RECENT IMPROVEMENTS IN THE PARMILA CODE*

H. Takeda, J. H. Billen, LANL, Los Alamos NM 87545

Abstract

We discuss capabilities, computational procedures, and recent improvements in the accelerator design code PARMILA. The name PARMILA stands for Phase and Radial Motion in Ion Linear Accelerators. We discuss the algorithms used in PARMILA, how the code designs individual linac sections to achieve efficient acceleration, and how it determines the distance between linac segments. We also discuss the restructuring and transformation of code from the Fortran 77 standard to the Fortran 90 standard.

INTRODUCTION

Program PARMILA originated in the early 1950s for the design drift-tube linear accelerators.[1] The code came to Los Alamos with D. E. Swenson in the mid 1960s and since then many author have contributed to the development of PARMILA. We report here on the current status of the code distributed by the Los Alamos Accelerator Code Group (LAACG) since 1993.

PARMILA performs two distinct tasks: accelerator design and beam-dynamics simulation. Though these tasks are closely related, the code can be used for either one separately or for both together in a single run. In a previous paper [2], we described code organization from the user's point of view. In this paper we concentrate more on computational algorithms.

ALGORITHM FOR GEOMETRICAL β

To achieve efficient acceleration, cell lengths must continuously increase as particles gain energy. However, to build the actual hardware, it is easier to design and engineer sections with a few identical cells. In this case, the rf fields are not exactly synchronous with the particles' arrival at the gap centers. The synchronous particle would have a constant velocity that we call β_g, or the "geometrical beta." To determine β_g for a sequence of equal-length cells, PARMILA uses an extension of Lapostolle's original formulation [3]. For simplicity in this discussion, we assume that the geometrical center of each gap coincides with its electrical center, which has equal potential differences on both sides of the gap.

We describe the acceleration through a cell by a "gap transformation" consisting of three elements: a field-free drift, an impulse kick at the gap center, followed by a field-free drift. The gap transformation derives from the approximation of constant particle velocity across the cell, which is accurate for ions subjected to electric fields achievable in typical accelerating structures. The impulse in each gap has no length and applies the energy gain through the entire cell instantaneously at the gap center.

Particle acceleration computed by PARMILA agrees closely with results of time-step integration using 3-dimensional field maps. The transit-time factor and its derivatives with respect to wave number represent the cumulative effect of the electromagnetic field through the cell. Particle velocities at the entry, electrical center, and exit of a cell are β_i, β_c and β_o. The phases of the rf fields "seen" by the design particle at these locations are ϕ_i, ϕ_c, and ϕ_o.

Consider the coupled-cavity drift-tube linac (CCDTL) shown in Figure 1 and a particle traveling from left to right. The drift tube centered in the cavity creates two asymmetric cells. For each cell, we designate the left and right halves by L and R subscripts. For the first cell, the drift lengths are $\beta_g\lambda n_L$ and $\beta_g\lambda n_R$, where n_L=1/4 and n_R=1/2. If a particle is synchronous with the rf for this structure, the phase difference between the entry and exit must satisfy the condition $\phi_o - \phi_i = 2\pi(n_L + n_R)$.

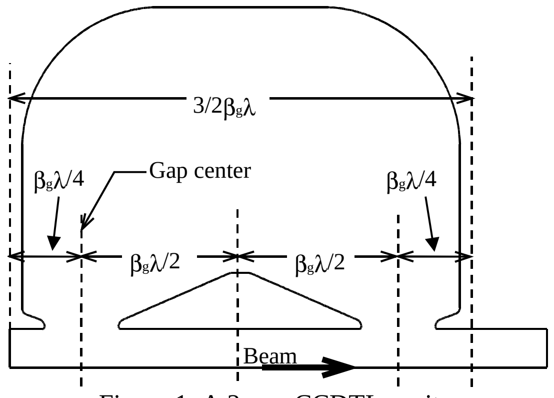

Figure 1. A 2-gap CCDTL cavity.

The structure design accounts for the different entry and exit velocities through the energy-gain equation. For cell j, the energy gain is

$$\Delta W_j = E_0 T_j(n_{j,L} + n_{j,R})\beta_g\lambda \ \cos(\phi_d + \Delta\phi_{Pr}),$$

where E_0 is the average axial electric field, T_j is the transit-time factor, ϕ_d is the design phase, and $\Delta\phi_{Pr}$ is the Promé phase correction term [4]. For a single cell with drift lengths Δz_L and Δz_R, the total phase change is

$$\phi_o - \phi_i = 2\pi(n_L + n_R) = \Delta\phi_L + \Delta\phi_R + \Delta\phi_{Pr}, \text{ where}$$

$$\Delta\phi_L = 2\pi\Delta z_L/\beta_i\lambda, \text{ and } \Delta\phi_R = 2\pi\Delta z_R/\beta_o\lambda.$$

PARMILA interpolates the transit-time factor T_j from a table of values computed for a series of

*Work supported by the US Department of Energy, Office of Science, Division of High Energy Physics.

representative cavities of increasing length. To design a symmetric structure containing several identical cavities we can satisfy the requirement for synchronism from end to end of the entire structure as follows:

$$2\pi\sum_j \left(n_{j,L} + n_{j,R}\right) =$$
$$\sum_j \left(\frac{2\pi}{\beta_{j,i}\lambda} n_{j,L}\beta_g\lambda + \frac{2\pi}{\beta_{j,o}\lambda} n_{j,R}\beta_g\lambda \right) + \sum_j \Delta\phi_{j,Pr}.$$

The Promé correction depends on the particle velocity $\beta_{j,c}$ at the center of the gap, and is computed for each cell. This equation can be written as follows:

$$\beta_g = \frac{\sum_j \left(n_{j,L} + n_{j,R}\right)}{\sum_j \left(\frac{n_{j,L}}{\beta_{j,i}} + \frac{n_{j,R}}{\beta_{j,o}} \right) + \sum_j \frac{\Delta\phi_{j,Pr}}{2\pi\beta_g}},$$

which can be solved iteratively for β_g in a few cycles. Because the Promé correction term is normally positive, its effect is to reduce slightly the result for β_g.

REFERENCE PARTICLE DYNAMICS AND RF STRUCTURE DESIGN

PARMILA designs individual drift-tube linac cells and also rf structure segments separated by drift spaces. We use the term segment to refer to a contiguous section of one or more cavities, usually of the same length. The drift space may contain transverse focusing elements. There are differences in design procedures for normal-conducting and superconducting structures.

Normal-conducting structures often consist of many cavities of varying length, but locked in phase because they are driven by single power source. Examples include the drift-tube linac (DTL) and the coupled cavity linac (CCL). The CCL for the Spallation Neutron Source (SNS) has 48 eight-cavity segments, each segment of a different length, and spanning the velocity range from β_g = 0.40 to 0.55.

In contrast, a superconducting linac typically has only a few different cavity types, with an rf power source for each individual cavity or for small groups of cavities. The layout of the accelerator comes from practical considerations related to the packaging of cavities within cryostats. For SNS, there are two superconducting cavity types, one with β_g = 0.61 and the other with β_g = 0.81.

Transit-time factors

The transit-time factor T for internal cavities of a multicell segment may differ from T for cavities at the end of the segment, especially if the segment has a large bore radius. The user supplies PARMILA with multiple sets of transit-time factor tables, one table for each type of boundary between cells and at the ends of segments. For example, the extended bore tube on the right side of Figure 1 implies the end of a segment while the left side abuts another cavity whose fields are of opposite sign.

For normal-conducting structures, the transit-time data corresponds to a series of representative cavities, each with a different βg. For superconducting structures, the data correspond to many particle velocities β for each individual cavity.

PARMILA linearly interpolates needed values from the supplied transit-time factor tables and warns if a requested β value is outside the table range. During the design part of the PARMILA run, the code stores interpolated transit-time data for each cell for later use in the dynamics simulation. If β_g of a designed segment differs appreciably from the particle velocity, PARMILA expands the equation for the transit-time factor and its derivatives in terms of the particle wave number k (= 2π/βλ) to compute T, ∂T/∂k, etc. for the β of interest.

Normal-conducting structures

To design a segment, the user specifies a design phase ϕ_d and an effective accelerating field E_0T. The code performs single-particle dynamics simulations iteratively to compute cell lengths or segment lengths and the spacing between them. This particle is called the reference particle. For segments containing same-length cavities, the reference particle arrives at each cell center at a different phase of the rf fields. The goal in PARMILA is to make the average of these phases for the segment equal to the user-specified design phase ϕ_d. The code iterates a calculation of the reference particle dynamics over the segment to achieve this goal. The iteration loop can include additional restrictions, for example, if the user requires a particular energy gain for the segment. Convergence of this process determines the segment length, at which point the code records the entry phase ϕ_i and exit phase ϕ_o for the next step: determining the intersegment spacing.

If two consecutive segments are part of the same coupled structure, the cavity fields are locked in phase. Therefore, the particle phase advance over the intersegment length is restricted to mπ/2, where m is an integer. For a given segment n, phases $\phi_{i,n}$ and $\phi_{o,n}$ generally differ from one another and from phases $\phi_{i,n+1}$ and $\phi_{o,n+1}$ of the next segment. PARMILA computes the spacing between segments n and n+1 after it has determined both segment lengths and makes a small correction equal to $(\phi_{i,n+1} - \phi_{o,n})\beta\lambda/2\pi$, where βc is the particle velocity between segments. Because of this method, the user normally designs one extra segment intersegment length between two different rf structures.

Superconducting structures

For a superconducting linac, the design procedure does not include finding cavity lengths. The layout of the linac is fixed, and we know each segment's β_g, and the spacing between segments. In the simplest case, a single independently-phased power source drives each segment. For efficient acceleration, we adjust the cavity phases for each segment.

Because a particular superconducting cavity may accelerate particles over a wide range of velocities, the individual cell lengths differ significantly from $\beta\lambda/2$ at the ends of the velocity range. PARMILA uses the exit phase $\phi_{o,n}$ from cavity n and the entry phase $\phi_{i,n+1}$ for cavity n+1 to calculate the drive phase for cavity n+1 that satisfies the design requirements.

CODE IMPROVEMENTS

3-D space charge

During the particle dynamics simulation, PARMILA applies space charge forces in a linac. Previous versions of PARMILA used exclusively the 2-D space-charge subroutine SCHEFF. The latest version of the code includes an option to use either SCHEFF or the 3-D code PICNIC.[5] PARMILA applies space-charge impulses once per cell at the center of each gap for the entire cell. In a quadrupole lens, the space charge is applied once at the center of each quadrupole for the entire lens. In transport structures, the code applies space charge at least once per $\beta\lambda$ in each drift. The user can specify the number of space charge kicks in the input file.

Code modernization

Recently, PARMILA went through a large structural reorganization. The source code now conforms to the Fortran 90 standard with free format. We eliminated obsolete language features and use Fortran modules rather than the older common block for storing variables. All of the code modules, subroutines, and functions declare IMPICIT NONE and define the type and kind of all variables. We use more descriptive names for important variables. With these changes, the code is more robust and easier to maintain.

File names are no longer limited to eight characters. PARMILA reads an initialization file, which may include names of input and output files, and also some program limits. The user can declare the maximum numbers of linac cells, tanks, and simulated particles.

Code organization

The input file for PARMILA contains keywords to define linac properties, initiate the linac design, and start the particle simulation. (We use quoted lower case strings for these keywords in this discussion.) At the highest level, a PARMILA input file is divided into sections by the "structure" keyword. Some structure sections design and simulate the performance of a particular accelerator type (for example, DTL, CCL, or superconducting linac). A separate structure section may define components for a transport beam line. Before the first "structure" line, the input file defines the beam particle distribution or provides the name of a binary file containing a distribution. Certain global keywords define parameters that pertain to the entire design and simulation calculation.

In structure sections that design a linac, keyword lines placed before the transit-time factor tables define properties of the linac. These properties include accelerating field E_0, design phase ϕ_d, and the transverse focusing lattice. After reading all the transit-time factor data, PARMILA designs the linac. At his point in the file, the user can modify the some of the linac properties (e.g., quadrupole lens settings or accelerating field) before the code starts the simulation part of the run..

The "scheff" line provides information for applying space charge impulses. The beam dynamics simulation starts after the "begin" line. The code generates an output text file that logs progress of the calculation. Certain options may generate other text files. PARMILA creates a binary file containing particle coordinates at locations specified by the "output" line. The postprocessor LINGRAF plots data read from the binary file. This file can be read as an input distribution to another PARMILA run.

CONCLUSION

We have described the latest version of the PARMILA program. For more information and to obtain a copy of PARMILA, send email to laacg@lanl.gov.

REFERENCES

[1] W. K. H. Panofsky, "Linear Accelerator Beam Dynamics," UCRL-1216 (February 1951).

[2] H. Takeda et al., "Recent Developments in the Accelerator Design Code PARMILA," Proceedings of the 1998 Linear Accelerator Conference, Chicago, IL, August 23-28, 1998.

[3] P. Lapostolle et al., Numerical Methods. Acceleration by a Gap, Linear Accelerators p. 747-783, J. Wiley & Sons, 1970.

[4] M. Promé, "Effects of the Space Charge in Proton Linear Accelerators," CEA-N-1457, 1971.

[5] N. Pichoff et al., "Simulation Results with an Alternate 3D Space Charge Routine, PICNIC," Proceedings of the 1998 Linear Accelerator Conference, Chicago, IL, August 23-28,1998.