



PAUL SCHERRER INSTITUT

PSI-PR-08-02

The OPAL Framework

(Object Oriented Parallel Accelerator Library)

Version 1.1.9¹

User's Reference Manual

Andreas Adelmann, Achim Gsell, Christof Kraus (PSI)
Yves Ineichen (IBM),
Steve Russell (LANL),
Yuanjie Bi, Chuan Wang, Jianjun Yang (CIAE),
Hao Zha (Tsinghua University)
Suzanne Sheehy, Chris Rogers (RAL) and
Christopher Mayes (Cornell)


Abstract

OPAL is a tool for charged-particle optics in accelerator structures and beam lines. Using the MAD language with extensions, OPAL is derived from MAD9P and is based on the **CLASSIC** class library, which was started in 1995 by an international collaboration. IPPL (Independent Parallel Particle Layer) is the framework which provides parallel particles and fields using data parallel ansatz. OPAL is built from the ground up as a parallel application exemplifying the fact that HPC (High Performance Computing) is the third leg of science, complementing theory and the experiment. HPC is made possible now through the increasingly sophisticated mathematical models and evolving computer power available on the desktop and in super computer centres. OPAL runs on your laptop as well as on the largest HPC clusters available today.

The OPAL framework makes it easy to add new features in the form of new C++ classes. It comes in the following flavours:

OPAL-CYCL tracks particles with 3D space charge including neighbouring turns in cyclotrons with time as the independent variable.

OPAL-T is a superset of IMPACT-T [40] and can be used to model guns, injectors and complete XFEL's excluding the undulator.

It should be noted that not all features of OPAL are available in all flavours. The icon  OPAL-T means that a feature is not yet available in OPAL-T. Similar icons are used for the other flavours.

¹Release Date: February 13, 2013

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 13 |
| 1.1 | Aim of OPAL and History | 13 |
| 1.2 | Parallel Processing Capabilities | 13 |
| 1.3 | Quality Management | 14 |
| 1.4 | Field Maps from the Femaxx 3D Eigenmode Solver | 15 |
| 1.5 | Output | 16 |
| 1.6 | Acknowledgements | 17 |
| 1.7 | Citation | 17 |
| 2 | Conventions | 19 |
| 2.1 | Physical Units | 19 |
| 3 | Tutorial | 21 |
| 3.1 | Starting OPAL | 21 |
| 3.2 | Restart Mode | 21 |
| 3.3 | Autophase Example | 21 |
| 3.4 | Examples of Beam Lines | 22 |
| 3.4.1 | PSI XFEL 250 MeV Injector | 23 |
| 3.4.2 | PSI Injector II Cyclotron | 24 |
| 3.4.3 | PSI Ring Cyclotron | 28 |
| 4 | OPAL-T | 31 |
| 4.1 | Introduction | 31 |
| 4.2 | Variables in OPAL-T | 31 |
| 4.3 | Integration of the Equation of Motion | 32 |
| 4.4 | Envelope Tracker | 32 |
| 4.5 | Space Charge | 32 |
| 4.6 | Wake Fields | 33 |
| 4.7 | Multiple Species | 33 |
| 5 | OPAL-CYCL | 35 |
| 5.1 | Introduction | 35 |
| 5.2 | Tracking modes | 35 |
| 5.2.1 | Single Particle Tracking mode | 35 |
| 5.2.2 | Tune Calculation mode | 36 |
| 5.2.3 | Multi-particle tracking mode | 36 |
| 5.3 | Variables in OPAL-CYCL | 36 |
| 5.3.1 | The initial distribution in the local reference frame | 37 |
| 5.4 | Field Maps | 37 |
| 5.4.1 | CARBONCYCL type | 38 |

| | | |
|----------|--|-----------|
| 5.4.2 | CYCIAE type | 39 |
| 5.4.3 | BANDRF type | 40 |
| 5.4.4 | Default PSI format | 40 |
| 5.4.5 | user's own fieldmap | 40 |
| 5.5 | RF field | 41 |
| 5.5.1 | Read RF voltage profile | 41 |
| 5.5.2 | Read 3D RF fieldmap | 41 |
| 5.6 | Particle Tracking and Acceleration | 41 |
| 5.7 | Space Charge | 41 |
| 5.8 | Multi-bunches Issues | 42 |
| 5.9 | Input | 42 |
| 5.10 | Output | 43 |
| 6 | Command Format | 45 |
| 6.1 | Statements and Comments | 45 |
| 6.2 | Identifiers or Labels | 46 |
| 6.3 | Command Attribute Types | 46 |
| 6.4 | String Attributes | 47 |
| 6.5 | Logical Expressions | 48 |
| 6.6 | Real Expressions | 49 |
| 6.7 | Operators | 50 |
| 6.8 | Operands in Expressions | 52 |
| 6.8.1 | Literal Constants | 52 |
| 6.8.2 | Symbolic constants | 52 |
| 6.8.3 | Variable labels | 52 |
| 6.8.4 | Element or command attributes | 53 |
| 6.8.5 | Deferred Expressions and Random Values | 53 |
| 6.8.6 | Table References | 54 |
| 6.9 | Element Selection | 54 |
| 6.9.1 | Element Selection | 54 |
| 6.9.2 | Range Selection | 55 |
| 6.10 | Constraints | 56 |
| 6.11 | Variable Names | 56 |
| 6.12 | Regular Expressions | 56 |
| 6.13 | Token List | 57 |
| 6.14 | Arrays | 57 |
| 6.14.1 | Logical Arrays | 57 |
| 6.14.2 | Real Arrays | 57 |
| 6.14.3 | String Arrays | 60 |
| 6.14.4 | Token List Arrays | 60 |
| 7 | Control Statements | 61 |
| 7.1 | Getting Help | 61 |
| 7.1.1 | HELP Command | 61 |
| 7.1.2 | SHOW Command | 61 |
| 7.1.3 | WHAT Command | 62 |
| 7.2 | STOP / QUIT Statement | 62 |
| 7.3 | OPTION Statement | 62 |
| 7.4 | Parameter Statements | 64 |
| 7.4.1 | Variable Definitions | 64 |
| 7.4.2 | Symbolic Constants | 66 |

| | | |
|-----------|---|-----------|
| 7.4.3 | Vector Values | 66 |
| 7.4.4 | Assignment to Variables | 67 |
| 7.4.5 | VALUE: Output of Expressions | 67 |
| 7.4.6 | H5merge | 68 |
| 7.5 | Miscellaneous Commands | 68 |
| 7.5.1 | ECHO Statement | 68 |
| 7.5.2 | SYSTEM: Execute System Command | 68 |
| 7.5.3 | SYSTEM Command under UNIX | 68 |
| 7.6 | TITLE Statement | 69 |
| 7.7 | File Handling | 69 |
| 7.7.1 | CALL Statement | 69 |
| 7.7.2 | SAVE Statement | 69 |
| 7.7.3 | MAKESEQ Statement | 69 |
| 7.8 | IF: Conditional Execution | 70 |
| 7.9 | WHILE: Repeated Execution | 70 |
| 7.10 | MACRO: Macro Statements (Subroutines) | 70 |
| 8 | Elements | 73 |
| 8.1 | Element Input Format | 73 |
| 8.2 | Common Attributes for all Elements | 73 |
| 8.3 | Drift Spaces | 75 |
| 8.4 | Bending Magnets | 76 |
| 8.5 | Quadrupole | 80 |
| 8.6 | Sextupole | 81 |
| 8.7 | Octupole | 81 |
| 8.8 | General Multipole | 81 |
| 8.9 | Solenoid | 83 |
| 8.10 | Cyclotron | 84 |
| 8.11 | RF Cavities (OPAL-T and OPAL-CYCL) | 86 |
| 8.11.1 | OPAL-Tmode | 86 |
| 8.11.2 | OPAL-CYCLmode | 86 |
| 8.12 | Traveling Wave Structure | 88 |
| 8.13 | Monitors | 89 |
| 8.14 | Collimators | 91 |
| 8.14.1 | OPAL-Tmode | 91 |
| 8.14.2 | OPAL-CYCLmode | 92 |
| 8.15 | Septum (OPAL-CYCL) | 93 |
| 8.16 | Probe (OPAL-CYCL) | 93 |
| 8.17 | Stripper (OPAL-CYCL) | 94 |
| 9 | Beam Lines | 97 |
| 9.1 | Simple Beam Lines | 97 |
| 9.2 | Sub-lines | 97 |
| 10 | Physics Commands | 99 |
| 10.1 | BEAM Command | 99 |

| | |
|--|------------|
| 11 Distribution Command | 101 |
| 11.1 Correlations for Gaussian Distribution (Experimental) | 103 |
| 11.1.1 Example | 104 |
| 11.2 Thermal Emittance | 104 |
| 11.3 Flattop Distribution | 105 |
| 11.3.1 Legacy Mode | 106 |
| 12 Fieldsolver | 107 |
| 12.1 Fieldsolver Command | 107 |
| 12.2 Define the Fieldsolver to be used | 107 |
| 12.3 Define Domain Decomposition | 107 |
| 12.4 Define Number of Gridpoints | 107 |
| 12.5 Define Boundary Conditions | 107 |
| 12.6 Define Greens Function | 107 |
| 12.7 Define Bounding Box Enlargement | 108 |
| 12.8 Define Geometry | 108 |
| 12.9 Define Iterative Solver | 108 |
| 12.10 Define Interpolation for Boundary Points | 109 |
| 12.11 Define Tolerance | 109 |
| 12.12 Define Maximal Iterations | 109 |
| 12.13 Define Preconditioner Behaviour | 109 |
| 12.14 Define the number of Energy Bins to use | 109 |
| 13 Wakefields | 111 |
| 13.1 Wakefield Command | 112 |
| 13.2 Define the Wakefield to be used | 112 |
| 13.3 Define the wakefield type | 112 |
| 13.4 Define the number of bins | 113 |
| 13.5 Define the bunch length to be constant | 113 |
| 13.6 Define the conductivity | 113 |
| 13.7 Define the impedance | 113 |
| 13.8 Define the form of the beam pipe | 113 |
| 13.9 Define the radius of the beam pipe | 113 |
| 13.10 Define the σ of the beam pipe | 113 |
| 13.11 Define the relaxation time (τ) of the beam pipe | 113 |
| 13.12 Import a wakefield from a file | 113 |
| 13.13 Wake Functions | 114 |
| 13.14 Filters | 114 |
| 14 Geometry | 117 |
| 14.1 Geometry Command | 117 |
| 14.2 Define the Geometry File | 117 |
| 14.3 Define the Length | 117 |
| 14.4 Define the Start | 118 |
| 14.5 Define the Semi-Major Axis | 118 |
| 14.6 Define the Semi-Minor Axis | 118 |
| 15 Tracking | 119 |
| 15.1 Track Mode | 119 |
| 15.1.1 Track a Random Machine | 121 |

| | |
|---|------------|
| 16 Field Emission | 123 |
| 16.1 Field Emission Command | 124 |
| 17 Multipacting | 125 |
| 17.1 Commands Related to Multipacting Simulation | 128 |
| 17.2 Run Parallel Plate Benchmark | 131 |
| 17.3 PostProcessing | 133 |
| 18 Physics Models Used in the Particle Matter Interaction Model | 135 |
| 18.1 The Energy Loss | 135 |
| 18.2 The Coulomb Scattering | 136 |
| 18.2.1 Multiple Coulomb Scattering | 136 |
| 18.2.2 Large Angle Rutherford Scattering | 136 |
| 18.3 The Flow Diagram of <i>CollimatorPhysics</i> Class in OPAL | 137 |
| 18.3.1 The Substeps | 140 |
| 18.4 Example of an Input File | 140 |
| 18.5 A Simple Test | 140 |
| A Installation | 143 |
| A.1 Build and install OPAL on a Mac & Linux | 143 |
| A.1.1 Supporting Libraries | 143 |
| A.1.2 Environment Variables | 143 |
| A.1.3 Installing OPAL | 145 |
| A.2 Cray XE6 Installation | 145 |
| A.2.1 .bash_profile.ext File | 145 |
| A.2.2 .bashrc.ext File | 146 |
| A.2.3 OPAL | 146 |
| A.3 Using pre-build Binaries | 148 |
| A.4 Enabling the Multigrid Space Charge Solver | 148 |
| A.5 Debug Flags | 150 |
| A.6 OPAL as a Library | 150 |
| A.7 Examples | 151 |
| B OPAL Language Syntax | 153 |
| C OPAL-T Field Maps | 163 |
| C.1 Introduction | 163 |
| C.2 Types and Format | 164 |
| C.3 1DMagnetoStatic | 169 |
| C.4 AstraMagnetostatic | 170 |
| C.5 1DDynamic | 171 |
| C.6 AstraDynamic | 172 |
| C.7 1DProfile1 & 1DProfile2 | 173 |
| C.8 2DElectroStatic | 176 |
| C.9 2DMagnetoStatic | 177 |
| C.10 2DDynamic | 178 |
| C.11 3DDynamic | 179 |

List of Tables

| | | |
|------|---|-----|
| 2.1 | Physical Units | 20 |
| 6.1 | String Operator in OPAL | 48 |
| 6.2 | String Function in OPAL | 48 |
| 6.3 | Logical Operators in OPAL | 49 |
| 6.4 | Real Operators in OPAL | 50 |
| 6.5 | Real Functions in OPAL | 50 |
| 6.6 | Real Functions with one in OPAL | 51 |
| 6.7 | Real Functions of Arrays in OPAL | 52 |
| 6.8 | Predefined Symbolic Constants | 53 |
| 6.9 | Real Array Functions in OPAL(acting component-wise) | 59 |
| 7.1 | Default Settings for Options | 65 |
| 11.1 | Different distributions specified by a single parameter m | 101 |
| 11.2 | Parameters for the DISTRIBUTION command | 102 |
| 11.3 | Parameters of the distribution command | 103 |
| 12.1 | Fieldsolver command summary | 108 |
| 12.2 | Preconditioner behaviour command summary | 109 |
| 13.1 | Wakefield command summary | 112 |
| 14.1 | Geometry command summary | 117 |
| 15.1 | Commands accepted in Tracking Mode | 119 |
| 16.1 | Field Emission Command summary | 124 |
| 17.1 | Multipacting Related Command Summary | 130 |
| A.1 | Debug flags. | 150 |

List of Figures

| | | |
|------|--|-----|
| 1.1 | Parallel efficiency and particles pushed per μs as a function of cores | 14 |
| 1.2 | Comparison of energy and emittance in x between IMPACT-Tand OPAL-T | 15 |
| 1.3 | Tetrahedral mesh of a pillbox shaped cavity | 16 |
| 3.1 | Reference orbit(left) and tune diagram(right) in Injector II | 26 |
| 3.2 | Radial and vertical eigenellipse at 2 MeV of Injector II | 27 |
| 3.3 | Energy Vs. time (left) and external B field Vs. trackstep (Right, only show for about 2 turns) . . . | 29 |
| 3.4 | Vertical phase at different energy from left to right: 0.87 MeV, 15 MeV and 35 MeV | 29 |
| 3.5 | Reference orbit(left) and tune diagram(right) in Ring cyclotron | 29 |
| 5.1 | 2D field map on the median plane with primary direction corresponding to the azimuthal direction, secondary direction to the radial direction | 38 |
| 8.1 | Visualisation of angles used to rotate the bend relative to the incoming beam, where \mathbf{n} is the normal of the face. | 79 |
| 8.2 | Schematic of the simplified geometry of a cavity gap and parameters | 87 |
| 8.3 | The on-axis field of an S-band TRAVELINGWAVE structure | 88 |
| 11.1 | OPAL Gauss-Flatop-Distribution | 105 |
| 17.1 | Typical SEY curve | 126 |
| 17.2 | Sketch map of the secondary emission process. | 126 |
| 17.3 | Time evolution of electron number predicted by theoretical model and OPAL simulation using Furman-Pivi's secondary emission model with both constant simulation particle approach and real emission particle approach at $f = 200MHz$, $V_0 = 120V$, $d = 5mm$ | 131 |
| 17.4 | Time evolution of electron number predicted by theoretical model and OPAL simulation using Vaughan's secondary emission model with both constant simulation particle approach and real emission particle approach at $f = 1640MHz$, $V_0 = 120V$, $d = 1mm$ | 131 |
| 18.1 | The comparison of stopping power with PSTAR. | 135 |
| 18.2 | The comparison of Coulomb scattering with Jackson's book. | 137 |
| 18.3 | The diagram of CollimatorPhysics in OPAL. | 138 |
| 18.4 | The diagram of CollimatorPhysics in OPAL(Continued). | 139 |
| 18.5 | The passage of protons through the collimator. | 141 |
| 18.6 | The energy spectrum and scattering angle at $z=0.1$ m | 141 |
| C.2 | The longitudinal phase space after a gun simulation using a 1D field map (on-axis field) of the gun, a 1D field map (on-axis field) of the gun in combination with the FAST switch, and a 2D field map of the gun generated by Poisson/Superfish. | 168 |
| C.3 | Example of a 1DMagnetoStatic field map | 169 |

| | | |
|------|--|-----|
| C.4 | Example of an ASTRA compatible magnetostatic field map | 170 |
| C.5 | Example of a 1DDynamic field map | 171 |
| C.6 | Example of an ASTRA compatible dynamic field map | 172 |
| C.7 | The profile of a rectangular bend and its corresponding design path. | 173 |
| C.8 | FIXME The location and definitions of the Enge function coefficients for a sector bend and its corresponding design path. | 174 |
| C.9 | Example of a 1DProfile1 field map | 174 |
| C.10 | Example of a 1DProfile2 field map | 175 |
| C.11 | Example of a 2DElectroStatic field map | 176 |
| C.12 | Example of a 2DMagnetoStatic field map | 177 |
| C.13 | Example of a 2DDynamic field map | 178 |
| C.14 | Example of a 3DDynamic field map | 179 |

Chapter 1

Introduction

1.1 Aim of OPAL and History

OPAL is a tool for charged-particle optics in accelerator structures and beam lines. Using the MAD language with extensions, OPAL is derived from MAD9P and is based on the [CLASSIC](#) class library, which was started in 1995 by an international collaboration. IPPL (Independent Parallel Particle Layer) is the framework which provides parallel particles and fields using data parallel ansatz. OPAL is built from the ground up as a parallel application exemplifying the fact that HPC (High Performance Computing) is the third leg of science, complementing theory and the experiment. HPC is made possible now through the increasingly sophisticated mathematical models and evolving computer power available on the desktop and in super computer centers. OPAL runs on your laptop as well as on the largest HPC clusters available today.

The OPAL framework makes it easy to add new features in the form of new C++ classes.

OPAL comes in the following flavours:


- OPAL-MAP (More details will be given in Version 1.1.9)
- OPAL-CYCL
- OPAL-T
- OPAL-E

OPAL-MAP tracks particles with 3D space charge using split operator techniques, and is a proper subset of MAD9P. In the future a linear space charge mode will become available allowing the user to track moments of the distribution.

OPAL-CYCL tracks particles with 3D space charge including neighbouring turns in cyclotrons with time as the independent variable.

OPAL-T is a superset of IMPACT-T [40] and can be used to model guns, injectors and complete XFEL's excluding the undulator.

It should be noted that not all features of OPAL are available in all flavours.

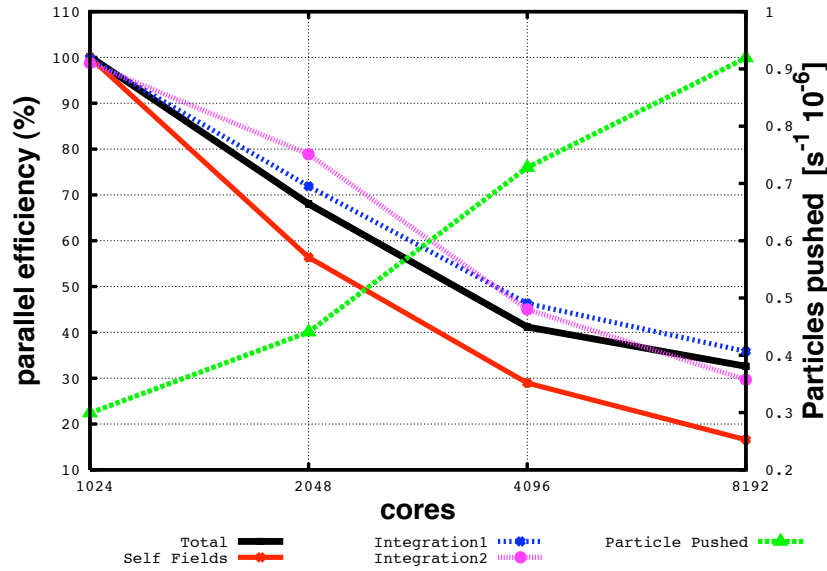
The following icon  OPAL-T means that a feature is not yet available in OPAL-T. Similar icons are used for the other flavours.

1.2 Parallel Processing Capabilities

OPAL is built to harness the power of parallel processing for an improved quantitative understanding of particle accelerators. This goal can only be achieved with detailed 3D modelling capabilities and a sufficient number of

Table 1.1: Parameters Parallel Performance Example

| Distribution | Particles | Mesh | Greens Function | Time steps |
|--------------|-----------|----------|-----------------|------------|
| Gauss 3D | 10^8 | 1024^3 | Integrated | 10 |

Figure 1.1: Parallel efficiency and particles pushed per μs as a function of cores

simulation particles to obtain meaningful statistics on various quantities of the particle ensemble such as emittance, slice emittance, halo extension etc.

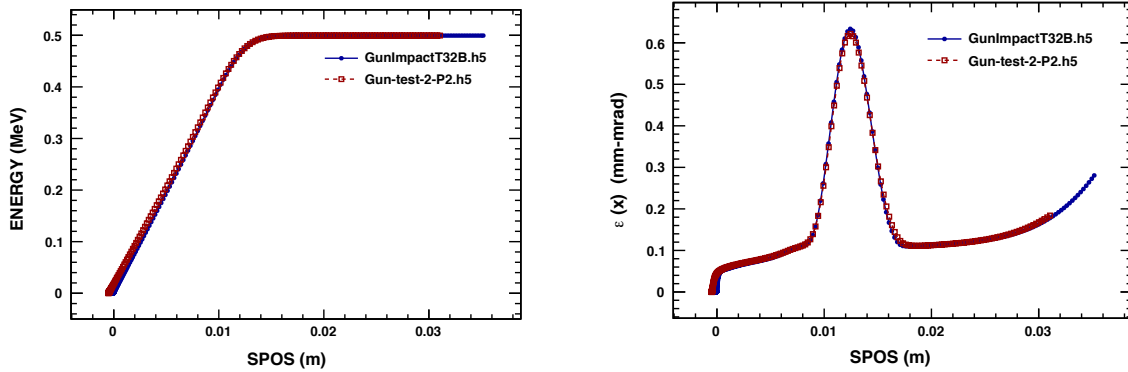
The following example is exemplifying this fact:

Figure 1.1 shows the parallel efficiency time as a function of used cores for a test example with parameters given in Tab. 1.1. The data were obtained on a Cray XT5 at the Swiss Center for Scientific Computing.

1.3 Quality Management

Documentation and quality assurance are given our highest attention since we are convinced that adequate documentation is a key factor in the usefulness of a code like OPAL to study present and future particle accelerators. Using tools such as a source code version control system ([subversion](#)), source code documentation (Doxygen) (check out, for instance, [ParallelTracker](#)) and the extensive user manual you are now enjoying, we are committed to providing users as well as co-developers with state-of-the-art documentation to OPAL.

One example of a non trivial test-example is the PSI DC GUN. In Figure 1.2 the comparison between IMPACT-T and OPAL-T is shown. This example is part of the regression test suite that is run every night. The inputfile is found in Section ??.

Figure 1.2: Comparison of energy and emittance in x between IMPACT-Tand OPAL-T

1.4 Field Maps from the Femaxx 3D Eigenmode Solver

[TODO: AA will rewrite]

Electromagnetic field maps for beam dynamics calculations originate from a number of different electromagnetic solvers, e.g. Superfish and similar codes.

Here, we describe the current status of work in progress which will, eventually, allow the usage of field maps that have been computed with the femaxx 3-dimensional electromagnetic eigenmodal solver [47, 48].

The femaxx code computes electromagnetic eigenmodes of resonant cavities of arbitrary 3-dimensional shape and boundary conditions.

Unlike Superfish and similar 2-dimensional codes, femaxx is not restricted in the kind of geometry it can model. It is therefore possible to consider arbitrary shapes and their inclusion in beam dynamics and particle tracking calculations.

Given a mesh of a 3-dimensional geometry femaxx computes eigenmodal field decompositions.

The user then specifies sampling locations for the electromagnetic eigenfields.

At present, sampling locations are specified in terms of a cylinder shape, i.e. the user indicates the cylinder axis, the radial cylinder vector and the number of sampling locations in axial, radial and azimuthal directions.

Once the eigenmodal solution has been computed the fields are sampled at these locations and stored in the T7 file format, for subsequent use in OPAL.

Considerable effort has been spent for the validation and benchmarking of beam dynamics calculations based on T7 field maps computed with femaxx.

A pillbox cavity, i.e. a cylinder shape with a radius $r = 4.7\text{cm}$ and height $h = 3\text{ cm}$, has been chosen for benchmarking purposes, due to the availability of an analytical solution.

The analytical resonance frequency of the dominant mode is 2.441 GHz.

We have compared two cases with OPAL: (1) The analytical solution has been sampled within a cylinder volume, stored into a T7 file and used in an OPAL run; (2) the same pillbox shaped geometry has been discretization into tetrahedra and the eigenmodal fields were calculated with femaxx. These two cases were then compared, resulting in the following conclusions: (1) Using a relatively coarse mesh with some 110'000 tetrahedra, the difference between the analytical and the numerical solution was usually smaller than 1 percent. (2) Using an adaptively refined mesh, the difference between analytical and numerical solutions decreased below 1 pro mille. The mesh is shown in the figure (1.3). (3) It is therefore imperative to use a tetrahedral mesh which has been refined around the beam axis. It is definitely more efficient to use local refinement, based on physical argument, than simply refine the complete mesh in a uniform manner.

We are now working towards benchmarking more complicated shapes in order to assess requirements w.r.t to meshes and modeling geometry so that we achieve the same or better accuracy as has been obtained from field maps that were computed with Superfish like solvers based on azimuthal symmetry.

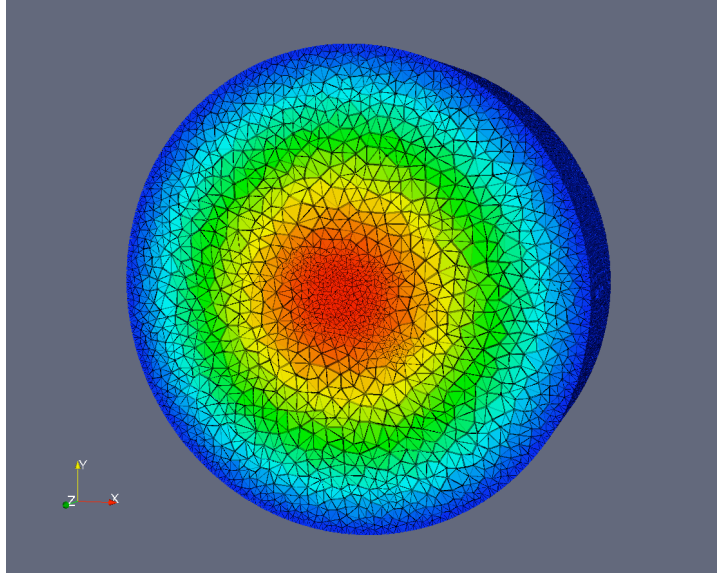


Figure 1.3: We show the discretization of a pillbox shaped cavity geometry into a tetrahedral mesh. The mesh has been adaptively refined so that the region around the cylinder axis is decomposed into smaller tetrahedra than those which are further away from the axis.

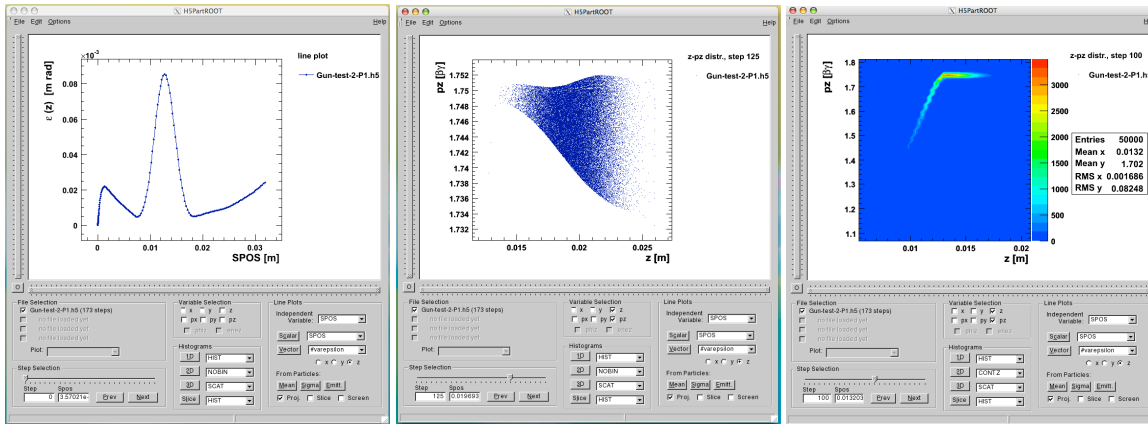


Figure 1.4: H5PartROOT enables a variety of data analysis and post processing task on OPAL data

1.5 Output

The data is stored in the H5hut file-format (<http://h5part.web.psi.ch/>) and can be analysed using the H5PartRoot (<http://amas.web.psi.ch/tools/H5PartROOT/index.html>). The frequency of the data output (phase space and some statistical quantities) can be set using the `OPTION` statement (see §7.3), statement and the flag `PSDUMPFREQ`. The file is named like in input file but the extension is `.h5`.

An ASCII file with statistical beam parameters is stored in a `.stat` file. An SDDS file output is in preparation.

1.6 Acknowledgements

The contributions of various individuals and groups are acknowledged in the relevant chapters, however a few individuals have or had considerable influence on the development of OPAL, namely Chris Iselin, John Jowett, Julian Cummings, Ji Qiang, Robert Ryne and Stefan Adam. For the H5PARTROOTvisualization tool credits go to Thomas Schietinger. The effort to couple FEMAXX to OPAL was led by Benedikt Oswald. Parts of the envelope tracker was contributed by Colwyn Gulliford.

Misprints and obscurity are almost inevitable in a document of this size. Comments and *active contributions* from readers are therefore most welcome. They may be sent to andreas.adelmann@psi.ch.

1.7 Citation

Please cite OPAL in the following way:

```
@techreport{Opal-User-Guide,  
title = "{The OPAL (Object Oriented Parallel Accelerator Library)  
Framework }",  
author = "A. Adelmann and Ch. Kraus and Y. Ineichen and S. Russell  
and Yuanjie Bi and J. Yang",  
institution = "Paul Scherrer Institut",  
number = "PSI-PR-08-02",  
year = {2008}}
```

Chapter 2

Conventions

2.1 Physical Units

Throughout the computations OPALuses international units (see Tab. 2.1), as defined by SI (Système International).

Table 2.1: Physical Units

| quantity | dimension |
|---------------------------------|------------------------|
| Length | m (metres) |
| Angle | rad (radians) |
| Quadrupole coefficient | m^{-2} |
| Multipole coefficient, 2n poles | m^{-n} |
| Electric voltage | MV (Megavolts) |
| Electric field strength | MV/m |
| Frequency | MHz (Megahertz) |
| Phase angles | 2π |
| Particle energy | GeV |
| Particle mass | GeV/c^2 |
| Particle momentum | GeV/c |
| Beam current | A (Amperes) |
| Particle charge | e (elementary charges) |
| Impedances | $M\Omega$ (Megohms) |
| Emittances | π m mrad |
| Emittances OPAL-T | m rad^{-1} |
| RF power | MW (Megawatts) |
| Higher mode loss factor | V/pc |

Chapter 3

Tutorial

This chapter will provide a jump start describing some of the most common used features of OPAL. The complete set of examples can be found and downloaded at <http://amas.web.psi.ch/download/TUTORIAL/opal-tutorial.tgz>. All examples run on a single core, but can be used efficiently on up to 8 cores. OPAL scales in the weak sense, hence for a higher concurrency one has to increase the problem size i.e. number of macro particles and the grid size, which is beyond this tutorial.

3.1 Starting OPAL

```
$ opal

Ippl> CommMPI: Parent process waiting for children ...
Ippl> CommMPI: Initialization complete.
>
>      / _ \ | _ \ / \ | |
>      | | | | | ) / \ | |
>      | | | | | / \ | |
>      | | | | | / _ \ | |
>      \ _ / | _ / \ \ _ \
OPAL >
OPAL > This is OPAL (Object Oriented Parallel Accelerator Library)
      Version 1.1.9 SVN version 13640 (c) PSI, http://amas.web.psi.ch
OPAL >
OPAL > Please send cookies, goodies or other motivations (wine and beer ... )
      to the OPAL developers opal@lists.psi.ch
OPAL > Time: 15.41.41 date: 29/03/2012
==>
```

3.2 Restart Mode

At the moment, we always restart from the last time step in the `restartfn`:

```
opal input.in -restartfn input.h5 -restart -1
```

3.3 Autophase Example

This is a partial complete example. First we have to set OPAL in AUTOPHASE mode, as described in Section 7.3 and for example set the nominal phase to -3.5 (deg).

```
Option, AUTOPHASE=4;

FINSS_RGUN_phi= (-3.5/180*Pi);
```

The cavity would be defined like

```
FINSS_RGUN: RFCavity, L = 0.17493, VOLT = 100.0,
    FMAPFN = "FINSS-RGUN.dat",
    ELEMEDGE =0.0, TYPE = "STANDING", FREQ = 2998.0,
    LAG = FINSS_RGUN_phi;
```

with FINSS_RGUN_phi defining the off crest phase. Now a normal TRACK command can be executed. A file containing the values of maximum phases is created, and has the format like:

```
1
FINSS_RGUN
2.22793
```

with the first entry defining the number of cavities in the simulation.

3.4 Examples of Beam Lines

```
examples/OBLA-Gun/OBLA-Gun.in
Title,string="OBLA Gun";

Option, TFS=FALSE;
Option, ECHO=FALSE;
Option, PSDUMPFREQ=10;

Edes=1.0E-9;
gamma=(Edes+EMASS)/EMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*EMASS;
brho = (EMASS*1.0e9*gambet) / CLIGHT;

value,{gamma,brho,Edes,beta,gambet};

// L:          physical element length (real)
// KS:          field scaling factor (real)
// FMAPFN:      field file name (string)
// ELEMEDGE:    physical start of the element on the floor (real)
SP1: Solenoid, L=1.20, ELEMEDGE=-0.5335, FMAPFN="1T2.T7",
    KS=0.00011;
SP2: Solenoid, L=1.20, ELEMEDGE=-0.399, FMAPFN="1T3.T7", KS=0.0;
SP3: Solenoid, L=1.20, ELEMEDGE=-0.269, FMAPFN="1T3.T7", KS=0.0;

gun: RFCavity, L=0.011, VOLT=-102.28, FMAPFN="1T1.T7",
    ELEMEDGE=0.00, TYPE="STANDING", FREQ=1.0e-6;

l1:   Line = (gun,sp1,sp2,sp3);

rf=1498.956e6;
v0=beta*CLIGHT;
lz = 6.5E-12*v0;
value,{v0,lz};
```

```

Dist1:DISTRIBUTION, DISTRIBUTION=gungauss,
      sigma= 0.00030, sigma_px=0.0, corrx=0.0,
      sigma_y= 0.00030, sigma_py=0.0, corry=0.0,
      sigma_t= 1z, sigma_pt=1.0, corrt=0.0 ,
      TEMISSION=3.9e-11, NBIN=39, DEBIN=1;

MINSTEPFORREBIN=1000;

Fs1:FIELDSOLVER, FSTYPE=FFT, MX=32, MY=32, MT=32,
      PARFFTX=true, PARFFTY=true, PARFFT=false,
      BCFFTX=open, BCFFTY=open, BCFFT=open,
      BBOXINCR=1, GREENSF=STANDARD;

beam1: BEAM, PARTICLE=ELECTRON, pc=P0, NPART=50000,
      BCURRENT=0.008993736, BFREQ=rf, CHARGE=-1;

Select, Line=11;
track,line=11, beam=beam1, MAXSTEPS=2000, DT=1.0e-13;
  run, method = "PARALLEL-T", beam=beam1, fieldsolver=Fs1,
    distribution=Dist1;
endtrack;
Stop;

```

3.4.1 PSI XFEL 250 MeV Injector

```

examples/diagnostic.in
Option, ECHO=FALSE;
Option, TFS=FALSE;
Option, PSDUMPFREQ=10;

Title,string="OPAL Diagnostics test";

Edes=0.0307; //GeV
gamma=(Edes+EMASS)/EMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*EMASS;
brho = (EMASS*1.0e9*gambet) / CLIGHT;
value,{gamma,brho,Edes,beta,gambet};

FINLB02_MSLAC40: Solenoid, L=0.001, KS=0.05,
                  FMAPFN="FINLB02-MSLAC.T7", ELEMEDGE=4.554;

FIND1_MQ10: Quadrupole, L=0.1, K1=2.788, ELEMEDGE=5.874;
FIND1_MQ20: Quadrupole, L=0.1, K1=-3.517, ELEMEDGE=6.074;

SCREEN: Monitor, L=0.01, ELEMEDGE=7.3867, OUTFN="Screen.h5";

FIND1: Line = (FINLB02_MSLAC40, FIND1_MQ10, FIND1_MQ20, SCREEN);

Dist1:DISTRIBUTION, DISTRIBUTION=gauss,
      sigmax= 1.0e-03, sigmapx=1.0e-4, corrx=0.5,
      sigmay= 2.0e-03, sigmapy=1.0e-4, corry=-0.5,
      sigmat= 3.0e-03, sigmapt=1.0e-4, corrt=0.0;

Fs2:FIELDSOLVER, FSTYPE=FFT, MX=32, MY=32, MT=64,
     PARFFTX=false, PARFFTY=false, PARFFTT=true,
     BCFFTX=open, BCFFTY=open, BCFFTT=open,
     BBOXINCR=1.0, GREENSF=INTEGRATED;

beam1: BEAM, PARTICLE=ELECTRON, pc=P0, NPART=1e5,
       BREQ=1498.953425154e6, BCURRENT=0.299598, CHARGE=-1;

Select, Line=FIND1;

track,line=FIND1, beam=beam1, MAXSTEPS=10000, DT=1.0e-12;
run, method = "PARALLEL-T", beam=beam1, fieldsolver=Fs2,
distribution=Dist1;
endtrack;
Stop;

```

3.4.2 PSI Injector II Cyclotron

Injector II is a separated sector cyclotron specially designed for preacceleration (inject: 870 keV, extract: 72 MeV) of high intensity proton beam for Ring cyclotron. It has 4 sector magnets, two double-gap acceleration cavities (represented by 2 single-gap cavities here) and two single-gap flat-top cavities.

Following is an input file of **Single Particle Tracking mode** for PSI Injector II cyclotron.

```

examples/opal-cycl.in
Title,string="OPAL-CyclT test";

Option, TFS=FALSE;
Option, ECHO=FALSE;
Option, PSDUMPFREQ=100000;
Option, SPTDUMPFREQ=5;

```

```

// define some physical parameters
Edes=0.002;
gamma=(Edes+PMASS)/PMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*PMASS;
brho = (PMASS*1.0e9*gambet) / CLIGHT;
value,{gamma,brho,Edes,beta,gambet};
phi01= 48.4812-15.0;
phi02=phi01+200.0;
phi03=phi01+1800.0;
phi04=phi01+2000.0;
phi05=(phi01+820.0)*3.0;
phi06=(phi01+2620.0)*3.0;

turns = 106;
nstep=5000;

// define elements and beamline
inj2: Cyclotron, TYPE="Injector2", CYHARMON=10, PHIINIT=30.0,
    PRINIT=-0.0067, RINIT=392.5, SYMMETRY=1.0,
    RFFREQ=frequency, FMAPFN=".. /ZYKL9Z.NAR";

rf0: RFCavity, VOLT=0.25796, FMAPFN=".. /av1.dat",
    TYPE="SINGLELEGAP", FREQ=frequency, ANGLE=35.0, PHI0=phi01,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf1: RFCavity, VOLT=0.25796, FMAPFN=".. /av1.dat",
    TYPE="SINGLELEGAP", FREQ=frequency, ANGLE=55.0, PHI0=phi02,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf2: RFCavity, VOLT=0.25796, FMAPFN=".. /av1.dat",
    TYPE="SINGLELEGAP", FREQ=frequency, ANGLE=215.0, PHI0=phi03,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf3: RFCavity, VOLT=0.25796, FMAPFN=".. /av1.dat",
    TYPE="SINGLELEGAP", FREQ=frequency, ANGLE=235.0, PHI0=phi04,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf4: RFCavity, VOLT=0.06380, FMAPFN=".. /av3.dat",
    TYPE="SINGLELEGAP", FREQ=frequency3, ANGLE=135.0, PHI0=phi05,
    RMIN = 830.0, RMAX = 3330.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf5: RFCavity, VOLT=0.06380, FMAPFN=".. /av3.dat",
    TYPE="SINGLELEGAP", FREQ=frequency3, ANGLE=315.0, PHI0=phi06,
    RMIN = 830.0, RMAX = 3330.0, PDIS = 0.0, GAPWIDTH = 0.0;

l1: Line = (inj2,rf0,rf1,rf2,rf3,rf4,rf5);

// define particles distribution, read from file
Dist1:DISTRIBUTION, DISTRIBUTION=fromfile,
    FNAME="PartDatabase.dat";

// usset define fieldsolver, useless for single particle track mode
Fs1:FIELDSOLVER, FSTYPE=NONE, MX=64, MY=64, MT=64,
    PARFFTX=true, PARFFTY=true, PARFFTT=false,
    BCFFTX=open, BCFFTY=open, BCFFTT=open;

// define beam parameters
beam1: BEAM, PARTICLE=PROTON, pc=P0, SPACECHARGE=true, NPART=1, BCURRENT=0.0;

// select beamline
Select, Line=l1;

```

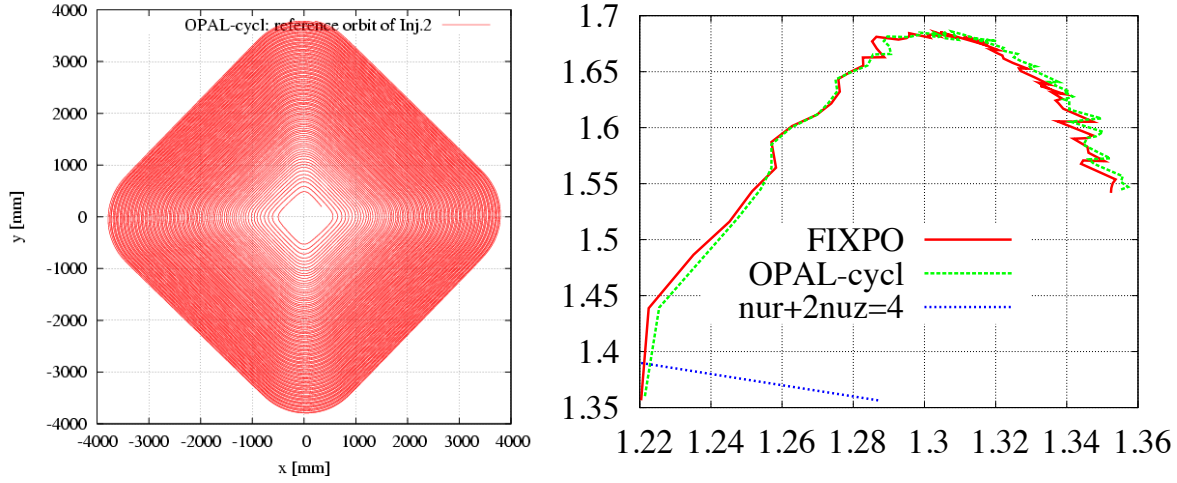


Figure 3.1: Reference orbit(left) and tune diagram(right) in Injector II

```
// start tracking
track,line=11, beam=beam1, MAXSTEPS=nstep*turns, STEPSPERTURN=nstep, TIMEINTEGRATOR="RK-4";
run, file = "track_output", turns = 1, method = "CYCLOTRON-T",
    beam=beam1, fieldsolver=Fsl, distribution=Dist1;
endtrack;
Stop;
```

To run opal on a single node, just use this command:

```
opal testinj2-1.in
```

Here shows some pictures using the resulting data from single particle tracking using OPAL-CYCL.

Left plot of Figure 3.1 shows the accelerating orbit of reference particle. After 106 turns, the energy increases from 870 keV at the injection point to 72.16 MeV at the deflection point.

From theoretic view, there should be an eigen ellipse for any given energy in stable area of a fixed accelerator structure. Only when the initial phase space shape matches its eigen ellipse, the oscillation of beam envelop amplitude will get minimal and the transmission efficiency get maximal. We can calculate the eigen ellipse by single particle tracking using betatron oscillation property of off-centered particle as following: track an off-centered particle and record its coordinates and momenta at the same azimuthal position for each revolution. Figure 3.2 shows the eigen ellipse at symmetric line of sector magnet for energy of 2 MeV in Injector II.

Right plot of Figure 3.1 shows very good agreement of the tune diagram by OPAL-CYCL and FIXPO. The trivial discrepancy should come from the methods they used. In FIXPO, the tune values are obtained according to the crossing points of the initially displaced particle. Meanwhile, in OPAL-CYCL, the Fourier analysis method is used to manipulate orbit difference between the reference particle and an initially displaced particle. The frequency point with the biggest amplitude is the betatron tune value at the given energy.

Following is the input file for single bunch tracking with space charge effects in Injector II.

```
// file name ''testinj2-2.in'' examples/opal-cycl-sc.in
// Define Option parameters
Option, ECHO=FALSE;
Option, PSDUMPFREQ=10;
Option, SPTDUMPFREQ=5;
```

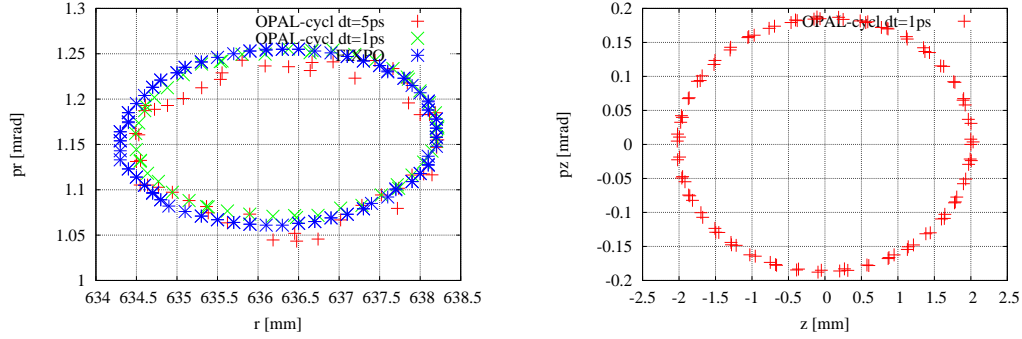


Figure 3.2: Radial and vertical eigenellipse at 2 MeV of Injector II

```

Option, REPARTFREQ=10;

// Define title
Title,string="OPAL-CyclT-SC test";

// define some physical parameters
Edes=0.000870;
gamma=(Edes+PMASS)/PMASS;
beta=sqrt(1-(1/gamma^2));
gambet=gamma*beta;
P0 = gamma*beta*PMASS;
brho = (PMASS*1.0e9*gambet) / CLIGHT;

phi01= 48.4812-15.0;
phi02=phi01+200.0;
phi03=phi01+1800.0;
phi04=phi01+2000.0;
phi05=(phi01+820.0)*3.0;
phi06=(phi01+2620.0)*3.0;

frequency=50.6370;
frequency3=3.0*frequency;

turns = 106;
nstep=5000;

// define elements and beamline
inj2: Cyclotron, TYPE="Injector2", CYHARMON=10, PHIINIT=30.0,
    PRINIT=-0.0067, RINIT=392.5, SYMMETRY=1.0,
    RFFREQ=50.6370, FMAPFN=".../ZYKL9Z.NAR";

rf0: RFCavity, VOLT=0.25796, FMAPFN=".../Cav1.dat",
    TYPE="SINGLELEGAP", FREQ=50.637, ANGLE=35.0, PHI0=phi01,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf1: RFCavity, VOLT=0.25796, FMAPFN=".../Cav1.dat",
    TYPE="SINGLELEGAP", FREQ=50.637, ANGLE=55.0, PHI0=phi02,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf2: RFCavity, VOLT=0.25796, FMAPFN=".../Cav1.dat",
    TYPE="SINGLELEGAP", FREQ=50.637, ANGLE=215.0, PHI0=phi03,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

```

```

rf3: RfCavity, VOLT=0.25796, FMAPFN="../../Cav1.dat",
    TYPE="SINGLELEGAP", FREQ=50.637, ANGLE=235.0, PHI0=phi04,
    RMIN = 350.0, RMAX = 3350.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf4: RfCavity, VOLT=0.06380, FMAPFN="../../Cav3.dat",
    TYPE="SINGLELEGAP", FREQ=151.911, ANGLE=135.0, PHI0=phi05,
    RMIN = 830.0, RMAX = 3330.0, PDIS = 0.0, GAPWIDTH = 0.0;

rf5: RfCavity, VOLT=0.06380, FMAPFN="../../Cav3.dat",
    TYPE="SINGLELEGAP", FREQ=151.911, ANGLE=315.0, PHI0=phi06,
    RMIN = 830.0, RMAX = 3330.0, PDIS = 0.0, GAPWIDTH = 0.0;

l1:   Line = (inj2,rf0,rf1,rf2,rf3,rf4,rf5);

// define particles distribution, generate Gaussion type
Dist1:DISTRIBUTION, DISTRIBUTION=gauss,
    sigma= 2.0e-03, sigmaPx=1.0e-7, corrx=0.0,
    sigmaY= 2.0e-03, sigmaPy=1.0e-7, corry=0.0,
    sigmaZ= 2.0e-03, sigmaPt=1.8e-4, corrt=0.0;

// define parallel FFT fieldsolver, parallel in x y direction
Fsl:FIELDSOLVER, FSTYPE=FFT, MX=32, MY=32, MT=32,
    PARFFTX=true, PARFFTY=true, PARFFT=false,
    BCFFTX=open, BCFFTY=open, BCFFT=open, BBOXINCR=5;

// define beam parameters
beam1: BEAM, PARTICLE=PROTON, pc=P0, SPACECHARGE=true,
    NPART=1e5, BCURRENT=3.0E-3, CHARGE=1.0, BFREQ= frequency;

// select beamline
Select, Line=l1;

// start tracking
track,line=l1, beam=beam1, MAXSTEPS=nstep*turns, STEPSPERTURN=nstep, TIMEINTEGRATOR="LF-2";
run, file = "track_output", turns = 1, method = "CYCLOTRON-T",
    beam=beam1, fieldsolver=Fsl, distribution=Dist1;
endtrack;
Stop;

```

To run opal on single node, just use this command:

```
# opal testinj2-2.in
```

To run opal on N nodes in parallel environment interactively, use this command instead:

```
# mpirun -np N opal testinj2-2.in
```

If restart a job from the last step of an existing .h5 file, add a new argument like this:

```
# mpirun -np N opal testinj2-2.in -restart -1
```

Figure 3.3 and 3.4 are simulation results, shown by H5ROOT code.

3.4.3 PSI Ring Cyclotron

From the view of numerical simulation, the difference between Injector II and Ring cyclotron comes from two aspects:

B Field The structure of Ring is totally symmetric, the field on median plain is periodic along azimuthal direction, OPAL-CYCLtake this advantage to only store 1/8 field data to save memory.

RF Cavity In the Ring, all the cavities are typically single gap with some parallel displacement from its radial position. OPAL-CYCLhave an argument PDIS to manipulate this issue.

Figure 3.5 shows a single particle tracking result and tune calculation result in the PSI Ring cyclotron. Limited by size of the user guide, we don't plan to show too much details as in Injector II.

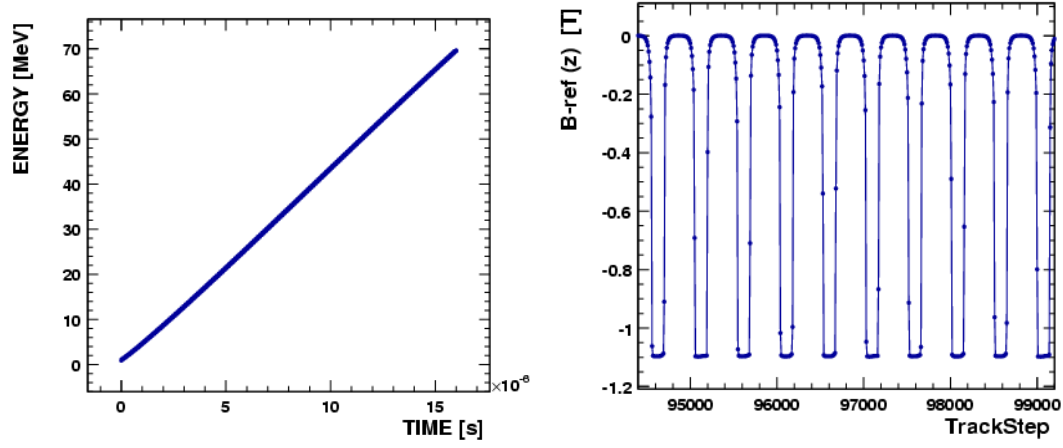


Figure 3.3: Energy Vs. time (left) and external B field Vs. trackstep (Right, only show for about 2 turns)

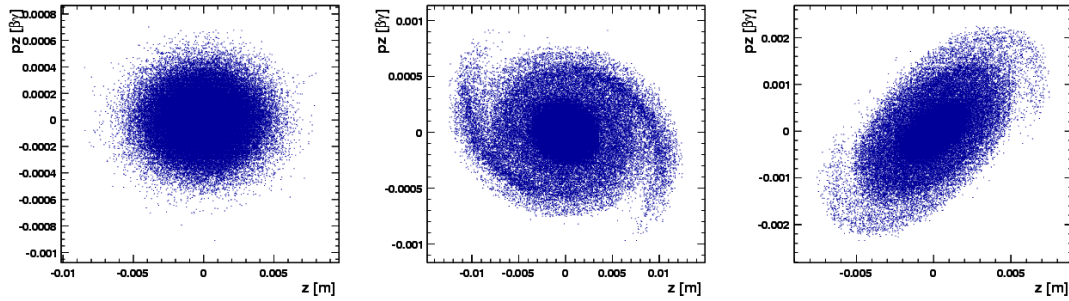


Figure 3.4: Vertical phase at different energy from left to right: 0.87 MeV, 15 MeV and 35 MeV

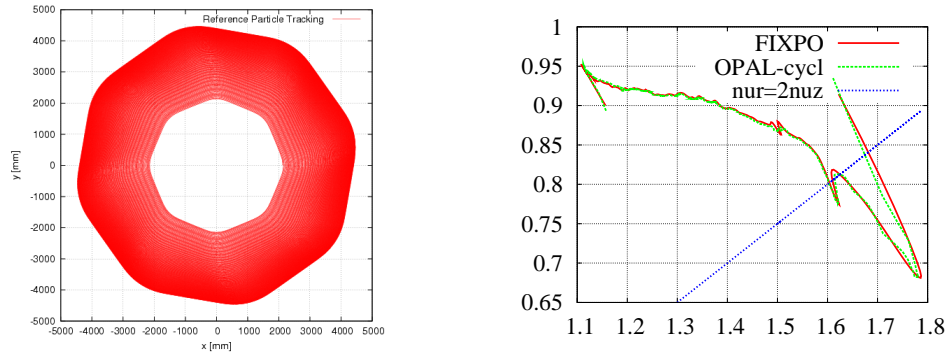


Figure 3.5: Reference orbit(left) and tune diagram(right) in Ring cyclotron

Chapter 4

OPAL-T

4.1 Introduction

OPAL-T is a fully three-dimensional program to track in time, relativistic particles taking into account space charge forces, self-consistently in the electrostatic approximation, and short-range longitudinal and transverse wake fields. OPAL-T is one of the few codes that is implemented using a parallel programming paradigm from the ground up. This makes OPAL-T indispensable for high statistics simulations of various kinds of existing and new accelerators. It has a comprehensive set of beamline elements, and furthermore allows arbitrary overlap of their fields, which gives OPAL-T a capability to model both the standing wave structure and traveling wave structure. Beside IMPACT-T it is the only code making use of space-charge solvers based on an integrated Green [40] function to efficiently and accurately treat beams with large aspect ratio, and a shifted Green function to efficiently treat image charge effects of a cathode [39, 41]. For simulations of particles sources i.e. electron guns OPAL-T uses the technique of energy binning in the electrostatic space-charge calculation to model beams with large energy spread. In the very near future a parallel Multi Grid solver taking into account the exact geometry will be implemented.

4.2 Variables in OPAL-T

OPAL-T uses the following canonical variables to describe the motion of particles. The physical units are listed in square brackets.

X Horizontal position x of a particle relative to the axis of the element [m].

PX $\beta_x \gamma$ Horizontal canonical momentum [1].

Y Horizontal position y of a particle relative to the axis of the element [m].

PY $\beta_y \gamma$ Horizontal canonical momentum [1].

Z Longitudinal position z of a particle in floor co-ordinates [m].

PZ $\beta_z \gamma$ Longitudinal canonical momentum [1].

The independent variable is **t** [s].

4.3 Integration of the Equation of Motion

OPAL-T integrates the relativistic Lorentz equation

$$\frac{d\gamma\mathbf{v}}{dt} = \frac{q}{m} [\mathbf{E}_{\text{ext}} + \mathbf{E}_{\text{sc}} + \mathbf{v} \times (\mathbf{B}_{\text{ext}} + \mathbf{B}_{\text{sc}})] \quad (4.1)$$

where γ is the relativistic factor, q is the charge, and m is the rest mass of the particle. \mathbf{E} and \mathbf{B} are abbreviations for the electric field $\mathbf{E}(\mathbf{x}, t)$ and magnetic field $\mathbf{B}(\mathbf{x}, t)$.

4.4 Envelope Tracker

The OPAL-e Envelope Tracker is an algorithm used to solve the envelope equations of a beam propagating through external fields and under the influence of its own space charge. The algorithm is based on the multi-slice analysis approach used in the theory of emittance compensation [62]. The space-charge model used can be switched between an analytic model derived and used in HOMDYN [63] and a similar model developed at PSI called Beam Envelope Tracker (BET).

4.5 Space Charge

Space-charge effects will be included in the simulation by specifying a field solver described in Chapter 12 and using the solver in the track command described in Chapter 15. By default, the code does not assume any symmetry i.e. full 3D. In the near future it is planned to implement also a slice (2D) model. This will allow the use of less numbers of macro-particles in the simulation which reduces the computational time significantly.

The space-charge forces are calculated by solving the 3D Poisson equation with open boundary conditions using a standard or integrated Green function method. The image charge effects of the conducting cathode are also included using a shifted Green function method. If more than one Lorentz frame is defined, the total space-charge forces are then the summation of contributions from all Lorentz frames. More details will be given in Version 1.1.9

FFT Based Particle-Mesh (PM) Solver

More details will be given in Version 1.1.9

Interpolation Schemes

More details will be given in Version 1.1.9

Iterative Space Charge Solver

This is a scalable parallel solver for the Poisson equation within a Particle-In-Cell (PIC) code for the simulation of electron beams in particle accelerators of irregular shape. The problem is discretized by Finite Differences. Depending on the treatment of the Dirichlet boundary the resulting system of equations is symmetric or ‘mildly’ nonsymmetric positive definite. In all cases, the system is solved by the preconditioned conjugate gradient algorithm with smoothed aggregation (SA) based algebraic multigrid (AMG) preconditioning. More details are given in [51].

Energy Binning

The beam out of a cathode or in a plasma wake field accelerator can have a large energy spread. In this case, the static approximation using one Lorentz frame might not be sufficient. Multiple Lorentz frames can be used so that within each Lorentz frame the energy spread is small and hence the electrostatic approximation is valid. More details will be given in Version 1.1.9

4.6 Wake Fields

Longitudinal and transverse short range wakefields wake fields are described in Chapter 13.

4.7 Multiple Species

In the present version only one particle species can be defined (Chapter 10), however due to the underlying general structure, the implementation of a true multi species version of OPAL is trivial.

Chapter 5

OPAL-CYCL

5.1 Introduction

OPAL-CYCL, as one of the flavors of the OPAL framework, is a fully three-dimensional parallel beam dynamics simulation program dedicated to future high intensity cyclotrons and FFAG. It tracks multiple particles which takes into account the space charge effects. For the first time in the cyclotron community, OPAL-CYCL has the capability of tracking multiple bunches simultaneously and take into account the beam-beam effects of the radially neighboring bunches (we call it neighboring bunch effects for short) by using a self-consistent numerical simulation model.

Apart from the multiparticle simulation mode, OPAL-CYCL also has two other serial tracking modes for conventional cyclotron machine design. One mode is the single particle tracking mode, which is a useful tool for the preliminary design of a new cyclotron. It allows one to compute basic parameters, such as reference orbit, phase shift history, stable region, and matching phase ellipse. The other one is the tune calculation mode, which can be used to compute the betatron oscillation frequency. This is useful for evaluating the focusing characteristics of a given magnetic field map.

In additions, the widely used plugin elements, including collimator, radial profile probe, septum, trim-coil field and charge stripper, are currently implemented in OPAL-CYCL. These functionalities are very useful for designing, commissioning and upgrading of cyclotrons and FFAGs.

5.2 Tracking modes

According to the number of particles defined by the argument NPART in BEAM (see Section 10.1), OPAL-CYCL works in one of the following three operation modes automatically.

5.2.1 Single Particle Tracking mode

In this mode, only one particle is tracked, either with acceleration or not. Working in this mode, OPAL-CYCL can be used as a tool during the preliminary design phase of a cyclotron.

The 6D parameters of a single particle in the initial local frame must be read from a file. To do this, in the OPAL input file, the command line DISTRIBUTION(See section 11) should be defined like this:

```
Dist1: DISTRIBUTION, DISTRIBUTION=fromfile, FNAME="PartDatabase.dat";
```

where the file *PartDatabase.dat* should have two lines:

```
1  
0.001 0.001 0.001 0.001 0.001 0.001
```

The number in the first line is the total number of particles. In the second line the data represents x, p_x, y, p_y, z, p_z in the local reference frame. Their units are described in section 5.3.

Please don't try to run his mode in parallel environment. You should believe that a single processor of the 21st century is capable of doing the single particle tracking.

5.2.2 Tune Calculation mode

In this mode, two particles are tracked, one with all data is set to zero is the reference particle and another one is an off-centering particle which is off-centered in both r and z directions. Working in this mode, OPAL-CYCL can calculate the betatron oscillation frequency ν_r and ν_z for different energies to evaluate the focusing characteristics for a given magnetic field.

Like the single particle tracking mode, the initial 6D parameters of the two particles in the local reference frame must be read from a file, too. In the file should has three lines:

| | | | | | | |
|-------|-----|-----|-----|-------|-----|--|
| 2 | | | | | | |
| 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 0.001 | 0.0 | 0.0 | 0.0 | 0.001 | 0.0 | |

When the total particle number equals 2, this mode is triggered automatically. Only the element CYCLOTRON in the beam line is used and other elements are omitted if any exists.

Please don't try to run his mode in parallel environment, either.

5.2.3 Multi-particle tracking mode

In this mode, large scale particles can be tracked simultaneously, either with space charge or not, either single bunch or multi-bunches, either serial or parallel environment, either reading the initial distribution from a file or generating a typical distribution, either running from the beginning or restarting from the last step of a former simulation.

Because this is the main mode as well as the key part of OPAL-CYCL, we will describe this in detail in Section 5.8.

5.3 Variables in OPAL-CYCL

OPAL-CYCL uses the following canonical variables to describe the motion of particles:

X Horizontal position x of a particle in given global Cartesian coordinates [m].

PX Horizontal canonical momentum [eV/c].

Y Longitudinal position y of a particle in global Cartesian coordinates [m].

PY Longitudinal canonical momentum [eV/c].

Z Vertical position z of a particle in global Cartesian coordinates [m].

PZ Vertical canonical momentum [eV/c].

The independent variable is: **t** [s].

NOTE: unit conversion of momentum in OPAL-T and OPAL-CYCL

Convert $\beta_x \gamma$ [dimensionless] to [mrad],

$$(\beta \gamma)_{\text{ref}} = \frac{P}{m_0 c} = \frac{P c}{m_0 c^2}, \quad (5.1)$$

$$P_x [\text{mrad}] = 1000 \times \frac{(\beta_x \gamma)}{(\beta \gamma)_{\text{ref}}}. \quad (5.2)$$

Convert from [eV/c] to $\beta_x \gamma$ [dimensionless],

$$(\beta_x \gamma) = \sqrt{\left(\frac{P_x [\text{eV}/c]}{m_0 c} + 1\right)^2 - 1}. \quad (5.3)$$

This may be deduced by analogy for the y and z directions.

5.3.1 The initial distribution in the local reference frame

To ensure compatibility with OPAL-T, the initial distribution of the bunch, either read from file or generated by distribution generator see Section 11, is specified in the local reference frame which is an Cartesian coordinates. During run time, the 6 phase space variables (x, y, z, p_x, p_y, p_z) are transformed to the global Cartesian coordinates before particle tracking starts.

$$\begin{aligned} X &= x \cos(\theta_0) - y \sin(\theta_0) + r_0 \cos(\theta_0) \\ Y &= x \sin(\theta_0) + y \cos(\theta_0) + r_0 \sin(\theta_0) \\ Z &= z \end{aligned}$$

$$\begin{aligned} PX &= (p_x + p_{r0}) \cos(\theta_0) - (p_y + p_{\theta0}) \sin(\theta_0) \\ PY &= (p_x + p_{r0}) \sin(\theta_0) + (p_y + p_{\theta0}) \cos(\theta_0) \\ PZ &= p_z \end{aligned}$$

5.4 Field Maps

In OPAL-CYCL, the magnetic field on the median plane is read from an ASCII type file. The field data should be stored in the cylinder coordinates frame (because the field map on the median plane of the cyclotron is usually measured in this frame).

There are two possible situations. One is the real field map on median plane of the exist cyclotron machine using measurement equipment. Limited by the narrow gaps of magnets, in most cases with cyclotrons, only vertical field B_z on the median plane ($z = 0$) is measured. Since the magnetic field data off the median plane field components is necessary for those particles with $z \neq 0$, the field need to be expanded in Z direction. According to the approach given by Gordon and Taivassalo, by using a magnetic potential and measured B_z on the median plane, at the point (r, θ, z) in cylindrical polar coordinates, the 3th order field can be written as

$$\begin{aligned} B_r(r, \theta, z) &= z \frac{\partial B_z}{\partial r} - \frac{1}{6} z^3 C_r, \\ B_\theta(r, \theta, z) &= \frac{z}{r} \frac{\partial B_z}{\partial \theta} - \frac{1}{6} \frac{z^3}{r} C_\theta, \\ B_z(r, \theta, z) &= B_z - \frac{1}{2} z^2 C_z, \end{aligned} \quad (5.4)$$

where $B_z \equiv B_z(r, \theta, 0)$ and

$$\begin{aligned} C_r &= \frac{\partial^3 B_z}{\partial r^3} + \frac{1}{r} \frac{\partial^2 B_z}{\partial r^2} - \frac{1}{r^2} \frac{\partial B_z}{\partial r} + \frac{1}{r^2} \frac{\partial^3 B_z}{\partial r \partial \theta^2} - 2 \frac{1}{r^3} \frac{\partial^2 B_z}{\partial \theta^2}, \\ C_\theta &= \frac{1}{r} \frac{\partial^2 B_z}{\partial r \partial \theta} + \frac{\partial^3 B_z}{\partial r^2 \partial \theta} + \frac{1}{r^2} \frac{\partial^3 B_z}{\partial \theta^3}, \\ C_z &= \frac{1}{r} \frac{\partial B_z}{\partial r} + \frac{\partial^2 B_z}{\partial r^2} + \frac{1}{r^2} \frac{\partial^2 B_z}{\partial \theta^2}. \end{aligned} \quad (5.5)$$

All the partial differential coefficients are on the median plane and can be calculated by interpolation. OPAL-CYCL uses Lagrange's 5-point formula.

The other situation is to calculate the field on the median plane or the 3D fields of the working gap for interesting region numerically by creating 3D model using commercial software, such as TOSCA, ANSOFT and ANSYS during the design phase of a new machine. If the field on the median plane is calculated, the field off the median plane can be obtained using the same expansion approach as the measured field map as described above. However, the 3D fields of the entire working gap should be more accurate than the expansion approach especially at the region not so close to the median plane in Z direction.

In the current version, we implemented the three specific type field-read functions *Cyclotron :: getFieldFromFile()* of the median plane fields. That which function is used is controlled by the parameters TYPE of CYCLOTRON (see Section 8.10) in the input file.

5.4.1 CARBONCYCL type

If TYPE=CARBONCYCL, the program requires the B_z data which is stored in a sequence shown in Fig.5.1. We

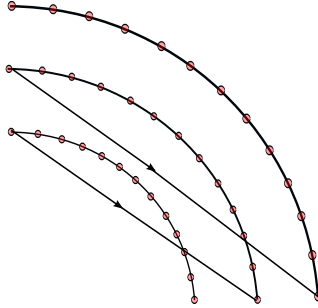


Figure 5.1: 2D field map on the median plane with primary direction corresponding to the azimuthal direction, secondary direction to the radial direction

need to add 6 parameters at the header of a plain B_z (kGauss) data file, namely, r_{min} [mm], Δr [mm], θ_{min} [°], $\Delta\theta$ [°], N_θ (total data number in each arc path of azimuthal direction) and N_r (total path number along radial direction). If Δr or $\Delta\theta$ is decimal, one can set its negative opposite number. For instance, if $\Delta\theta = \frac{1}{3}^\circ$, the fourth line of the header should be set to -3.0. Example showing the above explained format:

```

3.0e+03
10.0
0.0
-3.0
300
161
1.414e-03  3.743e-03  8.517e-03  1.221e-02  2.296e-02
3.884e-02  5.999e-02  8.580e-02  1.150e-01  1.461e-01
1.779e-01  2.090e-01  2.392e-01  2.682e-01  2.964e-01

```

```
3.245e-01  3.534e-01  3.843e-01  4.184e-01  4.573e-01
.....
```

5.4.2 CYCIAE type

If TYPE=CYCIAE, the program requires data format given by ANSYS 10.0. This function is originally for the 100 MeV cyclotron of CIAE, whose isochronous fields is numerically computed by by ANSYS. The median plane fields data is output by reading the APDL (ANSYS Parametric Design Language) script during the post-processing phase (you may need to do minor changes to adapt your own cyclotron model):

```
/post1
resume,solu,db
csys,1
nsel,s,loc,x,0
nsel,r,loc,y,0
nsel,r,loc,z,0
PRNSOL,B,COMP

CSYS,1
rsys,1

*do,count,0,200
  path,cyc100_Ansys,2,5,45
  ppath,1,,0.01*count,0,,1
  ppath,2,,0.01*count/sqrt(2),0.01*count/sqrt(2),,1

  pdef,bz,b,z
  paget,data,table

  *if,count,eq,0,then
    /output,cyc100_ANSYS,dat
    *STATUS,data,,,5,5
    /output
  *else
    /output,cyc100_ANSYS,dat,,append
    *STATUS,data,,,5,5
    /output
  *endif
*enddo
finish
```

By running this in ANSYS, you can get a fields file with the name *cyc100_ANSYS.data*. You need to put 6 parameters at the header of the file, namely, r_{min} [mm], Δr [mm], θ_{min} [°], $\Delta\theta$ [°], N_θ (total data number in each arc path of azimuthal direction) and N_r (total path number along radial direction). If Δr or $\Delta\theta$ is decimal, one can set its negative opposite number. This is useful if the decimal is unlimited. For instance, if $\Delta\theta = \frac{1}{3}^\circ$, the fourth line of the header should be -3.0. In a word, the fields are stored in the following format:

```
0.0
10.0
0.0e+00
1.0e+00
90
201
PARAMETER STATUS= DATA  ( 336 PARAMETERS DEFINED)
                        (INCLUDING      17 INTERNAL PARAMETERS)

LOCATION              VALUE
   1         5         1  0.537657876
   2         5         1  0.538079473
```

```

      3      5      1  0.539086731
      .....
     44      5      1  0.760777286
     45      5      1  0.760918663
     46      5      1  0.760969074

PARAMETER STATUS- DATA  ( 336 PARAMETERS DEFINED)
                      (INCLUDING 17 INTERNAL PARAMETERS)

LOCATION              VALUE
  1      5      1  0.704927299
  2      5      1  0.705050993
  3      5      1  0.705341275
      .....

```

5.4.3 BANDRF type

If TYPE=BANDRF, the program requires the B_z data format which is same as CARBONCYCL. But with BANDRF type, the program can also read in the 3D electric field(s). For the detail about its usage, please see Section 8.10.

5.4.4 Default PSI format

If the value of TYPE is other string rather than above mentioned, the program requires the data format like PSI format field file ZYKL9Z.NAR and SO3AV.NAR, which was given by the measurement. We add 4 parameters at the header of the file, namely, r_{min} [mm], Δr [mm], θ_{min} [°], $\Delta\theta$ [°], If Δr or $\Delta\theta$ is decimal, one can set its negative opposite number. This is useful if the decimal is unlimited. For instance, if $\Delta\theta = \frac{1}{3}^\circ$, the fourth line of the header should be -3.0.

```

1900.0
20.0
0.0
-3.0
LABEL=SO3AV
CFELD=FIELD      NREC= 141      NPAR= 3
LPAR= 7          IENT= 1        IPAR= 1
                3          141      135          30          8
                8          70
LPAR= 1089       IENT= 2        IPAR= 2
0.100000000E+01 0.190000000E+04 0.200000000E+02 0.000000000E+00 0.333333343E+00
0.506500015E+02 0.600000000E+01 0.938255981E+03 0.100000000E+01 0.240956593E+01
0.282477260E+01 0.340503168E+01 0.419502926E+01 0.505867147E+01 0.550443363E+01
0.570645094E+01 0.579413509E+01 0.583940887E+01 0.586580372E+01 0.588523722E+01
      .....

```

5.4.5 user's own fieldmap

You should revise the function or write your own function according to the instructions in the code to match your own field format if it is different to above types. For more detail about the parameters of CYCLOTRON, please refer to Section 8.10.

5.5 RF field

5.5.1 Read RF voltage profile

The RF cavities are treated as straight lines with infinitely narrow gaps and the electric field is a δ function plus a transit time correction. the two-gap cavity is treated as two independent single-gap cavities. the spiral gap cavity is not implemented yet. For more detail about the parameters of cyclotron cavities, see Section 8.11.2.

The voltage profile of a cavity gap is read from ASCII file. Here is an example:

| | | |
|------|-------|-------|
| 6 | | |
| 0.00 | 0.15 | 2.40 |
| 0.20 | 0.65 | 2.41 |
| 0.40 | 0.98 | 0.66 |
| 0.60 | 0.88 | -1.59 |
| 0.80 | 0.43 | -2.65 |
| 1.00 | -0.05 | -1.71 |

The number in the first line means 6 sample points and in the following lines the three values represent the normalized distance to the inner edge of the cavity, the normalized voltage and its derivative respectively.

5.5.2 Read 3D RF fieldmap

The 3D RF fieldmap can be read from h5part type file. This is useful for modeling the central region electric fields which usually has complicate shapes. For the detail about its usage, please see Section 8.10.

Please note that in this case, the E field is treated as a part of CYCLOTON element, rather than a independent RFCAVITY element.

5.6 Particle Tracking and Acceleration

The precision of the tracking methods is vital for the entire simulation process, especially for long distance tracking jobs. OPAL-CYCL uses a 4th order Runge-Kutta algorithm and the second order Leap-Frog scheme. The 4th order Runge-Kutta algorithm needs 4 the external magnetic field evaluation in each time step τ . During the field interpolation process, for an arbitrary given point the code first interpolates Formula B_z for its counterpart on the median plane and then expands to this give point using (5.4).

After each time step i , the code detects whether the particle crosses any one of the RF cavities during this step. If it does, the time point t_c of crossing is calculated and the particle return back to the start point of step i . Then this step is divided into three substeps: first, the code tracks this particle for $t_1 = \tau - (t_c - t_{i-1})$; then it calculates the voltage and adds momentum kick to the particle and refreshes its relativistic parameters β and γ ; and then tracks it for $t_2 = \tau - t_1$.

5.7 Space Charge

OPAL-CYCL uses the same solvers as OPAL-T to calculate the space charge effects (see Section 4.5). The difference is that, in a cyclotron, both the origin and orientation of the local Cartesian frame are changed from time to time. So the coordinates are transformed from the global frame to the local frame first, then the space charge fields are calculated in the local frame. After the space charge fields are solved, both coordinates and self electric and magnetic field are transformed back to global frame.

Typically, the space charge field is calculated once per time step. This is no surprise for the second-order Boris-Buneman time integrator (leapfrog-like scheme) which has per default only one force evaluation per step. The fourth-order Runge-Kutta integrator keeps the space charge field constant for one step, although there are

four external field evaluations. There is an experimental multiple-time-stepping (MTS) variant of the Boris-Buneman/leapfrog-method, which evaluates space charge only every N -th step, thus greatly reducing computation time while usually being still accurate enough.

5.8 Multi-bunches Issues

The neighboring bunches problem is motivated by the fact that for high intensity cyclotrons with small turn separation, single bunch space charge effects are not the only contribution. Along with the increment of beam current, the mutual interaction of neighboring bunches in radial direction becomes more and more important, especially at large radius where the distances between neighboring bunches get increasingly small and even they can overlap each other. One good example is PSI 590 MeV Ring cyclotron with a current of about 2mA in CW operation and the beam power amounts to 1.2 MW. A upgrade project for Ring is in process with the goal of 1.8 MW CW on target by replacing four old aluminum resonators by four new copper cavities with peak voltage increasing from about 0.7 MV to above 0.9 MV. After upgrade, the total turn number is reduced from 200 turns to less then 170 turns. Turn separation is increased a little bit, but still are at the same order of magnitude as the radial size of the bunches. Hence once the beam current increases from 2 mA to 3 mA, the mutual space charge effects between radially neighboring bunches can have significant impact on beam dynamics. In consequence, it is important to cover neighboring bunch effects in the simulation to quantitatively study its impact on the beam dynamics.

In OPAL-CYCL we developed a new fully consistent algorithm of multi-bunches simulation. We implemented two working modes, namely , `AUTO` and `FORCE`. In the first mode, only a single bunch is tracked and accelerated at the beginning, until the radial neighboring bunch effects become an important factor to the bunches' behavior. Then the new bunches will be injected automatically to take these effects into account. In this way, we can save time and memory sufficiently, and more important, we can get higher precision for the simulation in the region where neighboring bunch effects are neglectable. In the other mode, multi-bunches simulation starts from the injection points. This mode is appropriate for the machines in which this effects is unneglectable since the injection point.

In the space charge calculation for multi-bunches, the computation region covers all bunches. Because the energy of the bunches is quite different, it is inappropriate to use only one particle rest frame and a single Lorentz transformation any more. So the particles are grouped into different energy bins and in each bin the energy spread is relatively small. We apply Lorentz transforming, calculate the space charge fields and apply the back Lorentz transforming for each bin separately. Then add the field data together. Each particle has a ID number to identify which energy bin it belongs to.

5.9 Input

All the three working modes of OPAL-CYCL use an input file written in MAD language which will be described in detail in the following chapters.

For the **Tune Calculation mode**, one additional auxiliary file with the following format is needed.

| | | |
|--------|--------|--------|
| 72.000 | 2131.4 | -0.240 |
| 74.000 | 2155.1 | -0.296 |
| 76.000 | 2179.7 | -0.319 |
| 78.000 | 2204.7 | -0.309 |
| 80.000 | 2229.6 | -0.264 |
| 82.000 | 2254.0 | -0.166 |
| 84.000 | 2278.0 | 0.025 |

In each line the three values represent energy E , radius r and P_r for the SEO (Static Equilibrium Orbit) at starting point respectively and their units are MeV, mm, mrad.

A bash script *tuning.sh* is shown on the next page, to execute OPAL-CYCL for tune calculations.

```
#!/bin/bash
rm -rf tempfile
rm -f plotdata
rm -f tuningresult

exec 6<FIXPO_SEO
N="260"
j="1"
while read -u 6 E1 r pr
# read in Energy, initil R and intial Pr of each
# SEO from FIXPO output
do
rm -rf tempfile
echo -n j =
echo "$j"
echo -n energy= > tempfile
echo -n "$E1" >> tempfile
echo ";" >> tempfile

echo -n r= >> tempfile
echo -n "$r" >> tempfile
echo ";" >> tempfile

echo -n pr= >> tempfile
echo -n "$pr" >> tempfile
echo ";" >> tempfile
# execute OPAL to calculate tuning frquency and store
opal testcycl.in --commlib mpi --info 0 | \
    grep "Max" >>tuningresult
j=$((j+1))
done
exec 6<&-
rm -rf tempfile

# post porcess
exec 8<tuningresult
rm -f plotdata
i="0"

while [ $i -lt $N ]
do
read -u 8 a b url d
read -u 8 aa bb uz1 dd
echo "$url $uz1" >>plotdata
i=$((i+1))
done

exec 8<&-
rm -f tuningresult
```

To start execution, just run *tuning.sh* which uses the input file *testcycl.in* and the auxiliary file *FIXPO_SEO*. The output file is *plotdata* from which one can plot the tune diagram.

5.10 Output

Single Particle Tracking mode

The intermediate phase space data is stored in an ASCII file which can be used to the plot the orbit. The file's name is combined by input file name (without extension) and *-trackOrbit.dat*. The data are stored in the global Cartesian coordinates. The frequency of the data output can be set using the option SPTDUMPFREQ of `OPTION` statement (see §7.3)

The phase space data per STEPPERTURN (a parameter in the TRACK command) steps is stored in an ASCII file. The file's name is a combination of input file name (without extension) and *-afterEachTurn.dat*. The data is stored in the global cylindrical coordinate system. Please note that if the field map is ideally isochronous, the reference particle of a given energy take exactly one revolution in STEPPERTURN steps; Otherwise, the particle may not go through a full 360° in STEPPERTURN steps.

There are 3 ASCII files which store the phase space data around 0 , $\pi/8$ and $\pi/4$ azimuths. Their names are the combinations of input file name (without extension) and *-Angle0.dat*, *-Angle1.dat* and *-Angle2.dat* respectively. The data is stored in the global cylindrical coordinate system, which can be used to check the property of the closed orbit.

Tune calculation mode

The tunes ν_r and ν_z of each energy are stored in a ASCII file with name *tuningresult*.

Multi-particle tracking mode

The intermediate phase space data of all particles and some interesting parameters, including RMS envelop size, RMS emittance, external field, time, energy, length of path, number of bunches and tracking step, are stored in the H5hut file-format (<http://h5part.web.psi.ch/>) and can be analyzed using the H5PartRoot (<http://amas.web.psi.ch/tools/H5PartROOT/index.html>). The frequency of the data output can be set using the PSDUMPFREQ option of OPTION statement (see §7.3). The file is named like the input file but the extension is *.h5*.

The intermediate phase space data of central particle (with ID of 0) and an off-centering particle (with ID of 1) are stored in an ASCII file. The file's name is combined by the input file name (without extension) and *-trackOrbit.dat*. The frequency of the data output can be set using the SPTDUMPFREQ option of OPTION statement (see §7.3).

Chapter 6

Command Format

All flavours of OPAL using the same input language the MAD language. The language dialect here is ajar to MAD9, for hard core MAD8 users there is a conversion guide.

It is the first time that machines such as cyclotrons, proton and electron linacs can be described within the same language in the same simulation framework.

6.1 Statements and Comments

Input for OPAL is free format, and the line length is not limited. During reading, input lines are normally printed on the echo file, but this feature can be turned off for long input files. The input is broken up into tokens (words, numbers, delimiters etc.), which form a sequence of commands, also known as statements. Each statement must be terminated by a semicolon (;), and long statements can be continued on any number of input lines. White space, like blank lines, spaces, tabs, and newlines are ignored between tokens. Comments can be introduced with two slashes (//) and any characters following the slashes on the same line are ignored.

The C convention for comments (/ * . . . */) is also accepted. The comment delimiters / * and */ can be nested; this allows to “comment out” sections of input.

In the following descriptions, words in lower case stand for syntactic units which are to be replaced by actual text. UPPER CASE is used for keywords or names. These must be entered as shown. Ellipses (. . .) are used to indicate repetition.

The general format for a command is

```
keyword, attribute, . . ., attribute;  
label: keyword, attribute, . . ., attribute;
```

It has three parts:

1. The `label` is required for a definition statement. Its must be an identifier (see §6.2) and gives a name to the stored command.
2. The `keyword` identifies the action desired. It must be an identifier (see §6.2).
3. Each `attribute` is entered in one of the forms

```
attribute-name  
attribute-name=attribute-value  
attribute-name:=attribute-value
```

and serves to define data for the command, where:

- The `attribute-name` selects the attribute, it must be an identifier (see §6.2).
- The `attribute-value` gives it a value (see §6.3). When the attribute value is a constant or an expression preceded by the delimiter `=` it is evaluated immediately and the result is assigned to the attribute as a constant. When the attribute value is an expression preceded by the delimiter `:=` the expression is retained and re-evaluated whenever one of its operands changes.

Each attribute has a fixed attribute type (see §6.3).

The `attribute-value` can only be left out for logical attributes, this implies a `true` value.

When a command has a `label`, OPAL keeps the command in memory. This allows repeated execution of the same command by entering its label only:

```
label;
```

or to re-execute the command with modified attributes:

```
label,attribute,...,attribute;
```

If the label of such a command appears together with new attributes, OPAL makes a copy of the stored command, replaces the attributes entered, and then executes the copy:

```
QF:QUADRUPOLE,L=1,K1=0.01; // first definition of QF
QF,L=2;                    // redefinition of QF

MATCH;
...
LMD:LMDIF,CALLS=10;        // first execution of LMD
LMD;                      // re-execute LMD with
                          // the same attributes
LMD,CALLS=100,TOLERANCE=1E-5; // re-execute LMD with
                          // new attributes
ENDMATCH;
```

6.2 Identifiers or Labels

An identifier refers to a keyword, an element, a beam line, a variable, an array, etc.

A label begins with a letter, followed by an arbitrary number of letters, digits, periods (`.`), underscores (`_`). Other special characters can be used in a label, but the label must then be enclosed in single or double quotes. It makes no difference which type of quotes is used, as long as the same are used at either end. The preferred form is double quotes. The use of non-numeric characters is however strongly discouraged, since it makes it difficult to subsequently process a OPAL output with another program.

When a name is not quoted, it is converted to upper case; the resulting name must be unique. An identifier can also be generated from a string expression (see §6.4).

6.3 Command Attribute Types

An object attribute is referred to by the syntax

```
object-name->attribute-name
```

If the attribute is an array (see §6.14), one of its components is found by the syntax

`object-name->attribute-name[index]`

The following types of command attributes are available in OPAL:

- String (see §6.4),
- Logical (see §6.5),
- Real expression (see §6.6),
- Deferred expression (see §6.8.5),
- Place (see §6.9.1),
- Range (see §6.9.2),
- Constraint (see §6.10),
- Variable Reference (see §6.11)
- Regular expression (see §6.12),
- Token list (see §6.13).
- Array (see §6.14) of
 - Logical (see §6.14.1),
 - Real (see §6.14.2),
 - String (see §6.14.3),
 - Token lists (see §6.14.4),

See also:

- Operators (see Tab. 6.4),
- Functions (see Tab. 6.5),
- Array functions (see Tab. 6.7),
- Real functions of arrays (see Tab. 6.9),
- Operand (see §6.8),
- Random generators (see §6.8.5).

6.4 String Attributes

A string attribute makes alphanumeric information available, e.g. a title, file name, element class name, or an option. It can contain any characters, enclosed in single (') or double (") quotes. However, if it contains a quote, this character must be doubled. Strings can be concatenated using the & operator (see Tab. 6.1). An operand in a string can also use the function `STRING` (see Tab. 6.2). String values can occur in string arrays (see §6.14).

Examples:

Table 6.1: String Operator in OPAL

| Operator | Meaning | result type | operand types |
|----------|--|-------------|---------------|
| X & Y | concatenate the strings X and Y. String concatenations are always evaluated immediately when read. | string | string,string |

Table 6.2: String Function in OPAL

| Function | Meaning | result type | argument type |
|------------|---|-------------|---------------|
| STRING (X) | return string representation of the value of the numeric expression X | string | real |

```
TITLE,"This is a title for the program run ""test""";
CALL,FILE="save";
```

```
X=1;
TWISS,LINE=LEP&STRING(X+1);
```

The second example converts the value of the expression “X+1” to a string and appends it to “LEP”, giving the string “LEP2”.

6.5 Logical Expressions

Many commands in OPAL require the setting of logical values (flags) to represent the on/off state of an option. A logical value is represented by one of the values TRUE or FALSE, or by a logical expression. A logical expression can occur in logical arrays (see §6.14.1).

A logical expression has the same format and operator precedence as a logical expression in C. It is built from logical operators (see Tab. 6.3) and logical operands:

```
relation      ::= "TRUE" |
                  "FALSE" |
                  real-expr rel-operator real-expr

rel-operator  ::= "==" | "!=" | "<" | ">" | ">=" | "<="

and-expr      ::= relation | and-expr "&&" relation

logical-expr  ::= and-expr | logical-expr "||" and-expr
```

Example:

```
OPTION,ECHO=TRUE; // output echo is desired
```

When a logical attribute is not entered, its default value is always false. When only its name is entered, the value is set to TRUE:

```
OPTION,ECHO; // same as above
```

Example of a logical expression:

Table 6.3: Logical Operators in OPAL

| Operator | Meaning | result type | operand type |
|------------|--|-------------|-----------------|
| $X < Y$ | true, if X is less than Y | logical | real,real |
| $X \leq Y$ | true, if X is not greater than Y | logical | real,real |
| $X > Y$ | true, if X is greater than Y | logical | real,real |
| $X \geq Y$ | true, if X is not less than Y | logical | real,real |
| $X == Y$ | true, if X is equal to Y | logical | real,real |
| $X != Y$ | true, if X is not equal to Y | logical | real,real |
| $X \&\& Y$ | true, if both X and Y are true | logical | logical,logical |
| $X Y$ | true, if at least one of X and Y is true | logical | logical,logical |

$X > 10 \ \&\& \ Y < 20 \ || \ Z == 15$

6.6 Real Expressions

To facilitate the definition of interdependent quantities, any real value can be entered as an arithmetic expression. When a value used in an expression is redefined by the user or changed in a matching process, the expression is re-evaluated. Expression definitions may be entered in any order. OPAL evaluates them in the correct order before it performs any computation. At evaluation time all operands used must have values assigned. A real expression can occur in real arrays (see §6.14.2).

A real expression is built from operators (see Tab. 6.4) and operands (see §6.8):

```

real-ref  ::= real-variable |
              real-array "[" index "]" |
              object "->" real-attribute |
              object "->" real-array-attribute "[" index "]" |

table-ref ::= table "@" place "->" column-name

primary   ::= literal-constant |
              symbolic-constant |
              "#" |
              real-ref |
              table-ref |
              function-name "(" arguments ")" |
              (real-expression)

factor    ::= primary |
              factor "^" primary

term      ::= factor |
              term "*" factor |
              term "/" factor

real-expr ::= term |
              "+" term |
              "-" term |

```

```

real-expr "+" term |
real-expr "-" term |

```

It may contain functions (see Tab. 6.5), Parentheses indicate operator precedence if required. Constant sub-expressions are evaluated immediately, and the result is stored as a constant.

6.7 Operators

An expression can be formed using operators (see Tab. 6.4) and functions (see Tab. 6.5) acting on operands (see §6.8).

Table 6.4: Real Operators in OPAL

| Operator | Meaning | result type | operand type(s) |
|---|---|-------------|-----------------|
| Real operators with one operand | | | |
| + X | unary plus, returns X | real | real |
| - X | unary minus, returns the negative of X | real | real |
| Real operators with two operands | | | |
| X + Y | add X to Y | real | real,real |
| X - Y | subtract Y from X | real | real,real |
| X * Y | multiply X by Y | real | real,real |
| X / Y | divide X by Y | real | real,real |
| X ^ Y | power, return X raised to the power Y ($Y > 0$) | real | real,real |

Table 6.5: Real Functions in OPAL

| Function | Meaning | result type | argument type(s) |
|---|---|-------------|------------------|
| Real functions with no arguments | | | |
| RANF () | random number, uniform distribution in [0,1) | real | - |
| GAUSS () | random number, Gaussian distribution with $\sigma = 1$ | real | - |
| USER0 () | random number, user-defined distribution | real | - |
| SI () | arc length from start of ring to the entry of the current element. This function is only available in the EALIGN command (see §??) | real | - |
| SC () | arc length from start of ring to the centre of the current element. This function is only available in the EALIGN command (see §??) | real | - |
| SO () | arc length from start of ring to the exit of current the element. This function is only available in the EALIGN command (see §??) | real | - |

Table 6.7: Real Functions of Arrays in OPAL

| Function | Meaning | result type | operand type |
|----------------|---|-------------|--------------|
| VMAX (X, Y) | return largest array component | real | real array |
| VMIN (X, Y) | return smallest array component | real | real array |
| VRMS (X, Y) | return rms value of an array | real | real array |
| VABSMAX (X, Y) | return absolute largest array component | real | real array |

6.8 Operands in Expressions

A real expression may contain the operands listed in the following subsections.

6.8.1 Literal Constants

Numerical values are entered like FORTRAN constants. Real values are accepted in INTEGER or REAL format. The use of a decimal exponent, marked by the letter D or E, is permitted.

Examples:

```
1, 10.35, 5E3, 314.1592E-2
```

6.8.2 Symbolic constants

OPAL recognises a few built-in mathematical and physical constants (see Tab. 6.8). Their names must not be used for user-defined labels. Additional symbolic constants may be defined (see §7.4.2) to simplify their repeated use in statements and expressions.

6.8.3 Variable labels

Often a set of numerical values depends on a common variable parameter. Such a variable must be defined as a global variable (see §7.4.1) defined by one of

```
X=expression;
X:=expression;
VECTOR X=vector-expression;
VECTOR X:=vector-expression;
```

When such a variable is used in an expression, OPAL uses the current value of the variable. When the value is a constant or an expression preceded by the delimiter = it is evaluated immediately and the result is assigned to the variable as a constant. When the value is an expression preceded by the delimiter := the expression is retained and re-evaluated whenever one of its operands changes. Example:

```
L=1.0;
X:=L;
D1:DRIFT,L:=X;
D2:DRIFT,L:=2.0-X;
```

When the value of X is changed, the lengths of the drift spaces are recalculated as X and 2-X respectively.

Table 6.8: Predefined Symbolic Constants

| OPALname | Mathematical symbol | Value | Unit |
|----------|---------------------|------------------|---------|
| PI | π | 3.1415926535898 | 1 |
| TWOPI | 2π | 6.2831853071796 | 1 |
| DEGRAD | $180/\pi$ | 57.295779513082 | deg/rad |
| RADDEG | $\pi/180$ | .017453292519943 | rad/deg |
| E | e | 2.7182818284590 | 1 |
| EMASS | m_e | .51099906e-3 | GeV |
| PMASS | m_p | .93827231 | GeV |
| HMASS | m_{h^-} | .939277 | GeV |
| CMASS | m_c | 12*0.931494027 | GeV |
| UMASS | m_u | 238*0.931494027 | GeV |
| MASS | m_μ | 0.1057 | GeV |
| DMASS | m_d | 2*0.931494027 | GeV |
| XEMASS | m_{xe} | 124*0.931494027 | GeV |
| CLIGHT | c | 299792458 | m/s |

6.8.4 Element or command attributes

In arithmetic expressions the attributes of physical elements or commands can occur as operands. They are named respectively by

```
element-name->attribute-name
command-name->attribute-name
```

If they are arrays, they are denoted by

```
element-name->attribute-name[index]
command-name->attribute-name[index]
```

Values are assigned to attributes in element definitions or commands.

Example:

```
D1:DRIFT,L=1.0;
D2:DRIFT,L=2.0-D1->L;
```

D1->L refers to the length L of the drift space D1.

6.8.5 Deferred Expressions and Random Values

Definition of random machine imperfections requires evaluation of expressions containing random functions. These are not evaluated like other expressions before a command begins execution, but sampled as required from

the distributions indicated when errors are generated. Such an expression is known as a **deferred expression**. Its value cannot occur as an operand in another expression.

Example:

```
ERROR:EALIGN,CLASS=QUADRUPOLE,DX=SIGMA*GAUSS();
```

All elements in range are assigned independent random displacements sampled from a Gaussian distribution with standard deviation SIGMA. The quantity ERROR→DX must not occur as an operand in another expression.

6.8.6 Table References

Values can be extracted from a table with the syntax

```
table-name "@" place "->" column-name
```

Here `table-name` denotes a table (see Chapter ??), `place` denotes a table row (see §6.9.1), and `column-name` denotes the name of a column in the table.

Example:

```
TWISS@#E->BETX
```

denotes the horizontal beta function at the end of table TWISS.

6.9 Element Selection

Many OPAL commands allow for the possibility to process or display a subset of the elements occurring in a beam line or sequence. This is not yet available in:  OPAL-Tand  OPAL-CYCL.

6.9.1 Element Selection

A `place` denotes a single element, or the position **following** that element. It can be specified by one of the choices

object-name[`index`] The name verb'object-name' is the name of an element, line or sequence, and the integer `index` is its occurrence count in the beam line. If the element is unique, [`index`] can be omitted.

#S denotes the position before the first physical element in the **full** beam line. This position can also be written `#0`.

#E denotes the position after the last physical element in the **full** beam line.

Either form may be qualified by one or more beam line names, as described by the formal syntax:

```
place ::= element-name |
         element-name "[" integer "]" |
         "#S" |
         "#E" |
         line-name "::" place
```

An omitted index defaults to one. Examples: assume the following definitions:

```

M: MARKER;
S: LINE= (C, M, D) ;
L: LINE= (A, M, B, 2*S, A, M, B) ;
  SURVEY, LINE=L

```

The line L is equivalent to the sequence of elements

```
A, M, B, C, M, D, C, M, D, A, M, B
```

Some possible place definitions are:

C[1] The first occurrence of element C.

#S The beginning of the line L.

M[2] The second marker M at top level of line L, i. e. the marker between second A and the second B.

#E The end of line L

S[1]::M[1] The marker M nested in the first occurrence of S.

6.9.2 Range Selection

A range in a beam line (see §9) is selected by the following syntax:

```

range ::= place |
        place "/" place

```

This denotes the range of elements from the firstplace to the second place. Both positions are included. A few special cases are worth noting:

- When place1 refers to a LINE (see §9). the range starts at the **beginning** of this line.
- When place2 refers to a LINE (see §9). the range ends at the **ending** of this line.
- When both place specifications refer to the same object, then the second can be omitted. In this case, and if place refers to a LINE (see §9) the range contains the whole of the line.

Examples: Assume the following definitions:

```

M: MARKER;
S: LINE= (C, M, D) ;
L: LINE= (A, M, B, 2*S, A, M, B) ;

```

The line L is equivalent to the sequence of elements

```
A, M, B, C, M, D, C, M, D, A, M, B
```

Examples for range selections:



#S/#E The full range or L.

A[1]/A[2] A[1] through A[2], both included.

S::M/S[2]::M From the marker M nested in the first occurrence of S, to the marker M nested in the second occurrence of S.

S[1]/S[2] Entrance of first occurrence of S through exit of second occurrence of S.

6.10 Constraints

Please note this is not yet available in:  OPAL-Tand  OPAL-CYCL.

In matching it is desired to specify equality constraints, as well as lower and upper limits for a quantity. OPAL accepts the following form of constraints:

```
constraint          ::= array-expr constraint-operator array-expr
constraint-operator ::= "=" | "<" | ">"
```

6.11 Variable Names

A variable name can have one of the formats:

```
variable name ::= real variable |
                object->"real attribute"
```

The first format refers to the value of the `global variable` (see §7.4.1), the second format refers to a named `attribute` of the named `object`. `object` can refer to an `element` or a `command`

6.12 Regular Expressions

Some commands allow selection of items via a `regular-expression`. Such a pattern string **must** be enclosed in single or double quotes; and the case of letters is significant. The meaning of special characters follows the standard UNIX usage: utility:

`.` Stands for a single arbitrary character,

[letter...letter] Stands for a single character occurring in the bracketed string, Example: “[abc]” denotes the choice of one of a, b, c.

[character-character] Stands for a single character from a range of characters, Example: “[a-zA-Z]” denotes the choice of any letter.

`*` Allows zero or more repetitions of the preceding item, Example: “[A-Z]*” denotes a string of zero or more upper case letters.

`\character` Removes the special meaning of `character`, Example: “*” denotes a literal asterisk.

All other characters stand for themselves. The pattern

```
"[A-Za-z][A-Za-z0-9_']*"
```

illustrates all possible unquoted identifier formats (see §6.2). Since identifiers are converted to lower case, after reading they will match the pattern



```
"[a-z][a-z0-9_']*"
```

Examples for pattern use:

```
SELECT, PATTERN="D.. "
SAVE, PATTERN="K.*QD.*\R1"
```

The first command selects all elements whose names have exactly three characters and begin with the letter D. The second command saves definitions beginning with the letter K, containing the string QD, and ending with the string .R1. The two occurrences of . * each stand for an arbitrary number (including zero) of any character, and the occurrence \ . stands for a literal period.

6.13 Token List

In some special commands LIST (see §??) an attribute cannot be parsed immediately, since some information may not yet be available during parsing. Such an attribute is entered as a “token list”, and it is parsed again when the information becomes available. Token lists can occur in token list arrays (see §6.14.4). This is not yet available in:  OPAL-Tand  OPAL-CYCL.

Example:

```
LIST, COLUMN={X:12:6, Y:12:6};
```

where X:12:6 and Y:12:6 are two token lists, and {X:12:6, Y:12:6} is a token list array.

6.14 Arrays

An attribute array is a set of values of the same attribute type (see §6.3). Normally an array is entered as a list in braces:

```
{value, ..., value}
```

The list length is only limited by the available storage. If the array has only one value, the braces () can be omitted:

```
value
```

6.14.1 Logical Arrays

For the time being, logical arrays can only be given as a list. The formal syntax is:

```
logical-array ::= "{" logical-list "}"

logical-list  ::= logical-expr |
                  logical-list "," logical-expr
```

Example:

```
{true,true,a==b,false,x>y && y>z,true,false}
```

6.14.2 Real Arrays

Real arrays have the following syntax:

```
array-ref      ::= array-variable |
                  object "->" array-attribute |

table-ref      ::= "ROW" "(" table "," place ")" |
                  "ROW" "(" table "," place "," column-list ")" |
                  "COLUMN" "(" table "," column ")" |
```

```

" COLUMN " "(" table "," column "," range ")"

columns      ::= column |
               "{" column-list "}"

column-list  ::= column |
               column-list "," column

column       ::= string

real-list    ::= real-expr |
               real-list "," real-expr

index-select ::= integer |
               integer "," integer |
               integer "," integer "," integer

array-primary ::= "{" real-list "}" |
                 "TABLE" "(" index-select "," real-expr ")" |
                 array-ref |
                 table-ref |
                 array-function-name "(" arguments ")" |
                 (array-expression)

array-factor ::= array-primary |
                 array-factor "^" array-primary

array-term   ::= array-factor |
                 array-term "*" array-factor |
                 array-term "/" array-factor

array-expr   ::= array-term |
                 "+" array-term |
                 "-" array-term |
                 array-expr "+" array-term |
                 array-expr "-" array-term |

```

Example:

```
{a, a+b, a+2*b}
```

There are also three functions allowing the generation of real arrays:

TABLE Generate an array of expressions:

```

TABLE(n2, expression)    // implies
                        // TABLE(1:n2:1, expression)
TABLE(n1:n2, expression) // implies
                        // TABLE(n1:n2:1, expression)
TABLE(n1:n2:n3, expression)

```

These expressions all generate an array with $n2$ components. The components selected by $n1:n2:n3$ are filled from the given expression; a C pseudo-code for filling is

```

int i;
for (i = n1; i <= n2; i += n3) a[i] = expression(i);

```

In each generated expression the special character hash sign (#) is replaced by the current value of the index i .

Example:

Table 6.9: Real Array Functions in OPAL(acting component-wise)

| Function | Meaning | result type | argument type |
|------------|---|-------------|---------------|
| TRUNC (X) | truncate X towards zero (discard fractional part) | real array | real array |
| ROUND (X) | round X to nearest integer | real array | real array |
| FLOOR (X) | return largest integer not greater than X | real array | real array |
| CEIL (X) | return smallest integer not less than X | real array | real array |
| SIGN (X) | return sign of X (+1 for X positive, -1 for X negative, 0 for X zero) | real array | real array |
| SQRT (X) | return square root of X | real array | real array |
| LOG (X) | return natural logarithm of X | real array | real array |
| EXP (X) | return exponential to the base e of X | real array | real array |
| SIN (X) | return trigonometric sine of X | real array | real array |
| COS (X) | return trigonometric cosine of X | real array | real array |
| ABS (X) | return absolute value of X | real array | real array |
| TAN (X) | return trigonometric tangent of X | real array | real array |
| ASIN (X) | return inverse trigonometric sine of X | real array | real array |
| ACOS (X) | return inverse trigonometric cosine of X | real array | real array |
| ATAN (X) | return inverse trigonometric tangent of X | real array | real array |
| TGAUSS (X) | random number, Gaussian distribution with $\sigma=1$, truncated at X | real array | real array |
| USER1 (X) | random number, user-defined distribution with one parameter | real array | real array |
| EVAL (X) | evaluate the argument immediately and transmit it as a constant | real array | real array |

```
// an array with 9 components, evaluates to
// {1,4,7,10,13}:
table(5:9:2,3*#+1) // equivalent to
                    // {0,0,0,0,16,0,22,0,28}
```

ROW Generate a table row:

```
ROW(table,place)           // implies all columns
ROW(table,place,column list)
```

This generates an array containing the named (or all) columns in the selected place.

COLUMN Generate a table column:

```
COLUMN(table,column)       // implies all rows
COLUMN(table,column,range)
```

This generates an array containing the selected (or all) rows of the named column.

6.14.3 String Arrays

String arrays can only be given as lists of single values. For permissible values String values (see §6.4).

Example:

```
{A, "xyz", A & STRING(X)}
```

6.14.4 Token List Arrays

Token list arrays are always lists of single token lists.

Example:

```
{X:12:8,Y:12:8}
```

Chapter 7

Control Statements

7.1 Getting Help

7.1.1 HELP Command

A user who is uncertain about the attributes of a command should try the command `HELP`, which has three formats:

```
HELP;                // Give help on the "HELP" command
HELP,NAME=label;     // List funct. and attr. types of
                     // "label"
HELP,label;          // Shortcut for the second format
```

`label` is an identifier (see §6.2). If it is non-blank, OPAL prints the function of the object `label` and lists its attribute types. Entering `HELP` alone displays help on the `HELP` command.

Examples:

```
HELP;
HELP,NAME=TWISS;
HELP,TWISS;
```

7.1.2 SHOW Command

The `SHOW` statement displays the current attribute values of an object. It has three formats:

```
SHOW;                // Give help on the "SHOW" command
SHOW,NAME=pattern;    // Show names matching of "pattern"
SHOW,pattern;         // Shortcut for the second format
```

`pattern` is a regular expression (see §6.12). If it is non-blank, OPAL displays all object names matching the pattern. Entering `SHOW` alone displays help on the `SHOW` command. Examples:

```
SHOW;
SHOW,NAME="QD.*\..L*";
SHOW,"QD.*\..L*";
```

7.1.3 WHAT Command

The `WHAT` statement displays all object names matching a given regular expression. It has three formats:

```
WHAT;                // Give help on the "WHAT" command
WHAT,NAME=label;     // Show definition of "label"
WHAT,label;          // Shortcut for the second format
```

`label` is an identifier (see §6.2). If it is non-blank, OPAL displays the object `label` in a format similar to the input statement that created the object. Entering `WHAT` alone displays help on the `WHAT` command. Examples:

```
WHAT;
WHAT,NAME=QD;
WHAT,QD;
```

7.2 STOP / QUIT Statement

The statement

```
STOP or QUIT;
```

terminates execution of the OPAL program, or, when the statement occurs in a `CALL` file (see §7.7.1), returns to the calling file. Any statement following the `STOP` or `QUIT` statement is ignored.

7.3 OPTION Statement

The `OPTION` command controls global command execution and sets a few global quantities:

```
OPTION,ECHO=logical,INFO=logical,TRACE=logical,
      VERIFY=logical,WARN=logical,
      SEED=real,TELL=logical,PSDUMPFREQ=integral,
      STATDUMPFREQ=integral,SPTDUMPFREQ=integral,
      REPARTFREQ=integral,REBINFREQ=integral;
```

The first five logical flags activate or deactivate execution options:

ECHO Controls printing of an echo of input lines on the standard error file.

INFO If this option is turned off, OPAL suppresses all information messages. It also affects the *gnu.out* and *eb.out* files in case of OPAL-CYCL simulations.

TRACE When the `TRACE` option is on, OPAL writes additional trace information on the standard error file for each executable command. This information includes the command name and elapsed CPU time before and after the command.

VERIFY If this option is on, OPAL gives a message for each undefined variable or element in a beam line.

WARN If this option is turned off, OPAL suppresses all warning messages.

SEED Selects a particular sequence of random values. A `SEED` value is an integer in the range [0...999999999] (default: 123456789). `SEED` can be an expression. If `SEED = -1`, the time is used as seed and the generator is not portable anymore. See also: random values (see §6.8.5).

PSDUMPFREQ Defines after how many time steps the phase space is dumped into the H5hut file. Default value is 10.

STATDUMPFREQ Defines after how many time steps we dump statistical data, such as RMS beam emittance, to the .stat file. The default value is 10. Currently only available for OPAL-T.

SPTDUMPFREQ Defines after how many time steps we dump the phase space of single particle. It is always useful to record the trajectory of reference particle or some specified particle for primary study. Its default value is 1.

REPARTFREQ Defines after how many time steps we do particles repartition to balance the computational load of the computer nodes. Its default value is 10.

REBINFREQ Defines after how many time steps we update the energy Bin ID of each particle. For the time being. Only available for multi-bunch simulation in OPAL-CYCL. Its default value is 100.

PSDUMPEACHTURN Control option of phase space dumping. If true, dump phase space after each turn. For the time being, this is only use for multi-bunch simulation in OPAL-CYCL. Its default set is false.

PSDUMLOCALFRAME Control option whether the phase space data is dumped in the global Cartesian frame or in the local Cartesian frame. If true, in local frame, otherwise in global Cartesian frame. Only available for OPAL-CYCL. Its default set is false. Note that restarting run cannot be launched by reading in phase space data in local frame.

SCSOLVEFREQ If the space charge field is slowly varying w.r.t. external fields, this option allows to change the frequency of space charge calculation, i.e. the space charge forces are evaluated every SCSOLVEFREQ step and then reused for the following steps. Affects integrators LF-2 and RK-4 of OPAL-CYCL. Its default value is 1. Note: as the multiple-time-stepping (MTS) integrator maintains accuracy much better with reduced space charge solve frequency, this option should probably not be used anymore.

MTSSUBSTEPS Only used for multiple-time-stepping (MTS) integrator in OPAL-CYCL. Specifies how many substeps for external field integration are done per step. Default value is 1. Making less steps per turn and increasing this value is the recommended way to reduce space charge solve frequency.

RHODUMP If true the scalar ρ field is saved each time a phase space is written. There exists a reader in Visit with versions greater or equal 1.11.1.

EFDUMP If true the electric field (from the space charge) is saved each time a phase space is written.

EBDUMP If true the electric and magnetic field on the particle is saved each time a phase space is written.

CSRDUMP If true the electric csr field component (E_z), line density and the derivative of the line density is written into the *data* directory.

AUTOPHASE A phase scan of all n rf-elements is performed, if AUTOPHASE is greater than zero. AUTOPHASE finds the maximum energy in a way similar to the code Astra.

1. find the phase ϕ_i for maximum energy of the i -th cavity.
2. track is continued with $\phi_i + LAG$ to the element $i + 1$.
3. if $i < n$ goto 1

For convenience a file (`inputfn.phases`) with the phases corresponding to the maximum energies is written. A AUTOPHASE value of 4 gives Astra comparable results. An example is given in (see §3.3).

SCAN If true one can simulate in a loop several machines where some variables can be random variables. Find an example at 15.1.1.

CZERO If true the distributions are generated such that the centroid is exactly zero and not statistically dependent.

RNGTYPE The default random number generator (RANDOM) is a portable 48-bit generator. Three quasi random generators are available:

1. HALTON
2. SOBOL
3. NIEDERREITER.

For details see the GSL reference manual (18.5).

FINEEMISSION During emission of the beam from a cathode, the time step of the integration is set to a new value until all of the particles are emitted. This new value is determined two different ways depending on whether this `OPTION` command is true or false.

1. **TRUE:** The longitudinal distribution of the initial beam is represented internally has a histogram with N bins. The number of bins is determined by $N = NBIN \times SBIN$ where $NBIN$ is the number of energy bins used to calculate the beam space charge and $SBIN$ is the number of sampling bins per energy bin (see Chapter 11). The integration time step is set so that one histogram bin is emitted each time step. Therefore it takes N steps to emit the entire beam.
2. **FALSE:** In this case the integration time step is set so that one entire energy bin is emitted each time step. This typical results in a very coarse emission time step with the entire beam being emitted in $NBIN$ steps.

Once the entire beam is emitted the time step is reset to the value defined in the input file.

By using this option one can obtain fine time step emission without using many energy bins. As the number of energy bins is increased so is the space charge calculation time. Additionally, a large number of energy bins is not always needed for an accurate simulation, depending on the type of injector being modeled.

ENABLEHDF5 If true (default), HDF5 read and write is enabled.

ASCIIDUMP If true, instead of HDF5, ASCII output is generated for the following elements: Probe, Collimator, Monitor, Stripper, Foil and global losses.

The last attribute requests listing of the current settings:

TELL If true, the current settings are listed.

Examples:

```
OPTION,ECHO=FALSE,TELL;
OPTION,SEED=987456321
```

7.4 Parameter Statements

7.4.1 Variable Definitions

OPAL recognises several types of variables.

Table 7.1: Default Settings for Options

| | | | | | |
|--------------|---------|----------------|---------|-----------------|-------------|
| ECHO | = true | INFO | = true | TRACE | = false |
| WARN | = true | VERIFY | = false | SEED | = 123456789 |
| PSDUMPFREQ | = 10 | SPTDUMPFREQ | = 1 | REPARTFREQ | = 10 |
| FINEEMISSION | = true | CZERO | = false | TELL | = false |
| VERIFY | = false | STATDUMPFREQ | = 10 | RNGTYPE | = RANDOM |
| EBDUMP | = false | RDUMP | = 0 | REBINFREQ | = 100 |
| RHODUMP | = false | SCSOLVEFREQ | = 1 | EFDUMP | = false |
| SCAN | = false | AUTOPHASE | = 0 | PPDEBUG | = false |
| SURFDUMPFREQ | = -1 | PSDUMPEACHTURN | = false | PSDUMLOCALFRAME | = false |
| CSRDUMP | = false | ENABLEHDF5 | = true | ASCIIDUMP | = false |

Real Scalar Variables

```
REAL variable-name=real-expression;
```

The keyword `REAL` is optional. For backward compatibility the program also accepts the form

```
variable-name:=real-expression;
```

This statement creates a new global variable `variable-name` and discards any old variable with the same name. Its value depends on all quantities occurring in `real-expression` (see §6.6). Whenever an operand changes in `real-expression`, a new value is calculated. The definition may be thought of as a mathematical equation. However, OPAL is not able to solve the equation for a quantity on the right-hand side.

An assignment in the sense of the FORTRAN or C languages can be achieved by using the `EVAL` function (see §7.4.4).

A reserved variable is the value `P0` which is used as the global reference momentum for normalising all magnetic field coefficients. Example:

```
REAL GEV=100;
P0=GEV;
```

Circular definitions are not allowed:

```
X=X+1;    // X cannot be equal to X+1
A=B;
B=A;      // A and B are equal, but of unknown value
```

However, redefinitions by assignment are allowed:

```
X=EVAL(X+1);
```

Real Vector Variables

```
REAL VECTOR variable-name=vector-expression;
```

The keyword `REAL` is optional. This statement creates a new global variable `variable-name` and discards any old variable with the same name. Its value depends on all quantities occurring in `vector-expression` (see §6.14) on the right-hand side. Whenever an operand changes in `vector-expression`, a new value is calculated. The definition may be thought of as a mathematical equation. However, OPAL is not able to solve the equation for a quantity on the right-hand side.

Example:

```
REAL VECTOR A = TABLE(10, #);
REAL VECTOR B = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

Circular definitions are not allowed, but redefinitions by assignment are allowed.

Logical Variables

```
BOOL variable-name=logical-expression;
```

This statement creates a new global variable `variable-name` and discards any old variable with the same name. Its value depends on all quantities occurring in `logical-expression` (see §6.5). Whenever an operand changes in `logical-expression`, a new value is calculated. The definition may be thought of as a mathematical equation. However, OPAL is not able to solve the equation for a quantity on the right-hand side.

Example:

```
BOOL FLAG = X != 0;
```

Circular definitions are not allowed, but redefinitions by assignment are allowed.

7.4.2 Symbolic Constants

OPAL recognises a few built-in mathematical and physical constants. (see Tab. 6.8) Additional constants can be defined by the command

```
REAL CONST label:CONSTANT=<real-expression>;
```

which defines a constant with the name `label`. The keyword `REAL` is optional, and `label` must be unique. An existing symbolic constant can never be redefined. The `real-expression` is evaluated at the time the `CONST` definition is read, and the result is stored as the value of the constant.

Example:

```
CONST IN=0.0254; // conversion of inches to metres
```

7.4.3 Vector Values

A vector of expressions is established by a statement

```
REAL VECTOR vector-name=vector-expression;
```

The keyword `REAL` is optional. It creates a new global vector `vector-name` and discards any old vector with the same name. Its value depends on all quantities occurring in `vector-expression` (see §6.14). Whenever an operand changes in `vector-expression`, a new value is calculated. The definition may be thought of as a mathematical equation. However, OPAL is not able to solve the equation for a quantity on the right-hand side.

Example:

```
VECTOR A_AMPL={2.5e-3,3.4e-2,0,4.5e-8};
VECTOR A_ON=TABLE(10,1);
```

Circular definitions are not allowed.

7.4.4 Assignment to Variables

A value is assigned to a variable or vector by using the function `EVAL(real-expression)`. When seen, this function is immediately evaluated and replaced by the result treated like a constant.

```
variable-name=EVAL(real-expression);
```

This statement acts like a FORTRAN or C assignment. The `real-expression` or `vector-expression` is **evaluated**, and the result is assigned as a constant to the variable or vector on the left-hand side. Finally the expression is discarded. The `EVAL` function can also be used within an expression, e. g.:

```
vector-name=TABLE(range,EVAL(real-expression));
vector-name={...,EVAL(real-expression),...};
```

A sequence like the following is permitted:

```
...           // some definitions
X=0;          // create variable X with value
              // zero
WHILE (X <= 0.10) {
  TWISS,LINE=...; // uses X=0, 0.01, ..., 0.10
  X=EVAL(X+.01); // increment variable X by 0.01
                // CANNOT use: X=X+.01;
}
```

7.4.5 VALUE: Output of Expressions

The statement

```
VALUE,VALUE=expression-vector;
```

evaluates a set of expressions using the most recent values of any operands and prints the results on the standard error file.

Example:

```
A=4;
VALUE,VALUE=TABLE(5,#*A);
P1=5;
P2=7;
VALUE,VALUE={P1,P2,P1*P2-3};
```

These commands give the results:

```
value: {0*A,1*A,2*A,3*A,4*A} = {0,4,8,12,16}
value: {P1,P2,P1*P2-3} = {5,7,32}
```

This commands serves mainly for printing one or more quantities which depend on matched attributes. It also allows use of OPAL as a programmable calculator. One may also tabulate functions.

7.4.6 H5merge

With the H5merge utility (*H5hut/tools/H5PartMerge/src/H5merge.cpp*) I can prune h5-files. With this feature you can hence restart from every time step.

```
H5merge bigfile.h5[0:100] input.h5
```

The first 100 steps from the file `bigfile.h5` are copied into the file `input.h5` from which the simulation can be restarted, as shown above. The H5merge utility comes with H5hut.

7.5 Miscellaneous Commands

7.5.1 ECHO Statement

The ECHO statement has two formats:

```
ECHO,MESSAGE=message;
ECHO,message;           // shortcut
```

`message` is a string value (see §6.4). It is immediately transmitted to the ECHO stream.

7.5.2 SYSTEM: Execute System Command

During an interactive OPAL session the command SYSTEM allows to execute operating system commands. After execution of the system command, successful or not, control returns to OPAL. At present this command is only available under UNIX or VM/CMS. It has two formats:

```
SYSTEM,CMD=string;
SYSTEM,string;         // shortcut
```

The string (see §6.4) `string` must be a valid operating system command.

7.5.3 SYSTEM Command under UNIX

Most UNIX commands can be issued directly.

Example:

```
SYSTEM,"ls -l"
```

causes a listing of the current directory in long form on the terminal.

7.6 TITLE Statement

The TITLE statement has three formats:

```
TITLE,STRING=page-header;    // define new page header
TITLE,page-header;          // shortcut for first format
TITLE,STRING="";            // clear page header
```

page-header is a string value (see §6.4). It defines the page header which will be used as a title for subsequent output pages. Before the first TITLE statement is encountered, the page header is empty. It can be redefined at any time.

7.7 File Handling

7.7.1 CALL Statement

The CALL command has two formats:

```
CALL,FILE=file-name;
CALL,file-name;
```

file-name is a string (see §6.4). The statement causes the input to switch to the named file. Input continues on that file until a STOP or an end of file is encountered. Example:

```
CALL,FILE="structure";
CALL,"structure";
```

7.7.2 SAVE Statement

The SAVE command has two formats:

```
SAVE,FILE=file-name
```

file-name is a string (see §6.4). The command causes all beam element, beam line, and parameter definitions to be written on the named file. Examples:

```
SAVE,FILE="structure";
SAVE,"structure";
```

7.7.3 MAKESEQ Statement

A file containing a machine sequence can be generated in OPAL by the command

```
MAKESEQ,LINE=string,NAME=string,FILE=string;
```

Please note this is not yet supported for OPAL-Tand OPAL-CYCL.

The named beam line (see §9) or sequence (see §??) is written as a flat SEQUENCE (see §??) with the given name on the named file. All required elements and parameters are also written. All expressions are evaluated and only their values appear in the output. The command has the following attributes:

LINE The line for which a flat sequence is to be written.

NAME The name to be given to the sequence written.

FILE The name of the file to receive the output.

7.8 IF: Conditional Execution

Conditional execution can be requested by an `IF` statement. It allows usages similar to the C language `if` statement:

```
IF (logical) statement;
IF (logical) statement; ELSE statement;
IF (logical) { statement-group; }
IF (logical) { statement-group; }
    ELSE { statement-group; }
```

Note that all statements must be terminated with semicolons (;), but there is no semicolon after a closing brace. The statement or group of statements following the `IF` is executed if the condition is satisfied. If the condition is false, and there is an `ELSE`, the statement or group following the `ELSE` is executed.

7.9 WHILE: Repeated Execution

Repeated execution can be requested by a `WHILE` statement. It allows usages similar to the C language `while` statement:

```
WHILE (logical) statement;
WHILE (logical) { statement-group; }
```

Note that all statements must be terminated with semicolons (;), but there is no semicolon after a closing brace. The condition is re-evaluated in each iteration. The statement or group of statements following the `WHILE` is repeated as long as the condition is satisfied. Of course some variable(s) must be changed within the `WHILE` group to allow the loop to terminate.

7.10 MACRO: Macro Statements (Subroutines)

Subroutine-like commands can be defined by a `MACRO` statement. It allows usages similar to C language function call statements. A macro is defined by one of the following statements:

```
name(formals): MACRO { token-list }
name(): MACRO { token-list }
```

A macro may have formal arguments, which will be replaced by actual arguments at execution time. An empty formals list is denoted by (). Otherwise the `formals` consist of one or more names, separated by commas. The `token-list` consists of input tokens (strings, names, numbers, delimiters etc.) and is stored unchanged in the definition.

A macro is executed by one of the statements:

```
name(actuals);
name();
```

Each actual consists of a set of tokens which replaces all occurrences of the corresponding formal name. The actuals are separated by commas. Example:

```
// macro definitions:
SHOWIT(X): MACRO {
    SHOW, NAME = X;
```

```
}  
DOIT(): MACRO {  
    DYNAMIC, LINE=RING, FILE="DYNAMIC.OUT";  
}  
  
// macro calls:  
SHOWIT(PI);  
DOIT();
```

Chapter 8

Elements

8.1 Element Input Format

All physical elements are defined by statements of the form

```
label:keyword, attribute,..., attribute
```

where

label

Is the name to be given to the element (in the example QF), it is an identifier (see §6.2).

keyword

Is a keyword (see §6.2), it is an element type keyword (in the example QUADRUPOLE),

attribute

normally has the form

```
attribute-name=attribute-value
```

attribute-name

selects the attribute from the list defined for the element type keyword (in the example L and K1). It must be an identifier (see §6.2)

attribute-value

gives it a value (see §6.3) (in the example 1.8 and 0.015832).

Omitted attributes are assigned a default value, normally zero.

Example:

```
QF: QUADRUPOLE, L=1.8, K1=0.015832;
```

8.2 Common Attributes for all Elements

The following attributes are allowed on all elements:

TYPE A string value (see §6.4). It specifies an “engineering type” and can be used for element selection.

APERTURE A real vector with an arbitrary length which describes the element aperture. It is ignored by OPAL, but it can be used in other programs.

WAKEF Defines the type of wake to be applied: *WT*, *WL* or *WTL* for transverse, longitudinal or both.

other All elements can have arbitrary additional attributes which are not defined below. Such attributes must have a name different from all defined attributes and single real values.

Only the *TYPE* attribute is used by OPAL, but the *SAVE* command (see §7.7.2) saves all attributes.

8.3 Drift Spaces


```
label:DRIFT, TYPE=string, APERTURE=real-vector, L=real;
```

A DRIFT space has one real attribute:

L The drift length (default: 0 m)

Examples:

```
DR1:DRIFT, L=1.5;
DR2:DRIFT, L=DR1->L, TYPE=DRF;
```

The length of DR2 will always be equal to the length of DR1. The reference system for a drift space is a Cartesian coordinate system (see Fig. ??). This is a restricted feature:  OPAL-CYCL. In OPAL-Tdrifts are implicitly given, if no field is present.

8.4 Bending Magnets

Two different type keywords are recognised for bending magnets, they are distinguished only by the reference system used:

RBEND is a rectangular bending magnet. It has parallel pole faces and is based on a Cartesian reference system (see Fig. ??).

SBEND is a sector bending magnet. Its pole faces meet at the centre of curvature of the curved reference system (see Fig. ??).

Using a RBEND or SBEND in OPAL-Tmode requires a field map, so it is a quite different implementation than that found in OPAL-MAP. The following commands define these elements in OPAL-Tmode:

```

Label: SBEND, FMAPFN=string, L=real, GAP=real, ANGLE=real,
             K0=real, K0S=real, K1=real, ROTATION=real
             E1=real, E2=real, BETA=real, ELEMEDGE=real,
             DESIGNENERGY=real;
Label: RBEND, FMAPFN=string, L=real, GAP=real ANGLE=real,
             K0=real, K0S=real, K1=real, ROTATION=real,
             E1=real, BETA=real, ELEMEDGE=real,
             DESIGNENERGY=real;

```

FMAPFN Field maps in the T7 format can be specified. This field map has to be of type 3DMagnetoStatic (not yet implemented), 1DProfile1 or 1DProfile2. (See C.2.)

It is possible also to use a default field map, FMAPN = "1DPROFILE1-DEFAULT". In this case, internal values for the Enge function coefficients are used. These are obtained from [44]. The fringe field is calculated using the function:

$$F(z) = \frac{1}{1 + e^{\sum_n c_n (z/D)^{n-1}}} \quad (8.1)$$

where D is the full gap of the magnet, n is the Enge function order and z is the distance from the zero of the Enge function perpendicular to the edge of the dipole. For this default case, $n = 5$, and the Enge coefficients are, from [44]:

$$\begin{aligned}
c_0 &= 0.478959 \\
c_1 &= 1.911289 \\
c_2 &= -1.185953 \\
c_3 &= 1.630554 \\
c_4 &= -1.082657 \\
c_5 &= 0.318111
\end{aligned}$$

If using the default, one must set GAP to declare the full gap of the magnet and one must set L. This last parameter sets the distance between Enge function zeros for the entrance and exit of the dipole. This effectively sets the magnet length. See Figure C.7 and Figure C.8. Both the entrance and the exit use the same Enge function for the fringe fields.

When using the default, OPAL-Twill adjust the extend of the fringe field calculation to be "reasonable" values. That is, it extends the fringe fields outside the magnet until the field is 10^{-4} the field at the center of the magnet. The fringe calculation is extended inside the magnet in a similar fashion. Be careful if using a large gap. This can lead to funny magnet field profiles if it is on the order of the magnet L.

Finally, this default option is implemented in order to make it easier to include bends in a simulation without having to generate field maps for each one. This should give you reasonable results. But be aware that at some point you will want to replace the default with profiles from your own magnets as the your design moves forward.

L This parameter is ignored if the element has a field map defined. If `FMAPN = "1DPROFILE1-DEFAULT"` then `L` must be defined and is the distance between the entrance and exit Enge polynomial zeros in meters. See Figure C.7 and Figure C.8.

GAP This parameter is ignored if the element has a field map defined. If `FMAPN = "1DPROFILE1-DEFAULT"` then `GAP` must be defined and is the full gap of the magnet in meters.

ANGLE Since OPAL-Tuses a field map, we must specify a scaling factor for that map. There are two ways to do this. First, we can specify the peak value of that field (see the `K0` and `K0S` parameters below). Second, we can set the `ANGLE` parameter. If the `ANGLE` is set then OPAL-Twill attempt to calculate the required strength of the magnet to bend the beam through that angle using the average energy of the beam when it enters the magnet. The `ANGLE` parameter overrides the field magnitude set by `K0` and/or `K0S`. The convention for a positive bend angle is to bend the beam to the right, in the negative x direction.

If a negative bend angle is defined, this is interpreted as a positive bend angle rotated by 180° about the z axis. In the case where a negative bend angle is defined along with a magnet `ROTATION` (see below) this 180° rotation is added to the defined `ROTATION`.

K0 Rather than set the bend angle, we can also declare the field strength of the magnet via `K0`. `K0` is the peak field in the vertical direction in Tesla. A positive value of `K0` will bend a positive charge in the negative x direction (positive bend angle). `K0`, in conjunction with `K0S` can also be used to rotate the magnet about the z axis. (See `ROTATION` parameter.)

K0S `K0S` is analogous to `K0` but sets the peak field in the horizontal, or x, direction. In conjunction with `K0` it can also be used to rotate the magnet about the z axis. (See `ROTATION` parameter.)

K1 `K1` defines the field gradient index of the magnet $K_1 = \frac{1}{B\rho} \frac{\partial B_y}{\partial x}$. This introduces quadrupole focusing. This gradient will always be defined in the bend plane of the magnet, even when the magnet is rotated.

ROTATION When setting the bend angle, we assume a positive bend in the x plane. If one wants to bend in a plane other than the horizontal, or x plane, the magnet needs to be rotated about the z axis. This is set via the `ROTATION` parameter.

A bend magnet is defined to have zero rotation if it has a purely y field and bends particles in the positive direction (toward the negative x axis). A magnet rotated by 90° will have a field in the x direction and bend particles in the positive y direction. A magnet rotated by 180° will have a field in the y direction and bend particles in the negative direction (toward the positive x axis). And so on. Angles that are not multiples of 90° will bend in both planes.

Another way to set the magnet rotation is via the `K0` and `K0S` parameters. However, in this case the rotation of the bend is defined by the direction of the field. For instance, a positive value for `K0` and a zero value of `K0S` gives a 0° rotation. A positive value of `K0S` and a zero value of `K0` gives a rotation of -45° . And so on. When inputting magnet parameters in this way one must be careful because the signs of the entrance and exit angles may not be consistent with bend the direction of the magnet, especially when bending negative particles.

E1 Edge angle for the magnet entrance. This is the angle between the design beam trajectory and the face of the bend. `E1` = 0 for a perpendicular face and `E1` > 0 if the face is rotated anticlockwise from this perpendicular position (as seen from above, y > 0, on to the plane). See Figure ??, Figure ?? and Figure 8.1

E2 Edge angle for the magnet exit. This parameter is not used for an `RBEND` element as it is fixed by the entrance angle, `E1`, and the rectangular geometry of the magnet. `E2` > 0 if the exit face of the magnet is rotated anticlockwise from the position where the design trajectory is perpendicular to the exit face. See Figure ??.

BETA Analogous to `E1`, but slightly more complicated. This is a rotation of the magnet about the x axis. See Figure 8.1.

ELEMEDGE The edge of the field is specified in floor coordinates from the cathode in m. If one uses `1DProfile1` or `1DProfile2` field files, this is the location of the zero of the Enge function for the entrance fringe field (see C.2). For a `3DMagnetoStatic` map, it is the location of the start of that map.

DESIGNENERGY Energy of the reference particle. If the magnet strength is set With `K0` and/or `K0S`, this is set as the energy of the reference trajectory through the magnet, and therefore sets the reference beam bend angle. It is used to calculate the radius of the circular path in the $y = 0$ plane. The radius in turn is needed in the calculation of the CSR wakefield. The design energy is also used to track a single particle through the bend while initializing the elements. `DESIGNENERGY` fixes the speed while `E1` and `BETA` define the initial direction of the velocity in local coordinates. From the path this particle travels a map between the path length (s-coordinate) and the local z-coordinate is constructed. If the bend angle, `ANGLE`, is set, `DESIGNENERGY` is automatically defined as the average energy of the beam when it enters the field and the element is re-initialized at this time.

FINT The field integral (default =0).

HGAP The half gap of the magnet (default: 0 m).

The pole face rotation angles are referred to the magnet model for a `RBEND` (see Fig. ??) and `SBEND` (see Fig. ??) respectively. The quantities `FINT` and `HGAP` specify the finite extent of the fringe fields as defined in SLAC-75 [8] as follows:

$$\text{FINT} = \int_0^\infty \frac{B_y(s)(B_0 - B_y(s))}{B_0^2 g} ds, \quad \text{HGAP} = 2g.$$

The default values of zero corresponds to the hard-edge approximation, i.e. a rectangular field distribution. For other approximations, enter the correct value of the half gap, and one of the following values for `FINT`:

| Typical values for FINT | |
|--------------------------------------|------|
| Linear Field drop-off | 1/6 |
| Clamped "Rogowski" fringing field | 0.4 |
| Unclamped "Rogowski" fringing field | 0.7 |
| "Square-edged" non-saturating magnet | 0.45 |

A reasonable average value for `FINT` is 0.5. All these dipole examples have the same bend angle:

```
BR:RBEND, L=5.5, ANGLE=+0.001; // Deflection to the right
BR:RBEND, L=5.5, K0=+0.001/5.5; // Deflection to the right
// This magnet has a
// straight reference
BL:SBEND, L=5.5, ANGLE=-0.001; // Deflection to the left
BL:SBEND, L=5.5, K0=-0.001/5.5; // Deflection to the left
// This magnet has a
// straight reference
```

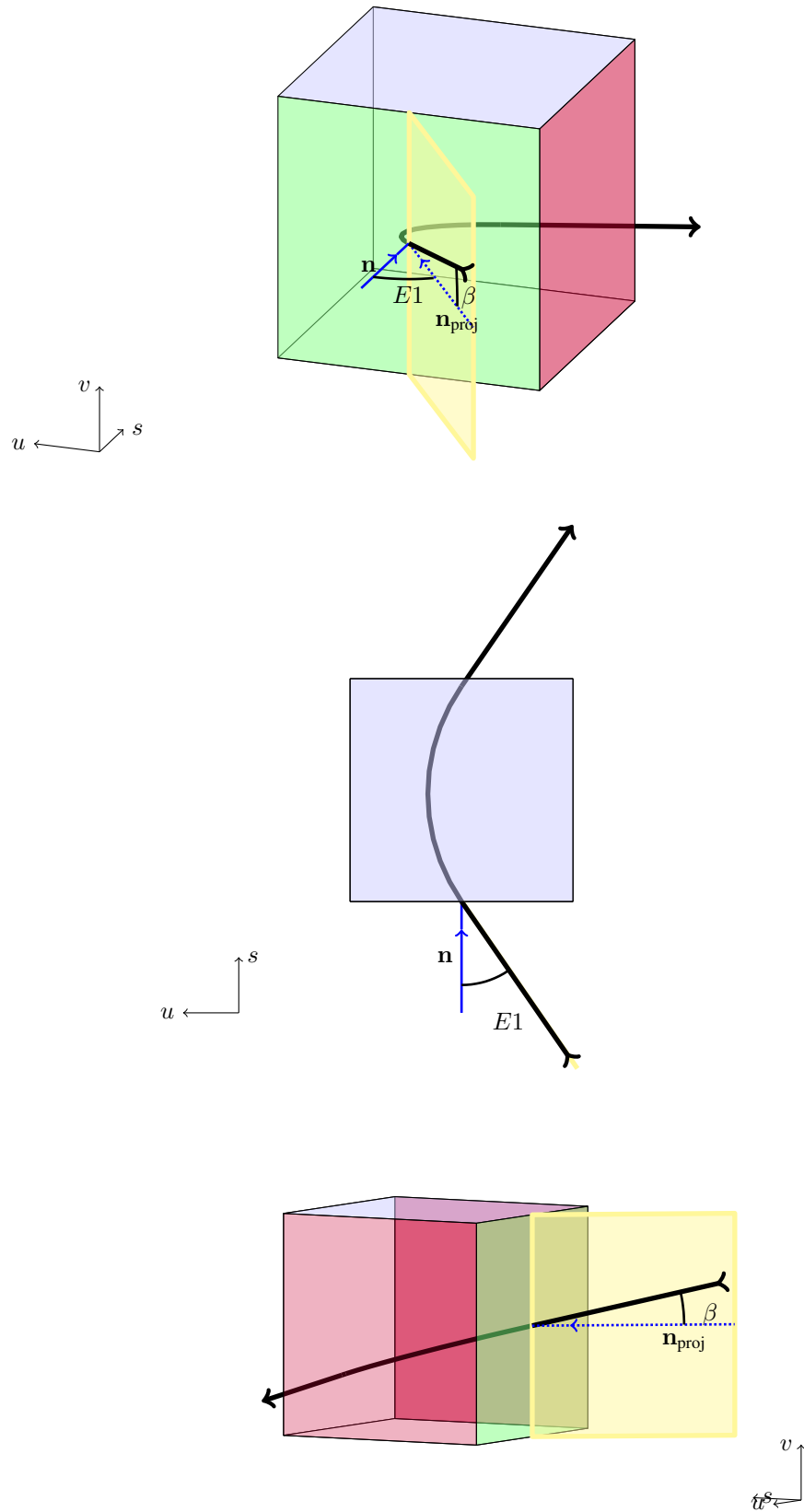



Figure 8.1: Visualisation of angles used to rotate the bend relative to the incoming beam, where \mathbf{n} is the normal of the face.

8.5 Quadrupole

```
label:QUADRUPOLE, TYPE=string, APERTURE=real-vector,
      L=real, K1=real, K1S=real;
```

The reference system for a quadrupole is a Cartesian coordinate system (see Fig. ??). This is a restricted feature:  OPAL-CYCL. A QUADRUPOLE has three real attributes:

L The quadrupole length (default: 0 m).

K1 The normal quadrupole component $K_1 = \frac{\partial B_y}{\partial x}$.

K1S The skew quadrupole component. $K_{1s} = \frac{\partial B_x}{\partial x}$.

FMAPFN Field maps in the *T7* format can be specified. This field map has to be of type 3DMagnetoStatic.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m.

Example:

```
QP1: Quadrupole, L=1.20, ELEMEDGE=-0.5265,
      FMAPFN="1T1.T7", K1=0.11;
```

8.6 Sextupole

[TODO: IN OPAL-T?]

```
label:SEXTUPOLE, TYPE=string, APERTURE=real-vector,
      L=real, K2=real, K2S=real;
```

A SEXTUPOLE has three real attributes:

L The sextupole length (default: 0 m). A thin sextupole (see §??) is defined by setting the length to zero.

K2 The normal sextupole component $K_2 = \frac{1}{B\rho} \frac{\partial^2 B_y}{\partial x^2}$. The default is 0m^{-3} . The component is positive, if B_y is positive on the positive x -axis.

K2S The skew sextupole component $K_{2s} = \frac{1}{B\rho} \frac{\partial^2 B_x}{\partial x^2}$. The default is 0m^{-3} . The component is negative, if B_x is positive on the positive x -axis.

Example:

```
S:SEXTUPOLE, L=0.4, K2=0.00134;
```

The reference system for a sextupole is a Cartesian coordinate system (see Fig. ??).

8.7 Octupole

[TODO: IN OPAL-T?]

```
label:OCTUPOLE, TYPE=string, APERTURE=real-vector,
      L=real, K3=real, K3S=real;
```

An OCTUPOLE has three real attributes:

L The octupole length (default: 0 m). A thin octupole (see §??) is defined by setting the length to zero.

K3 The normal sextupole component $K_3 = \frac{1}{B\rho} \frac{\partial^3 B_y}{\partial x^3}$. The default is 0m^{-4} . The component is positive, if B_y is positive on the positive x -axis.

K3S The skew sextupole component $K_{3s} = \frac{1}{B\rho} \frac{\partial^3 B_x}{\partial x^3}$. The default is 0m^{-4} . The component is negative, if B_x is positive on the positive x -axis.

Example:

```
O3:OCTUPOLE, L=0.3, K3=0.543;
```

The reference system for an octupole is a Cartesian coordinate system (see Fig. ??).

8.8 General Multipole

[TODO: IN OPAL-T?] A MULTIPOLE is thin lens of arbitrary order, including a dipole:

```
label:MULTIPOLE, TYPE=string, APERTURE=real-vector,
      L=real, KNORMAL=real-vector, KSKEW=real-vector;
```

L The multipole length (default: 0 m). A thin multipole (see §??) is defined by setting the length to zero.

KN A real vector (see §6.14), containing the normal multipole coefficients. A component is positive, if B_y is positive on the positive x -axis.

KS A real vector (see §6.14), containing the skew multipole coefficients. A component is negative, if B_x is positive on the positive x -axis.

The multipole coefficients are defined as $K_n = \frac{1}{B\rho} \frac{\partial^n B_y}{\partial x^n}$. (default: 0m^{-n}). The order n is unlimited, but all components up to the maximum must be given, even if zero. The number of poles of each component is $(2n + 2)$. The most important error components of fully symmetric quadrupoles are: `KNORMAL[5]`, the 12-pole, and `KNORMAL[9]`, the twenty-pole. Superposition of many multipole components is permitted. The reference system for a multipole is a Cartesian coordinate system (see Fig. ??).

Example:

```
M27:MULTIPOLE, L=1, KNORMAL[3]=0.0001, KSKEW[2]=0.0001;
```

A multipole with no dipole component has no effect on the reference orbit, i.e. the reference system at its exit is the same as at its entrance. If it includes a dipole component, it has the same effect on the reference orbit as a `SBEND` with the same length and deflection angle `KNORMAL[0]*L`.

8.9 Solenoid

```
label:SOLENOID, TYPE=string, APERTURE=real-vector,
      L=real, KS=real;
```

A SOLENOID has two real attributes:

L The length of the solenoid (default: 0 m)

KS The solenoid strength K_s (default: 0 rad/m). For positive KS and positive particle charge, the solenoid field points in the direction of increasing s .

The reference system for a solenoid is a Cartesian coordinate system (see Fig. ??).
Using a solenoid in OPAL-t mode, the following additional parameters are defined:

FMAPFN Field maps in the $T7$ format can be specified.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m.

Example:

```
SP1: Solenoid, L=1.20, ELEMEDGE=-0.5265, KS=0.11,
      FMAPFN="1T1.T7";
```

8.10 Cyclotron

```
label:CYCLOTRON, TYPE=string, CYHARMON=int,
      PHIINIT=real, PRINIT=real, RINIT=real,
      SYMMETRY=real, RFFREQ=real, FMAPFN=string;
```

A CYCLOTRON object includes the main characteristics of a cyclotron, the magnetic field, and also the initial condition of the injected reference particle, and it has currently the following attributes:

TYPE The data format of field map, Currently three formats are implemented: CARBONCYCL, CYCIAE, AVFEQ, FFAG, BANDRF and default PSI format. For the details of their data format, please read Section 5.4.

CYHARMON The hamonic number of the cyclotron h .

RFFREQ The RF system f_{rf} (unit:MHz, default: 0). The particle revolution frequency $f_{rev} = f_{rf} / h$.

FMAPFN Filename for the magnetic field map.

SYMMETRY Defines symmetrical fold number of the B field map data.

RINIT The initial radius of reference particle (unit: mm, default: 0)

PHINIT The initial azimuth of reference particle (unit: degree, default: 0)

PRINIT Initial radial momenta of reference particle $P_r = \beta\gamma$ (default : 0)

MINZ The minimal vertical extent of the machine (unit: mm, default : -10000.0)

MAXZ The maximal vertical extent of the machine (unit: mm, default : 10000.0)

MINR Minimal radial extent of the machine (unit: mm, default : 0.0)

MAXR Minimal radial extent of the machine (unit: mm, default : 10000.0)

During the tracking, the particle (r, z, θ) will be deleted if $\text{MINZ} < z < \text{MAXZ}$ or $\text{MINR} < r < \text{MAXR}$, and it will be recorded in the ASCII file *inputfilename.loss*. Example:

```
ring: Cyclotron, TYPE="RING", CYHARMON=6, PHIINIT=0.0,
      PRINIT=-0.000240, RINIT=2131.4 , SYMMETRY=8.0,
      RFFREQ=50.650, FMAPFN="s03av.nar",
      MAXZ=10, MINZ=-10, MINR=0, MAXR=2500;
```

If TYPE is set to BANDRF, the 3D electric field map of RF cavity will be read from external h5part file and 4 extra arguments need to specified:

RFFMAPFN The filename for the electric fieldmap in h5part binary format.

RFPHI The Initial phase of the electric field map (rad)

ESCALE The maximal value of the electric fieldmap (MV/m)

SUPERPOSE An option whether all of the electric field maps are superposed, The is valid when more than one electric field map is read. (default: true)

Example for single electric field map:

```
COMET: Cyclotron, TYPE="BANDRF", CYHARMON=2, PHIINIT= -71.0,
PRINIT=pr0, RINIT= r0 , SYMMETRY=1.0, FMAPFN="Tosca_map.txt",
RFPHI=Pi, RFFREQ=72.0, RFMAPFN="efield.h5part",
ESCALE=1.06E-6;
```

We can have more than one RF field maps.

Example for multiple RF field maps:

```
COMET: Cyclotron, TYPE="BANDRF", CYHARMON=2, PHIINIT=-71.0,
PRINIT=pr0, RINIT=r0 , SYMMETRY=1.0, FMAPFN="Tosca_map.txt",
RFPHI= {Pi,0,Pi,0}, RFFREQ={72.0,72.0,72.0,72.0},
RFMAPFN={"e1.h5part","e2.h5part","e3.h5part","e4.h5part"},
ESCALE={1.06E-6, 3.96E-6,1.3E-6,1.E-6}, SUPERPOSE=true;
```

In this example SUPERPOSE is set to true. Therefore, if a particle locates in multiple field regions, all the field maps are superposed. if SUPERPOSE is set to false, then only one field map, which has highest priority, is used to do interpolation for the particle tracking. The priority ranking is decided by their sequence in the list of RFMAPFN argument, i.e., "e1.h5part" has the highest priority and "e4.h5part" has the lowest priority.

For RF cavity, another way is reading the RF voltage profile in the RFAVITY element (see Section 8.11) and just do a momentum kick when a particle crosses the RF gap. In the center region of the compact cyclotron, the electric field shape of complicated and pay significant impact on the transverse beam dynamics, hence a simple momentum kick is not enough. In this case, we need to read 3D field map to do precise simulation.

In additions, the simplified trim-coil field model is also implemented so as to do fine tuning on the magnetic field. A trim-coil can be defined by 4 arguments:

TCR1 Tthe inner radius of the trim coil (mm)

TCR2 Tthe outer radius of the trim coil (mm)

MBTC The maximal B field of trim coil (kG)

SLPTC The slope of the rising edge (kG/mm)

This is a restricted feature: OPAL-CYCL.

8.11 RF Cavities (OPAL-T and OPAL-CYCL)

For an RFCAVITY the three modes have four real attributes in common:

```
label:RFCAVITY, APERTURE=real-vector, L=real,
      VOLT=real, LAG=real;
```

L The length of the cavity (default: 0 m)

VOLT The peak RF voltage (default: 0 MV). The effect of the cavity is $\delta E = \text{VOLT} \cdot \sin(2\pi(\text{LAG} - \text{HARMON} \cdot f_0 t))$.

LAG The phase lag [rad] (default: 0).

8.11.1 OPAL-Tmode

Using a RF Cavity in OPAL-t mode, the following additional parameters are defined:

FMAPFN Field maps in the *T7* format can be specified.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m.

TYPE Type specifies STANDING [default] or SINGLE GAP structures.

FREQ Defines the frequency of the RF Cavity in units of MHz. A warning is issued when the frequency of the cavity card does not correspond to the frequency defined in the FMAPFN file. The frequency of the cavity card overrides the frequency defined in the FMAPFN file.

APVETO If TRUE this cavity will not be auto-phased, instead the actual phase on the element is used.

Example standing wave cavity which mimics a DC gun:

```
gun: RFCavity, L=0.018, VOLT=-131/(1.052*2.658),
      FMAPFN="1T3.T7", ELEMEDGE=0.00,
      TYPE="STANDING", FREQ=1.0e-6;
```

Example of a two frequency standing wave cavity:

```
rf1: RFCavity, L=0.54, VOLT=19.961, LAG=193.0/360.0,
      FMAPFN="1T3.T7", ELEMEDGE=0.129, TYPE="STANDING",
      FREQ=1498.956;
rf2: RFCavity, L=0.54, VOLT=6.250, LAG=136.0/360.0,
      FMAPFN="1T4.T7", ELEMEDGE=0.129, TYPE="STANDING",
      FREQ=4497.536;
```

8.11.2 OPAL-CYCLmode

Using a RF Cavity (standing wave) in OPAL-CYCLmode, the following parameters are defined:

FMAPFN Defines name of file which stores normalized voltage amplitude curve of cavity gap in ASCII format.
(See data format in Section 5.4)

VOLT Sets peak value of voltage amplitude curve in MV.

TYPE Defines Cavity type, SINGLEGAP represents cyclotron type cavity.

FREQ Sets the frequency of the RF Cavity in units of MHz.

RMIN Sets the radius of the cavity inner edge in mm.

RMAX Sets the radius of the cavity outer edge in mm.

ANGLE Sets the azimuthal position of the cavity in global frame in degree.

PDIS Set shift distance of cavity gap from center of cyclotron in mm. If its value is positive, the shift direction is clockwise, namely, shift towards the smaller azimuthal angle.

GAPWIDTH Set gap width of cavity in mm.

PHI0 Set initial phase of cavity in degree.

Example of a RF cavity of cyclotron:

```
rf0: RfCavity, VOLT=0.25796, FMAPFN="Cav1.dat",
      TYPE="SINGLE GAP", FREQ=50.637, RMIN = 350.0,
      RMAX = 3350.0, ANGLE=35.0,   PDIS = 0.0,
      GAPWIDTH = 0.0, PHI0=phi01;
```

Fig. 8.2 shows the simplified geometry of a cavity gap and its parameters.

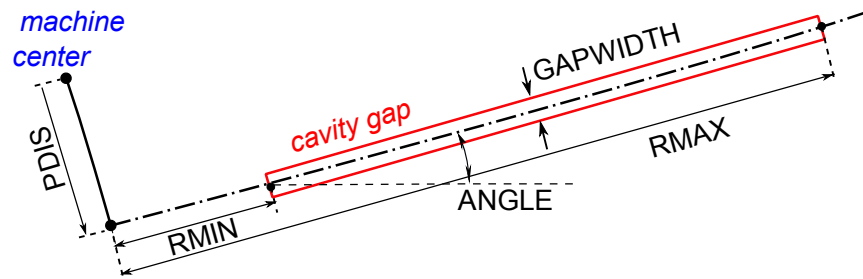


Figure 8.2: Schematic of the simplified geometry of a cavity gap and parameters

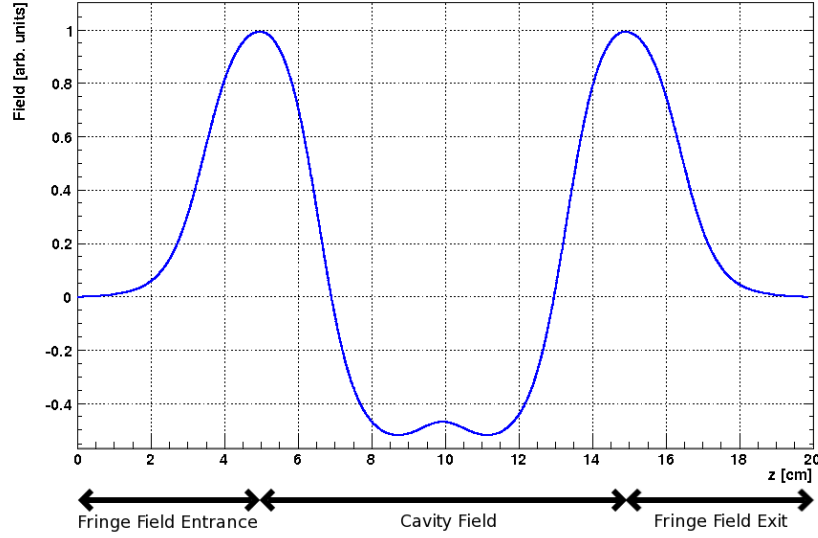


Figure 8.3: The on-axis field of an S-band (2997.924 MHz) TRAVELINGWAVE structure. The field of a single cavity is shown between its entrance and exit fringe fields. The fringe fields extend one half wavelength ($\lambda/2$) to either side.

8.12 Traveling Wave Structure

An example of a 1D TRAVELINGWAVE structure field map is shown in Figure 8.3. This map is a standing wave solution generated by Superfish and shows the field on axis for a single accelerating cavity with the fringe fields of the structure extending to either side. OPAL-Treads in this field map and constructs the total field of the TRAVELINGWAVE structure in three parts: the entrance fringe field, the structure fields and the exit fringe field.

The fringe fields are treated as standing wave structures and are given by:

$$\begin{aligned} \mathbf{E}_{\text{entrance}}(\mathbf{r}, t) &= \mathbf{E}_{\text{from-map}}(\mathbf{r}) \cdot \text{VOLT} \cdot \cos(2\pi \cdot \text{FREQ} \cdot t + \phi_{\text{entrance}}) \\ \mathbf{E}_{\text{exit}}(\mathbf{r}, t) &= \mathbf{E}_{\text{from-map}}(\mathbf{r}) \cdot \text{VOLT} \cdot \cos(2\pi \cdot \text{FREQ} \cdot t + \phi_{\text{exit}}) \end{aligned}$$

where VOLT and FREQ are the field magnitude and frequency attributes (see below). $\phi_{\text{entrance}} = \text{LAG}$, the phase attribute of the element (see below). ϕ_{exit} is dependent upon both LAG and the NUMCELLS attribute (see below) and is calculated internally by OPAL-T.

The field of the main accelerating structure is reconstructed from the center section of the standing wave solution shown in Figure 8.3 using

$$\begin{aligned} \mathbf{E}(\mathbf{r}, t) &= \frac{\text{VOLT}}{\sin(2\pi \cdot \text{MODE})} \\ &\times \left\{ \mathbf{E}_{\text{from-map}}(\mathbf{x}, \mathbf{y}, \mathbf{z}) \cdot \cos\left(2\pi \cdot \text{FREQ} \cdot t + \text{LAG} + \frac{\pi}{2} \cdot \text{MODE}\right) \right. \\ &\quad \left. + \mathbf{E}_{\text{from-map}}(\mathbf{x}, \mathbf{y}, \mathbf{z} + \mathbf{d}) \cdot \cos\left(2\pi \cdot \text{FREQ} \cdot t + \text{LAG} + \frac{3\pi}{2} \cdot \text{MODE}\right) \right\} \end{aligned}$$

where \mathbf{d} is the cell length and is defined as $\mathbf{d} = \lambda \cdot \text{MODE}$. MODE is an attribute of the element (see below). When calculating the field from the map ($\mathbf{E}_{\text{from-map}}(\mathbf{x}, \mathbf{y}, \mathbf{z})$), the longitudinal position is referenced to the

start of the cavity fields at $\frac{\lambda}{2}$ (In this case starting at $z = 5.0$ cm). If the longitudinal position advances past the end of the cavity map ($\frac{3\lambda}{2} = 15.0$ cm in this example), an integral number of cavity wavelenths is subtracted from the position until it is back within the map's longitudinal range.

A TRAVELINGWAVE structure has seven real attributes, one integer attribute, one string attribute and one boolean attribute:

```
label:TRAVELINGWAVE, APERTURE=real-vector, L=real,
      VOLT=real, LAG=real, FMAPFN=string,
      ELEMEDGE=real, FREQ=real, NUMCELLS=integer,
      MODE=real;
```

L The length of the cavity (default: 0 m). In OPAL-Tthis attribute is ignored, the length is defined by the field map and the number of cells.

VOLT The peak RF voltage (default: 0 MV). The effect of the cavity is $\delta E = \text{VOLT} \cdot \sin(\text{LAG} - 2\pi \cdot \text{FREQ} \cdot t)$.

LAG The phase lag [rad] (default: 0).

FMAPFN Field maps in the T7 format can be specified.

ELEMEDGE The edge of the field is specified absolute (floor space co-ordinates) in m. This is the position of the front edge of the first TRAVELINGWAVE cavity. (In the simulation the actual field will extend $\frac{1}{2}$ wavelength in front of this position.)

FREQ Defines the frequency of the traveling wave structure in units of MHz. A warning is issued when the frequency of the cavity card does not correspond to the frequency defined in the FMAPFN file. The frequency defined in the FMAPFN file overrides the frequency defined on the cavity card.

NUMCELLS Defines the number of cells in the tank. (The cell count should not include the entry and exit half cell fringe fields.)

MODE Defines the mode in units of 2π , for example $\frac{1}{3}$ stands for a $\frac{2\pi}{3}$ structure.

FAST If FAST is true and the provided field map is in 1D then a 2D field map is constructed from the 1D on-axis field, see section C.2. To track the particles the field values are interpolated from this map instead of using an FFT based algorithm for each particle and each step. (default: FALSE)

APVETO If TRUE this cavity will not be auto-phased, instead the actual phase on the element is used.

Use of a traveling wave requires the particle momentum **P** and the particle charge **CHARGE** to be set on the relevant optics command before any calculations are performed.

Example of a L-Band travelling wave structure:


```
lrf0: TravelingWave, L=0.0253, VOLT=14.750,
      NUMCELLS=40, ELEMEDGE=2.73066,
      FMAPFN="INLB-02-RAC.Ez", MODE=1/3,
      FREQ=1498.956, LAG=248.0/360.0;
```

8.13 Monitors

A MONITOR detects all particles passing it and writes the position, the momentum and the time when they hit it into an H5hut file. Furthermore the exact position of the monitor is stored. It has always a length of 1 cm consisting of 0.5 cm drift, the monitor of zero length and another 0.5 cm drift. This is to prevent OPAL-Tfrom missing any particle. The positions of the particles on the monitor are interpolated from the current position and momentum one step before they would passe the monitor.

OUTFN The filename into which the monitor should write the collected data. The file is an H5hut file.

ELEMEDGE The position of the monitor is specified absolute (floor space co-ordinates) in m. This is the position at which the data is collected.

This is a restricted feature:  OPAL-CYCL.

8.14 Collimators

Three types of collimators are defined:

ECOLLIMATOR Elliptic aperture,

RCOLLIMATOR Rectangular aperture.

CCOLLIMATOR Radial rectangular collimator in cyclotrons

```
label:ECOLLIMATOR, TYPE=string, APERTURE=real-vector,
      L=real, XSIZE=real, YSIZE=real;
label:RCOLLIMATOR,TYPE=string, APERTURE=real-vector,
      L=real, XSIZE=real, YSIZE=real;
```

Either type has three real attributes:

L The collimator length (default: 0 m).

XSIZE The horizontal half-aperture (default: unlimited).

YSIZE The vertical half-aperture (default: unlimited).

For elliptic apertures, XSIZE and YSIZE denote the half-axes respectively, for rectangular apertures they denote the half-width of the rectangle. Optically a collimator behaves like a drift space, but during tracking, it also introduces an aperture limit. The aperture is checked at the entrance. If the length is not zero, the aperture is also checked at the exit.

Example:

```
COLLIM:ECOLLIMATOR, L=0.5, XSIZE=0.01, YSIZE=0.005;
```

The reference system for a collimator is a Cartesian coordinate system (see Fig. ??).

8.14.1 OPAL-Tmode

The RCOLLIMATOR and CCOLLIMATOR are not supported at the moment. A ECOLLIMATOR detects all particles which are outside the aperture defined by XSIZE and YSIZE. The lost particles are saved into an H5hut file defined by OUTFN. The ELEMEDGE defines the location of the collimator and L the length.

OUTFN The filename into which the monitor should write the collected data. The file is an H5hut file.

ELEMEDGE The position of the monitor is specified absolute (floor space co-ordinates) in m. This is the position at which the data is collected.

Example:

```
Col1:ECOLLIMATOR, L=1.0E-3, ELEMEDGE=3.0E-3, XSIZE=5.0E-4,
      YSIZE=5.0E-4, OUTFN="Coll.h5";
```

8.14.2 OPAL-CYCLmode

Only `CCOLLIMATOR` is available for OPAL-CYCL. This element is radial rectangular collimator which can be used to collimate the radial tail particles. So when a particle hit this collimator, it will be absorbed or scattered, the algorithm is based on the Monte Carlo method. Please note when a particle is scattered, it will not be recorded as the lost particle. If this particle leave the bunch, it will be removed during the integration afterwards, so as to maintain the accuracy of space charge solving.

XSTART The x coordinate of the start point. [mm]

XEND The x coordinate of the end point. [mm]

YSTART The y coordinate of the start point. [mm]

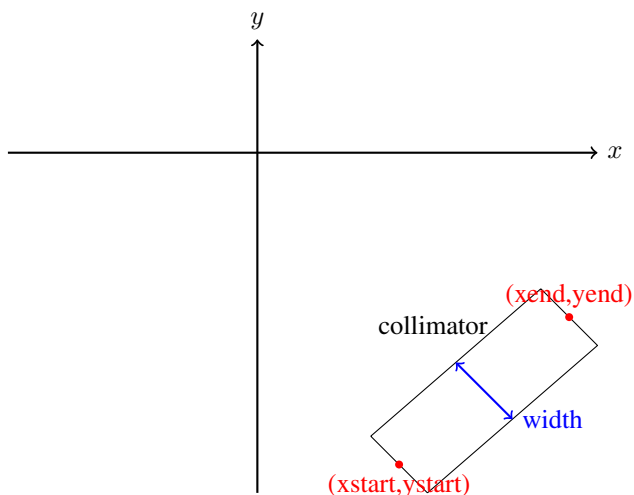
YEND The y coordinate of the end point. [mm]

ZSTART The vertical coordinate of the start point [mm]. Default value is -100 mm.

ZEND The vertical coordinate of the end point. [mm]. Default value is -100 mm.

WIDTH The width of the septum. [mm]

SURFACEPHYSICS Surfacephysics is an attribute of the element. Collimator physics is a only a kind of surfacephysics. It can be applied to any element. If the type of Surfacephysics is "Collimator", the material is defined here. The material "Cu", "Be", "Graphite" and "Mo" are defined until now. If this is not set, the surface physics module will not be activated. The particle hitting collimator will be recorded and directly deleted from the simulation.




Example:

```
y1=-0.0;
y2=0.0;
y3=200.0;
y4=205.0;
x1=-215.0;
x2=-220.0;
x3=0.0;
x4=0.0;
cmphys:surfacephysics, TYPE="Collimator", MATERIAL="Cu";
cmal: CCollimator, XSTART=x1, XEND=x2, YSTART=y1, YEND=y2,
```

```
ZSTART=2, ZEND=100, WIDTH=10.0, SURFACEPHYSICS=cmphys ;
cma2: CCollimator, XSTART=x3, XEND=x4, YSTART=y3, YEND=y4,
ZSTART=2, ZEND=100, WIDTH=10.0, SURFACEPHYSICS=cmphys;
```

The particles lost on the CCLLIMATOR are recorded in the ASCII file *inputfilename.loss*

8.15 Septum (OPAL-CYCL)

This is a restricted feature:  OPAL-T. The particles hitting on the septum is removed from the bunch. There are 5 parameters to describe a septum.

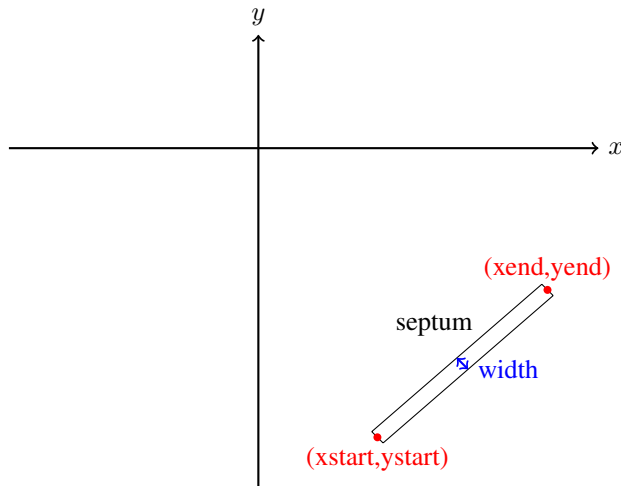
XSTART The x coordinate of the start point. [mm]

XEND The x coordinate of the end point. [mm]

YSTART The y coordinate of the start point. [mm]

YEND The y coordinate of the end point. [mm]

WIDTH The width of the septum. [mm]



Example:

```
eec2: Septum, xstart=4100.0, xend=4300.0,
ystart=-1200.0, yend=-150.0, width=0.05;
```

The particles lost on the SEPTUM are recorded in the ASCII file *inputfilename.loss*

8.16 Probe (OPAL-CYCL)

The particles hitting on the probe is recorded. There are 5 parameters to describe a probe.

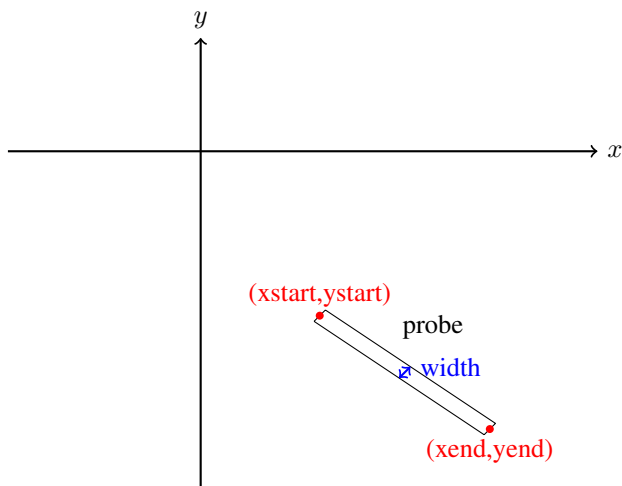
XSTART The x coordinate of the start point. [mm]

XEND The x coordinate of the end point. [mm]

YSTART The y coordinate of the start point. [mm]

YEND The y coordinate of the end point. [mm]

WIDTH The width of the probe, NOT used yet.



Example:

```
probl: Probe, xstart=4166.16, xend=4250.0,
ystart=-1226.85, yend=-1241.3;
```

The particles probed on the PROBE are recorded in the ASCII file *inputfilename.loss*. Please note that these particles are not deleted in the simulation, however, they are recorded in the “loss” file.

8.17 Stripper (OPAL-CYCL)

A stripper element strip the electron(s) from a particle. The particle hitting the stripper is recorded in the file, which contains the time, coordinates and momentum of the particle at the moment it hit the stripper. The charge and mass are changed. It has the same geometry as the PROBE element. Please note that the stripping physics is not included yet.

There are 9 parameters to describe a stripper.

XSTART The x coordinate of the start point. [mm]

XEND The x coordinate of the end point. [mm]

YSTART The y coordinate of the start point. [mm]

YEND The y coordinate of the end point. [mm]

WIDTH The width of the probe, NOT used yet.

OPCHARGE Charge number of the outgoing particle. Negative value represents negative charge.

OPMASS Mass of the outgoing particles. [GeV/c²]

OPYIELD Yield of the outgoing particle (the outcome particle number per income particle), the default value is 1.

STOP If STOP is true, the particle is stopped and deleted from the simulation; Otherwise, the outgoing particle continues to be tracked along the extraction path.

Example: H_2^+ particle stripping

```
probl: Stripper, xstart=4166.16, xend=4250.0,  
ystart=-1226.85, yend=-1241.3,  
opcharge=1, opmass=PMASS, opyield=2, stop=false;
```

No matter what the value of STOP is, the particles hitting on the STRIPPER are recorded in the ASCII file *input filename.loss*.

Chapter 9

Beam Lines

The accelerator to be studied is known to OPAL as a sequence of physical elements called a **beam line**. A beam line is built from simpler beam lines whose definitions can be nested to any level. A powerful syntax allows to repeat or to reflect pieces of beam lines. Formally a beam line is defined by a `LINE` command:

```
label:LINE=(member,...,member);
```

`label` (see §6.2) gives a name to the beam line for later reference.

Each `member` may be one of the following:

- An element label,
- A beam line label,
- A sub-line, enclosed in parentheses,

Beam lines can be nested to any level.

9.1 Simple Beam Lines

The simplest beam line consists of single elements:

```
label:LINE=(member,...,member);
```

Example:

```
L:LINE=(A,B,C,D,A,D);
```

9.2 Sub-lines

Instead of referring to an element, a beam line member can refer to another beam line defined in a separate command. This provides a shorthand notation for sub-lines which occur several times in a beam line. Lines and sub-lines can be entered in any order, but when a line is used, all its sub-lines must be known.

Example:

```
L:LINE=(A,B,S,B,A,S,A,B);  
S:LINE=(C,D,E);
```

This example produces the following expansion steps:

1. Replace sub-line S:

$$(A, B, (C, D, E), B, A, (C, D, E), A, B)$$

2. Omit parentheses:

$$\bar{A}, B, C, D, E, B, \bar{A}, C, D, E, \bar{A}, B$$

valuated to constants immediately.

Chapter 10

Physics Commands

10.1 BEAM Command

All OPAL commands working on a beam require the setting of various quantities related to this beam. These are entered by a BEAM command:

```
label:BEAM, PARTICLE=name, MASS=real, CHARGE=real,  
        ENERGY=real, PC=real, GAMMA=real, BCURRENT=real,  
        EX=real, EXN=real, EY=real, EYN=real, ET=real,  
        KBUNCH=integer, NPART=real, BUNCHED=logical,  
        RADIATE=logical, DAMP=logical, QUANTUM=logical;
```

The label is optional, it defaults to UNNAMED_BEAM. . The particle mass and charge are defined by:

PARTICLE The name of particles in the machine.

OPAL knows the mass and the charge for the following particles:

POSITRON The particles are positrons ($MASS=m_e$, $CHARGE=1$),

ELECTRON The particles are electrons ($MASS=m_e$, $CHARGE=-1$),

PROTON The particles are protons (default, $MASS=m_p$, $CHARGE=1$),

ANTIPROTON The particles are anti-protons ($MASS=m_p$, $CHARGE=-1$).

HMINUS The particles are h- protons ($MASS=m_{h^-}$, $CHARGE=-1$).

CARBON The particles are carbons ($MASS=m_c$, $CHARGE=12$).

URANIUM The particles are of type uranium ($MASS=m_u$, $CHARGE=35$).

MUON The particles are of type muon ($MASS=m_\mu$, $CHARGE=-1$).

DEUTERON The particles are of type deuteron ($MASS=m_d$, $CHARGE=1$).

XENON The particles are of type xenon ($MASS=m_{xe}$, $CHARGE=20$).

For other particle names one may enter:

MASS The particle mass in GeV.

CHARGE The particle charge expressed in elementary charges.

Chapter 11

Distribution Command

[TODO: AA will rewrite]

Particle distributions can be read in or generated by specifying rms beam quantities. The allowed parameters are described in Table 11.3.

Particle distributions are generated separately in all three phase space planes. There are limited correlations between planes e.g. between longitudinal and transverse ($r51, r52, r61, r62$). Besides an efficient parallel Gaussian distribution generator based on a parallelized “Method of Rejection”, a more general algorithm for generating distributions is available [42]. The shape of the binomial distribution is governed by one parameter m . By varying this single parameter one obtains the most commonly used distributions for our type of simulations as, listed in Table 11.1.

If NBIN is larger than one the distribution is binned in energy and for each bin a separate field solve is performed when using the electrostatic solvers.

Table 11.1: Different distributions specified by a single parameter m

| m | Distribution | Density | Profile |
|----------------------|--------------|---|--|
| 0.0 | Hollow shell | $\frac{1}{\pi} \delta(1 - r^2)$ | $\frac{1}{\pi} (1 - r^2)^{-0.5}$ |
| 0.5 | Flat profile | $\frac{1}{2\pi} (1 - r^2)^{-0.5}$ | $\frac{1}{2}$ |
| 1.0 | Uniform | $\frac{1}{\pi}$ | $\frac{2}{\pi} (1 - x^2)^{0.5}$ |
| 1.5 | Elliptical | $\frac{3}{2\pi} (1 - r^2)^{0.5}$ | $\frac{1}{4} (1 - x^2)$ |
| 2.0 | Parabolic | $\frac{2}{\pi} (1 - r^2)$ | $\frac{3}{8\pi} (1 - x^2)^{1.5}$ |
| $\rightarrow \infty$ | Gaussian | $\frac{1}{2\pi\sigma_x\sigma_y} \exp(-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2})$ | $\frac{1}{\sqrt{2\pi}\sigma_x} \exp(-\frac{x^2}{2\sigma_x^2})$ |

There are also special distribution commands.

1. *GUNGAUSSFLATTOPTH* will create a distribution that is uniform transversely. The longitudinal profile has a Gaussian rise and fall with a flat top distribution in between. More details are given in Section 11.3. This distribution has thermal emittance as defined at 11.2.
2. *ASTRAFLATTOPTH* is the same like *GUNGAUSSFLATTOPTH* except it uses a low noise Hammersley generator in the longitudinal direction.

The following example reads in a distribution from a file and scales the coordinates:

Table 11.2: Parameters for the DISTRIBUTION command

| Parameter | Purpose |
|--|---------|
| DISTRIBUTION FROMFILE or GAUSS or BINOMINAL GUNGAUSSFLATTOPTH or ASTRAFLATTOPTH FNAME Specifies the filename of a particle distribution to be read in XMULT Scales the x coordinate: $x = XMULT * x$ PXMULT Scales the px coordinate: $px = PXMULT * px$ YMULT Scales the y coordinate: $y = YMULT * y$ PYMULT Scales the py coordinate: $py = PYMULT * py$ TMULT Scales the t coordinate: $t = TMULT * t$ PTMULT Scales the pt coordinate: $pt = PTMULT * pt$ | |
| SIGMAX \tilde{x} see Chapter on Notation SIGMAPX \tilde{p}_x see Chapter on Notation SIGMAY \tilde{y} see Chapter on Notation SIGMAPY \tilde{p}_y see Chapter on Notation SIGMAT \tilde{t} see Chapter on Notation TRANSVCUTOFF Defines the transverse cut-off of GUNGAUSS3D in units of σ PT $\langle p_t \rangle$ see Chapter on Notation SIGMAPT \tilde{p}_t see Chapter on Notation | |
| mx Defines the transverse distribution (see Table 11.1) my Defines the transverse distribution (see Table 11.1) mt Defines the longitudinal distribution (see Table 11.1) | |
| CORRX Defines the x, p_x correlation CORRY Defines the y, p_y correlation CORRT Defines the t, p_t correlation | |

```

DistFile:DISTRIBUTION, DISTRIBUTION=FROMFILE,
  FNAME="../Dist/inpdist1finitecur.dat",
  XMULT=0.06816207, YMULT=0.06816207,
  TMULT=1.0*beta*0.06816207,
  PXMULT=1/gambet, PYMULT=1/gambet,
  PTMULT=1.0/beta^2/gamma;

```

The file with the data has to have the following format:

Table 11.3: Parameters of the distribution command

| Parameter | Purpose |
|--|---------|
| TEMISSION Defines the length of the emission process [s] | |
| NBIN How many energy bins begin used. For accurate results this should be around 10 for an RF photoinjector and around 40 for a DC photoinjector. | |
| SBIN How many samples per energy bin to use when constructing the time histogram of the distribution. The default value is 100. | |
| DEBIN Defines a energy band dE [MeV]. If the maximal energy difference between all bins are smaller than dE all bins are merged into one bin. | |
| ELASER Laser energy (eV) | |
| SIGLASER Sigma of (uniform) laser spot size (m) | |
| W Workfunction of material (eV) | |
| FE Fermi energy (eV) | |
| AG Acceleration Gradient (MV/m) | |

 N
 $x_1 \ px_1 \ y_1 \ py_1 \ z_1 \ pz_1$
 $x_2 \ px_2 \ y_2 \ py_2 \ z_2 \ pz_2$
 \cdot
 \cdot
 $x_N \ px_N \ y_N \ py_N \ z_N \ pz_N,$

where N is the number of particles, the vector (x_i, y_i, z_i) describes the position of the i -th particle and the vector (px_i, py_i, pz_i) its momentum in as defined in section 4.2 and section 5.3.

11.1 Correlations for Gaussian Distribution (Experimental)

To generate gaussian initial distribution with dispersion, first we generate the uncorrelated gaussian inputs matrix $R = (R_1, \dots, R_n)$. The mean of R_i is 0 and the standard deviation squared is 1. Then we correlate R . The correlation coefficient matrix σ in x, p_x, t, p_t phase space reads:

$$\begin{matrix} & x & px & t & pt \\ \begin{matrix} x \\ px \\ t \\ pt \end{matrix} & & & & \end{matrix}$$

$$\sigma = \begin{bmatrix} 1 & c_x & r51 & r61 \\ c_x & 1 & r52 & r62 \\ r51 & r52 & 1 & c_t \\ r61 & r62 & c_t & 1 \end{bmatrix}.$$

The Cholesky decomposition of the symmetric positive-definite matrix σ is $\sigma = C^T C$, then the correlated distribution is $C^T R$.

Note: This correlation works for the moment only with the gaussian distribution.

11.1.1 Example

Let the initial correlation coefficient matrix be:

$$\sigma = \begin{bmatrix} 1 & 0.756 & 0.023 & 0.496 \\ 0.756 & 1 & 0.385 & -0.042 \\ 0.023 & 0.385 & 1 & -0.834 \\ 0.496 & -0.042 & -0.834 & 1 \end{bmatrix}$$

then the corresponding distribution command read:

```
Dist:DISTRIBUTION, DISTRIBUTION=gauss,
  sigmax=4.796e-03, sigmapx=231.0585, corrx=0.756,
  sigmay=23.821e-03, sigmapy=1.6592e+03, corry=-0.999,
  t=0.466e-02, sigmat=0.466e-02, pt=72e6,
  sigmapt=74.7, corrt=-0.834,
  r61=0.496, r62=-0.042, r51=0.023, r52=0.385;
```

11.2 Thermal Emittance

The thermal emittance calculation is based on [49, 50] where $P(E_f, E_{ph} = \hbar\omega)$ the probability for a photon of energy E_{ph} exiting an electron to a final state energy E_f is

$$P(E_f, E_{ph} = \hbar\omega) \propto N_f(E_f) N_i(E_f - E_{ph} = \hbar\omega) \text{ with} \quad (11.1)$$

$N_f(E_f)$ is the density of final state and $N_i(E_f - E_{ph})$ is the density of initial state.

Two cases, no-scattering (non-equilibrium) and scattering (equilibrium, e-e and e-phonon collisions) can be distinguished. In OPAL the non-equilibrium case is considered and a uniform radial distribution is assumed hence: $x_{rms} = \frac{r}{2}$ ¹.

Photoemission from a metal involves first the absorption of a photon with:

$$\hbar\omega > \Phi_e \quad (11.2)$$

where $\Phi_e = \Phi - \Delta$ is the reduced work function. The reduction is a function of the applied electric field E_c :

$$\Delta = e\sqrt{eE_c/4\pi\epsilon_0}. \quad (11.3)$$

¹ Soon we can generate distributions from virtual cathode images

Electrons are emitted isotropic into the half-sphere with: $E_{kin} = \varepsilon_f + \hbar\omega$.

Particles with angel φ larger than $\varphi_{max} = \arccos \sqrt{(\varepsilon_f + \Phi_e/E_{kin})}$ will pass the potential barrier.

$$p_x = p \sin \varphi \cos \theta, \quad \varphi = [0 \dots \varphi_{max}], \quad \theta = [0 \dots \pi] \quad (11.4)$$

and

$$p = m_0 c \sqrt{\gamma^2 - 1}. \quad (11.5)$$

The following parameters defines the thermal emittance: r_{rms} , material such as Cu, Fe, Cs2Te $\rightarrow \Phi, \varepsilon_f$, the laser energy given by $\hbar\omega$ and the electric field E_c which enters in the Schottky effect calculation.

This is a example of an OPALdistribution definition with thermal emittance similar to the example in [50] p.199.

```
Dist1:DISTRIBUTION, DISTRIBUTION = GUNGAUSSFLATTOPH,
    sigmax=0.00054, sigmapx=0.0, corrx=0.0,
    sigmay=0.00054, sigmapy=0.0, corry=0.0,
    sigmat=0.0, pt=0.0, sigmapt=0.0, corrt=0.0,
    tRise=0.5e-12, tFall=0.5e-12, tPulseFWHM=9.9e-12,
    cutoff=3.0,
    NBIN=50, DEBIN=80,
    ELASER=4.6, SIGLASER=0.001, W=4.6, FE=7.0, AG=84;
```

11.3 Flattop Distribution

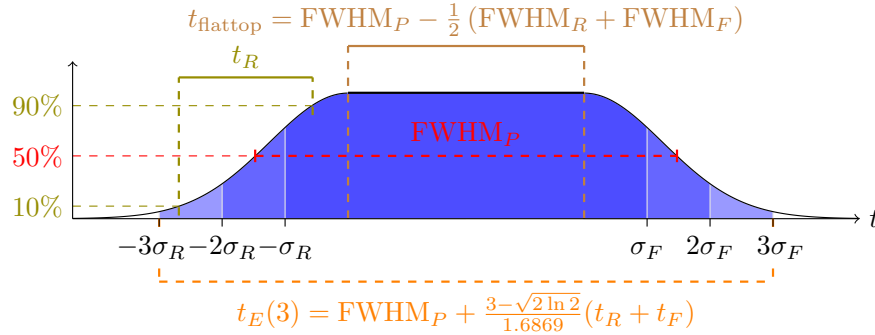


Figure 11.1: OPAL Gauss-Flattop-Distribution

Figure 11.1 depicts parameters needed to specify a Gauss-Flattop distribution in OPAL where c stands for cutoff, P for pulse, R for rise and F for fall. A Gauss-Flattop distribution is defined as half a Gauss plus a uniform (flat-top) part plus half a Gauss with the parameters σ_R , σ_F and t_{flatop} . For practical reasons we replace these three parameters with the following measurable parameters

$$t_R = \left(\sqrt{2 \ln(10)} - \sqrt{2 \ln\left(\frac{10}{9}\right)} \right) \sigma_R = 1.6869 \sigma_R, \quad t_F = 1.6869 \sigma_F \quad \text{and}$$

$$\text{FWHM}_P = t_{flatop} + \sqrt{2 \ln 2} (\sigma_R + \sigma_L).$$

The emission time (t_E) with a given cutoff c in terms of input parameters is given by

$$\begin{aligned} t_E(c) &= \text{FWHM}_P - \frac{1}{2}\text{FWHM}_R - \frac{1}{2}\text{FWHM}_F + c\sigma_R + c\sigma_F \\ &= \text{FWHM}_P + \frac{c - \sqrt{2\ln 2}}{1.6869}(t_R + t_F). \end{aligned}$$

In OPAL this distribution can be specified by the following distribution command (thermal emittance enabled)

```
Dist1:DISTRIBUTION, DISTRIBUTION = "GUNGAUSSFLATTOPTH",
  sigmax = 0.000275 * 2., sigmapx = 0.0, corrx = 0.0,
  sigmay = 0.000275 * 2., sigmapy = 0.0, corry = 0.0,
  sigmat = 0.0, pt = 0.0, sigmapt = 0.0, corrt = 0.0,
  tRise = 0.7e-12, tFall = 0.7e-12, tPulseFWHM = 9.9e-12,
  ekin = 0.63, NBIN = 50, DEBIN = 80;
```

where t_{Rise} (t_R , olive in Figure 1), t_{Fall} and $t_{\text{PulseFWHM}}$ (FWHM_P , red in Figure 1) in seconds [s] define the shape of the Gauss-FlatTop distribution. The default cutoff is 3.0. This can be changed by adding e.g. `cutoff = 4.0` to the distribution command.

11.3.1 Legacy Mode

We provide a legacy mode for flatTop distribution. The following old input specification

```
SRise = 0.7e-12;
SFall = 0.7e-12;
CRise = 3.0;
CFall = 3.0;
TFlatTop = 9.9e-12*0.925;
TEmis = TFlatTop + CRise* SRise + CFall* SFall;

value,{TFlatTop,TEmis};

Dist1:DISTRIBUTION, DISTRIBUTION = "GUNGAUSSFLATTOPTH",
  sigmax = 0.000270*2.0, sigmapx = 0.0, corrx = 0.0,
  sigmay = 0.000270*2.0, sigmapy = 0.0, corry = 0.0,
  sigmat = 0.0, pt = 0.0, sigmapt = 0.0, corrt = 0.0, ekin = 0.4,
  sigmarise = 0.7e-12, sigmafall = 0.7e-12, flattoptime = 9.9e-12*0.925,
  cutoffrise = CRise, cutofffall = CFall,
  TEMISSION = TEmis, NBIN = 10, DEBIN = 80;
```

can be replaced by

```
st1:DISTRIBUTION, DISTRIBUTION = "GUNGAUSSFLATTOPTH",
  sigmax = 0.000275 * 2., sigmapx = 0.0, corrx = 0.0,
  sigmay = 0.000275 * 2., sigmapy = 0.0, corry = 0.0,
  sigmat = 0.0, pt = 0.0, sigmapt = 0.0, corrt = 0.0, ekin = 0.4,
  tRise = 0.7e-12, tFall = 0.7e-12, tPulseFWHM = 9.9e-12*0.925,
  NBIN = 10, DEBIN = 80, LEGACYMODE=TRUE;
```

and should produce the same results. Make sure to enable the `LEGACYMODE` option to create the flatTop distribution in legacy mode!

Chapter 12

Fieldsolver

[TODO: AA will rewrite]

12.1 Fieldsolver Command

See Table 12.1 for a summary of the Fieldsolver command.

12.2 Define the Fieldsolver to be used

At present only a FFT based solver is available. Future solvers will include Finite Element solvers and a Multi Grid solver with Shortley-Weller boundary conditions for irregular domains.

12.3 Define Domain Decomposition

The dimensions in x , y and z can be parallel (TRUE) or serial FALSE. The default settings are: parallel in z and serial in x and y .

12.4 Define Number of Gridpoints

Number of grid points in x , y and z for a rectangular grid.

12.5 Define Boundary Conditions

Two boundary conditions can be selected independently among x , y namely: OPEN and for z OPEN & PERIODIC. In the case you select for z periodic you are about to model a DC-beam.

12.6 Define Greens Function

Two Greens functions can be selected: INTEGRGREEN, GREEN. The integrated Green's function is described in [40].

Table 12.1: Fieldsolver command summary

| Command | Purpose |
|--------------------------|--|
| <code>FIELDSOLVER</code> | Specify a fieldsolver |
| <code>FSTYPE</code> | Specify the type of field solver |
| <code>PARFFTX</code> | If TRUE, the dimension x is distributed among the processors |
| <code>PARFFTY</code> | If TRUE, the dimension y is distributed among the processors |
| <code>PARFFTZ</code> | If TRUE, the dimension z is distributed among the processors |
| <code>MX</code> | Number of grid points in x specifying rectangular grid |
| <code>MY</code> | Number of grid points in y specifying rectangular grid |
| <code>MZ</code> | Number of grid points in z specifying rectangular grid |
| <code>BCFFTX</code> | Boundary condition in x [OPEN] |
| <code>BCFFTY</code> | Boundary condition in y [OPEN] |
| <code>BCFFTZ</code> | Boundary condition in z [OPEN,PERIODIC] |
| <code>GREENSF</code> | Defines the Greens function for the FFT Solver |
| <code>BBOXINCR</code> | Enlargement of the bounding box in % |
| <code>GEOMETRY</code> | Geometry to be used as domain boundary |
| <code>ITSOLVER</code> | Type of iterative solver |
| <code>INTERPL</code> | Interpolation used for boundary points |
| <code>TOL</code> | Tolerance for iterative solver |
| <code>MAXITERS</code> | Maximum number of iterations of iterative solver |
| <code>PRECMODE</code> | Behaviour of the preconditioner |

12.7 Define Bounding Box Enlargement

The bounding box defines a minimal rectangular domain including all particles. With `BBOXINCR` the bounding box can be enlarged by a factor given in percent of the minimal rectangular domain.

12.8 Define Geometry

The list of geometries defining the beam line boundary. For further details see Chapter 14.

12.9 Define Iterative Solver

The iterative solver for solving the preconditioned system: CG, BiCGSTAB or GMRES.

12.10 Define Interpolation for Boundary Points

The interpolation method for gridpoints near the boundary: CONSTANT, LINEAR or QUADRATIC.

12.11 Define Tolerance

The tolerance for the iterative solver used by the MG solver.

12.12 Define Maximal Iterations

The maximal number of iterations the iterative solver performs.

12.13 Define Preconditioner Behaviour

The behaviour of the preconditioner can be: STD, HIERARCHY or REUSE. This argument is only relevant when using the MG solver and should **only be set if the consequences to simulation and solver are evident**. A short description is given in Table 12.2.

Table 12.2: Preconditioner behaviour command summary

| Value | Behaviour |
|-----------|--|
| STD | The preconditioner is rebuilt in every timestep (enabled by default) |
| HIERARCHY | The hierarchy (tentative prolongator) is reused |
| REUSE | The preconditioner is reused |

12.14 Define the number of Energy Bins to use

Suppose dE the energy spread in the particle bunch is too large, the electrostatic approximation is no longer valid. One solution to that problem is to introduce k energy bins and perform k separate field solves in which dE is again small and hence the electrostatic approximation valid. In case of a cyclotron (Section 8.10) the number of energy bins must be at minimum the number of neighbouring bunches (NNEIGHBB) i.e. $ENBINS \leq NNEIGHBB$

Chapter 13

Wakefields

Basically there are two different kind of wakefields that can be used. The first one is the wakefield of a round, metallic beam pipe that can be calculated numerically (see Sections 13.2 - 13.11). Since this also limits the applications of wakefields we also provide a way to import a discretized wakefield from a file (see Section 13.12).

The wakefield of a round, metallic beam pipe with radius a can be calculated by inverse FFT of the beam pipe impedance. There are known models for beam pipes with DC and AC conductivity. The DC conductivity of a metal is given by

$$\sigma_{DC} = \frac{ne^2\tau}{m} \quad (13.1)$$

with n the density of conduction electrons with charge e , τ the relaxation time, and m the electron mass. The AC conductivity, a response to applied oscillation fields, is given by

$$\sigma_{AC} = \frac{\sigma_{DC}}{1 - i\omega\tau} \quad (13.2)$$

with ω denoting the frequency of the fields.

The longitudinal impedance with DC conductivity is given by

$$Z_{Ldc}(k) = \frac{1}{ca} \frac{2}{\frac{\lambda}{k} - \frac{ika}{2}} \quad (13.3)$$

where

$$\lambda = \sqrt{\frac{2\pi\sigma|k|}{c}}(i + \text{sign}(k)) \quad (13.4)$$

with c denoting the speed of light and k the wave number.

The longitudinal wake can be obtained by an inverse Fourier transformation of the impedance. Since $\text{Re}(Z_L(k))$ drops at high frequencies faster than $\text{Im}(Z_L(k))$ the cosine transformation can be used to calculate the wake. The following equation holds in both, the DC and AC, case

$$W_L(s) = 10^{-12} \frac{2c}{\pi} \text{Re} \left(\int_0^\infty \text{Re}(Z_L(k)) \cos(ks) dk \right) \quad (13.5)$$

with $Z_L(k)$ either representing $Z_{LDC}(k)$ or $Z_{LAC}(k)$ depending on the conductivity. With help of the Panofsky-Wenzel theorem

$$Z_L(k) = \frac{k}{c} Z_T(k). \quad (13.6)$$

we can deduce the transversal wakefield from (13.5):

$$W_T(s) = 10^{-12} \frac{2c}{\pi} \text{Re} \left(\int_0^\infty \text{Re}\left(\frac{c}{k} Z_L(k)\right) \cos(ks) dk \right). \quad (13.7)$$

To calculate the integrals in (13.5) and (13.7) numerically the Simpson integration schema with equidistant mesh spacings is applied. This leads to an integration with small Δk with a big N which is computational not optimal with respect to efficiency. Since we calculate the wakefield usually just once in the initialization phase the overall performance will not be affected from this.

13.1 Wakefield Command

See Table 13.1 for a summary of the Wakefield command.

NOTE: Currently when using wakefields the domain can only be parallelized in z -direction! Please adapt the parallelization in the FieldSolver accordingly

```

Fs1:FIELDSOLVER, FSTYPE=FFT,
    MX=32, MY=32, MT=32,
    PARFFTX=false, PARFFTY=false, PARFFT=true,
    BCFFTX=open, BCFFTY=open, BCFFT=open,
    BBOXINCR=0, GREENSF=INTEGRATED;

```

Table 13.1: Wakefield command summary

| Command | Purpose |
|--------------|---|
| WAKE | Specify a wakefield |
| TYPE | Specify the wake function [1D-CSR, LONG-SHORT-RANGE, TRANSV-SHORT-RANGE, LONG-TRANSV-SHORT-RANGE] |
| NBIN | Number of bins used in the calculation of the line density |
| CONST_LENGTH | TRUE if the length of the bunch is considered to be constant |
| CONDUCT | Conductivity [AC, DC] |
| Z0 | Impedance of the beam pipe in [Ω] |
| FORM | The form of the beam pipe [ROUND] |
| RADIUS | The radius of the beam pipe in [m] |
| SIGMA | Material constant dependent on the beam pipe material in [$\Omega^{-1}m$] |
| TAU | Material constant dependent on the beam pipe material in [s] |
| FNAME | Specify a file that provides a wakefunction |

13.2 Define the Wakefield to be used

The WAKE statement defines data for a wakefunction on an element.

13.3 Define the wakefield type

Used to specify the wakefunction

13.4 Define the number of bins

The number of bins used in the calculation of the line density.

13.5 Define the bunch length to be constant

With the `CONST_LENGTH` flag the bunch length can be set to be constant.

13.6 Define the conductivity

The conductivity of the bunch which can be set to either AC or DC.

13.7 Define the impedance

The impedance Z_0 of the beam pipe in $[\Omega]$.

13.8 Define the form of the beam pipe

The form of the beam pipe can be set to `ROUND`.

13.9 Define the radius of the beam pipe

The radius of the beam pipe in $[m]$.

13.10 Define the σ of the beam pipe

The σ of the beam pipe (material constant), see (13.1).

13.11 Define the relaxation time (τ) of the beam pipe

The τ defines the relaxation time and is needed to calculate the impedance of the beam pipe (see 13.1).

13.12 Import a wakefield from a file

Since we only need values of the wakefunction at several discrete points to calculate the force on the particle it is also possible to specify these in a file. To get required datapoints of the wakefield not provided in the file we linearly interpolate the available function values. The files are specified in the SDDS¹ (Self Describing Data Sets).

Whenever a file is specified OPAL will use the wakefield found in the file and ignore all other commands related to round beam pipes.

¹http://www.aps.anl.gov/Accelerator_Systems_Division/Operations_Analysis/manuals/GettingStartedWithSDDS/HTML/GettingStartedWithSDDS.html

13.13 Wake Functions

Three types of wake functions are implemented so far: transverse and longitudinal geometric wakes and the CSR wake. The general input format is

```
label:WAKE, TYPE=string, NBIN=real, CONST_LENGTH=bool,
      CONDUCT=string, Z0=real, FORM=string, RADIUS=real,
      SIGMA=real, TAU=real, FILTERS=string-array
```

CONST_LENGTH, **CONDUCT**, **Z0**, **FORM**, **RADIUS**, **SIGMA** and **TAU** are only used in the geometric wakes.

TYPE The type of wake function; either TRANSV-SHORT-RANGE, LONG-SHORT-RANGE or 1D-CSR.

NBIN Not implemented yet; Number of bins to be used for line density if different from space charge solver grid.

CONST_LENGTH

CONDUCT

Z0

FORM

RADIUS

SIGMA

TAU

FILTERS Array of names of filters to be applied onto the longitudinal histogram of the bunch to get rid of the noise and to calculate derivatives. All the filters/smothers are applied to the line density in the order they appear in the array. The last filter is also used for calculating the derivatives. The actual filters have to be defined elsewhere.

13.14 Filters

Filters can be defined which then are applied to the line density of the bunch. The following smoothing filters are implemented: Savitzky-Golay, Stencil, FixedFFTLowPass, RelativFFTLowPass. The input format for them is

```
label:FILTER, TYPE=string, NFREQ=real, THRESHOLD=real,
      NPOINTS=real, NLEFT=real, NRIGHT=real,
      POLYORDER=real
```

TYPE The type of filter: Savitzky-Golay, Stencil, FixedFFTLowPass, RelativFFTLowPass

NFREQ Only used in FixedFFTLowPass: the number of frequencies to keep

THRESHOLD Only used in RelativeFFTLowPass: the minimal strength of frequency compared to the stronges to consider.

NPOINTS Only used in Savitzky-Golay: width of moving window in number of points

NLEFT Only used in Savitzky-Golay: number of points to the left

NRIGHT Only used in Savitzky-Golay: number of points to the right

POLYORDER Only used in Savitzky-Golay: polynomial order to be used in least square approximation

The Savitzky-Golay filter and the ones based on the FFT routine provide a derivative on a natural way. For the Stencil filter a second order stencil is used to calculate the derivative.

An implementation of the Savitzky-Golay filter can be found in the Numerical Recipes. The Stencil filter uses the following two stencil consecutively to smooth the line density:

$$f_i = \frac{7 \cdot f_{i-4} + 24 \cdot f_{i-2} + 34 \cdot f_i + 24 \cdot f_{i+2} + 7 \cdot f_{i+4}}{96}$$

and

$$f_i = \frac{7 \cdot f_{i-2} + 24 \cdot f_{i-1} + 34 \cdot f_i + 24 \cdot f_{i+1} + 7 \cdot f_{i+2}}{96}.$$

For the derivative a standard second order stencil is used:

$$f'_i = \frac{f_{i-2} - 8 \cdot f_{i-1} + 8 \cdot f_{i+1} - f_{i+2}}{h}$$

This filter was designed by Ilya Pogorelov for the ImpactT implementation of the CSR 1D model.

The FFT based smoothers calculate the Fourier coefficients of the line density. Then they set all coefficients corresponding to frequencies above a certain threshold to zero. Finally the back-transformation is calculate using this coefficients. The two filters differ in the way they identify coefficients which should be set to zero. FixedFFT-LowPass uses the n lowest frequencies whereas RelativeFFTLowPass searches for the coefficient which has the biggest absolut value. All coefficients which, compared to this value, are below a threshold (measure in percents) are set to zero. For the derivative the coefficients are multiplied with the following function (this is equivalent to a convolution):

$$g_i = \begin{cases} i \frac{2\pi i}{N \cdot L} & i < N/2 \\ -i \frac{2\pi i}{N \cdot L} & i > N/2 \end{cases}$$

where N is the total number of coefficients/sampling points and L is the length of the bunch.

Chapter 14

Geometry

At present the GEOMETRY command is still an **experimental feature** which is not to be used by the general user. It can only be used to specify boundaries for the MG Solver. The command can be used in two modes:

1. specify a H5FED file holding the surface mesh of a complicated boundary geometry
2. specify a cylinder with an elliptic base area

14.1 Geometry Command

Table 14.1: Geometry command summary

| Command | Purpose |
|-----------------------------|---|
| GEOMETRY Specify a geometry | |
| FGEOM | Specifies the H5FED geometry file |
| LENGTH | Specifies the length of the geometry |
| S | Specifies the start of the geometry |
| A | Specifies the semi-major axis of the elliptic base area |
| B | Specifies the semi-minor axis of the elliptic base area |

14.2 Define the Geometry File

The H5FED file containing the surface mesh of the geometry.

14.3 Define the Length

The length of the specified geometry in [m].

14.4 Define the Start

The start of the specified geometry in [m].

14.5 Define the Semi-Major Axis

The semi-major axis of the ellipse in [m].

14.6 Define the Semi-Minor Axis

The semi-minor axis of the ellipse in [m].

Chapter 15

Tracking

Table 15.1: Commands accepted in Tracking Mode

| Command | Purpose |
|-----------------|--|
| TRACK | Enter tracking mode |
| LINE | Label of LINE or SEQUENCE |
| BEAM | Label of BEAM |
| DT | Initial time step for tracking |
| MAXSTEPS | The maximal number of time steps |
| ZSTOP | Defines a z-location [m], after which the simulation stops when $SPOS > ZSTOP$ |
| STEPSPERTURN | The timesteps per revolution period |
| TIMEINTEGRATOR | Defines the time integrator used in OPAL-CYCL |
| name=expression | Parameter relation |
| START | Define initial conditions |
| RUN | Run particles for specified number of turns or steps |
| TSAVE | Save end conditions |
| ENDTRACK | Leave tracking mode |

15.1 Track Mode

Before starting to track, a beam line (see §9) or sequence (see §??) and a beam (see §10.1) must be selected. The time step (DT) and the maximal steps to track (MAXSTEPS) or ZSTOP should be set. This command causes OPAL to enter “tracking mode”, in which it accepts only the track commands (see Tab. 15.1). Several tracks can be defined in a sequence and all parameters are always local to the actual step.

The attributes of the command are:

LINE The label of a preceding **LINE** (see §9) or **SEQUENCE** (see §??) (no default).

BEAM The named **BEAM** command defines the particle mass, charge and reference momentum (default: `UNNAMED_BEAM`).

MAXSTEPS The maximal number of timesteps, default value is 10.

ZSTOP Defines a z-location [m], after which the simulation stops when $SPOS > ZSTOP$. The initial value for **ZSTOP** is $1E6$ [m].

TIMEINTEGRATOR Define the time integrator. Currently only available in **OPAL-CYCL**. The valid options are **RK-4**, **LF-2** and **MTS**:

- **RK-4**: the fourth-order Runge-Kutta integrator. This is the default integrator for **OPAL-CYCL**.
- **LF-2**: the second-order Boris-Buneman (leapfrog-like) integrator. Currently, **LF-2** is only available for multi-particles with/without space charge. For single particle tracking and tune calculations, use the **RK-4** for the time being.
- **MTS**: the multiple-time-stepping integrator. Considering that the space charge fields change much slower than the external fields in cyclotrons, the space charge can be calculated less frequently than the external field interpolation, so as to reduce time to solution. The outer step (determined by **STEPSPERTURN**) is used to integrate space charge effects. A constant number of substeps per outer step is used to query external fields and to move the particles. The number of substeps can be set with the option **MTSSUBSTEPS** and its default value is 1. When using this integrator, the input file has to be rewritten in the units of the outer step. For example, extracts of the inputfile suited for **LF-2** or **RK-4** read

```
Option, PSDUMPFREQ=100;
Option, REPARTFREQ=20;
Option, SPTDUMPFREQ=50;
turns=5;
nstep=3000;
TRACK, LINE=11, BEAM=beam1, MAXSTEPS=nstep*turns, STEPSPERTURN=nstep,
TIMEINTEGRATOR="LF-2";
    RUN, METHOD = "CYCLOTRON-T", BEAM=beam1, FIELDSOLVER=Fsl, DISTRIBUTION=Dist1;
ENDTRACK;
```

and should be transformed to

```
Option, MTSSUBSTEPS=10;
Option, PSDUMPFREQ=10;
Option, REPARTFREQ=2;
Option, SPTDUMPFREQ=5;
turns=5;
nstep=300;
TRACK, LINE=11, BEAM=beam1, MAXSTEPS=nstep*turns, STEPSPERTURN=nstep,
TIMEINTEGRATOR="MTS";
    RUN, METHOD = "CYCLOTRON-T", BEAM=beam1, FIELDSOLVER=Fsl, DISTRIBUTION=Dist1;
ENDTRACK;
```

In general all step quantities should be divided by **MTSSUBSTEPS**.

In our first experiments on PSI injector II cyclotron, simulations with reduced space charge solving frequency by a factor of 10 lie still very close to the original solution. How large **MTSSUBSTEPS** can be chosen of course depends on the importance of space charge effects.

DT Initial time step for tracking, default value is 1 ps.

STEPSPERTURN The timesteps per revolution period. Only available for **OPAL-CYCL**, default value is 720.

In OPAL-Tand OPAL-MAP, the command format is:

```
TRACK, LINE=name, BEAM=name, MAXSTEPS=value, DT=value;
```

In OPAL-CYCL, instead of setting time step, the timesteps per-turn should be set. The command format is:

```
TRACK, LINE=name, BEAM=name, MAXSTEPS=value, STEPSPERTURN=value;
```

Particles are tracked in parallel i.e. the coordinates of all particles are transformed at each beam element as it is reached.

OPAL leaves **track mode** when it sees the command

```
ENDTRACK;
```

15.1.1 Track a Random Machine

This example shows how to track a *random* machine i.e. some parameters are random variables. At the moment (Version 1.1.4) there seems to be a problem when having random variables in the Distribution command.

```
Option, SCAN=TRUE;

.....

I=0;
WHILE (I < 3) {

    rv1:= (RANF()*4.7);
    rv2:=0.0;
    rv3:=0.0;
    rv4:=0.0;
    rv5:=0.0;

    Ppo: PepperPot, L=200.0E-6, ELEMEDGE=6.0E-3,
        R=1.0E-4, PITCH=0.5E-4, NHOLX=20, NHOLY=20,
        XSIZE=5.0E-3, YSIZE=5.0E-3, OUTFN="ppo.h5";

    Col: ECollimator, L=3.0E-3, ELEMEDGE=7.0E-3,
        XSIZE=7.5E-4, YSIZE=7.5E-4, OUTFN="Coll.h5";
    SP1: Solenoid, L=1.20, ELEMEDGE=-0.5315,
        FMAPFN="1T2.T7", KS=8.246e-05 + rv2;
    SP2: Solenoid, L=1.20, ELEMEDGE=-0.397,
        FMAPFN="1T3.T7", KS=1.615e-05 + rv3;
    SP3: Solenoid, L=1.20, ELEMEDGE=-0.267,
        FMAPFN="1T3.T7", KS=1.016e-05 + rv4;
    SP4: Solenoid, L=1.20, ELEMEDGE=-0.157,
        FMAPFN="1T3.T7", KS=4.750e-05 + rv5;
    SP5: Solenoid, L=1.20, ELEMEDGE=-0.047,
        FMAPFN="1T3.T7", KS=0.0;

    gun: RFCavity, L=0.013, VOLT=(-47.51437343 + rv1),
        FMAPFN="1T1.T7", ELEMEDGE=0.00,
        TYPE="STANDING", FREQ=1.0e-6;

    value,{I, rv1, rv2, rv3, rv4, rv5};

    l1: Line=(gun, Ppo, sp1, sp2, sp3, sp4, sp5);

    SELECT, Line=l1;
```

```
TRACK, line=l1, beam=beam1, MAXSTEPS=500, DT=2.0e-13;
  RUN, method="PARALLEL-T", beam=beam1,
    fieldsolver=Fsl, distribution:=Dist1;
ENDTRACK;

SYSTEM, "mkdir -p scan0-" & STRING(I);
SYSTEM, "mv scan-0.h5 scan-0.stat scan-0.lbal scan0-"
  & STRING(I);
I=EVAL(I+1.0);
}
```

Chapter 16

Field Emission

Field emission is a major source of both dark current particles and primary incident particles in secondary emission. The Fowler-Nordheim (F-N) formula we use here to predict the emitted current density is given in (16.1) [52] [53]

$$J(\mathbf{r}, t) = \frac{A(\beta E)^2}{\varphi t(y)^2} \exp\left(\frac{-Bv(y)\varphi^{3/2}}{\beta E}\right) [\text{A/m}^2] \quad (16.1)$$

where $J(\mathbf{r}, t)$ stands for emitted electric current density in position \mathbf{r} and time t . The Greek letters φ and β denote the work function of the surface material and the local field enhancement factor respectively. The parameter E is the electric field in the normal direction of surface. The parameters A and B are empirical constants. The functions $v(y)$ and $t(y)$ representing the image charge effects [52] as a function of the Fowler-Nordheim parameter y with the following definition[54]

$$y = \sqrt{\frac{e^3}{4\pi\epsilon}} \frac{\sqrt{\beta E}}{\varphi} = 3.795 \times 10^{-5} \frac{\sqrt{\beta E}}{\varphi}. \quad (16.2)$$

In our model, we have chosen a simpler approximation originated by J. H. Han[54]

$$\begin{aligned} v(y) &= a - by^2 \\ t(y) &\approx 1. \end{aligned}$$

These approximations are valid for a large range of y , corresponding to typical applied electric field ranges in RF guns.

Whenever the normal components of an electric field are strong enough the field emission current density will be limited by space charge effect[52]. To cover this situation we incorporated the 1D Child-Langmuir law

$$\begin{aligned} J(\mathbf{r}, t) &= \frac{4\epsilon_0}{9} \sqrt{2\frac{e}{m}} \left(\frac{V^{3/2}}{d^2} \right) \\ &= \frac{4\epsilon_0}{9} \sqrt{2\frac{e}{m}} \left(\frac{E^{3/2}}{d^{1/2}} \right) [\text{A/m}^2] \end{aligned} \quad (16.3)$$

into our field emission model. $J(\mathbf{r}, t)$ denotes space charge limited emission current density in position \mathbf{r} and time t , ϵ_0 the permittivity in vacuum, E the normal component of electric field on the surface and d the distance from the position where E is evaluated. Currently we choose d to be equal to the distance traveled by emitted particles in one time step, i.e., $d = \frac{eE\Delta t^2}{2m_0}$ where Δt is simulation time step.

16.1 Field Emission Command

To perform field emission related simulation, a triangulated surface geometry defined by `GEOMETRY` command (see Chapter 14) should be specified and attached to the elements (currently only `RFCavity` element is valid for field emission). A `SURFACEEMISSION` type of distribution, defined in `DISTRIBUTION` command should be attached to the `GEOMETRY` command. And users can customize dark current simulation by specifying the value of the work function φ , local field enhancement factor β and other parameters present in (16.1) and (16.2) in the `SURFACEEMISSION` type of distribution definition in input file. See the following example input file and Table 16.1 for a summary of the field emission related command in the `SURFACEEMISSION` type of distribution definition.

```
DistSurf: DISTRIBUTION, DISTRIBUTION = "SURFACEEMISSION",
          NPDARKCUR = 0, INWARDMARGIN=0.0,
          FNBETA = 30, FNMAXEMI = 2,
          FNFIELDTHR = -0.1;
ge:      GEOMETRY, FGEOM="../New_Gun.h5",
          S=0.0, DISTR=DistSurf,
          ZSHIFT=0.0;
FINSSGUN: RFCavity, L = 0.175,
          VOLT = 100.0, FMAPFN = "../RF_GUN_PSI-fieldmap.T7" ,
          GEOMETRY = ge, ELEMEDGE =0.0,
          TYPE = "STANDING", FREQ = 2997.922938148;
\ldots
```

Table 16.1: Field Emission Command summary

| Command | Purpose (Default) |
|------------|---|
| FNA | Empirical constant A for F-N emission model (1.54×10^{-6}) |
| FNB | Empirical constant B for F-N emission model (6.83×10^9) |
| FNY | Constant for image charge effect parameter $y(E)$ (3.795×10^{-5}) |
| FNVYZERO | Zero order constant for $v(y)$ function (0.9632) |
| FNVYSECOND | Second order constant for $v(y)$ function (1.065) |
| FNPFIW | Work function of gun surface material (4.65 eV) |
| FNBETA | Field enhancement factor β for F-N emission (50.0) |
| FNFIELDTHR | Field threshold for F-N emission (30.0 MV/m) |
| FNMAXEMI | Maximum Number of electrons emitted from a single triangle in each time step (10) |

Chapter 17

Multipacting

Multiple electron impacting (multipacting) is a phenomenon in radio frequency (RF) structure that under certain conditions (material and geometry of the RF structure, frequency and level of the electromagnetic field, with or without the appearance of the magnetic field ...), electrons secondary emission yield (SEY) coefficient will be larger than one and lead to exponential multiplication of electrons.

Besides the particle tracker in OPAL, the computational model for solving multipacting problem contains an accurate representation of 3D geometry of RF structure by using triangulated surface mesh (see Chapter 14 and Chapter 16), an efficient particle-boundary collision test scheme, two different secondary emission models, and necessary post-processing scripts.

As we use a triangulated surface mesh to represent the RF structure, our particle-boundary collision test scheme is based on line segment-triangle intersection test. An axis aligned boundary box combined with surface triangle inward normal method is adopted to speedup the particle-boundary collision test [55].

The SEY curve is a very important property of the surface material for the development of a multipacting in a RF structure. Figure 17.1 shows a typical SEY curve. Here, the horizontal axis is the energy of impacting electron, the vertical axis is the SEY value δ , defined as [56]:

$$\delta = \frac{I_s}{I_0} \quad (17.1)$$

where I_0 is the incident electron beam current and I_s is the secondary current, i.e., the electron current emitted from the surface. Usually the SEY value δ appeared in an SEY curve is the measured SEY with normal incident, i.e., the impacting electron is perpendicular to the surface. The energy E_1 and E_2 are the first crossover energy and the second crossover energy respectively, where the SEY value δ exceed and fall down to $\delta = 1$ at the first time. Obviously, only the energy range of $\delta > 1$, i.e., $E \in (E_1, E_2)$ can contribute to multipacting.

Both Furman-Pivi's probabilistic secondary emission model [56] and Vaughan's formula based secondary emission model [57] have been implemented in OPALand have been benchmarked (see Section 17.2).

The Furman and Pivi's secondary emission model calculates the number of secondary electrons that result from an incident electron of a given energy on a material at a given angle (see Figure 17.2). For each of the generated secondary electrons the associated process: *true secondary*, *rediffused* or *backscattered* is recorded, as is sketched in Figure 17.2. This model is mathematically self-consistent, which means that (1) when averaging over an infinite number of secondary-emission events, the reconstructed δ and $d\delta/dE$ are guaranteed to agree with the corresponding input quantities; (2) the energy integral of $d\delta/dE$ is guaranteed to equal δ ; (3) the energy of any given emitted electron is guaranteed not to exceed the primary energy; and (4) the aggregate energy of the electrons emitted in any multielectron event is also guaranteed not to exceed the primary energy. This model contains built-in SEY curves for copper and stainless steel and the only thing user need to set is to choose the material type, i.e., copper or stainless steel, as long as the surface material of user's RF structure has the same SEY curve as built-in SEY curves.

Although a set of parameters in the model can be adjusted to model different SEY curves without breaking the

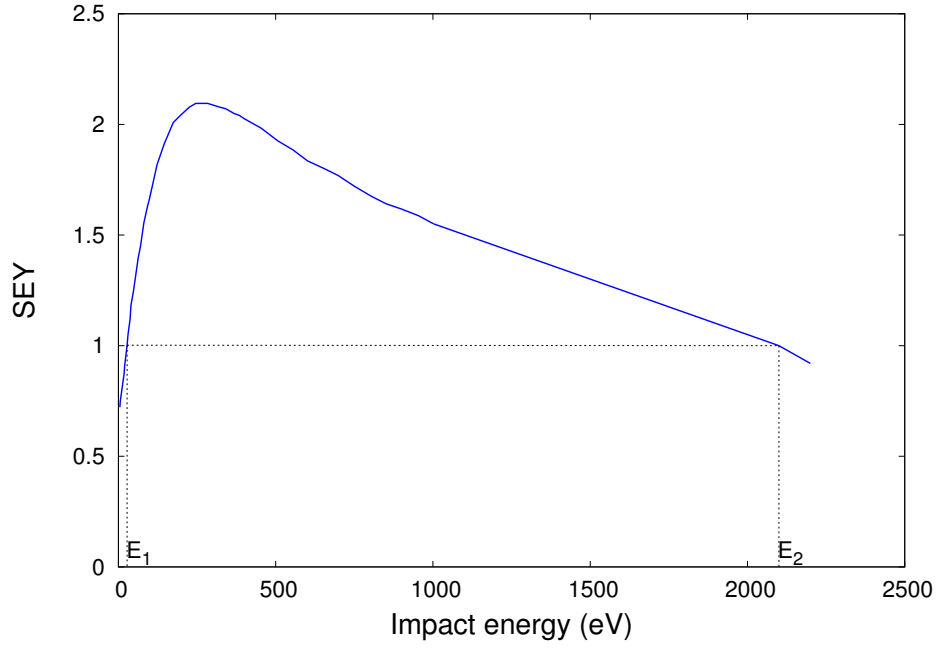


Figure 17.1: Typical SEY curve

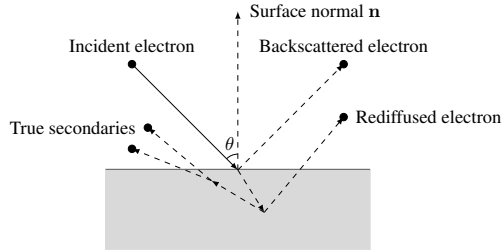


Figure 17.2: Sketch map of the secondary emission process.

above mentioned mathematical self-consistency, it is easier to use Vaughan's formula based secondary emission model if user has to model a different SEY curve.

The Vaughan's secondary emission model is based on a secondary emission yield formula [57, 58]:

$$\delta(E, \theta) = \delta_{max}(\theta) \cdot (ve^{1-v})^k, \text{ for } v \leq 3.6 \quad (17.2a)$$

$$\delta(E, \theta) = \delta_{max}(\theta) \cdot 1.125/v^{0.35}, \text{ for } v > 3.6 \quad (17.2b)$$

$$\delta(E, \theta) = \delta_0, \text{ for } v \leq 0 \quad (17.2c)$$

where

$$v = \frac{E - E_0}{E_{max}(\theta) - E_0},$$

$$k = 0.56, \text{ for } v < 1,$$

$$k = 0.25, \text{ for } 1 < v \leq 3.6,$$

$$\delta_{max}(\theta) = \delta_{max}(0) \cdot (1 + k_\theta \theta^2 / 2\pi),$$

$$E_{max}(\theta) = E_{max}(0) \cdot (1 + k_E \theta^2 / 2\pi).$$

The secondary emission yield value for an impacting electron energy E and incident angle θ w.r.t the surface normal is denoted as $\delta(E, \theta)$. Parameter k_θ and k_E denotes the dependence on surface roughness. Both should be assigned a default value of 1.0, which appears appropriate for typical dull surfaces in a working tube environment, and lower values down to zero or higher values, up to about 2.0, are only valid for specified cases [57]. $E_{max}(0)$ is the impacting energy when incident angle is zero and secondary yield reaches its maximum value. E_0 is an adjustable parameter to make the first crossover energy be fitted to the experiment data [59]. δ_0 is the user specified constant denote the SEY of low impacting energy.

The emission energy obeys the thermal energy distribution:

$$f(\varepsilon) = \frac{\varepsilon}{(k_B T)^2} \exp\left(\frac{-\varepsilon}{k_B T}\right) \quad (17.3)$$

The polar angles of emitted secondaries $\theta \in [0, \pi/2]$ are with probability density $\cos^\alpha \theta$, and the azimuthal angles $\phi \in [0, 2\pi]$ are with uniform probability density. These emission angles of the secondary electrons is relative to the local coordinate system which is centered at the collision point and whose z axis is along the inward normal of the surface.

Motivated by the fact that the population of particles in simulation domain may continually grow exponentially and lead to tens of magnitude larger within limited simulation time steps, which may cause the exhaust of computing memory, a re-normalization of simulation particle number approach is also implemented. In each electron impacting events, instead of emitting the real number of simulation particles predicted by secondary emission models, this re-normalization approach emit only one particle, and the current of newly emitted particle will be the current of incident particle multiplied by SEY value δ .

17.1 Commands Related to Multipacting Simulation

To perform multipacting simulation, a triangulated surface geometry defined in the `GEOMETRY` command (see Chapter 14) must be specified and attached to the elements (currently only `RFCavity`, `ParallelPlate` and `Drift` elements are available for multipacting simulation).

A distribution array, containing `SURFACEEMISSION` and `SURFACERANDCREATE` type of distributions, defined in the `DISTRIBUTION` command must be attached to the `GEOMETRY`. Users can use commands contained in `SURFACERANDCREATE` type of distribution to specify the position of initial *seed* electrons. And commands within `SURFACEEMISSION` type of distribution can be used to customize the type and the parameters of secondary emission model in input file.

A summary of multipacting simulation related parameters are given in Table 17.1.

The following example shows the usage of the multipacting simulation related command.

```
Title, string="Cyclotron_Multipacting_Simulation_example";

Option, TFS=FALSE;
Option, ECHO=FALSE;
Option, INFO=FALSE;

Option, PSDUMPFREQ=1;
Option, STATDUMPFREQ=1;
Option, PPDEBUG=FALSE;
Option, SURFDUMPFREQ=100;

// Set an upper limit of simulation particle number
// to prevent memory overflow.

MAXPARTSNUM=1000000;

// SECONDARYFLAG = 1: Using Furman-Pivi's model
// SURFMATERIAL=0: surface material is copper
// Set NEMISSIONMODE=false will use re-normalize
// simulation particle approach.
// Set the field enhancement factor FNBETA to
// a very small number to prevent field
// emission.

DistSurf: DISTRIBUTION, DISTRIBUTION = "SURFACEEMISSION",
                                NPDARKCUR =0, INWARDMARGIN = 0.0,
                                FNBETA = 0.1, FNMAXEMI = 2,
                                SECONDARYFLAG = 1,
                                NEMISSIONMODE=false,
                                SURFMATERIAL=0;

// INWARDMARGIN: seed electron positions along the
// inward normal w.r.t the boundary surface.
DistSurf1: DISTRIBUTION,
            DISTRIBUTION = "SURFACERANDCREATE",
            INWARDMARGIN = 0.0, NPDARKCUR =10000,
            EINITHR = 0.2;

// For multipacting study of cyclotron cavity,
// the axis z in geometry file is actually axis y
// in ParallelTTracker, so we need to shift z
// coordinates of the geometry by specifying ZSHIFT to make
// sure that the z coordinates read in by ParallelTTracker
// will be correct.

ge: GEOMETRY, FGEOM="../10.h5", S=0.0,
     ZSHIFT=0.631, DISTRs={DistSurf, DistSurf1};
```



```

Box: RFCavity, PLENGTH = 1.262, VOLT = 1,
      GEOMETRY = ge, FMAPFN = "../CyciaeEM.h5",
      ELEMEDGE =0, FAST=true,
      FREQ =44.6, LAG = 0.0,
      DX = 0, DY = 0, DZ = 0;

// This element is used to model the magnetic field in
// the valley of a cyclotron, where the RF cavity is installed.

Mag: CYCLOTRONVALLEY, FMAPFN = "../CyciaeMagReal.h5",
      ELEMEDGE =0, DX = 0, DY = 0, DZ = 0;

Benchmark: Line = (Box, Mag);

Fs1:FIELDSOLVER, FSTYPE = NONE, MX = 32,
      MY = 32, MT = 256,
      PARFFTX = true, PARFFTY = true,
      PARFFTT = false, BCFFTX = open,
      BCFFTY = open, BCFFTT = open,
      BBOXINCR = 0.1, GREENSF = INTEGRATED;

qb=0.2e-9;
bfreq=300;
bcurrent=qb*bfreq;

beam1: BEAM, PARTICLE = ELECTRON, pc = P0,
      NPART = 2000, BFREQ = bfreq ,
      BCURRENT = bcurrent, CHARGE =-1;

Select, Line=Benchmark;

track, line= Benchmark, beam=beam1,
      MAXSTEPS=23000, DT=4e-12, ZSTOP=3;

run, method = "PARALLEL-T", beam = beam1,
      fieldsolver = Fs1, MULTIPACTING=true;
endtrack;

Quit;

```

Table 17.1: Multipacting Related Command Summary

| Command | Purpose (Default) |
|---------------|---|
| VW | Velocity scalar in Maxwellian Dist (1.0 m/s) |
| VVTHERMAL | Thermal velocity in Maxwellian Dist (7.268929821×10^5 m/s) |
| SECONDARYFLAG | Secondary model type, 0:none, 1:Furman-Pivi, 2:Vaughan (0) |
| NEMISSIONMODE | Emit real No. secondaries or not (true) |
| VEZERO | SEY will be δ_0 , if energy is less than VEZERO in Vaughan's model (12.5 eV) |
| VSEYZERO | δ_0 in Vaughan's model (0.5) |
| VSEYMAX | δ_{max} in Vaughan's model (2.22) |
| VEMAX | Energy related to δ_{max} in Vaughan's model (165 eV) |
| VKENERGY | The roughness of surface for impact energy in Vaughan's model(1.0) |
| VKTHETA | The roughness of surface for impact angle in Vaughan's model(1.0) |
| SURFMATERIAL | The material type for Furman-Pivi model, 0: copper; 1: stainless steel (0) |

17.2 Run Parallel Plate Benchmark

Both the Furman-Pivi's model and Vaughan's model have been carefully benchmarked in both re-normalize simulation particle approach and real simulation particles approach against a non-stationary multipacting theory [60]. The OPALsimulation results and the theory match very well (see figure 17.3 and figure 17.4).

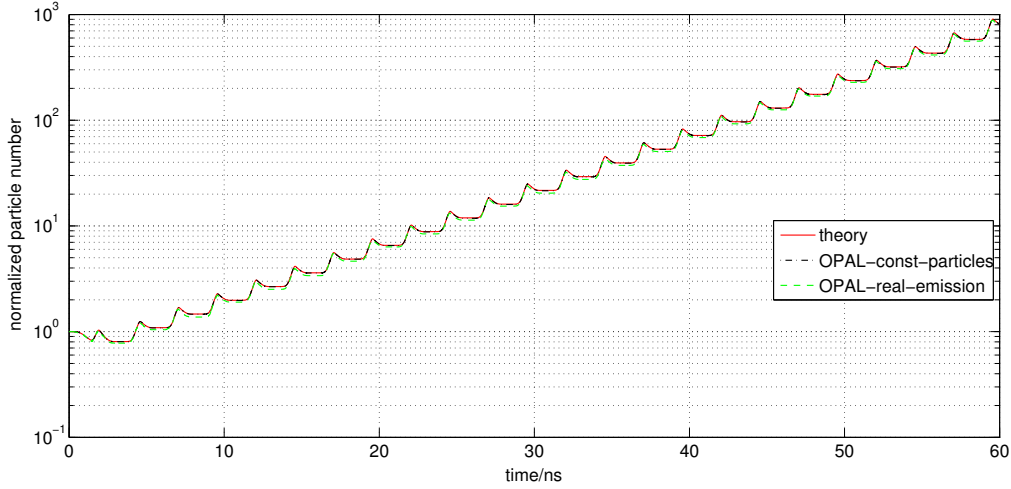


Figure 17.3: Time evolution of electron number predicted by theoretical model and OPAL simulation using Furman-Pivi's secondary emission model with both constant simulation particle approach and real emission particle approach at $f = 200\text{MHz}$, $V_0 = 120\text{V}$, $d = 5\text{mm}$

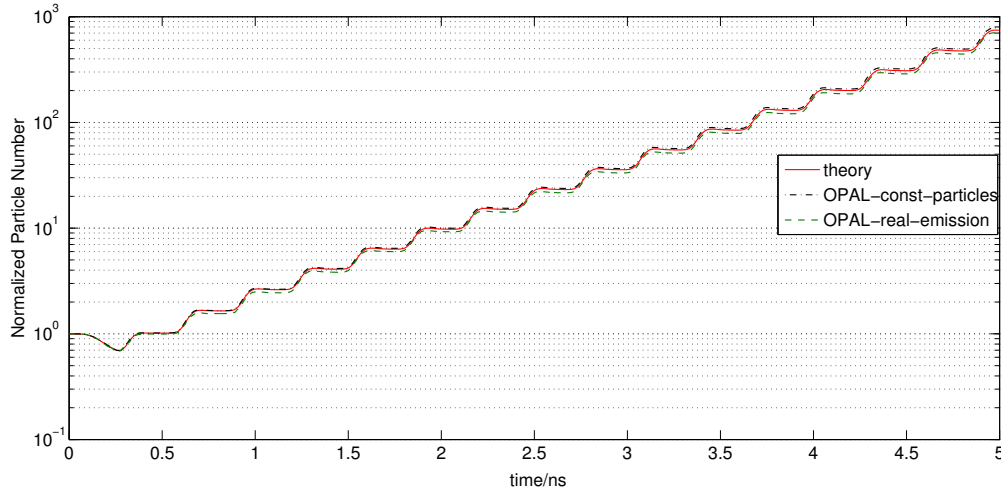


Figure 17.4: Time evolution of electron number predicted by theoretical model and OPAL simulation using Vaughan's secondary emission model with both constant simulation particle approach and real emission particle approach at $f = 1640\text{MHz}$, $V_0 = 120\text{V}$, $d = 1\text{mm}$.

To run the parallel plate benchmark simulation, user need to set the option `PPDEBUG` in the input file `true`.

The input file and the geometry file needed by the parallel plate benchmark simulation can be found in the regression test folder.

17.3 PostProcessing

In the general case (not only in multipacting simulations), OPAL will dump the 6D phase space and statistical information of the particles in the simulation domain, into a `h5` file. The dump frequency, i.e., after how many time steps the particle information will be saved can be specified with the option `PSDUMPFREQ`. Setting `Option, PSDUMPFREQ=1` dumps the information in each time step.

A utility tool `h5ToVtk` converts the `h5` file to the Visualization Toolkit (VTK) legacy format. The number of VTK files equals to the number of time steps in `h5` file. These VTK files together with a VTK file automatically generated by the geometry class of OPAL which contains the geometry of the RF structure under study can be visualized using for example with Paraview [61]. The animation and clip feature of Paraview is very useful to visualize the particle motion inside the RF structure.

For simulations involving the geometry (multipacting and field emission), OPAL will also dump the position and current of incident particles into another `h5` file with the name `*_Surface.h5`, where the asterisk stands for the base name of the user's OPAL input file. If we need this surface loss data during post processing, we should specify the dump frequency in the option `SURFDUMPFREQ` with a positive integer in the OPAL input file, otherwise, the default value of the option is `SURFDUMPFREQ=-1`, and the `*_Surface.h5` will not be generated. Another utility tool `h5SurfaceVtk` convert the `*_Surface.h5` file to VTK files. For multipacting simulation, these VTK files can be used to visualize the *hot spots* of the RF structure where multipacting happens.

The above mentioned utility tools are based on H5hut library, and will soon be available in the distribution.

Some of the boundary geometry related simulations, like the multipacting simulation using re-normalizing particle number approach, or dark current simulations where the current of field emitted particles from a single triangle has been re-normalized as the model predicted current has exceeded the user defined upper limit, the current (weight) of simulation particles varies and each simulation particle stands for more physical particles than the initial simulation particles. In these cases, instead of using simulation particles, we count the number of *effective particles* defined as the ratio of total current in simulation over the current of a single initial particle.

An ASCII file named `Part_statistics.dat` containing the simulation time, the number of impacts and associated total SEY value as well as the number of *effective particles* in each time step. This makes the analysis of the time evolution of particle density feasible with tools like GNUPLOT.

Chapter 18

Physics Models Used in the Particle Matter Interaction Model

18.1 The Energy Loss

The energy loss is simulated using the Bethe-Bloch equation.

$$-dE/dx = \frac{Kz^2Z}{A\beta^2} \left[\frac{1}{2} \ln \frac{2m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} - \beta^2 \right], \quad (18.1)$$

where Z is the atomic number of absorber, A is the atomic mass of absorber, m_e is the electron mass, z is the charge number of the incident particle, $K = 4\pi N_A r_e^2 m_e c^2$, r_e is the classical electron radius, N_A is the Avogadro's number, I is the mean excitation energy. β and γ are kinematic variables. T_{max} is the maximum kinetic energy which can be imparted to a free electron in a single collision.

$$T_{max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma m_e/M + (m_e/M)^2}, \quad (18.2)$$

where M is the incident particle mass.

The stopping power is compared with PSTAR program of NIST in Fig. 18.1.

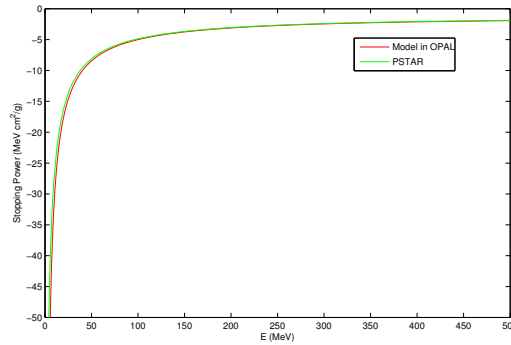


Figure 18.1: The comparison of stopping power with PSTAR.

Energy straggling: For relatively thick absorbers such that the number of collisions is large, the energy loss distribution is shown to be Gaussian in form. For nonrelativistic heavy particles the spread σ_0 of the Gaussian

distribution is calculated by:

$$\sigma_0^2 = 4\pi N_A r_e^2 (m_e c^2)^2 \rho \frac{Z}{A} \Delta s, \quad (18.3)$$

where ρ is the density, Δs is the thickness.

18.2 The Coulomb Scattering

The Coulomb scattering is treated as two independent events: the multiple Coulomb scattering and the large angle Rutherford scattering.

Using the distribution given in Classical Electrodynamics, by J. D. Jackson, the multiple- and single-scattering distributions can be written:

$$P_M(\alpha) d\alpha = \frac{1}{\sqrt{\pi}} e^{-\alpha^2} d\alpha, \quad (18.4)$$

$$P_S(\alpha) d\alpha = \frac{1}{8 \ln(204Z^{-1/3})} \frac{d\alpha}{\alpha^3}, \quad (18.5)$$

where $\alpha = \frac{\theta}{\langle \Theta^2 \rangle^{1/2}} = \frac{\theta}{\sqrt{2}\theta_0}$.

the transition point is $\theta = 2.5\sqrt{2}\theta_0 \approx 3.5\theta_0$,

$$\theta_0 = \frac{13.6 \text{ MeV}}{\beta c p} z \sqrt{\Delta s / X_0} [1 + 0.038 \ln(\Delta s / X_0)], \quad (18.6)$$

where p is the momentum, Δs is the stepsize, and X_0 is the radiation length.

18.2.1 Multiple Coulomb Scattering

Generate two independent Gaussian random variables with mean zero and variance one: z_1 and z_2 . If $z_2\theta_0 > 3.5\theta_0$, start over. Otherwise,

$$x = x + \Delta s p_x + z_1 \Delta s \theta_0 / \sqrt{12} + z_2 \Delta s \theta_0 / 2, \quad (18.7)$$

$$p_x = p_x + z_2 \theta_0. \quad (18.8)$$

Generate two independent Gaussian random variables with mean zero and variance one: z_3 and z_4 . If $z_4\theta_0 > 3.5\theta_0$, start over. Otherwise,

$$y = y + \Delta s p_y + z_3 \Delta s \theta_0 / \sqrt{12} + z_4 \Delta s \theta_0 / 2, \quad (18.9)$$

$$p_y = p_y + z_4 \theta_0. \quad (18.10)$$

18.2.2 Large Angle Rutherford Scattering

Generate a random number ξ_1 , if $\xi_1 < \frac{\int_{2.5}^{\infty} P_S(\alpha) d\alpha}{\int_0^{2.5} P_M(\alpha) d\alpha + \int_{2.5}^{\infty} P_S(\alpha) d\alpha} = 0.0047$, sampling the large angle Rutherford scattering.

The cumulative distribution function of the large angle Rutherford scattering is

$$F(\alpha) = \frac{\int_{2.5}^{\alpha} P_S(\alpha) d\alpha}{\int_{2.5}^{\infty} P_S(\alpha) d\alpha} = \xi, \quad (18.11)$$

where ξ is a random variable. So

$$\alpha = \pm 2.5 \sqrt{\frac{1}{1-\xi}} = \pm 2.5 \sqrt{\frac{1}{\xi}}. \quad (18.12)$$

Generate a random variable P_3 ,
 if $P_3 > 0.5$

$$\theta_{Ru} = 2.5 \sqrt{\frac{1}{\xi}} \sqrt{2} \theta_0, \quad (18.13)$$

else

$$\theta_{Ru} = -2.5 \sqrt{\frac{1}{\xi}} \sqrt{2} \theta_0. \quad (18.14)$$

The angle distribution after Coulomb scattering is shown in Fig. 18.2. The line is from Jackson's formula, and the points are simulations with Matlab. For a thickness of $\Delta s = 1e-4$ m, $\theta = 0.5349\alpha$ (in degree).

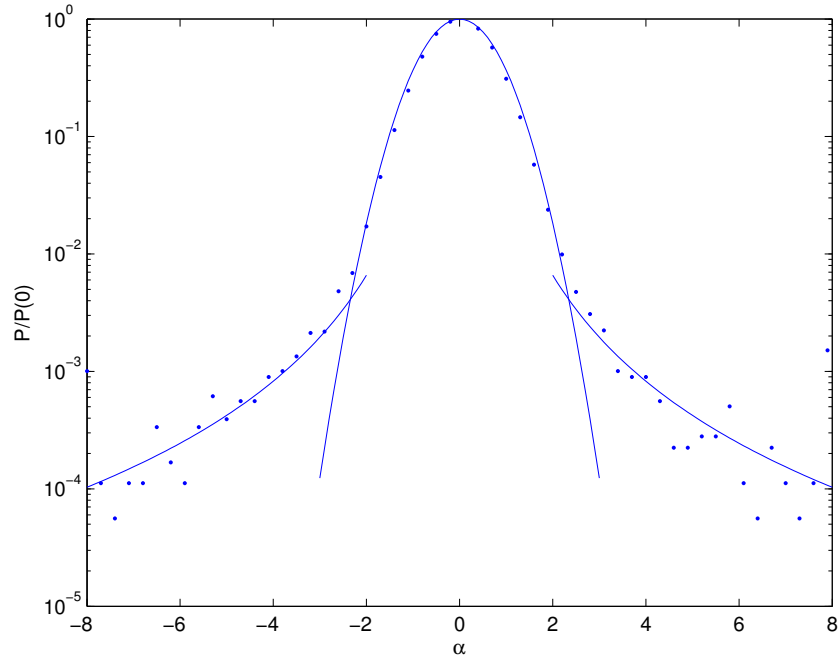


Figure 18.2: The comparison of Coulomb scattering with Jackson's book.

18.3 The Flow Diagram of *CollimatorPhysics* Class in OPAL

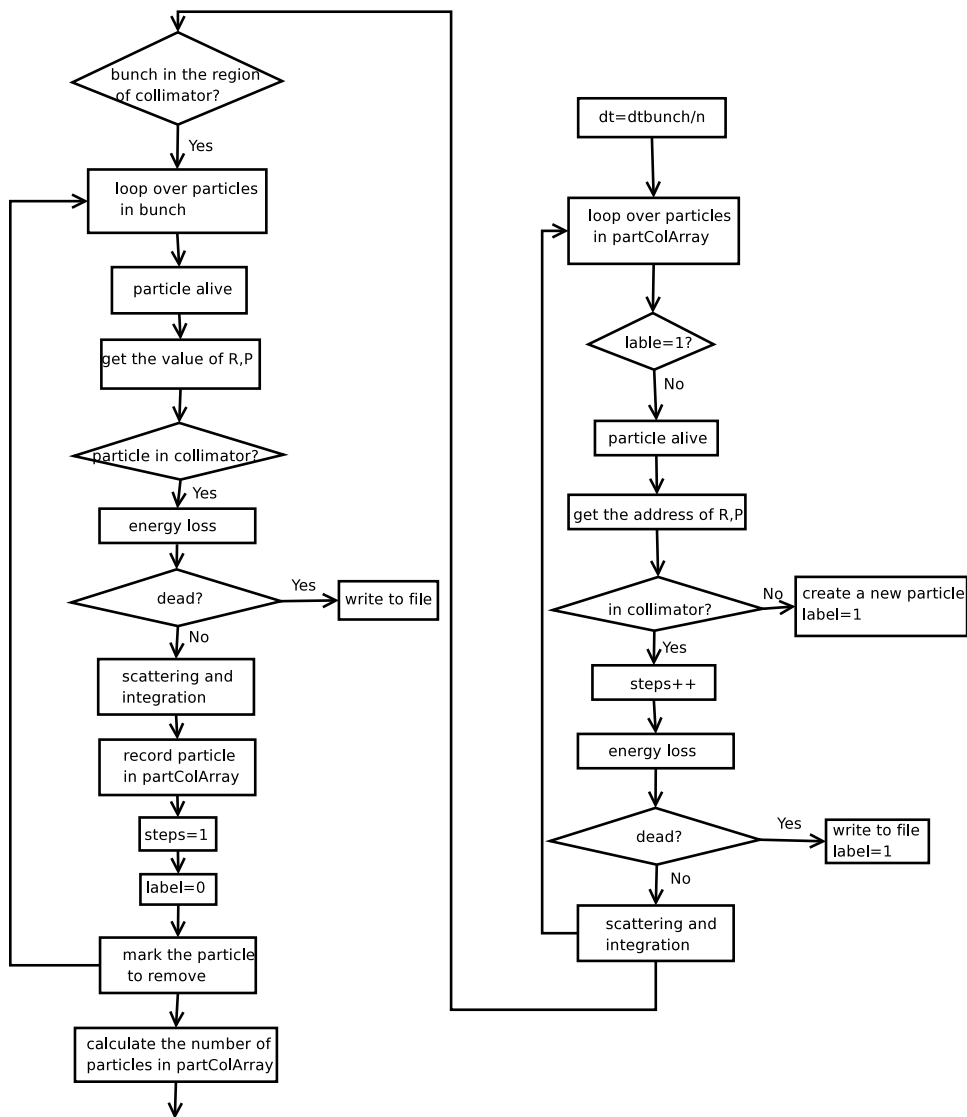


Figure 18.3: The diagram of CollimatorPhysics in OPAL.

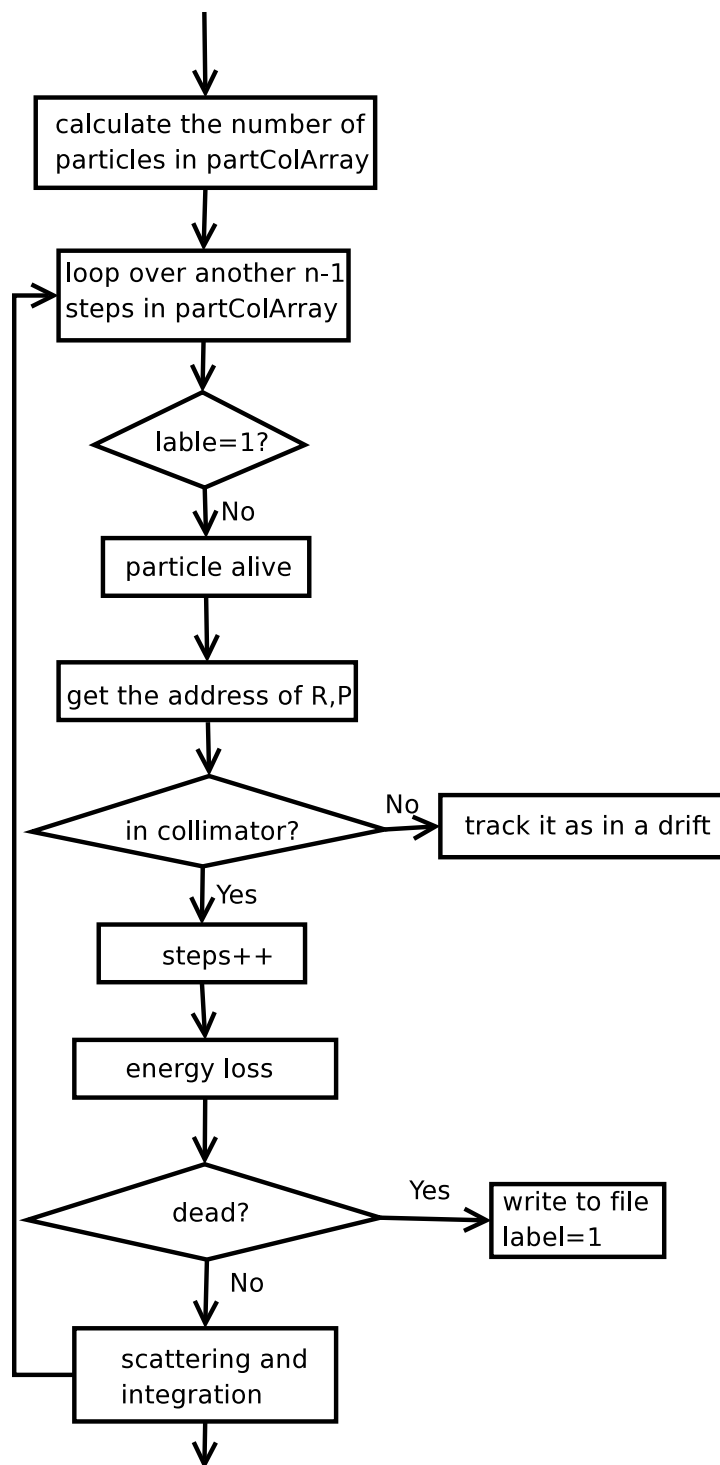


Figure 18.4: The diagram of CollimatorPhysics in OPAL(Continued).

18.3.1 The Substeps

Small step is needed in the routine of CollimatorPhysics.

If a large step is given in the main input file, in the file CollimatorPhysics.cpp, it is divided by a integer number n to make the stepsize using for the calculation of collimator physics less than $1.01\text{e-}12$ s. As shown by Fig. 18.3 and Fig. 18.4 in the previous section, first we track one step for the particles already in the collimator and the newcomers, then another $(n-1)$ steps to make sure the particles in the collimator experience the same time as the ones in the main bunch.

Now, if the particle leave the collimator during the $(n-1)$ steps, we track it as in a drift and put it back to the main bunch when finishing $(n-1)$ steps.

18.4 Example of an Input File

```
KX1IPHYS: SurfacePhysics, TYPE="Collimator", MATERIAL="Cu";
KX2IPHYS: SurfacePhysics, TYPE="Collimator", MATERIAL="Graphite";
KX0I: ECollimator, L=0.09, ELEMEDGE=0.01, APERTURE={0.003,0.003},OUTFN="KX0I.h5", SURFACE-
PHYSICS='KX1IPHYS';
FX5: Slit, L=0.09, ELEMEDGE=0.01, APERTURE={0.005,0.003}, SURFACEPHYSICS='KX2IPHYS';
FX16: Slit, L=0.09, ELEMEDGE=0.01, APERTURE={-0.005,-0.003}, SURFACEPHYSICS='KX2IPHYS';
```

FX5 is a slit in x direction, the APERTURE is **POSITIVE**, the first value in APERTURE is the left part, the second value is the right part.

FX16 is a slit in y direction, the APERTURE is **NEGATIVE**, the first value in APERTURE is the down part, the second value is the up part.

18.5 A Simple Test

A cold Gaussian beam with $\sigma_x = \sigma_y = 5$ mm. The position of the collimator is from 0.01 m to 0.1 m, the half aperture in y direction is 3 mm. Fig. 18.5 shows the trajectory of particles which are either absorbed or deflected by a copper slit. As a benchmark of the collimator model in OPAL, Fig. 18.6 shows the energy spectrum and angle deviation at $z=0.1$ m after an elliptic collimator.

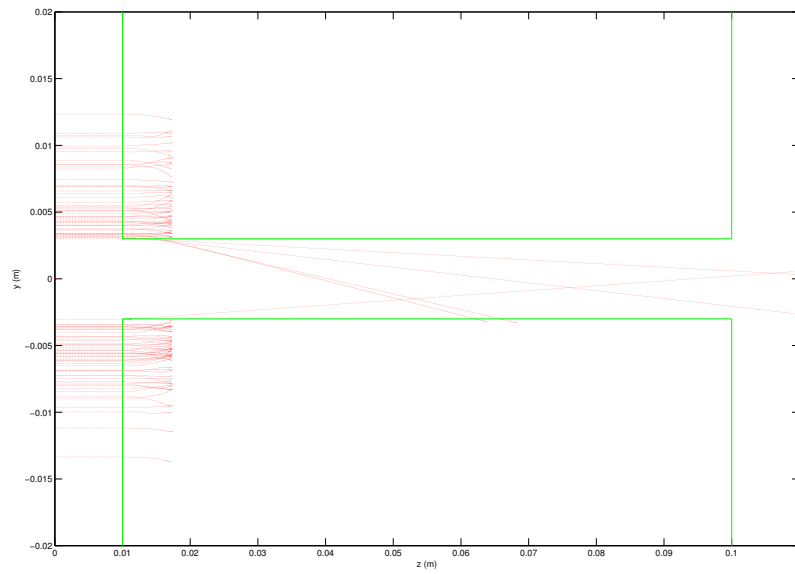
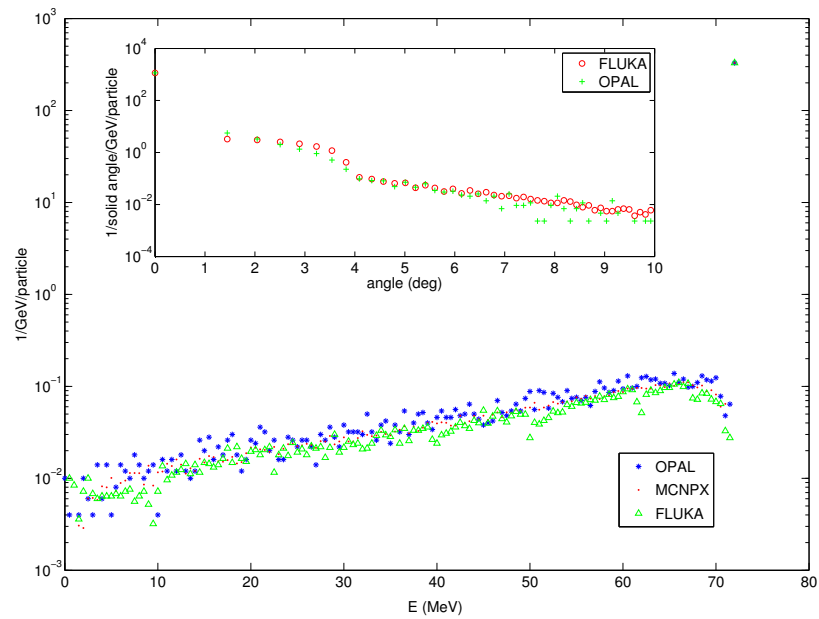


Figure 18.5: The passage of protons through the collimator.

Figure 18.6: The energy spectrum and scattering angle at $z=0.1$ m

Appendix A

Installation

OPAL and all its flavours are based on several packages which are all installable using `cmake` or the `configure-make-install` trilogy.

OPAL is also preinstalled on several HPC clusters including the FELSIM and Merlin clusters at PSI. The preinstalled version can be accessed using the module command:

```
module load opal
```

Due to some incompatibilities, between the Intel compiler and the Gnu libraries, you have to use GCC 4.5 in combination with Intel 12.1. Next we describe the installation process for the GNU 4.6.3 compiler.

A.1 Build and install OPAL on a Mac & Linux

A.1.1 Supporting Libraries

Several libraries and tools must be present before starting with the actual OPAL installation process described in (A.1.3). The following packages are maybe already installed. Please check the versions carefully and do not use older ones. For a Linux installation you can skip `macport` and `Xcode` related software.

- `macport` from www.macports.org
- `Xcode` <http://developer.apple.com/TOOLS/Xcode/>
- `automake-1.10.2`
- `autoconf-2.64`
- `libtool-2.2`
- `cmake-2.8.4`

A.1.2 Environment Variables

Assuming IPPL and OPAL resides in `$HOME/svnwork`, the following environment variables must be set accordingly:

```
export OPAL_ROOT=$HOME/svnwork/OPAL/  
export CLASSIC_ROOT=$OPAL_ROOT/classic/5.0/  
  
export H5hut=$HOME/svnwork/H5hut  
export IPPL_ROOT=$HOME/svnwork/ippl/  
export IPPL_PREFIX=$IPPL_ROOT/build/
```

Build/Install gcc (4.6.3) Mac

```
sudo port install gcc46
```

Build/Install gsl (1.14) Mac

```
sudo port install gsl
```

Build/Install OpenMPI (openmpi-1.4.3) Mac & Linux

```
CC=gcc CXX=g++ F77=gfortran ./configure
```

Build/Install HDF5 1.8.7 Mac & Linux

```
./configure --enable-parallel --prefix=/usr/local
```

```
Libraries have been installed in:
  /usr/local/lib
```

```
If you ever happen to want to link against installed libraries
in a given directory, LIBDIR, you must either use libtool, and
specify the full pathname of the library, or use the '-LLIBDIR'
flag during linking and do at least one of the following:
```

- add LIBDIR to the 'DYLD_LIBRARY_PATH' environment variable
 during execution

```
See any operating system documentation about shared libraries
for more information, such as the ld(1) and ld.so(8) manual
pages.
```

H5hut (1.99.9) Mac & Linux

The tarball can be found at:

svn+ssh://savannah02.psi.ch/repos/H5hut/src/tags/1.99.9

Now we can build and install the package:

```
./configure --enable-parallel CFLAGS=-std=c99
make
make install
```

Install IPPL Mac & Linux

1.

```
cd $HOME/svnwork
```

2. Get source code:

```
svn co --username username \
  svn+ssh://savannah02.psi.ch/repos/ippl/src/trunk ippl
```

3. `cd ippl`¹
4.

```
mkdir build
export IPPL_ROOT=$HOME/svnwork/ippl
export IPPL_PREFIX=$IPPL_ROOT/build
rm -f CMakeCache.txt
CXX=mpicxx cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=$IPPL_PREFIX $IPPL_ROOT
```

5. `make`
6. `make install`
7. Add the `export` statements to your `.bashrc` file.

A.1.3 Installing OPAL

The following `svn` checkout²

```
cd $OPAL_ROOT
svn+ssh://savannah02.psi.ch/repos/opal/src .
```

will get you the trunk of the repository. Now install OPAL³

```
mkdir build
cd build
CXX=mpicxx cmake -DCMAKE_BUILD_TYPE=RELEASE $OPAL_ROOT
```

Use `-DCMAKE_BUILD_TYPE=DEBUG` to enable `-g`.

A.2 Cray XE6 Installation

These notes are for installing OPAL on Hopper at the National Energy Research Scientific Computing Center (NERSC). NERSC is an open science computing center sponsored by the U.S. Department of Energy and located at Lawrence Berkely Laboratory (LBL). Hopper is a Cray XE6 system and, at the time of this writing, is their premier super computer.

The instructions below work, but you may want to modify them depending on where you prefer various codes and libraries to be located or if you are on a similar Cray system.

A.2.1 .bash_profile.ext File

Edit the `.bash_profile.ext` file in your home directory. Add the following lines.

```
if [ $NERSC_HOST == "hopper" ]
then
  export PATH=$PATH:$HOME/hopper/bin
  # Module files.
  module load cmake
  module load gsl
  module load hdf5-parallel
  module load zlib
```

¹Note, we encountered in some cases a problem when the build directory is under `$IPPL_ROOT`

²If you can not checkout the sources send an email to andreas.adelmann@psi.ch.

³We encountered in some cases a problem when the build directory is under `$OPAL_ROOT`

```

module swap PrgEnv-pgi PrgEnv-gnu
fi

source $HOME/.bashrc

```

A.2.2 .bashrc.ext File

Edit the `.bashrc.ext` file in your home directory. Add the following lines.

```

if [ $NERSC_HOST == "hopper" ]
then
  # For OPAL.
  export HDF5_INCLUDE_PATH=$CRAY_HDF5_DIR/hdf5-parallel-gnu/include
  export HDF5_LIBRARY_PATH=$CRAY_HDF5_DIR/hdf5-parallel-gnu/lib
  export H5hut=$HOME/hopper/extlib/H5hut-1.99.8
  export GSL_PREFIX=$GSL_DIR
  export IPPL_ROOT=$HOME/hopper/extlib/ippl
  export OPAL_ROOT=$HOME/svnwork/OPAL
  export CLASSIC_ROOT=$OPAL_ROOT/classic/5.0
fi

```

These environment variables need to be set so we can build OPAL. When you are done, issue the command:

```

source $HOME/.bash_profile

```

A.2.3 OPAL

To install OPAL you will need to perform the following steps:

1. Install H5hut.
2. Install IPPL.
3. Install OPAL.

Install H5hut

1. `mkdir -p hopper/extlib`
2. `mkdir svnwork`
3. `cd $HOME/svnwork`
4. Get the source code:

```

\url{svn+ssh://savannah02.psi.ch/repos/H5hut/src/tags/1.99.9 H5hut-1.99.9}

```

5. `cd H5hut-1.99.9`
6. `./autogen.h`

```

export HDF5ROOT=$CRAY_HDF5_DIR/hdf5-parallel-gnu

```

7. `./configure --prefix=$HOME/hopper/extlib/H5hut-1.99.9 \`
`--enable-parallel CFLAGS=-std=c99 CC=cc`
-

8. make
9. make install

Install IPPL

1.

`cd $HOME/svnwork`

2. Get source code:

`svn co --username username \
svn+ssh://savannah02.psi.ch/repos/ippl/src/trunk ippl`

3. `cd ippl`
4.

`mkdir build
export IPPL_ROOT=$HOME/svnwork/ippl
export IPPL_PREFIX=$IPPL_ROOT/build
rm -f CMakeCache.txt
CXX=mpicxx cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=$IPPL_PREFIX $IPPL_ROOT`

5. make
6. make install
7. Add the export statements to your .bashrc file.

Install OPAL

1. `cd`
2. `cd svnwork`
3. Get the source code:

`svn co --username username \
svn+ssh://savannah02.psi.ch/repos/opal/src OPAL`

4.

`cd $OPAL_ROOT`

5.

`CXX=CC cmake $OPAL_ROOT`

6. `make\verb`
7.

`mkdir $HOME/hopper/bin`

8.

`cp src/opal $HOME/hopper/bin`

A.3 Using pre-build Binaries

Pre-build binaries are available for SL Linux (2.16) and Mac OS X (10.6.7 & 10.7.3) at the following download page: <http://amas.web.psi.ch/download/OPAL/>.

A.4 Enabling the Multigrid Space Charge Solver

Please note: The Multigrid space charge solver is not yet capable of emitting a beam from the cathode. You are advised to use the FFT space charge solver for studies with particle emission.

The following packages must be pre-installed: mkl-10.0-em64t, parmetis, SuperLUDist.

If using CMake you can enable the solver with

`cmake -D ENABLE_ML_SOLVER=TRUE $OPAL_ROOT` and make sure the `TRILINOS_INCLUDE_PATH` environment variable points to the directory containing the Trilinos header files.

If no Trilinos version (>10.6) is available, download and build the source code from the Trilinos webpage.⁴ The following Trilinos packages are required:

- epetra and epetraext
- ml and ml_parmetis3x
- amesos and amesos-superludist
- ifpack
- teuchos and teuchos-extended
- aztecco and aztecco-teuchos
- galeri
- belos

To enable these packages run `cmake` with the following arguments:

```
CC=mpicc
CXX=mpicxx
CPP="mpicxx -E"
F77=mpif77

cmake \
--prefix=/path/to/install \
-DCMAKE_INSTALL_PREFIX:PATH=/path/to/install \
-DCMAKE_CXX_FLAGS:STRING="-DMPICH_IGNORE_CXX_SEEK -fPIC" \
-DCMAKE_C_FLAGS:STRING="-DMPICH_IGNORE_CXX_SEEK -fPIC" \
-DCMAKE_Fortran_FLAGS:STRING="-fPIC" \
-D CMAKE_BUILD_TYPE:STRING=DEBUG \
-D TPL_ENABLE_SuperLUDist:BOOL=ON \
-D TPL_ENABLE_MPI:BOOL=ON \
-D TPL_ENABLE_BLAS:BOOL=ON \
-D TPL_ENABLE_LAPACK:BOOL=ON \
-D TPL_ENABLE_ParMETIS:BOOL=ON \
-D TPL_ENABLE_METIS:BOOL=OFF \
-D BLAS_LIBRARY_DIRS:PATH=${MKL_LIB_DIR} \
-D BLAS_INCLUDE_DIRS:PATH=${MKL_INCLUDE} \
-D BLAS_LIBRARY_NAMES:STRING="mkl_blas95_lp64; \
    mkl_intel_lp64;mkl_intel_thread;mkl_core;pthread;guide" \
```

⁴<http://trilinos.sandia.gov>

```

-D LAPACK_LIBRARY_DIRS:PATH=${MKL_LIB_DIR} \
-D LAPACK_INCLUDE_DIRS:PATH=${MKL_INCLUDE} \
-D LAPACK_LIBRARY_NAMES:STRING="mkl_lapack" \
-D ParMETIS_INCLUDE_DIRS:PATH="${PARMETIS_INCLUDE_PATH}" \
-D ParMETIS_LIBRARY_DIRS:PATH="${PARMETIS_LIBRARY_PATH}" \
-D ParMETIS_LIBRARY_NAMES:STRING="parmetis;metis" \
-D ParMETIS_LIBRARIES:STRING="parmetis;metis" \
-D TPL_BLACS_INCLUDE_DIRS:FILEPATH="/gpfs/homefelsim/kraus/include" \
-D BLACS_LIBRARY_DIRS:FILEPATH="${MKL_LIB_DIR}" \
-D TPL_BLACS_LIBRARIES="/opt/intel-mkl/mkl-10.2/lib/em64t/libmkl_blacs_lp64.a; \
/opt/intel-mkl/mkl-10.2/lib/em64t/libmkl_blacs_ilp64.a" \
-D SuperLUDist_INCLUDE_DIRS:FILEPATH= \
"/gpfs/homefelsim/kraus/extlib/SuperLU_DIST_2.5/include" \
-D SuperLUDist_LIBRARY_DIRS:FILEPATH= \
"/gpfs/homefelsim/kraus/extlib/SuperLU_DIST_2.5/lib" \
-D SuperLUDist_LIBRARY_NAMES:STRING="superlu_dist_2.5" \
-D SuperLUDist_LIBRARIES="superlu_dist_2.5" \
-D TPL_SuperLUDist_LIBRARIES= \
"/gpfs/homefelsim/kraus/extlib/SuperLU_DIST_2.5/lib/libsuperlu_dist_2.5.a" \
-D SCALAPACK_INCLUDE_DIRS:FILEPATH="${MKL_INCLUDE}" \
-D SCALAPACK_LIBRARY_DIRS:FILEPATH="${MKL_LIB_DIR}" \
-D SCALAPACK_LIBRARY_NAMES:STRING="mkl_scalapack_lp64" \
-D Trilinos_ENABLE_Belos:BOOL=ON \
-D Trilinos_ENABLE_Epetra:BOOL=ON \
-D Trilinos_ENABLE_EpetraExt:BOOL=ON \
-D Trilinos_ENABLE_Impack:BOOL=ON \
-D Trilinos_ENABLE_ML:BOOL=ON \
-D Trilinos_ENABLE_Amesos:BOOL=ON \
-D Amesos_ENABLE_BLACS:BOOL=ON \
-D Amesos_ENABLE_SuperLUDist:BOOL=ON \
-D Amesos_ENABLE_SCALAPACK:BOOL=ON \
-D Trilinos_ENABLE_Amesos-superlu:BOOL=ON \
-D Trilinos_ENABLE_AztecOO:BOOL=ON \
-D Trilinos_ENABLE_Teuchos:BOOL=ON \
-D Trilinos_ENABLE_AztecOO-Teuchos:BOOL=ON \
-D Trilinos_ENABLE_Teuchos-Extended:BOOL=ON \
-D Trilinos_ENABLE_Isorropia:BOOL=ON \
-D Trilinos_ENABLE_Isorropia-Epetraext:BOOL=ON \
-D Trilinos_ENABLE_Didasko:BOOL=OFF \
-D Didasko_ENABLE_TESTS=OFF \
-D Didasko_ENABLE_EXAMPLES=OFF \
-D Trilinos_ENABLE_TESTS:BOOL=OFF \
$EXTRA_ARGS \
$TRILINOS_ROOT

```

Finally execute the following configure command (i.e. on the FELSIM cluster)

```

CXX=mpicxx ./configure \
--with-classic-includedir=$CLASSIC_ROOT/src \
--with-classic-libdir=$CLASSIC_ROOT/src \
--with-doom-includedir=$DOOM_ROOT \
--with-doom-libdir=$DOOM_ROOT \
--with-ippl-includedir=$IPPL_ROOT/src \
--with-ippl-libdir=$IPPL_ROOT/lib/$IPPL_ARCH \
--with-h5part-includedir=$H5hut/src \
--with-h5part-libdir=$H5hut/src \
--with-hdf5-includedir=$HDF5HOME/include \
--with-hdf5-libdir=$HDF5HOME/lib \
--with-libdir="-L/opt/parmetis/parmetis-3.1 \
-L/opt/intel-mkl/mkl-10.0/lib/em64t \
-L/opt/intel/intel-10.0/fce-10.0/lib" \
--with-libs="-lsuperlu_dist_2.0 -lifcore \
-lparmetis -lmetis" \
--with-blas=mkl --with-lapack=mkl \

```

```
--with-trilinos-includedir=$TRILINOS_ROOT/include \
--with-trilinos-libdir=$TRILINOS_ROOT/lib \
--enable-ml-solver
```

A.5 Debug Flags

Table A.1: Debug flags.

| Name | Description | Default |
|-----------------|---|---------|
| DBG_SCALARFIELD | dumps scalar potential on the grid | not set |
| DBG_STENCIL | dumps stencil (MG solver) to a Matlab readable file | not set |
| DBG_CSR | dump information regarding the 1D CSR calculation | not set |

DBG_SCALARFIELD dumps the field to a file called rho_scalar. The structure of the data can be deduced from the following Matlab script:

```
function scalfield(RHO)

rhosize=size(RHO)
for i=1:rhosize(1)
    x = RHO(i,1);
    y = RHO(i,2);
    z = RHO(i,3);
    rhoxyz(y,z) = RHO(i,4);
    rhoxy(x,y) = RHO(i,4);
    rhoxz(x,z) = RHO(i,4);
    rho(x,y,z) = RHO(i,4);
end
```

DBG_STENCIL dumps the discretization stencil to a file (A.dat). The following Matlab code will read and store the sparse matrix in the variable 'A'.

```
load A.dat;
A = spconvert(A);
```

DBG_CSR dumps a text file of the calculated, 1D CSR field. The first line gives the average position of the beam bunch. Subsequent lines list z position of longitudinal mesh (with respect to the head of the beam bunch), E_z , line density and the derivative of the line density. Note that currently the line density derivative needs to be scaled by the inverse of the mesh spacing to get the correct value. The CSR field is dumped at each time step of the calculation. Each text file is named "Bend Name" (from input file) + "-CSRWake" + "time step number in that bend (starting from 1)" + ".txt".

A.6 OPAL as a Library

An OPAL library can be build by specifying `-DBUILD_LIBOPAL` in the cmake process. The OPAL libraey is currently used in the opt-pilot, a multi-objective optimization package [64].

A.7 Examples

When checking out the OPAL framework you will find the *opal-Tests* directory and moreover a subdirectory called *RegressionTests*. There several input files can be found which are run every day to check the validity of the current version of OPAL. This is a good starting-point to learn how to model accelerators with the various flavours of OPAL. More examples will be given in subsequent chapters, enjoy!

Appendix B

OPAL Language Syntax

Words in *italic font* are syntactic entities, and characters in `monospaced font` must be entered as shown. Comments are given in **bold font**.

Statements:

| | | |
|-----------------------|---|--|
| <i>comment</i> | : | <code>// anything-except-newline</code> |
| | | <code>/* anything-except-*/ */</code> |
| <i>identifier</i> | : | <code>[a-zA-Z][a-zA-Z0-9-]</code> |
| <i>integer</i> | : | <code>[0-9]+</code> |
| <i>string</i> | : | <code>' anything-except-single-quote'</code> |
| | | <code>"anything-except-double-quote"</code> |
| <i>command</i> | : | <code>keyword attribute-list</code> |
| | | <code>label : keyword attribute-list</code> |
| <i>keyword</i> | : | <i>identifier</i> |
| <i>label</i> | : | <i>identifier</i> |
| <i>attribute-list</i> | : | <i>empty</i> |
| | | <i>attribute-list , attribute</i> |

attribute : *attribute-name* // **only for logical attribute**
| *attribute-name* = *attribute-value*
| **// expression evaluated**
| *attribute-name* := *attribute-value*
| **// expression retained**

attribute-name : *identifier*

attribute-value : *string-expression*
| *logical-expression*
| *real-expression*
| *array-expression*
| *constraint*
| *variable-reference*
| *place*
| *range*
| *token-list*
| *token-list-array*
| *regular-expression*

Real expressions:

| | | |
|------------------------|---|--|
| <i>real-expression</i> | : | <i>real-term</i> |
| | | $+ \textit{real-term}$ |
| | | $- \textit{real-term}$ |
| | | $\textit{real-expression} + \textit{real-term}$ |
| | | $\textit{real-expression} - \textit{real-term}$ |
| <i>real-term</i> | : | <i>real-factor</i> |
| | | $\textit{real-term} * \textit{real-factor}$ |
| | | $\textit{real-term} / \textit{real-factor}$ |
| <i>real-factor</i> | : | <i>real-primary</i> |
| | | $\textit{real-factor} ^ \textit{real-primary}$ |
| <i>real-primary</i> | : | <i>real-literal</i> |
| | | <i>symbolic-constant</i> |
| | | $\#$ |
| | | <i>real-name</i> |
| | | $\textit{array} \ [\ \textit{index} \]$ |
| | | $\textit{object-name} \rightarrow \textit{real-attribute}$ |
| | | $\textit{object-name} \rightarrow \textit{array-attribute} \ [\ \textit{index} \]$ |
| | | <i>table-reference</i> |
| | | $\textit{real-function} \ (\)$ |
| | | $\textit{real-function} \ (\ \textit{real-expression} \)$ |
| | | $\textit{real-function} \ (\ \textit{real-expression} \ , \ \textit{real-expression} \)$ |
| | | $\textit{function-of-array} \ (\ \textit{array-expression} \)$ |
| | | $(\ \textit{real-expression} \)$ |

| | | |
|----------------------|---|--------|
| <i>real-function</i> | : | RANF |
| | | GAUSS |
| | | USER0 |
| | | SI |
| | | SC |
| | | SO |
| | | ABS |
| | | TRUNC |
| | | ROUND |
| | | FLOOR |
| | | CEIL |
| | | SIGN |
| | | SQRT |
| | | LOG |
| | | EXP |
| | | SIN |
| | | COS |
| | | ABS |
| | | TAN |
| | | ASIN |
| | | ACOS |
| | | ATAN |
| | | TGAUSS |
| | | USER1 |
| | | ATAN2 |
| | | MAX |
| | | MIN |
| | | MOD |
| | | POW |

function-of-array : VMIN
 | VMAX
 | VRMS
 | VABSMAX

Real variables and constants:

real-prefix : empty
 | REAL
 | REAL CONST
 | CONST

real-definition : *real-prefix real-name = real-expression*
// expression evaluated
 | *real-prefix real-name := real-expression*
// expression retained

symbolic-constant : PI
 | TWOPI
 | DEGRAD
 | RADDEG
 | E
 | EMASS
 | PMASS
 | CLIGHT
 | *real-name*

real-name : *identifier*

object-name : *identifier*

table-reference : *table-name @ place -> column-name*

table-name : *identifier*

column-name : *identifier*

Logical expressions:

logical-expression : *and-expression*
| *logical-expression* || *and-expression*

and-expression : *relation*
| *and-expression* && *relation*

relation : *logical-name*
| TRUE
| FALSE
| *real-expression* *relation-operator* *real-expression*

logical-name : *identifier*

relation-operator : ==
| !=
| <
| >
| >=
| <=

Logical variables:

```

logical-prefix          :  BOOL
                           |  BOOL CONST

logical-definition     :  logical-prefix logical-name = logical-expression
                           // expression evaluated
                           |  logical-prefix logical-name := logical-expression
                           // expression retained

```

String expressions:

```

string-expression      :  string
                           |  identifier // taken as a string
                           |  string-expression & string

```

String constants:

```

string-prefix          :  STRING

string-definition     :  string-prefix string-name = string-expression
                           // expression evaluated
                           |  string-prefix string-name := string-expression
                           // expression retained

```

Real array expressions:

| | | |
|-------------------------|---|---|
| <i>array-expression</i> | : | <i>array-term</i> |
| | | + <i>array-term</i> |
| | | – <i>array-term</i> |
| | | <i>array-expression</i> + <i>array-term</i> |
| | | <i>array-expression</i> – <i>array-term</i> |
| <i>array-term</i> | : | <i>array-factor</i> |
| | | <i>array-term</i> * <i>array-factor</i> |
| | | <i>array-term</i> / <i>array-factor</i> |
| <i>array-factor</i> | : | <i>array-primary</i> |
| | | <i>array-factor</i> ^ <i>array-primary</i> |
| <i>array-primary</i> | : | { <i>array-literal</i> } |
| | | <i>array-reference</i> |
| | | <i>table-generator</i> |
| | | <i>row-reference</i> |
| | | <i>column-reference</i> |
| | | <i>real-function</i> (<i>array-expression</i>) |
| | | (<i>array-expression</i>) |
| <i>table-generator</i> | | TABLE (<i>last</i> , <i>real-expression</i>) |
| | : | TABLE (<i>first</i> : <i>last</i> , <i>real-expression</i>) |
| | : | TABLE (<i>first</i> : <i>last</i> : <i>step</i> , <i>real-expression</i>) |
| <i>first</i> | : | <i>integer</i> |
| <i>last</i> | : | <i>integer</i> |
| <i>step</i> | : | <i>integer</i> |
| <i>table-row</i> | : | <i>table-name</i> @ <i>place</i> |

| | | |
|-------------------------|---|---|
| <i>row-reference</i> | : | ROW (<i>table-name</i> , <i>place</i>) |
| | | ROW (<i>table-name</i> , <i>place</i> , { <i>column-list</i> }) |
| <i>column-reference</i> | : | COLUMN (<i>table-name</i> , <i>column-name</i>) |
| | | COLUMN (<i>table-name</i> , <i>column-name</i> , <i>range</i>) |
| <i>column-list</i> | : | <i>column-name</i> |
| | | <i>column-list</i> , <i>column-name</i> |
| <i>array-literal</i> | : | <i>real-expression</i> |
| | | <i>array-literal</i> , <i>real expression</i> |
| <i>array-reference</i> | : | <i>array-name</i> |
| | | <i>object-name</i> -> <i>array-attribute</i> |
| <i>array-name</i> | : | <i>identifier</i> |

Real array definitions:

| | | |
|-------------------------|---|--|
| <i>array-prefix</i> | : | REAL VECTOR |
| <i>array-definition</i> | : | <i>array-prefix</i> <i>array-name</i> = <i>array-expression</i> |
| | | <i>array-prefix</i> <i>array-name</i> := <i>array-expression</i> |

Constraints:

| | | |
|----------------------------|---|--|
| <i>constraint</i> | : | <i>array-expression</i> <i>constraint-operator</i> <i>array-expression</i> |
| <i>constraint-operator</i> | : | == |
| | | < |
| | | > |

Variable references:

| | | |
|---------------------------|---|---|
| <i>variable-reference</i> | : | <i>real-name</i> |
| | | <i>object-name</i> -> <i>attribute-name</i> |

Places:

place : *element-name*
| *element-name* [*integer*]
| #S
| #E
| #integer
| *line-name* :: *place*

Ranges:

range : *place*
| *place* / *place*

Token lists:

token-list : *anything-except-comma*

token-list-array : *token-list*
| *token-list-array* , *token-list*

Regular expressions:

regular-expression : "UNIX-regular-expression"

Appendix C

OPAL-T Field Maps

C.1 Introduction

In this chapter details of the different types of field maps in OPAL-T are presented. The possibility to add comments (almost) everywhere in the files is common to all field maps. Comments are initiated by a # and contain the rest of a line. Comments are accepted at the beginning of the file, between the lines and at the end of a line. If in the following sections two values are shown on one line then they have to be on the same line. They should not be separated by a comment and, consequently, be on different lines. Three examples of valid comments:

```
# This is valid a comment
1DMagnetoStatic 40 # This is an other valid comment
-60.0 60.0 9999
  # and this is also a valid comment
0.0 2.0 199
```

The following examples will break the parsing of the field maps:

```
1DMagnetoStatic # This is an invalid comment
40
-60.0 60.0 # This is an other invalid comment # 9999
0.0 2.0 199
```

If OPAL-T encounters an error while parsing a field map it disables the corresponding element, outputs a warning message and continues the simulation. The following messages may be output:

```
***** W A R N I N G *****
THERE SEEMS TO BE SOMETHING WRONG WITH YOUR FIELD MAP file.t7.
There are only 10003 lines in the file, expecting more.
Please check the section about field maps in the user manual.
*****
```

In this example there is something wrong with the number of grid spacings provided in the header of the file. Make sure that you provide the number of grid **spacings** and not the number of grid **points**! The two numbers always differ by 1.

```
***** W A R N I N G *****
THERE SEEMS TO BE SOMETHING WRONG WITH YOUR FIELD MAP file.t7.
There are too many lines in the file, expecting only 10003 lines.
Please check the section about field maps in the user manual.
*****
```

| | |
|----------------------------------|---|
| Didn't find enough values! | If OPAL-T expects more values on this line. |
| Found more values than expected! | If OPAL-T expects less values on this line. |
| Found wrong type of values! | If OPAL-T found e.g. characters instead of an integer number. |

Again there seems to be something wrong with the number of grid spacings provided in the header. In this example OPAL-T found more lines than it expected. Note that comments and empty lines at the end of a file are ignored such that **they don't cause** this warning.

```
***** W A R N I N G *****
THERE SEEMS TO BE SOMETHING WRONG WITH YOUR FIELD MAP file.t7.
_error_msg_
expecting: '_expecting_' on line 3,
found instead: '_found_'.
*****
```

Where `_error_msg_` is either `'_expecting_'` is replaced by the types of values OPAL-T expects on the line. E.g. it could be replaced by `'double double int'`. Finally `'_found_'` is replaced by the actual content of the line without any comment possibly following the values. If line 3 of a file consists of `'-60.0 60.0 # This is an other invalid comment # 9999'` OPAL-T will output `'-60.0 60.0'`.

```
***** W A R N I N G *****
DISABLING FIELD MAP file.t7 SINCE FILE COULDN'T BE FOUND!
*****
```

This warning could be issued if the filename is mistyped or otherwise if the file couldn't be read.

```
***** W A R N I N G *****
THERE SEEMS TO BE SOMETHING WRONG WITH YOUR FIELD MAP file.t7.
Could not determine the file type.
Please check the section about field maps in the user manual.
*****
```

In this case OPAL-T didn't recognise the string of characters which identify the type of field map stored in the file.

C.2 Types and Format

The file format used for the field maps is derived from the T7 file format as produced by Superfish [43] but also ASTRA type of field maps are supported. A valid field map consists of a few header lines and a rest representing regularly spaced interpolation points either in 1D, 2D or 3D. ASTRA field maps consist of one or two header lines (not supported by ASTRA) and possibly non-equidistant interpolation points. In the case of 2D and 3D field maps the fields at a given position are calculated by linear interpolation from the nearest grid points. In the case of 1D field maps a Fourier transformation is used to calculate the longitudinal derivatives of the on-axis field. From these the fields at any position in space are calculated.

To calculate the off-axis field the 1st, 2nd and 3rd derivative have to be computed. OPAL-T uses a Fourier transformation in conjunction with a low pass filter (only a number of low frequency Fourier coefficients are kept). In order to apply a Fast Fourier Transformation the data has to be equidistant. To this end the sampling

values of an ASTRA field map are resampled using a cubic spline interpolation. The number of sampling points is preserved in this process. From the resulting sampling the Fourier coefficients are calculated.

It is important to note that when a field map is read into OPAL-T, it is normalized so that the peak field value is equal to either 1 MV/m, in the case of electric field maps, or 1 T, in the case of magnetic field maps. So, when using a field map in an accelerating cell, for instance, the peak field in the simulation will be equal to the field scaling factor you specify in your input file.

At the beginning of the first line information of the kind of field map has to be provided in form of a string. This can either be:

1DElectroStatic

1D electrostatic field map. 1D field maps are described by the on-axis field.

Not implemented yet, use a 1DDynamic field map with very low frequency instead.

1DMagnetoStatic

if the file describes a 1D magnetostatic field map.

AstraMagnetoStatic

if the file describes a 1D magnetostatic field map with possibly non-equidistant sampling. This file type is compatible with ASTRA field maps with small changes.

1DDynamic

if the file describes a 1D dynamic electromagnetic field map.

AstraDynamic

if the file describes a 1D dynamic electromagnetic field map with possibly non-equidistant sampling. This file type is compatible with ASTRA field maps with small changes.

1DProfile1

if the file contains Enge coefficients (see [44]) which describe the fringe field of an element. From these the correct field at any position is calculated. This type of field map is special in the sense that the class processing these files doesn't return the actual field at a position but rather the on-axis field profile and its first and second derivatives. The classes (elements) supporting this kind of field map have to deal with this appropriately. At the moment only the rectangular bend (RBEND) and the sector bend (SBEND) elements in OPAL-t can use this type of field file.

1DProfile2

if the file describes a mid plane on-axis field profile which is processed to get the corresponding Enge coefficients. Otherwise this type is the same as 1DProfile1.

2DElectroStatic

if the file describes a 2D electrostatic field map. 2D field maps are described by the electromagnetic field in one half-plane.

2DMagnetoStatic

if the file describes a 2D magnetostatic field map.

2DDynamic

if the file describes a 2D dynamic electromagnetic field map.

3DElectroStatic

if the file describes a 3D electrostatic field map.

Not implemented yet.

3DMagnetoStatic

if the file describes a 3D magnetostatic field map.

Not implemented yet.

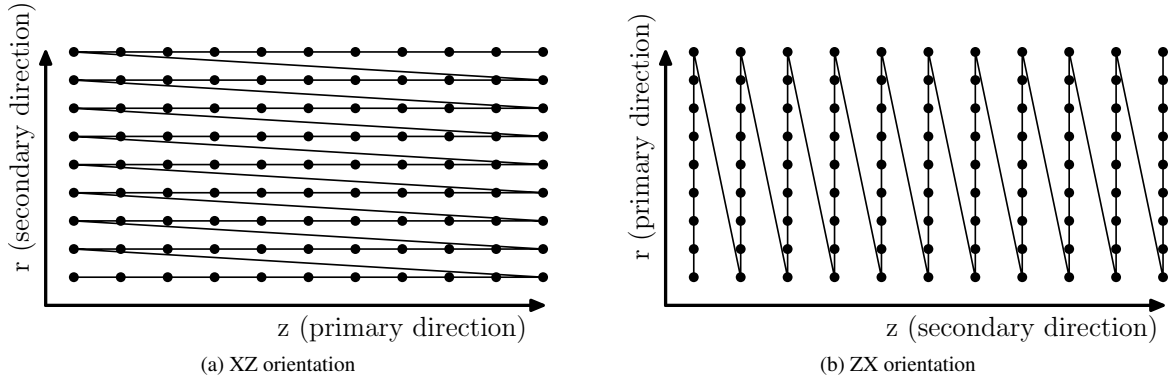


Figure C.1: Ordering of points for 2D field maps in T7 files

3DDynamic

if the file describes a 3D dynamic electromagnetic field map.

In the case of the 1DDynamic, 1DMagnetoStatic, 1DElectroStatic, AstraMagnetostatic and the AstraDynamic field maps one finds in addition one integer number on the first line of the file describing the number of Fourier coefficients to be used in the calculation of the derivative of the on-axis field.

In the case of 2D and 3D field maps an additional string has to be provided describing the orientation of the field map.

For 2D field maps this can either be

XZ

if the primary direction is in z direction and the secondary in r direction.

ZX

if the primary direction is in r direction and the secondary in z direction.

For 3D field maps this can be

XYZ

if the primary direction is in z direction, the secondary in x direction and the tertiary in y direction

Each line after the header corresponds to a grid point of the field map. This point can be referred to by two indices in the case of a 2D field map and three indices in the case of a 3D field map respectively. Each column describes either E_z , E_r , B_z , B_r or H_t in the 2D case and E_x , E_y , E_z , B_x , B_y , or B_z in the 3D case.

By primary, secondary and tertiary direction is meant the following (see also Figure C.1a and Figure C.1b):

- the index of the primary direction increases the fastest, the index of the tertiary direction the slowest.
- the order of the columns is accordingly: if the z direction in an electrostatic field map is the primary direction then E_z is on the first column, E_r on the second. For all other cases it's analogous.

For the 2D dynamic case in XZ orientation there are four columns: E_z , E_r , an unused column and H_t in this order. In the other orientation the first and the second column and the third and fourth column are interchanged.

On the second line of the header of a 1D, 2D or 3D T7 type of field map the beginning and the end of the electromagnetic field relative to the physical element in primary direction is written (in centimeters!). Also written on the second line is the number of **mesh spacings**, corresponding to the **number of grid points minus 1**. For the ASTRA compatible field maps this line is omitted.

On the third line is the frequency. For static cases this line is omitted. The frequency is on the second line of a AstraDynamic field map.

The fourth line corresponds to the second line but in secondary direction and the fifth accordingly for the tertiary direction. In the case of a 1D or 2D the fifth line is omitted since there is no tertiary direction. Those two lines are omitted in the ASTRA compatible field maps.

On the sixth line follows the first line with field values as described above.

Even though there is no secondary direction in the 1D case the header of a 1D field map is equal to its 2D equivalent: the elements can be provided with a boolean attribute FAST which has only an effect in conjunction with a 1D field map. The code then generates internally a 2D field map and the field strengths are interpolated as in the case of 2D and 3D field maps instead of being calculated using the Fourier coefficients. The fourth line determines the transverse dimension and the grain size of the produced mesh.

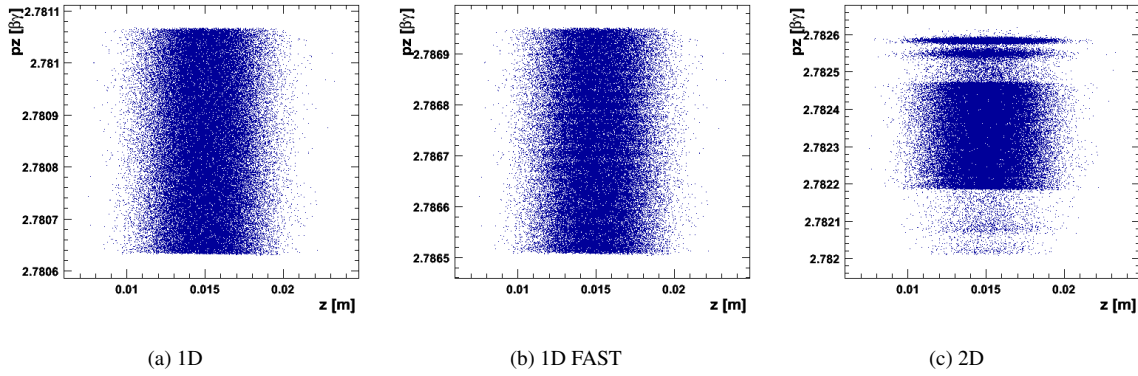


Figure C.2: The longitudinal phase space after a gun simulation using a 1D field map (on-axis field) of the gun, a 1D field map (on-axis field) of the gun in combination with the FAST switch, and a 2D field map of the gun generated by Poisson/Superfish.

⚠ As a general warning: be wise when you choose the type of field map to be used! The following three pictures show the longitudinal phase space after three gun simulations using different types of field maps. In the first picture, Figure C.2a, we used a 1D field map which stores a sampling of the electric field in longitudinal direction, E_z . From these values E_z , E_r and B_t off-axis are calculated resulting in a smooth field. All field maps we used are made of the same solution file from a Poisson/Superfish simulation.

In Figure C.2b we used the same on-axis field map as in the first but we used the FAST switch which constructs a 2D field map from the on-axis field maps. Between the grid points the fields are calculated using a linear interpolation. Here we see a structure which can be influence using more grid points in transverse direction.

In the last picture, Figure C.2c, we generated directly a 2D field map from the solution file of Poisson/Superfish. Here we could observe two different structures: first the fine structure stemming from the linear interpolation and secondly a much stronger structure of unknown origin.

C.3 1DMagnetoStatic

```

1DMagnetoStatic 40
-60.0 60.0 9999
0.0 2.0 199
  0.000000e+00
  4.36222e-06
  8.83270e-06
+ 9'994 lines
  1.32490e-05
  1.73710e-05
  2.18598e-05

```

Figure C.3: A 1D field map describing a magnetostatic field using 10'000 grid points (9'999 grid spacings) in longitudinal direction. If the FAST switch is set in the input deck 200 values in transvers direction for each longitudinal grid point are calculated. The field is non-negligible from -60.0 cm to $+60.0\text{ cm}$ relative to ELEMEDGE in longitudinal direction. The 200 grid points span a length of 2.0 cm in radial direction. From the 10'000 field values 5'000 complex Fourier coefficients are calculated whereof only 40 are kept to calculate the off-axis field values. OPAL-Tnormalizes the field values internally such that $\max(B_{\text{onaxis}}) = 1.0\text{ T}$.

C.4 AstraMagnetostatic

```

AstraMagnetostatic 40
-3.0000000e-01 0.0000000e+00
-2.9800000e-01 2.9075045e-05
-2.9600000e-01 5.9367702e-05
-2.9400000e-01 9.0866460e-05
-2.9200000e-01 1.2374798e-04
-2.9000000e-01 1.5799850e-04
....
2.9000000e-01 1.5799850e-04
2.9200000e-01 1.2374798e-04
2.9400000e-01 9.0866460e-05
2.9600000e-01 5.9367702e-05
2.9800000e-01 2.9075045e-05
3.0000000e-01 0.0000000e+00

```

Figure C.4: A 1D field map describing a magnetostatic field using n non-equidistant grid points in longitudinal direction. From these values n equidistant field values are computed from which in turn $n/2$ complex Fourier coefficients are calculated. In this example only 40 Fourier coefficients are kept to calculate the off-axis field values. The z-position of the sampling is in the 1st column (in meters), the corresponding longitudinal on-axis magnetic field amplitude is in the 2nd column. As with the 1DMagnetoStatic field maps, OPAL-T normalizes the field values to $\max(B_{\text{onaxis}}) = 1.0$ T. In the header only the first line is headed since the information on the longitudinal dimension is contained in the first column. Furthermore OPAL-T does not provide a FAST version of this map type.

C.5 1DDynamic

```
1DDynamic 40
-3.0 57.0 4999
1498.953425154
0.0 2.0 199
  0.000000e+00
  4.36222e-06
  8.83270e-06
+ 4'994 lines
  1.32490e-05
  1.73710e-05
  2.18598e-05
```

Figure C.5: A 1D field map describing a dynamic field using 5'000 grid points in longitudinal direction. If the FAST switch is set in the input deck 200 values in transvers direction for each longitudinal grid point are calculated. The field is non-negligible from -3.0 cm to 57.0 cm relative to ELEMEDGE in longitudinal direction. The 200 grid points span a length of 2.0 cm in radial direction. From the 5'000 field values 2'500 complex Fourier coefficients are calculated whereof only 40 are kept to calculate the off-axis field values.

C.6 AstraDynamic

```

AstraDynamic 40
2997.924
  0.00000000e+00    0.00000000e+00
  5.0007941e-04    2.8090000e-04
  9.9991114e-04    5.6553000e-04
  1.4996762e-03    8.4103000e-04
  . . . .
  1.9741957e-01    1.4295000e-03
  1.9792448e-01    1.1306000e-03
  1.9841987e-01    8.4103000e-04
  1.9891525e-01    5.6553000e-04
  1.9942016e-01    2.8090000e-04
  1.9991554e-01    0.0000000e+00

```

Figure C.6: A 1D field map describing a dynamic field using n non-equidistant grid points in longitudinal direction. From the n non-equidistant field values n equidistant field values are computed from which in turn $n/2$ complex Fourier coefficients are calculated. In this example only 40 Fourier coefficients are kept to calculate the off-axis field values. The z -position is in the 1st column (in meters), the corresponding longitudinal on-axis electric field amplitude is in the 2nd column. OPAL-T normalizes the field values such that $\max(E_{\text{onaxis}}) = 1$ MV/m. In the header only the first and the third line of a corresponding 1DDynamic field map is needed since the information on the longitudinal dimension is contained in the first column. OPAL-T does not provide a FAST version of this map type.

C.7 1DProfile1 & 1DProfile2

The field maps **1DProfile1** and **1DProfile2** are different from the rest since no actual fields are stored but a profile. They are used to represent the fringe fields of various elements. The actual fields used in an OPAL-Tsimulation are calculated using Enge functions [44]. In turn, the Enge functions are declared in two different ways. A **1DProfile1** field map gives the coefficients of the Enge function explicitly. A **1DProfile2** field map stores the profile itself. From this profile the Enge coefficients are calculated by solving a least-square equation.

On the first line after the string describing the type of field map two integer numbers and a floating point number follow. The first integer number describes the number of Enge coefficients to be used for the entry fringe field, the second the equivalent for the exit fringe field. The floating point number specifies the full gap height. On the second line the first value describes the beginning of the entry fringe field (in local coordinates; corresponds to *zbegin_entry* in Figure C.7 and Figure C.8), the second the origin of the Enge polynomial, the third the end of the fringe field (*zend_entry*). The last number of this line is only used in **1DProfile2** and specifies the number of grid points minus 1 of the field profile. The third line looks identical to the second, but the last value on this line is not used yet. The values on this line correspond to *zbegin_exit*, *origin of Enge function* and *zend_exit* in Figure C.7 and Figure C.8. The following lines are the coefficients of the Enge function in the case of **1DProfile1** and the actual field profile in **1DProfile2**, see appendix C.1.

Note that *zbegin_entry*, *zend_entry*, *zbegin_exit* and *zend_exit* have slightly different definitions when defining a rectangular bend as opposed to a sector bend (Figure C.7 and Figure C.8).

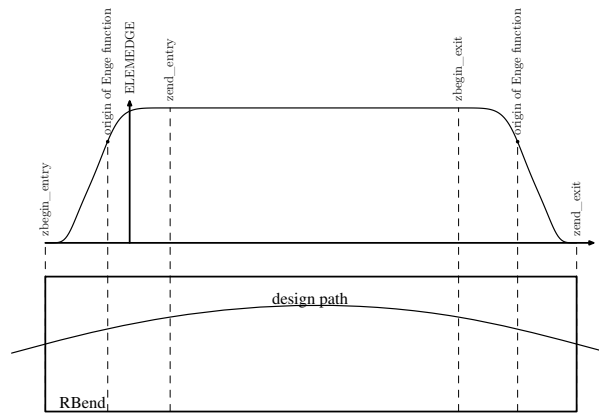


Figure C.7: The profile of a rectangular bend and its corresponding design path.

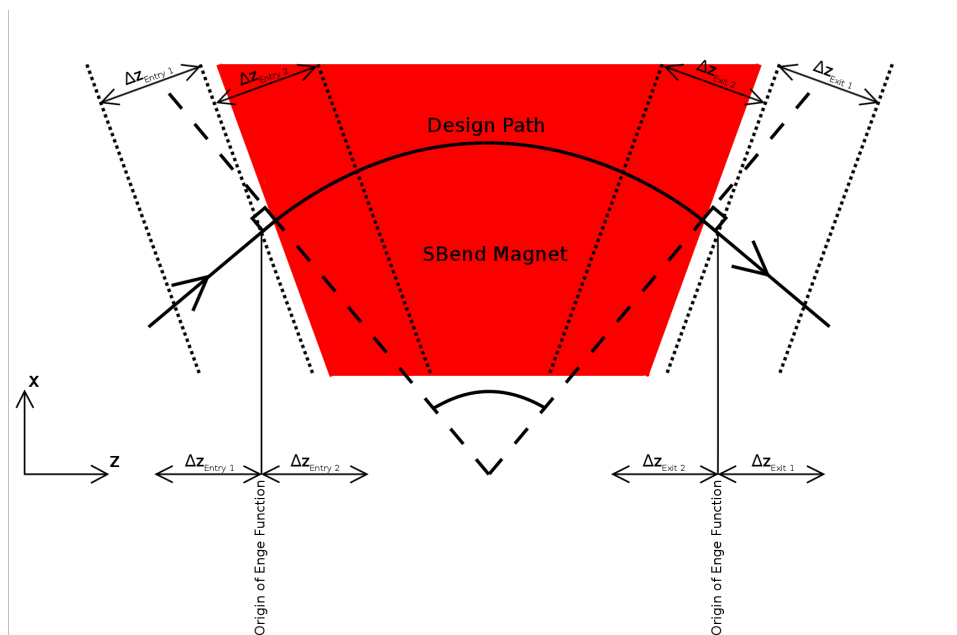


Figure C.8: FIXME The location and definitions of the Enge function coefficients for a sector bend and its corresponding design path.

```

1DProfile1 6 7 3.0
-6.0 -2.0 2.0 1000
24.0 28.0 32.0 0
0.00000e+00
4.36222e-06
8.83270e-06
+ 9 lines
1.32490e-05
1.73710e-05
2.18598e-05

```

Figure C.9: A 1D field map describing the fringe field of an element using 7 Enge coefficients for the entrance fringe field and 8 Enge coefficients for the exit fringe field (polynomial order 6 and 7 respectively). The element has a gap height of 3.0 cm, the entrance fringe field is non-negligible from -6.0 cm and reaches the core strength at 2.0 cm relative to ELEMEDGE. The origin of the Enge function for the entrance fringe field is at -2.0 cm relative to ELEMEDGE. The exit fringe field is non-negligible up to 32.0 cm from ELEMEDGE and starts to deviate from the core strength after 24.0 cm from ELEMEDGE. The origin of the Enge function for the exit fringe field is at 28.0 cm from ELEMEDGE. The values 1000 in line 2 and 0 in line 3 do not have any meaning.

```

1DProfile2 6 7 3.0
-6.0 -2.0 2.0 1000
24.0 28.0 32.0 0
0.000000e+00
4.36222e-06
8.83270e-06
+ 995 lines
1.32490e-05
1.73710e-05
2.18598e-05

```

Figure C.10: A 1D field map describing the fringe field of an element. The file provides field values at 1001 equidistant sampling points. To calculate the field between the grid points a Enge function is fitted to these values. To calculate the field the corresponding Enge coefficients, 7 for the entrance fringe field and 8 for the exit fringe field (polynomial order 6 and 7 respectively). The element has a gap height of 3.0 cm. The length of the entrance and the exit fringe field are determined by the field values and the total length, $32.0 \text{ cm} - (-6.0) \text{ cm} = 38.0 \text{ cm}$.

C.8 2DElectroStatic

```

2DElectroStatic XZ
-3.0 51.0 4999
0.0 2.0 199
  0.000000e+00  0.000000e+00
  4.36222e-06  0.000000e+00
  8.83270e-06  0.000000e+00
+ 999'994 lines
  1.32490e-05  0.000000e+00
  1.73710e-05  0.000000e+00
  2.18598e-05  0.000000e+00

```

Figure C.11: A 2D field map describing an electrostatic field using 5'000 grid points in longitudinal direction times 200 grid points in transvers direction. The field between the grid points is calculated with a bilinear interpolation. The field is non-negligible from -3.0 cm to 51.0 cm relative to ELEMEDGE and the 200 grid points in transverse direction span a length of 2.0 cm . The field values are ordered in XZ orientation, the index in longitudinal direction changes fastest, on the first column the E_z values are stored, on the second the E_r values.

C.9 2DMagnetoStatic

```

2DMagnetoStatic ZX
0.0 2.0 199
-3.0 51.0 4999
  0.000000e+00  0.000000e+00
  0.000000e+00  4.36222e-06
  0.000000e+00  8.83270e-06
+ 999'994 lines
  0.000000e+00  1.32490e-05
  0.000000e+00  1.73710e-05
  0.000000e+00  2.18598e-05

```

Figure C.12: A 2D field map describing a magnetostatic field using 5'000 grid points in longitudinal direction times 200 grid points in transvers direction. The field between the grid points is calculated with a bilinear interpolation. The field is non-negligible from -3.0 cm to 51.0 cm relative to ELEMEDGE and the 200 grid points in transverse direction span a length of 2.0 cm . The field values are ordered in ZX orientation, the index in transvers direction changes fastest, on the first column the B_r values are stored, on the second the B_z values.

C.10 2DDynamic

```

2DDynamic XZ
-3.0 51.0 4121
1498.953425154
0.0 1.0 75
  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
  4.36222e-06  0.000000e+00  0.000000e+00  4.36222e-06
  8.83270e-06  0.000000e+00  0.000000e+00  8.83270e-06
+ 313'266 lines
  1.32490e-05  0.000000e+00  0.000000e+00  1.32490e-05
  1.73710e-05  0.000000e+00  0.000000e+00  1.73710e-05
  2.18598e-05  0.000000e+00  0.000000e+00  2.18598e-05

```

Figure C.13: A 2D field map describing a dynamic field oscillating with $1.498953425154\text{ GHz}$. The field map provides 4122 grid points in longitudinal direction times 76 grid points in transvers direction. The field between the grid points is calculated with a bilinear interpolation. The field is non-negligible between -3.0 cm and 51.0 cm relative to ELEMEDGE and the 76 grid points in transvers direction span a distance of 1.0 cm . The field values are ordered in XZ orientation, the index in longitudinal direction changes fastest, on the first column the E_z values are stored, on the second the E_r values, on the fourth the B_t values and the third column contains dummy values. For the ZX orientation the first column would contain the E_r values, the second the E_z values, the third column the B_t values and the fourth column dummy values. In addition the ordering of the values would be such that the index in transvers direction changes fastest and the second and fourth line of the file would be interchanged.

C.11 3DDynamic

```

3DDynamic XYZ
1498.953425154
-1.5 1.5 227
-1.0 1.0 151
-3.0 51.0 4121
0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00 0.00e+00
4.36e-06 0.00e+00 4.36e-06 0.00e+00 4.36e-06 0.00e+00
8.83e-06 0.00e+00 8.83e-06 0.00e+00 8.83e-06 0.00e+00
+ 142'852'026 lines
1.32e-05 0.00e+00 1.32e-05 0.00e+00 1.32e-05 0.00e+00
1.73e-05 0.00e+00 1.73e-05 0.00e+00 1.73e-05 0.00e+00
2.18e-05 0.00e+00 2.18e-05 0.00e+00 2.18e-05 0.00e+00

```

Figure C.14: A 3D field map describing a dynamic field oscillating with $1.498953425154\text{ GHz}$. The field map provides 4122 grid points in z-direction times 228 grid points in x-direction and 152 grid points in y-direction. The field between the grid points is calculated with a bilinear interpolation. The field is non-negligible between -3.0 cm to 51.0 cm relative to ELEMEDGE, the 228 grid points in x-direction range from -1.5 cm to 1.5 cm and the 152 grid points in y-direction range from -1.0 cm to 1.0 cm relative to the design path. The field values are ordered in XYZ orientation, the index in z-direction changes fastest, then the index in y-direction while the index in x-direction changes slowest. This is the only orientation that is implemented. The columns correspond to E_x , E_y , E_z , B_x , B_y and B_z .

Bibliography

- [1] *The Graphical Kernel System (GKS)*. ISO, Geneva, July 1985. International Standard ISO 7942.
- [2] B. Autin and Y. Marti. *Closed Orbit Correction of Alternating Gradient Machines using a small Number of Magnets*. CERN/ISR-MA/73-17, CERN, 1973.
- [3] D.P. Barber, K. Heinemann, H. Mais and G. Ripken, *A Fokker–Planck Treatment of Stochastic Particle Motion within the Framework of a Fully Coupled 6-dimensional Formalism for Electron-Positron Storage Rings including Classical Spin Motion in Linear Approximation*, DESY report 91-146, 1991.
- [4] R. Bartolini, A. Bazzani, M. Giovannozzi, W. Scandale and E. Todesco, *Tune evaluation in simulations and experiments*, CERN SL/95-84 (AP) (1995).
- [5] J. D. Bjorken and S. K. Mtingwa. *Particle Accelerators* **13**, pg. 115.
- [6] E.M. Bollt and J.D. Meiss, *Targeting chaotic orbits to the Moon through recurrence*, Phys. Lett. **A204**, 373 (1995).
- [7] P. Bramham and H. Henke. private communication and LEP Note LEP-70/107, CERN.
- [8] Karl L. Brown. *A First-and Second-Order Matrix Theory for the Design of Beam Transport Systems and Charged Particle Spectrometers*. SLAC 75, Revision 3, SLAC, 1972.
- [9] Karl L. Brown, D. C. Carey, Ch. Iselin, and F. Rothacker. *TRANSPORT — A Computer Program for Designing Charged Particle Beam Transport Systems*. CERN 73-16, revised as CERN 80-4, CERN, 1980.
- [10] A. Chao. *Evaluation of beam distribution parameters in an electron storage ring*. Journal of Applied Physics, 50:595–598, 1979.
- [11] A. W. Chao and M. J. Lee. *SPEAR II Touschek lifetime*. SPEAR-181, SLAC, October 1974.
- [12] M. Conte and M. Martini. *Particle Accelerators* **17**, 1 (1985).
- [13] E. D. Courant and H. S. Snyder. *Theory of the alternating gradient synchrotron*. Annals of Physics, 3:1–48, 1958.
- [14] Ph. Defert, Ph. Hofmann, and R. Keyser. *The Table File System, the C Interfaces*. LAW Note 9, CERN, 1989.
- [15] M. Donald and D. Schofield. *A User’s Guide to the HARMON Program*. LEP Note 420, CERN, 1982.
- [16] A. Dragt. *Lectures on Nonlinear Orbit Dynamics, 1981 Summer School on High Energy Particle Accelerators, Fermi National Accelerator Laboratory, July 1981*. American Institute of Physics, 1982.
- [17] D. A. Edwards and L. C. Teng. *Parametrisation of linear coupled motion in periodic systems*. IEEE Trans. on Nucl. Sc., 20:885, 1973.

- [18] M. Giovannozzi, *Analysis of the stability domain of planar symplectic maps using invariant manifolds*, CERN/PS 96-05 (PA) (1996).
- [19] H. Grote. *GXPLOT User's Guide and Reference Manual*. LEP TH Note 57, CERN, 1988.
- [20] LEP Design Group. *Design Study of a 22 to 130 GeV e^+e^- Colliding Beam Machine (LEP)*. CERN/ISR-LEP/79-33, CERN, 1979.
- [21] M. Hanney, J. M. Jowett, and E. Keil. *BEAMPARAM — A program for computing beam dynamics and performance of e^+e^- storage rings*. CERN/LEP-TH/88-2, CERN, 1988.
- [22] R. H. Helm, M. J. Lee, P. L. Morton, and M. Sands. *Evaluation of synchrotron radiation integrals*. IEEE Trans. Nucl. Sc., NS-20, 1973.
- [23] F. James. *MINUIT, A package of programs to minimise a function of n variables, compute the covariance matrix, and find the true errors*. program library code D507, CERN, 1978.
- [24] E. Keil. *Synchrotron radiation from a large electron-positron storage ring*. CERN/ISR-LTD/76-23, CERN, 1976.
- [25] D. E. Knuth. *The Art of Computer Programming*. Volume 2, Addison-Wesley, second edition, 1981. Semi-numerical Algorithms.
- [26] J. Laskar, C. Froeschlé and A. Celletti, *The measure of chaos by the numerical analysis of the fundamental frequencies. Application to the standard mapping*, Physica D **56**, 253 (1992).
- [27] H. Mais and G. Ripken, *Theory of Coupled Synchro-Betatron Oscillations*. DESY internal Report, DESY M-82-05, 1982.
- [28] M. Meddahi, *Chromaticity correction for the $108^\circ/60^\circ$ lattice*, CERN SL/Note 96-19 (AP) (1996).
- [29] J. Milutinovic and S. Ruggiero. *Comparison of Accelerator Codes for a RHIC Lattice*. AD/AP/TN-9, BNL, 1988.
- [30] B. W. Montague. *Linear Optics for Improved Chromaticity Correction*. LEP Note 165, CERN, 1979.
- [31] G. Ripken, *Untersuchungen zur Strahlführung und Stabilität der Teilchenbewegung in Beschleunigern und Storage-Ringen unter strenger Berücksichtigung einer Kopplung der Betatronschwingungen*. DESY internal Report R1-70/4, 1970.
- [32] F. Ruggiero, *Dynamic Aperture for LEP 2 with various optics and tunes*, Proc. Sixth Workshop on LEP Performance, Chamonix, 1996, ed. J. Poole (CERN SL/96-05 (DI), 1996), pp. 132–136.
- [33] L. C. Teng. *Concerning n -Dimensional Coupled Motion*. FN 229, FNAL, 1971.
- [34] U. Völkel. *Particle loss by Touschek effect in a storage ring*. DESY 67-5, DESY, 1967.
- [35] R. P. Walker. *Calculation of the Touschek lifetime in electron storage rings*. 1987. Also SERC Daresbury Laboratory preprint, DL/SCI/P542A.
- [36] P. B. Wilson. *Proc. 8th Int. Conf. on High-Energy Accelerators*. Stanford, 1974.
- [37] A. Wrulich and H. Meyer. *Life time due to the beam-beam bremsstrahlung effect*. PET-75-2, DESY, 1975.
- [38] Birdsall, Langdon. *Plasma Physics via computer simulation*. Page 340-341
- [39] G. Fubiani, J. Qiang, E. Esarey, W. P. Leemans, G. Dugan. *Space charge modeling of dense electron beams with large energy spreads*. Accelerators and Beams, 9, 064402 (2006)

- [40] J. Qiang, S. Lidia, R. D. Ryne, and C. Limborg-Deprey *A Three-Dimensional Quasi-Static Model for High Brightness Beam Dynamics Simulation* Lawrence Berkeley National Laboratory. Paper LBNL-59098. <http://repositories.cdlib.org/lbnl/LBNL-59098>
- [41] J. Qiang, S. Lidia, R. D. Ryne, C. Limborg-Deprey, *Phys. Rev. Special Topics - Accel. Beams* 9, 044204, (2006).
- [42] W. Joho private discussions
- [43] J. H. Billen, L. M. Young *POISSON SUPERFISH*. LA-UR-96-1834, Los Alamos National Laboratory, 2004
- [44] J. E. Spencer, H. A. Enge *Split-Pole Magnetic Spectrograph for Precision Nuclear Spectroscopy* Nucl. Instr. Meth. 49 (1967) 181-193
- [45] J.M. Sanz-Sern and M.P. Calvo *Numerical Hamiltonian Problems* Chapman and Hall 1994
- [46] E. Forest. *Phys. Lett. A* 158, p99, 1991
- [47] P. Arbenz, R. Geus and S. Adam *Solving Maxwell eigenvalue problems for accelerating cavities*, *Physical Review Special Topics - Accelerators and Beams*, 4, pp. 022001-1:022001-10, 2001.
- [48] P. Arbenz, M. Becka, R. Geus, U. Hetmaniuk and T. Mengotti *On a Parallel Multilevel Preconditioned Maxwell Eigensolver*, *PARCO*, 32(2), pp. 157-165, 2006, doi: 10.1016/j.parco.2005.06.005
- [49] K. Flöttmann, *Note on the thermal emittance of electrons emitted by Cesium Telluride photo cathodes, TESLA FEL-Report*, 1997-01.
- [50] J. E. Clendenin, T. Kotseroglou, G. A. Mulhollan, D. T. Palmer, J. F. Schmerge, *Reduction of thermal emittance of RF guns NIM A*, 455 (2000) 198-201.
- [51] A Adelmann and P Arbenz and Y Ineichen, *A Fast Parallel Poisson Solver on Irregular Domains Applied to Beam Dynamic Simulations 0907.4863v1 arXiv* (2009)
- [52] Y. Feng and J. P. Verboncoeur, *Transition from Fowler-Nordheim field emission to space charge limited current density Phys.Plasmas*, 13, 073105 (2006)
- [53] R. H. Fowler and L. Nordheim, *Electron Emission in Intense Electric Fields Proc. R. Soc. London, Ser. A* 119, 173 (1928)
- [54] J. H. Han, *Dynamics of Electron Beam and Dark Current in Photocathode RF Guns* PhD Thesis, Desy, 2005 Available Online on <http://www-library.desy.de/preparch/desy/thesis/desy-thesis-05-045.pdf>
- [55] C. Wang and A. Adelmann and Y. Ineichen, *A Field Emission and Secondary Emission Model in OPAL Proc. of HB2010*, MOPD55 (2010)
- [56] M. A. Furman and M. T. F. Pivi, *Probabilistic model for the simulation of secondary electron emission Phys. Rev. ST Accel. Beams*, 5, 124404 (2002)
- [57] J. Rodney M. Vaughan, *A New Formula for Secondary Emission Yield IEEE Trans. on Electron Devices*, 36:9, 1963 (1989)
- [58] J. Rodney M. Vaughan, *Secondary Emission Formulas IEEE Trans. on Electron Devices*, 40:4, 830 (1993)
- [59] C. Vicente and M. Mattes and D. Wolk and H. L. Hartnagel and J. R. Mosig and D. Raboso, *FEST3D - A Simulation Tool for Multipactor Prediction Proc. of MULCOPIIM 2005*, (2005)

- [60] S. Anza and C. Vicente and J. Gil and V. E. Boria and B. Gimeno and D. Raboso, *Nonstationary Statistical Theory for Multipactor Phys. Plasmas*, 17, 062110 (2010)
- [61] A. Henderson, *ParaView Guide: A Parallel Visualization Application Kitware Inc.*, (2007)
- [62] L. Serafini and J. B. Rosenzweig, *Phys. Rev. E* **55** (1996).
- [63] M. Ferrario, J. E. Clendenin, D. T. Palmer, J. B. Rosenzweig, L. Serafini, *HOMDYN Study for the LCLS RF Photo-Injector*, SLAC-PUB-8400, March, 2000
- [64] arxiv