phi_i^2 W_i^2\right)^2} \ ,$$

where ϕ_i is the phase of the particle and W_i is the particle energy. The longitudinal emittance may also be expressed in terms of positions and momenta. In the following equation δz_i and $\delta p_{z,i}$ refer to difference of the particle coordinates from the beam's average position and momentum:

$$\varepsilon_{z,rms} = \frac{1}{N}\sqrt{\sum_{i=1}^{N}\left(\delta z_i\right)^2\sum_{i=1}^{N}\left(\frac{\delta p_{z,i}}{mc}\right)^2 - \left(\sum_{i=1}^{N}\left(\delta z_i\right)\left(\frac{\delta p_{z,i}}{mc}\right)\right)^2} \ .$$

In the calculation of the rms emittances the effects of the average values of the particle coordinates are removed from the result. Note that some people define rms emittances with an additional factor of 4 in the above expressions.

For different SUBNUM options, Pargraf prints either unnormalized emittance ε_{rms} or normalized emittance $\varepsilon_{rms,n}$, but the α and β ellipse parameters are always unnormalized quantities in the lab frame. The headings indicate which convention is used for emittance, and what the units are. The conversion between the normalized quantities and the unnormalized laboratory-frame quantities is as follows:

$$\alpha_{rms} = \alpha_{rms,n} \ ,$$

$$\beta_{rms} = \left(1000\beta\gamma\right)\beta_{rms,n} \ ,$$

and

$$\varepsilon_{rms} = \frac{\varepsilon_{rms,n}}{1000\beta\gamma} \ .$$

where β without a subscript refers to the particle velocity in units of the velocity of light and γ is the particle energy in units of its rest mass. The Courant-Snyder α_{rms} is a dimensionless quantity and is not affected by the normalization.

VI. Utility Programs for Use with Parmela

We recommend that users of Parmela use the Poisson Superfish codes to compute rf and static fields for input to Parmela. (The Poisson Superfish codes must be downloaded by FTP and installed separately.) In addition to Poisson Superfish, there are a few companion programs for Parmela. Programs EFLD and EFLDTR calculate the Fourier coefficients used by CELL and TRWAVE lines, respectively. These two codes read interpolated fields from Superfish written by program SF7. Program ThreeDin reads a file of 3-D rf fields generated by MatLab and writes a binary file readable by the CField line in Parmela. Program MK_LAINI creates a new copy of the LANL.INI file.

A. EFLD, Fourier coefficients for CELL and DTCELL lines

Note: The EFLD program is an obsolescent code that we will eventually remove from the Parmela distribution. The recommended method for supplying field data to Parmela is to supply a field map using the CField line.

Parmela can generate electromagnetic fields for linac cells using an expansion of the field in a Fourier series. Program EFLD can produce a set of coefficients for any structure for which you can compute the fields. The directory LANL\Examples\Parmela\Fourier.C contains files that illustrate how to use program EFLD to generate the Fourier coefficients to include on the CELL line in Parmela. Input data for EFLD consist of fields interpolated in the bore region of the accelerating structure.

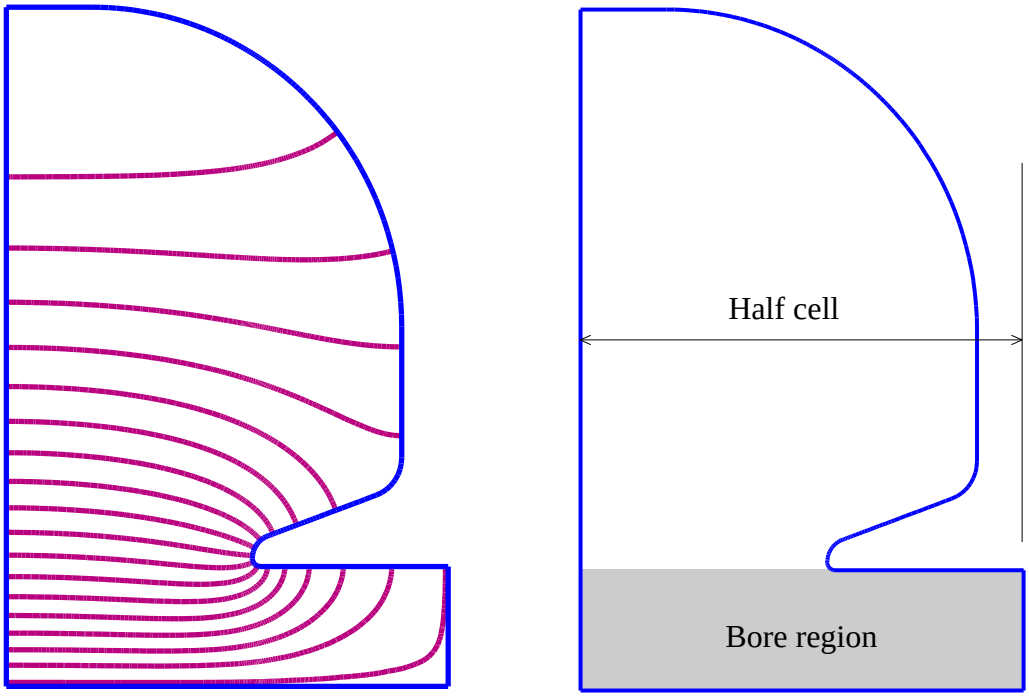

Figure VI-1. Coupled-cavity linac cell.
The left side shows the field contours calculated by Superfish in half of the coupled-cavity linac cell. Program SF7 interpolates fields in the shaded area at right for use in program EFLD.

The file 1300C is an input file for the Poisson Superfish tuning program CCLfish, which creates the input file 1300C1.AM for Automesh and other input files for Poisson Superfish postprocessors. (You must install Poisson Superfish before running this example.) After running CCLfish, program WSFplot will display briefly the field contours shown at left in Figure VI-1. The geometry is the right half of a typical coupled-cavity linac structure operating at 1300 MHz. With a Neumann boundary condition on the left edge and a Dirichlet boundary condition on the right edge of this geometry, Superfish computes fields for the π mode of the structure.

The next step after running the Superfish program is to run the postprocessor SF7 to compute the fields in the bore region of the accelerator. The region of interest is shown as the shaded area on the right of Figure VI-1. The batch file Run1300.BAT starts SF7 with input from file 1300C1.IN7. SF7 will write the file 1300C101.T7 (1300C1.T7 for Poisson Superfish versions earlier than 3.24) for input to Parmela. File 1300C1.IN7 specifies the corners of a rectangle (Z_{min}, R_{min}) and (Z_{max}, R_{max}) on which to interpolate the fields. For Parmela, R_{min} must be zero, and R_{max} should be the bore radius. The file uses $N_z = 50$ increments in the z direction and $N_r = 20$ increments in the r direction. You may select different increments, but you must be consistent among all the cell types in a Parmela input file. Table under the discussion of the CField line in Parmela lists the contents of a T7 file for SF7.

After running SF7, the batch file runs program EFLD with input from file 1300C101.T7. EFLD prompts for the resonant frequency of the accelerating structure. If the resonant

frequency computed by Superfish appears in the input file, the code uses it as the default value of the frequency. Press Enter in response to the prompt. The program writes the computed set of Fourier coefficients to the file FORCOEFF.TXT. All 14 Fourier coefficients (near the end of the file) must appear on the CELL or DTCELL line as parameters 20 through 30.

The SF7 output file with extension T7 contains data that can be supplied directly to Parmela using the CField line. This is the recommended method for supplying field data to Parmela. For this use, Parmela requires fields normalized to 1.0 MV/m. SF7 will provide this normalization automatically provided that the SFO program has already been run. No particular field normalization is required by ELFD.

B. EFLDTR, Fourier coefficients for TRWAVE lines

Note: The EFLDTR program is an obsolescent code that we will eventually remove from the Parmela distribution. The recommended method for supplying traveling-wave field data to Parmela is to supply field maps using the TRWCField line.

Parmela can generate electric and magnetic fields from an expansion in a Fourier-Bessel series (see G. A. Loew and R. B. Neal in "Linear Accelerators" edited by Lapostolle and Septier, pages 44 to 95). Directory LANL\Examples\Parmela\Fourier.TR contains files that illustrate how to use program EFLDTR to generate the Fourier coefficients to include on the TRWAVE line in Parmela. Input data for EFLDTR consist of fields interpolated in the bore region of the traveling-wave accelerating structure.

Batch file Run2PO3.BAT first runs Autofish with input from file 2PIOVR3.AF. (You must install Poisson Superfish before running this example.) After running Autofish, WSFplot displays briefly the field contours shown at left in Figure VI-2. The geometry is one and a half cells of a typical traveling-wave structure operating at 3 GHz. With Neumann boundary conditions on both the left and right edges of this geometry, Superfish computes fields for the $2\pi/3$ mode of the structure. Next, the batch file runs the postprocessor SF7 with input from file 2PIOVR3.IN7 to compute the fields in the bore region of the accelerator. The region of interest is shown as the shaded area on the right of Figure VI-2. SF7 will write the file 2PIOVR01.T7 for input to Parmela. File 2PIOVR3.IN7 specifies the corners of a rectangle (Z_{min}, R_{min}) = (0, 0) and (Z_{max}, R_{max}) = (5.0, 0.8643) on which to interpolate the fields. The file uses N_z = 100 increments in the z direction and N_r = 20 increments in the r direction. Table under the discussion of the CField line in Parmela lists the contents of a T7 file.

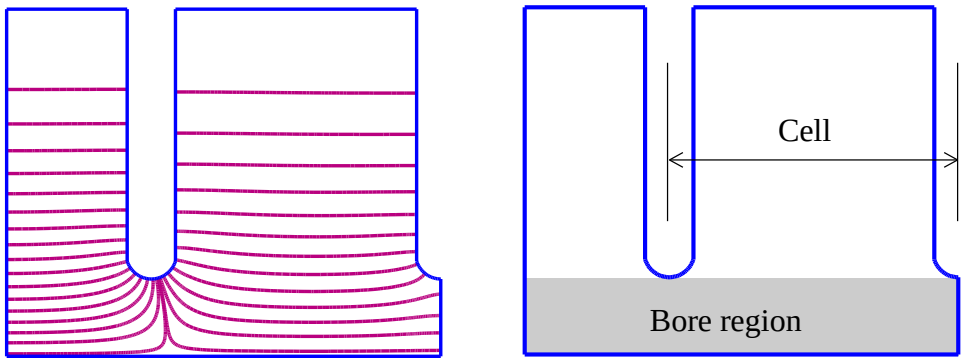

Figure VI-2. Traveling-wave accelerator.
The left side shows the field contours computed by Superfish for the 2π/3 mode. Program SF7 interpolates fields in the shaded area at right for use in program EFLDTR.

Finally the batch file runs program EFLDTR with input from file 2PIOVR01.T7. EFLDTR prompts for additional information listed in Table VI-1. Accept the default settings for these prompts, except or the cell length, which is 3.333 cm. The resonant frequency computed by Superfish, which appears in the input file, is the default value of the frequency. The default setting for the cell-to-cell phase shift is 120 degrees, which corresponds to the 2π/3 mode commonly used in traveling wave accelerators.

Table VI-1. Program prompts in EFLDTR.

Prompt	Default value
Input file	none
Frequency in MHz	f_{res} in input file
Cell length in cm	none
Phase shift in degrees per cell	120
Lowest order coefficient N_L	–5
Highest order coefficient N_U	5

The EFLDTR code generates a maximum of 11 consecutive coefficients for either the cosine or sine series expansion. The recommended input to Parmela is a symmetric set of coefficients for the cosine series. Table VI-2 lists the combinations of coefficients N_L and N_U that are recommended for Parmela. The variables N_L and N_U are the lower and upper dimensions of the Fourier coefficient array $C(N_L:N_U)$. For example, if C has terms in the range (–5, –4, …, 0, 1, …, 5), then N_L = –5 and N_U = +5. The last column in Table VI-2 gives the total number of coefficients that must be included on a TRWAVE line for each choice. EFLDTR writes the computed set of Fourier coefficients to the file FORCOEFF.

Table VI-2. Combinations of N_L and N_U recommended for Parmela.

N_L	N_U	Total
–5	+5,	11
–4	+4,	9
–3	+3	7
–2	+2	5
–1	+1	3
0	0	1

C. ThreeDin, writes 3-D field data readable by CField line

The Parmela installation includes a utility program called ThreeDin. File ThreeDin.EXE is an executable code that reads a file of 3-D rf fields generated by MatLab and writes a binary file readable by the CField line in Parmela. File ThreeDin.FOR in directory LANL\DeveloperFiles\Parmela is the self-documented source code for this utility program. The user can modify the code to read 3-D fields in other formats as needed. The CField line reads 3-D field maps created by ThreeDin if the number "3" appears on the line after the *CellType*.

If the file has extension TXT, then Parmela will open the file named after CField lines or EM3DField lines as a text file. Otherwise, the type of file depends on the setting of LANL.INI variable 3dTextFile. If you change the code or write your own code to produce text files, be sure to include enough significant digits in the numerical entries. In Fortran, we recommend a format such as G12.7, which will always contain 7 significant digits.

D. MK_LAINI: Create or update file LANL.INI.

Program MK_LAINI creates a new copy of the LANL.INI file. The installation program runs this code to update or create file LANL.INI. The installer leaves a copy of MK_LAINI in the installation directory. To start MK_LAINI use the following command line:

MK_LAINI [*directory*]

where *directory* is the optional destination directory. The destination directory must already exist before running MK_LAINI. If no directory appears the MK_LAINI creates the file in the current directory. To create file LANL.INI in the C:\LANL directory enter the following command:

MK_LAINI C:\LANL

The content of the resulting file depends upon whether a copy of LANL.INI already exists in the specified directory. If LANL.INI does not already exist in the destination directory, the new file will contain the default settings for a new installation. If LANL.INI does exist, then MK_LAINI will retain settings from the present copy and only add missing entries from the default installation set. The code also retains additional comment lines added by the user. Comment lines start with a semicolon (;) or exclamation mark (!).

Program MK_LAINI ignores any keywords and comments that appear after an invalid section heading. A section heading is the name of a program in square brackets (e.g., [Parmela]).

VII. Sample Input Files

Directory LANL\Examples\Parmela and several subdirectories listed in Table VII-1 contain example input files for Parmela and Pargraf as well as some sample files used with Poisson Superfish codes to create input files for Parmela. (You must install Poisson Superfish separately.) The interested user can run Parmela on the input files provided and study the output produced by the code for the various input options.

Each directory includes a batch file for running the codes. After Pargraf starts, continue to press Enter to step through all the displays. Some directories include a special copy of the initialization files LANL.INI or SF.INI (for Poisson Superfish) if the programs being run in that directory require a setting that differs from the default settings in these files as distributed.

Table VII-1. Contents of the LANL\Examples\Parmela directories.

Subdirectory name	Description
DEMO	Generic Parmela file for a photo injector followed by a traveling-wave linac. The INPUT.ACC file uses several available options in Parmela.
ATF	A photocathode RF gun problem using 3-D space charge. The example reads rf fields from a file produced by program SF7.
EPMIX	Sample files demonstrating the simultaneous transport of electrons and positrons.
Fourier.C	Sample files for generating Fourier coefficients for CELL or DTCELL lines in the Parmela input file.
Fourier.TR	Sample files for generating Fourier coefficients for TRWAVE lines in the Parmela input file.
SOLENOID	Sample files that show how to use program Poisson to produce an input file for the Poisson line in the Parmela input file.
TRWAVE	A simple traveling-wave linac example.

A. Parmela and Pargraf example files

The subdirectories described in Table VII-2 contain sample files for running the codes Parmela and Pargraf. The following sections describe briefly some interesting features of these examples.

1. The DEMO directory

The DEMO directory includes a sample input file for Parmela that demonstrates several features in the code. The file uses the default name Input.Acc and contains a one-cell photo injector followed by a traveling-wave linac. The batch file RUN_INP.BAT runs Parmela and Pargraf, each with input from the default input file: Input.Acc for Parmela, and SIMPLE.PGF for Pargraf. After running the batch file, the files in Table VII-2 will also be found in the DEMO directory. To save recalculation time, Parmela uses data in file SCGRID if it exists and if it corresponds to the current problem. If necessary, Parmela creates a new SCGRID file.

Table VII-2. Files produced by running RUN_INP.BAT.

File	Description
BEAMLOAD	Input for another code not part of the Parmela distribution.
OUTPAR.TXT	Parmela output file.
OUTGRAF.TXT	Pargraf output file.
SAVECORA	File of saved particle coordinates from the “SAVE 1” line.
SAVECORB	File of saved particle coordinates from the “SAVE 2” line.
SCGRID	Space-charge grid data used in Parmela.
TAPE2.T2	Parmela output of coordinates at the end of elements.
TAPE3.T3	Parmela output of coordinates at the end of time steps.
RFFLDnnn.TBL	Tablplot input file at rf phase nnn degrees (nnn = 000 to 315, steps of 045).

Figure VII-2 (parts a and b) shows the sample input file SIMPLE.PGF for program Pargraf. The Pargraf control files do not allow comment lines like Parmela input files. Blank lines are ignored. With a printed copy of this file in hand, you can run Pargraf after completing the Parmela run on file Input.Acc and identify the lines in the file that produce the successive screens.

```
run        1 1 1300. -0.002806 0.00001 9
title
 Example of one cell photo injector followed by a traveling wave acc.
; This example is only meant to demonstrate some of the elements
; in a parmela input file. It does not represent a real accelerator.
drift      0.0 2.0 1
cell       6.665 1.2 1 0.000 26. 1 5. -1 0 0 0
0.1551113E+01,-.1297884E+00,-.7242164E-01,0.4755327E-01,
-0.1684164E-01,0.4277525E-02,-.6665214E-03,-.1467097E-04,
0.8780015E-04,-.4094113E-04,0.1824233E-04,-.6261408E-05,
-0.1619545E-05,0.3140922E-05
drift      20 2.0 1
;  This is a comment line and is ignored.
cell       8.65 2.54 1 150.000 4.75 2 5. 1 0 0 0 ;this is a comment too
0.1819523E+01,0.8274025E+00,0.1129347E+00,-.5197137E-01, :this is a comment
-0.3271770E-01,-.7732846E-02,0.8754907E-03,0.1490549E-02,!this is too.
0.4951317E-03,-.1471407E-04,-.6959761E-04,-.2246184E-04,
0.8182504E-07,0.2409466E-05
trwave   4.325 3.19 1  60.000 7.52 2 5 1300 4 -5 5 0.75 5 0 0 0 0 0 0
0.000020  0.000897 -0.006674 -0.018529  0.557918  0.829216  0.023465
-0.013701  0.000897  0.000125 -0.000026
trwave   8.651 3.19 1  60.0 7.42 2 5 1300 4
trwave   8.651 3.17 1  60.0 7.43 2 5 1300 4
trwave   8.651 3.15 1  60.0 7.42 2 5 1300 4
trwave   8.651 3.13 1  60.0 7.47 2 5 1300 4
drift      10 1.4 1
drift      10 1.4 1
drift      10 1.4 1
drift      6 1.4 1
quad       7 1.4 1   123
drift      3 1.4 1
quad       14 1.4 1 -104.5
drift      3 1.4 1
quad       7 1.4 1   123
drift      10 1.4 1
drift      10 1.4 1
drift      10 1.4 1
drift      5 1.4 1
wiggler  1.25 0.5 1 0.5 40 -3041. 0.7071 0 0 0
wiggler  2.5 0.5 1 1 40 6082 0.7071 0 0 0
wiggler  10 0.5 1 4 40 6082 0.7071 0 0 0
wiggler  1.25 0.5 1 0.5 40 3041. 0.7071 0 0 0
```

Figure VII-1a. First part of the Parmela input file INPUT.ACC.

```
coil       -5. 7.0 -25000 0. 100.
coil       5. 7.0  25000
coil       50  10.  12000
coil       -50  10.  -12000
zout       500 45 0 75 1.
input      9 111 10. 0.500 1 5. 0.
input       111 10. 0.500 1 5. 4.4
input      9 111 10. 0.500 1 5. -4.4
input      9 111 10. 0.500 1 5. 3.3
input      9 111 10. 0.500 1 5. -3.3
input      9 111 10. 0.500 1 5. 2.2
input      9 111 10. 0.500 1 5. -2.2
input      9 111 10. 0.500 1 5. 1.1
input      9 111 10. 0.500 1 5. -1.1
output     5
; This part of input from this comment to next comment can be removed to
; restart with saved coordinates...
scheff     6.5 1. 1.5 12 200 0 0 5 0 0
start      21. 1 92 1 10
scheff     6.5 1. 1.5 12 100 0 0 3 0 0
continue    5. 200. 2 0
save       1
; ...to here. If the above input is removed or put after the END line,
; the problem can be restarted.
scheff     6.5 1.1 1.5 20 100 0 0 3 0 0
restart    30. 52. 2 0 0 1
save       2
scheff     6.5 0.3 1.5 20 100 0 0 3 0 0
continue    30 600. 2 0
end
```

Figure VII-1b. Remainder of the Parmela input file INPUT.ACC.

```
subnum  7
output    2 1 1 50
optcon    2 1. 1 100  0 10 0 100 0
begin
subnum  7
output    2 1 1 50
optcon    1 1. 1 100  0 10 0 100 0
begin
subnum  5
output    2 1 1 200
optcon     0. 8.0 8 1. 2 1. 2
begin
subnum  7
output    2 1 1 15
optcon    5 1. 1 100  0 20 0 100 0
begin
```

Figure VII-2a. First part of Pargraf control file SIMPLE.PGF.

```
subnum  7
output    2 1 1 15
optcon    6 1. 1 100 0 20 0 100 0
begin
end
subnum  9
output    2 1 1 50
optcon     0 150 15 0.75 3 180 0 500 0
begin
subnum  3
output    2 1 1 32
optcon    360 720 1000 0.5 1 100 0
begin
subnum  7
output    1 1 29
optcon    1 0.50 1 5 1 90 0 1000 0 12
begin
subnum  7
output    1 1 29
optcon    1 0.50 1 5 1 90 0 4000 0 17
begin
subnum  7
output    1 1 29
optcon    3 0.50 1 5 1 90 0 1000 0
begin
subnum  7
output    2 1 2 3
optcon    1 0.50 2 100 0 180 1 100 0 12
begin
subnum  7
output    2 1 4 6
optcon    1 0.50 2 100 0 45 1 200 0 12
begin
subnum  7
output    2 1 7 20
optcon    1 0.5 2 5 1 45 0 500 0 12
begin
subnum  7
output    2 1 21 32
optcon    1 0.5 2 5 1 45 0 1000 0 12
begin
subnum  7
output    2 1 1 5
optcon    2 0.5 2 200 0 180 0 100 0
begin
end
```

Figure VII-2b. Remainder of Pargraf control file SIMPLE.PGF.

2. The ATF directory

Subdirectory ATF contains input files listed in Table VII-3 for Parmela and Pargraf for a photocathode gun problem similar to one built for the Advanced Test Facility at

Brookhaven National Laboratory. The file ATFNM.ACC includes an option on the SCHEFF line that makes the code calculate the space charge using a point-to-point 3D method. (The impact parameter b on the SCHEFF line is nonzero.) A nonzero value for the impact parameter b in cm signals a point-to-point space-charge calculation.

Table VII-3. Files in the ATF directory.

File	Description
ATFNM.ACC	Sample Parmela input for a photocathode gun
ATFNM.T7	A file from program SF7 read by Parmela after a CField line. The file contains field data for the photocathode gun problem.
ATFNM.PGF	Sample input file for Pargraf used with ATFNM.
LANL.INI	Special version of the initialization file that specifies the ATF directory as the path to the CField file ATFNM.T7.
RUN_ATF.BAT	Batch file for running the codes

```
run       1 2 2856. -0.00113 1.e-5
title
ATF PHOTOCATHODE RF GUN + TRANSPORT LINE, NO DIPOLES, 3-D POINT-TO-POINT SC
INPUT  9 999 0.3 0.9 2.06 6.18
DRIFT  0 1 1
CELL   2.625 1. 1 0. 63.75 1 5 -1
cfield    1
atfnm.t7
CELL   5.25 1. 1 180. 63.75 1 10.
drift     2.22 5 1
quad      12.065 5.08 1 -84.7
drift     0.6 5 1
quad      20.83 5.08 1 74.86
drift     0.6 5 1
quad      12.065 5.08 1 -84.7
drift     10.0      5 1
drift     10.0      5 1
drift     10.0      5 1
drift     10.0      5 1
drift     10.0      5 1
zout      500 90 0 8 0.8
output    5
scheff    2.86 1 3 10 100 0 0 5 1.5 0 0.01
start     39. 1. 330 1 10
scheff    0.86 2 4 10 100 0 0 3 1.5 0 0.01
continue    15 1500 1 0
end
```

Figure VII-3. The Parmela input file ATFNM.ACC.

3. The EPMIX directory

Subdirectory EPMIX contains input files listed in Table VII-4 for Parmela and Pargraf that show how to use feature that allows simultaneous transport of electrons and positrons. Please note that in this release a few features in Parmela and Pargraf have not yet been updated to work with two beams of opposite charge. In Pargraf, some of the

displays do not show the two charges in different colors. The colors in Pargraf may be added in a future release.

Table VII-4. Files in the EPMIX directory.

File	Description
EPMIX.ACC	Sample Parmela input with both electrons and positrons
EPMIX.PGF	Sample Pargraf control file for EPMIX.ACC
RUN_MIX.BAT	Batch file for running the codes

4. The TRWAVE directory

Table VII-5 lists files in subdirectory TRWAVE that simulate a simple traveling-wave linac with Parmela and Pargraf. Batch file RUN_TRW.BAT runs the codes necessary to produce input files for the CField and TRWCField lines in Parmela input file TRAVELWV.ACC (see Figure VII-4). The files SLAC1.AF and SLAC_C.AF, and SLAC_C.AF are Autofish input files for the SLAC accelerating structure, which operates in the $2\pi/3$ mode at 2856 MHz. Geometrical parameters for the SLAC structure are reported by A. L. Eldridge, A. V. Lisin, and V. G. Price, "Accelerating Structure Technology," in Linear Accelerators, edited by Pierre M. Lapostolle and Albert L Septier, North-Holland Publishing Company, pages 265 to 313 (1970). See Table 1 on page 270 of this article.

Table VII-5. Files in the TRWAVE directory.

File	Description
RUN_TRW.BAT	Double-click this file to run the codes.
SLAC1.AF	Autofish input file for the first half cell of the SLAC traveling-wave linac with a bore tube attached.
SLAC1.IN7	SF7 input file for interpolating fields in the bore tube of the SLAC1.AF geometry.
SLAC_C.AF	Autofish input file for computing fields for the cosine solution in the $2\pi/3$ mode of the SLAC traveling-wave linac. The cosine solution has Neumann boundary conditions on the left and right edges of the geometry.
SLAC_C.IN7	SF7 input file for interpolating fields in the bore tube of the first half cell of the SLAC_C.AF geometry.
SLAC_S.AF	Autofish input file for computing fields for the sine solution in the $2\pi/3$ mode of the SLAC traveling-wave linac. The sine solution has Dirichlet boundary conditions on the left and right edges of the geometry. Also, the value of ENORM has been adjusted to produce the same stored energy as the cosine solution.
SLAC_S.IN7	SF7 input file for interpolating fields in the bore tube of the first half cell of the SLAC_S.AF geometry. The file is identical to SLAC_C.IN7.
TRAVELWV.ACC	Sample Parmela input for a traveling-wave linac.
TRAVELWV.PGF	Sample Pargraf control file for TRAVELWV.ACC.

Files SLAC_C.AF and SLAC_S.AF contain the same geometry of one and a half cells of the traveling-wave structure. SLAC_C.AF uses boundary conditions appropriate for the cosine solution for the $2\pi/3$ mode and SLAC_S.AF uses boundary conditions for the sine solution. File SLAC1.AF contains a bore tube attached to a half cell of the structure to simulate the accelerating fields from the first cell that penetrate into the bore tube. Batch

file RUN_TRW.BAT first runs Autofish, WSFplot, SF7, and Tablplot on both the SLAC_C.AF and SLAC_S.AF problems. (You must have already installed the Poisson Superfish Version 7.12 or higher before running this example.) These steps produce files SLAC_C01.T7 and SLAC_S01.T7, which Parmela reads when it encounters the TRWCField line.

The next step runs Autofish, WSFplot, SF7, and Tablplot on the SLAC1.AF problem to generate the Superfish fields for a half cell of the structure with a beam tube attached. These steps produce file SLAC101.T7, which Parmela reads when it encounters the CField line.

After running the Parmela program on input file TRAVELWV.ACC, the batch file starts Pargraf to view the results of the beam-dynamics simulation. Press Next to step through all the plots. The batch file continues by displaying briefly in program Tablplot the fields used during the Parmela calculation. The ZOUT line in the Parmela input file causes Parmela to write a series of Tablplot input files named RFFLDnnn.TBL, where nnn is an rf phase in degrees. The setting $\Delta\phi = 30$ degrees on the TRWAVE line generates a plot file every 30 degrees in phase over an rf cycle. You can check that there is a smooth transition at z= 5.25 cm, which is the end of the CELL-line fields and the start of the TRWAVE-line fields. Try different values of E_0T on the CELL line to see the effect of a bad match in the field amplitudes. The setting $E_0T = 3.4$ gives an acceptable match. The 8th parameter on the CELL line is *Config* = +1, which indicates that the fields are for a nonsymmetric full cavity.

```
run        1 1 2856 -6.4 0.100 0
title
Small example of traveling wave accelerator

Cell       5.24637 1.311 1 90.000 3.4  1 5. 1
CField   1
SLAC101.T7
Trwave  1.74879 1.311  1 0.000 7.52 1 5 2856 5 -5 5 0.666667 7
TRWCField 5
SLAC_C01.T7
SLAC_S01.T7
trwave   3.49758  1.311 1 0.000 7.42 1 5 2856 5
trwave   3.49758  1.311 1 0.000 7.32 1 5 2856 5
trwave   3.49758  1.311 1 0.000 7.22 1 5 2856 5
trwave   3.49758  1.311 1 0.000 7.1  1 5 2856 5
trwave   3.49758  1.311 1 0.000 7.0  1 5 2856 5
trwave   3.49758  1.311 1 0.000 6.9  1 5 2856 5
coil 10. 20. 10000. 0. 60.
coil -10. 20. -10000.
zout 500 30 0 69. 1
zlimit -25.
input 6 4999 0. 60.0 0.01 0. 60. 0.01 180. 0.
output 5
scheff 2. 2. 12.642 10 20 1 0. 3
start -90. 10. 130. 1 10
end
```

Figure VII-4. The Parmela input file TRAVELWV.ACC.

B. Creating field input for Parmela

Subdirectory Solenoid contains files listed in Table VII-6 that show how to use program Poisson Superfish codes to produce a 2-D field map for input to Parmela. The file SOLL2.AM, taken from the LANL\Examples\Magnetostatic\Solenoid directory describes a solenoid magnet with an additional "bucking coil" that has the current adjusted to cancel the magnetic field from the other coil near R = 0 (which corresponds to Y = 0 for a Poisson problem in cylindrical coordinates). The SF7 run generates files BFLD.QKP and SOLL2.T7. File SOLL2.T7 is the Parmela input file. Enter its name after a Poisson line in the Parmela input file. File BFLD.QKP is a Quikplot input file for plotting some of the interpolated fields in SOLL2.T7. (These files also can be produced by running Poisson or Pandira if SF.INI contains the setting ParmelaFields = Yes.)

Table VII-6. Files in the Solenoid directory

File	Description
RUNSOLL2.BAT	Batch file for running codes Automesh, Poisson, WSFplot, SF7, and Quikplot.
SOLL2.AM	Automesh input file for a solenoid with bucking coil.
SOLL2.IN7	Input command file for postprocessor SF7.

C. Generating Fourier coefficients for Parmela

We provide two sets of example files for generating the Fourier coefficients. One example is for a 1300-MHz coupled-cavity linac problem for which the code EFLD computes Fourier coefficients for the CELL line in Parmela. The other example is for a traveling-wave structure at 3 GHz, operating in the $2\pi/3$ mode. The code EFLDTR computes Fourier coefficients for the TRWAVE line for this problem.

Table VII-7 lists the sample files in directory Fourier.C and Table VII-8 list files in directory Fourier.TR. For more information about these examples refer to the sections on utility programs EFLD and EFLDTR.

Table VII-7. Files in directory Fourier.C.

File	Description
1300C.CCL	Input file for the Poisson Superfish tuning program CCLfish. The problem is a 1300-MHz coupled-cavity linac cell.
1300C1.IN7	Input file for program SF7 for the problem 1300C1.AM generated by program CCLfish.
RUN1300.BAT	Batch file for running CCLfish, SF7, and EFLD to create Fourier coefficients for the CELL line.

Table VII-8. Files in directory Fourier.TR.

File	Description
2PIOVR3.AF	Input file for the Poisson Superfish code Autofish. The file describes one and a half cells of a traveling-wave linac operating at 3 GHz.
2PIOVR3.IN7	Input file for program SF7 for the problem 2PIOVR3.AF.
RUN1300.BAT	Batch file for running Autofish, SF7, and EFLDTR to create Fourier coefficients for the TRWAVE line.

VIII. Information about other codes

We refer to several other computer codes in this documentation. The sections below describe what we know about how to acquire some of these codes. The information is from "Computer Codes for Particle Accelerator Design and Analysis: A Compendium," by Helen K. Deaven and K. C. Dominic Chan, LA-UR-90-1766 (1990).

A. ISIS, particle-in-cell code

ISIS is a multidimensional, fully relativistic, electromagnetic, particle-in-cell code. It uses a charge-conserving algorithm to accumulate currents, eliminating the need for Poisson correction. Other features include body-fitted coordinates and some three-dimensional capability. According to the 1990 compendium, ISIS was last modified in July 1989. At that time the code was maintained and distributed, without charge, by:

Michael Jones
Los Alamos National Laboratory
Applied Theoretical Physics Division
Group X-1, Mail Stop E531
Los Alamos, NM 87545
Phone: 505-667-7760
Email: mej@lanl.gov

B. EGUN, electron-gun code

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

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

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

C. TBCI, wake-field interactions with beam bunches

TBCI, by Thomas Weiland, calculates wake fields and analyzes the electromagnetic interaction between bunched beams of charged particles moving through cylindrically symmetric cavities. The default Gaussian shape function for the bunch may be replaced by other shapes. Various postprocessors can subtract a tube wake field from the total wake, Fourier transform fields and wakes from data at every time step, and calculates the impedance.

The original stand-alone TBCI is apparently no longer available. The TBCI calculation is now part of the MAFIA family of codes, maintained and distributed, for a fee, by:

CST GmbH
Lauteschlaegerstr. 38
D-64289 Darmstadt
Germany
Phone: +49 (6151) 16 21 61
Fax: +49 6151 718057
Email: r137@temf00.temf.e-technik.th-darmstadt.de