LBL-18871

WOLF: A COMPUTER CODE PACKAGE FOR THE CALCULATION OF
ION BEAM TRAJECTORIES*

LBL--18871

M.S. Thesis

DE86 005459

Deon L. Vogel

Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

October 1985

MASLEN

* This work was supported by the Director, Office of Energy Research,
Office of Fusion Energy, Development and Technology Division of the
U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

I. Introduction

The function of the WOLF package is to solve for charged particle trajectories in arbitrary electrostatic fields where ion space-charge and electron space-charge from an emitting plasma play important roles in the calculation. The code has the capability to vary several parameters such as current density, electrode and emitter positions and shapes as well as electrode potentials, to find the self-consistent electric field at the emitter surface. This procedure can also be used to minimize the angular divergence of the emitted beam. This feature is very useful in optimizing the performance of ion accelerators where space-charge and electrode potentials significantly influence the ion motion.

The motivation for developing the programs was the desire to design optimum ion-source/accelerator systems in a logical fashion¹. Work on the code was started in the spring of 1973 by William S. Cooper, Klaus Halbach and Steven B. Magary. The group provided the information and understanding relating to plasma determined boundary conditions of the emitter, formulate mathematical models and, where necessary, algorithms, and then to translate these into a usable, efficient computer program. Development has continued ever since and the program was updated frequently as new ideas and suggestions were implemented. Contributions in the latter regard have been made by Chun Fai Chan, Ludmilla Soroka, Art Paul, Oscar A. Anderson and Deon Vogel. Inquires regarding the WOLF program should be directed to Oscar A.

Acknowledgements

I would like to express my gratitude to Professor Lawrence Ruby for his guidance throughout this project. Oscar Anderson, Chun Fai Chan and Ludmilla Soroka deserve much of the credit for familiarizing me with the program and some of its applications. Their willingness to offer assistance at all times has been much appreciated. Furthermore, thanks must go in general to all personnel in Building 4 at Lawrence Berkeley Laboratory for putting up with me during my enjoyable stay there. Appendices A, C and J are included in this report courtesy of Klaus Halbach from his report LBL-4444 (December 1975).

Contents

I.	Introduction.....	1
II.	The LATTICE Program.....	8
III.	The TRIPLOT Program.....	20
IV.	The WOLFCALC Program.....	21
V.	The WOLFPLLOT Program.....	29
VI.	Access and Implementation of the WOLF Code.....	34
	References.....	39
	Figure Captions.....	40
	Appendix A: The difference equations for solving the two- dimensional Poisson equation.....	41
	Appendix B: Input and Output characteristics of the WOLF package.....	48
	Appendix C: Evaluation of integrals for electron deposition..	52
	Appendix D: Summary of the LATTICE program.....	61
	Appendix E: Input to the LATTICE program.....	64
	Appendix F: Input for the WOLF code - WOLFCALC input.....	72
	Appendix G: PISA input to the WOLFCALC program.....	97
	Appendix H: Numerical Solution of Least Squares Optimization by Linearization.....	115
	Appendix I: Input/Output for the WOLFPLLOT program.....	126
	Appendix J: Ion trajectories.....	131
	Appendix K: Addenda and Errata.....	160

WOLF: A COMPUTER CODE PACKAGE FOR THE CALCULATION OF
ION BEAM TRAJECTORIES*

Deon L. Vogel

M.S. Thesis

Accelerator and Fusion Research Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

Abstract

The WOLF code solves POISSON'S equation within a user-defined problem boundary of arbitrary shape. The code is compatible with ANSI FORTRAN and uses a two-dimensional Cartesian coordinate geometry represented on a triangular lattice. The vacuum electric fields and equipotential lines are calculated for the input problem. The user may then introduce a series of emitters from which particles of different charge-to-mass ratios and initial energies can originate. These non-relativistic particles will then be traced by WOLF through the user-defined region. Effects of ion and electron space charge are included in the calculation. A subprogram PISA forms part of this code and enables optimization of various aspects of the problem. The WOLF package also allows detailed graphics analysis of the computed results to be performed.

*This work was supported by the Director, Office of Energy Research, Office of Fusion Energy, Development and Technology Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Anderson or William S. Cooper, Building 4, Lawrence Berkeley Laboratory, Berkeley, California 94720, FTS 451-5011, commercial (415)486-5011.

WOLF is a computer package that consists of four interdependent programs: LATTICE, TRIPLLOT, WOLFCALC and WOLFPLLOT. The LATTICE program generates an irregular triangular mesh which suitably describes the problem configuration. The output from LATTICE is required by all three remaining programs. TRIPLLOT produces a plot of the mesh and WOLFCALC solves Poisson's equation, calculates ion trajectories and deposits ion and electron charges over relevant problem areas. WOLFCALC contains routines that can be used to optimize this geometry.

WOLFCALC contains two controlling sections, FLOW and PISA, together with their respective subroutines. FLOW finds the non-relativistic set of two-dimensional ion trajectories and electric fields (including space-charge from the ions and electrons) for a given set of boundary and initial conditions for the particles and fields. PISA is a program for the optimization of the problem parameters. The combination of FLOW and PISA constitutes the program WOLFCALC that finds the shape of the emitter that is consistent with the plasma forming it, and also optimizes the physical characteristics of the system. The program WOLFPLLOT produces plotted output of the program results provided by WOLFCALC.

The codes comprising the WOLF package are designed for Cartesian geometry and are compatible with ANSI FORTRAN. The physical dimensions of the structures to be modeled by the code are of the order of centimeters rather than meters; thus volts, amperes,

seconds and centimeters are used as units.

I. A Formalism

The WOLF code works on an irregular triangular mesh. The positional coordinates of the vertices of each triangle are known, as are the fixed potential values for the electrode mesh points. This information is supplied to WOLFCALC by the output from LATTICE. As the calculation by WOLFCALC proceeds, the positive and negative charges and potentials are assigned at each triangle vertex. The potential values for the electrode (or emitter) mesh points are not changed. The electric field is assumed to be constant within a triangle and the trajectories of charged particles through a triangle are therefore segments of parabolas. Since the solution of the equations of motion in each triangle is so simple, the calculation of these charged particle trajectories proceeds very rapidly. Mathematical models and algorithms for WOLF can be found in the literature² and the most important of these are summarized in Appendices A, C and J of this report.

If electrons are included in the computation, they are deposited with a Boltzmann distribution and a magnitude determined by an initial fraction of the positive charge density on the emitting surface, and by the potential at the point in question. The negative charge from the electrons is deposited on the vertices of the triangles. This continues until some user specified cut-off value is reached where the charge is then sufficiently small to be neglected.

Since the density of the triangular mesh is to a certain extent user controlled, the computational accuracy can be varied in different regions of a problem. Good accuracy can be obtained by using a fine mesh in-and-around regions where the potentials are changing rapidly, or where the ion energies are low. Using a coarser mesh in areas where less accuracy is required, leads to economy of computer time and data storage.

The WOLF calculations proceed by initially solving Poisson's equation for the potentials due to the vacuum fields. Next, if desired, electrons are deposited on the mesh and Poisson's equation is solved again, taking into account the negative space charge. This cycle is repeated until the largest residual charge deposited along some mesh row is smaller than some predetermined value. This sequence of electron deposition, followed by solution for the resulting potentials, is repeated until convergence is attained. Symbolizing the solution of Poisson's equation by P and the deposition of electrons by E , then convergence after n cycles is achieved after $(PE)^n$ calculations have been performed. Hereafter the ions are traced in this potential distribution and the ion charge deposited on the mesh points. This entire sequence is repeated until the residual of the ion charge deposited is less than a predetermined value. If the ion tracing is referred to by the symbol B , and if the ion trajectories are traced m times, then the program has performed $((PE)^n B)^m$ calculations to yield a solution within the required convergence limits.

I. B. Restrictions and Characteristics

The following restrictions and characteristics apply to this package:

- i) Two dimensional x-y Cartesian geometry is used.
- ii) Flexible triangular mesh is generated.
- iii) The particle motion is non-relativistic.
- iv) Electrons are deposited with a Boltzmann distribution.
- v) Effects of ion and electron space charge are included.
- vi) A large number of independent emitters (425) of particles of different charge-to-mass ratios, q/m , is possible.
- vii) No magnetic fields are considered.

The WOLF capabilities can be further illustrated by application of the code to various types of problems. For example consider determining the electrostatic potentials and ion beam trajectories with or without a background of electrons, for problems which simulate different physical situations:

- i) Ion Source: the ion source produces an ion beam from an electrically neutral plasma. The shape of the self-consistent emitter surface and the space charge of the electrons can play an important role in the proper simulation of the source physics.
- ii) Simulation of Interacting Beams: beam loss because of charge exchange is sometimes important in these problems. The ion produced by charge exchange starts from rest and follows a trajectory affected by the space charge of the beam in which it is produced.

- iii) Beam travel through deflection plates: in this type of example where the ion beam enters the problem from outside the problem boundary, care must be taken to properly simulate the potential on the boundary to represent the space charge outside the problem.

Before the use of the WOLF package is described in detail, features common to all four program components need to be discussed. These include input conventions, mesh characteristics and data dumps. The discussion is found in Appendix B of this report.

I. C. General Information

The four programs of the WOLF code, together with the following three sample files are available for National Magnetic Fusion Energy Computer Centre (NMFECC) users:

TESTLAT - LATTICE input described in Appendix E.

PICALC - sample input for the WOLFCALC program as described in Appendix F.

PIPLOT - sample input for the WOLFPILOT program as described in Appendix I.

The above programs and input files can be used on the CDC 7600 or either of the CRAY-1 machines of the NMFECC system. The CHATR compiler on the 7600 and the CFT compiler on the CRAY-1 are generally used for compilation of the source coding.

The above files are stored under globally readable status in the public file space of certain MFE users. The user number and

directory name can be obtained from Oscar A. Anderson or William S. Cooper. A file may then be retrieved (for example, from user number 12345 and directory .DIRECT) by the command

```
FILEM READ .12345 .DIRECT filename
```

The program of input subroutines to read data (BRDATA) will also be found in this directory.

II. The LATTICE Program

II. A. Introduction

The LATTICE program⁴ processes the data that describe the physical, two-dimensional, Cartesian geometry of the problem and generates an output dump on a disc file called RECORD (by default). This file is then used as input to the TRIPLLOT and WOLFCALC programs. Note that this file RECORD should be renamed TAPE35 when used as input to the WOLFCALC program. The RECORD designation is, however, suitable as input to TRIPLLOT.

The LATTICE code constructs an irregular triangular mesh for the program. Input to the code is prepared by dividing the problem into regions of uniform material properties and by specifying the coordinates of selected points on the boundary of each region. The code then interpolates the omitted boundary points, locates the internal mesh points of each region by some formula derived from harmonic function theorems and assigns regional properties to each triangle. Except that a region boundary may not cross itself, there are no restrictions on the shape of a region. LATTICE also maps this irregular coordinate mesh space, called the logical space, onto real x-y coordinates. The appearance of the region boundaries on the logical diagram may look distorted from the actual x-y diagram, because of the restrictions placed on a geometry by the logical space.

Two-dimensional meshes composed of irregular triangles have certain advantages over rectangular meshes for the numerical solution of differential equations involving the Laplacian such

as Poisson's equation². Because of the irregularity of these meshes it is not practical to construct them by hand and so a computer code is required for this purpose. Such a code, called a generator, forms the mesh out of regional information. These regions are obtained by division of the mesh according to the properties of the physical system to be evaluated. Each region, characterized by uniform material properties is defined by specifying the logical and geometrical coordinates of selected points on its boundary. The generator interpolates the omitted points, labels various aspects of the mesh with regional and boundary information and prepares a tape on disc file which becomes the input for codes requiring the mesh structure. The structure of this LATTICE code is outlined in Appendix D.

II. B. The Logical mesh

Logical paper uses a triangular (isometric) grid as shown in Figure 3. Lines of constant K zig-zag upward starting off to the left of the base line. Lines of constant L are horizontal lines. The origin is at the point $(K, L) = (1,1)$. Logical (permitted) lines on this grid are:

- i) constant L (horizontal)
- ii) constant K (zig-zag, "left-right" i.e., the first segment starting at $L=1$ slopes upward and to the left.)
- iii) slanted (diagonal) lines to the right
- iv) slanted (diagonal) lines to the left.

Note that a right-left zig-zag is not a logical line.

Not every point (K,L) on the logical mesh needs to be specified since the mesh generator can produce values of (X,Y) for non-specified (K,L) points along logical lines. This requires the input of non-adjacent points to lie on a line in logical space. Boundaries are defined by the end points of line segments; each such point is identified by the coordinates (K,L,X,Y).

Specified adjacent points can be any of the six nearest neighbors to the preceding point. If the user wishes to change directions i.e. advance along one logical line for some distance and then change to another logical line, the point of discontinuity must be specified in addition to the starting and ending points of the lines. *It must be kept in mind that some skill must be developed by the user to properly specify the problem in this logical space.*

11. C. Specification of regions

The lattice into which the real problem is mapped by the program must be divided into regions of different type, some of which will overlap. The maximum number of regions allowed is 63. Regions are specified sequentially and have the following properties:

- A region may be
- i) a single point
 - ii) an unclosed line
 - iii) a closed area.

When several regions are overlaid, the last region type given will be used. There are three types of regions:

- i) Type 0
- ii) Type 1
- iii) Type 2

Type 2 regions are closed regions and specify the outer boundaries of the problem. The Type 2 region defining the problem area (the "universe") is usually entered as the first region in the data. It has rectangular logical space dimensions $KMAX * LMAX$, but can contain or enclose a problem area of arbitrary shape. The triangular mesh capability enables problems to be solved within a great variety of boundary shapes. These two constants represent respectively the maximum values of K and L used to describe the problem in logical space.

Type 1 regions may or may not be closed regions and these follow the Type 2 regions in the data stream. These regions describe the area inside of which the problem equations are to be solved. (These regions are usually referred to as Poisson regions). When executing, WOLF will calculate the potentials and ion orbits inside these Type 1 regions only and the ion trajectories will be terminated when a transition occurs between Type 1 and Type 2 regions.

Finally Type 0 regions are open regions and may not be closed. These are usually line regions on which the potential may be fixed or floating. These fixed or floating potentials may be set by the use of a special boundary sentinel which has the value -1 or 0.

Use of the -1 sentinel generates a Dirichlet boundary on which the potential is fixed. The equipotentials are parallel to these Dirichlet boundaries and the boundary potentials are not changed

during the solution of Poisson's equation. If the special boundary sentinel is 0, the potentials on the boundary may be changed during the solution process.

As an example, when describing electrodes which appear in the problem, use is made of Type 2 or Type 0 regions depending on whether the electrode is isolated from, or in some way attached to, the problem boundary.

An electrode that is attached to the boundary (Figure 16(a)) is described by an open Type 0 line region. Points on this line (for the circular electrode) are specified in a counter clockwise manner from point a to point b or clockwise from b to a. This line region traces the (K,L) and (X,Y) coordinates of selected points on the electrode outline. A special boundary sentinel of -1 is then used to set the potential on this electrode.

For a floating or isolated electrode (Fig. 16(b)) the procedure is more detailed. Firstly, the electrode is described by a closed Type 2 region of zero potential. No boundary sentinel is used for this region. Then the potential on this electrode is fixed by overlaying on the electrode boundary two separate Type 0 line regions (Figure 16(c)) with -1 boundary sentinels. Note that the two Type 0 line regions have common end points.

The following procedure has been found to yield good results for constructing a mesh to describe a problem:

- i) Draw the real problem boundaries on graph paper.
- ii) Draw the logical mesh on isometric paper. Try to keep regions of different mesh density more or less rectangular, and avoid drastic changes in the mesh density for adjacent regions.

Also avoid sharp or angular corners as they can lead to:

- a) unacceptable ("negative") triangles (such as when a corner projects into the mesh) and/or
 - b) inaccurate solution of Poisson's equation near this sharp corner.
- iii) Make the first region a closed Type 2 region following the outer boundaries of the problem. Give the (K,L) and (X,Y) coordinates for the corners and for every point where a line drawn on the logical mesh intersects the boundary.
 - iv) This region is followed by all the closed Type 2 regions describing the isolated electrodes (or other objects) in the problem.
 - v) Make the next region a closed Type 1 region following the inner (and outer, where appropriate) boundaries, to enclose the region in which Poisson's equation is to be solved and in which beams are to be traced.
 - vi) Add open Type 1 line regions within the closed type 1 region to delineate areas of different mesh density.
 - vii) Add open Type 0 line regions with special boundary sentinel -1 to put potentials on appropriate areas of the problem.
 - viii) Add the open Type 0 line regions with -1 sentinels to set potentials on any floating or isolated electrodes in the problem.

II. D. Outer Boundaries and Symmetry Planes

If Poisson's equation is solved in a region touching the outer boundary of the problem, the equipotentials will be assumed to be

parallel to the boundary (this is a Dirichlet boundary where the potential $V = 0$ or $V = \text{constant}$). This assumption can be over-ridden selectively to set the equipotentials perpendicular to any or all of the upper, lower, left or right boundaries. (This is a Neumann boundary condition). The default value of 0 for the boundary selects the Dirichlet condition whereas a value of 1 selects the Neumann boundary. If the equipotentials are set perpendicular to the left or right boundaries, beams are also reflected from them, making these boundaries symmetry planes of the problem. The left and right symmetry planes are assumed to be at $K=1$ and $K=KMAX$ respectively.

If any part of an outer boundary has been defined to be a Type 0 line region with a special boundary sentinel -1, this part of the boundary retains boundary conditions appropriate to a conductor (with fixed potential at all points), even if the boundary, or the rest of it, is defined to be a symmetry plane.

II.E. Input to LATTICE

The input format for the LATTICE program is presented in Appendix E.

II.F. Output from LATTICE

Upon termination of the LATTICE program, two output files, OUTPUT and RECORD are generated.

a) OUTPUT: This file is produced by the program whether the mesh is successfully generated or not. If the mesh is generated the run ends with the following message printed in OUTPUT,

ITERATION CONVERGED

DUMP NUMBER 0 HAS BEEN WRITTEN ON TAPE35

GENERATION COMPLETE

If the mesh generation is incomplete the relevant errors encountered during execution are listed. In summary the output from LATTICE onto the file OUTPUT consists of:

- i) a listing of the 124 elements of the PRBCON array.
- ii) a listing of all the regions with user specified values of (K,L) and (X,Y).
- iii) the LATTICE convergence table.
- iv) if desired, a tabulation of the K, L, X, Y values of all the mesh points in the problem.
- v) a message as to the success or failure of the program to generate the proper mesh mapping.

The LATTICE convergence table prints for every 20 iterations the sum of the X and Y residuals. The RMS value of these residuals is compared with the value of PRBCON (89) (by default 10^{-10}) and when less than this value will successfully terminate the mesh generation.

b) RECORD

A successful execution of LATTICE results in the generation of a disc file called, by default, RECORD. The data in this file is written under the heading DUMP 0. This file can be read by the plotting program TRIPLOF, which generates a hardcopy plot of the mesh and the problem boundaries. RECORD can also be read (as a file renamed TAPE35) by WOLFCALC as input for the solution of Poisson's equation. This file is discussed in detail in the section of this report on the WOLFCALC program.

II.G. Error messages

The following is a list of the various error messages that can be generated by LATTICE in the event that it is unable to complete the mesh generation.

All error messages are preceded by: 0 --- ERROR EXIT ---

i) THE NUMBER OF MESH POINTS IN THIS PROBLEM EXCEEDS PROGRAM DIMENSIONS

- this version of LATTICE requires that $(KMAX+2)*(LMAX+2) \leq 13000$.

These dimensions may be increased via the variables in the common block LAR. The integer MAXCNT in the program must also be changed to the new value. This new value is calculated by

$$VALUE = INTEGER * 64 + 8$$

where INTEGER is the smallest integer satisfying

$$(KMAX + 2)*(LMAX + 2) \leq INTEGER$$

The above relations must be satisfied in order to meet certain memory storage requirements for the CDC 7600 (see section VI for a sample calculation of INTEGER).

- ii) FIRST AND LAST POINTS OF REGION HAVE SAME K AND L, DIFFERENT COORDINATES.
 - the x and y coordinates of this point should then be entered correctly.
- iii) NEGATIVE OR ZERO L
 - the smallest value of L is 1
- iv) NEGATIVE OR ZERO K
 - the smallest value of K is 1
- v) LENGTH OF ROW EXCEEDS MAXIMUM
 - the value of K should not exceed 200. This maximum is controlled by the variable NOMAX. The variables in the common block BOUND must be altered if NOMAX is given a new value.
- vi) L, K AND LPRIME, KPRIME NOT ON SAME LOGICAL LINE
 - adjacent (as sequentially listed) mesh points must always be on the same logical line.
- vii) L EXCEEDS LMAX
K EXCEEDS KMAX
 - need to increase the values of LMAX or KMAX or check for errors in specifying L or K.
- viii) YOU HAVE EXCEEDED THE MAXIMUM NUMBER OF INPUT BOUNDARY POINTS PER REGION
 - the maximum number of points per region is 200. This maximum is controlled by the variable NOMAX. The variables in common block BOUND need to be altered if NOMAX is given a new value.

x) 0 ITERATION TERMINATED --- MAXIMUM NUMBER OF CYCLES

- the maximum number of cycles allowed for the mesh generation is 600. This limit is controlled by the value stored in PRBCON (30).

II. H. Negative triangles

The mesh coordinates input by the user may be such that the mesh generator is unable to produce suitable triangular meshes in certain areas. These triangles may become severely distorted or their sides (called zone lines) could cross over. As these "negative" triangles are not suitable for the solution of Poisson's equation by the WOLFCALC program, LATTICE returns with either (or both) of the following messages:

0 THE UPPER TRIANGLE OF POINT K,L HAS NEGATIVE OR ZERO AREA

0 THE LOWER TRIANGLE OF POINT K,L HAS NEGATIVE OR ZERO AREA

These errors will cause an exit from LATTICE.

Eliminating negative triangles or setting up the mesh input so that negative triangles will be avoided is not an easy task and requires a good deal of user skill and experience.

Some common errors that result in negative triangles are when too few points are specified in an area that is geometrically complex or when coordinates of the problem are incorrectly entered in the data. Often negative triangles appear when the configuration in the logical space differs too much from the real x-y geometry.

If the problem domain contains features (like circular electrodes) that are difficult to represent in the logical space

then it is usually necessary to experiment with different shapes on the logical mesh until one is obtained for which no negative triangles are generated.

III. The TRIPLOT Program

The TRIPLOT program is a plot program which needs the output tape (RECORD) generated by the LATTICE program, or the WOLFCALC output tape (TAPE36 switched to RECORD). This program produces a plot of the problem geometry showing the mesh triangles and the electrode and problem boundaries.

The only input required for the TRIPLOT program is either of the two tapes mentioned above.

One of the advantages of this program is the ability to produce a plot of the triangular mesh generated by LATTICE. Since the LATTICE program produces a dump on RECORD irrespective of whether the mesh has been successfully generated or not, TRIPLOT thus serves as a tool for locating and eliminating negative triangles. If LATTICE produces negative triangles, the TRIPLOT plot of the mesh can be used to examine the mesh distortion. Inspection of the mesh irregularity (or irregularities) generally leads the user to alternative methods of representing the input geometry so as to eliminate the distortion.

IV. The WOLFCALC Program

The WOLFCALC Program consists of the main sections FLOW and PISA together with their respective subroutines. The program FLOW finds the non-relativistic, self-consistent set of two dimensional ion trajectories and electric fields for a given set of initial conditions for the particles and fields. PISA is the optional optimisation section of the program.

The section below treats some of the key subroutines in the WOLFCALC program and outlines their functions in this package.

IV. A. 1. The Program FLOW

This program accomplishes the objectives stated above in the following way²: The two-dimensional space that is to be treated is covered with an irregular triangular mesh with boundary sections being given on either scalar potentials or on electric field lines. (The subroutine Poisson calculates the scalar potential at each of these mesh points).

It is assumed that the electrons in the plasma in the vicinity of the ion emitter surface have a Boltzmann distribution with temperature kT_e , in units of electron volts. The subroutine ELECR deposits charges at the mesh points which represent the space charge caused by these Boltzmann electrons.

Ions coming from the emitter will have various charge-to-mass ratios and will have a spread in energy and emission angle. This physical reality is modeled by letting a number of beams emerge from

each line segment that represents part of the emitter surface. Each such beamlet is characterized by a charge to mass ratio, the current it represents, and an initial energy and trajectory direction. The subroutine BTRACE traces each beamlet and deposits the appropriate charges at the mesh points. The trajectory calculation does not take into account relativistic effects nor radiation emitted and the self-produced magnetic field is also ignored.

IV. A. 2. The subroutine POISSON

The reasons for using this program (originally developed by A. Winslow⁴) to solve the electrostatic field problem are manifold:

The difference equations are solved on an irregular mesh that is produced by the mesh generator that is part of the WOLF program package (see the LATTICE program). This makes it possible to have a large density of mesh points where required and also has the advantage of having all surfaces like emitters, electrodes and field lines that are boundaries of the problem, defined by mesh lines without having to resort to interpolation. This in turn leads to comparatively simple difference equations. These are derived in Appendix A.

With given ion and electron space charges from the distribution in the mathematical model, this program solves for the scalar potential values at the mesh points with an over-relaxation method that finds its own optimum over-relaxation factor. The iteration process is terminated when a convergence criterion indicates that one is sufficiently close to the solution.

IV. A. 3. The subroutine ELECTR

Since the density of the Boltzmann electrons is proportional to $\exp(\text{scalar potential}/V_e)$, where $eV_e = kT_e$, the electron temperature, the effect to electron space charge is usually ignored beyond some distance from the emitter. In the region where they are considered, the electrons are taken into account by calculating for every triangle the total charge and the 'center-of-mass' of the charge. From that the charges are calculated for each corner of the triangle and are used with the appropriate under-relaxation factor to update the electron charge array. The relevant formulas are derived in, and summarized in Appendix C.

IV. A. 4. The subroutine BTRACE

Each beamlet, with a charge-to-mass ratio and a current, is traced through one triangle at a time starting with given initial energy and trajectory direction at the emitter. When the beamlet has been traced through a particular triangle, the charge and the charge center-of-mass are calculated and then the proper charges for the three vertices of the triangle are computed and used to update the corresponding array. The relevant formulas are derived in Appendix J.

IV. A. 5. Sequencing of Subroutine calls in FLOW

Every time after POISSON has calculated new scalar potentials at the mesh points, ELECTR is called to recalculate the electron charge

deposition. To avoid computational instabilities and to speed up convergence, the newly calculated electron charges are not deposited directly as calculated, but compared with previously existing electron charges. The recommended change is not fully applied, but first multiplied by an under-relaxation factor that lies between 0.05 and 0.3. After alternating between POISSON and ELECTR a sufficient number of times, the recommended changes in electron charge become small enough that one then calculates the beam trajectories and deposit the ion charges thus generated.

The POISSON-ELECTR iteration starts again until it has converged giving the signal to use BTRACE again. This process is repeated until the ion-charge changes are sufficiently small to indicate convergence of the FLOW calculation.

It is the opinion of all persons involved that this sequencing and under-relaxation scheme is, although it works at present, the weakest part of the program.² Substantial improvements may be possible, but will not be easily obtained. It should also be pointed out in this context that some well posed evaluation problems have been encountered where convergence was extremely slow, or could not be attained at all.

IV. A. 6. WOLFCALC Input

The WOLFCALC code requires two sets of input data. The first set is a file called TAPE35 which is produced by the program LATTICE when constructing the irregular triangular mesh and when calculating quantities required by WOLFCALC. The second file is an input file

drawn up by the user giving a complete description of the problem constant changes required (PRBCON () array) and of the ion beams, electrons and desired paper output; see section IV.A.8.

IV. A. 7. The files TAPE35 and TAPE36

This file contains information processed by LATTICE for WOLFCALC. This information consists of a listing of the complete PRBCON array and the x and y coordinates of all the triangle vertices for the problem. Electrode potentials at appropriate mesh points are also provided together with an array which contains data concerning the mesh. The file TAPE36 is also needed for continuing runs of the same problem.

IV. A. 8. User supplied input file

This input file, which must be prepared by the user, is detailed in Appendix F.

IV. B. 1. The program PISA

The design of a system can be interpreted as the task of finding a set of parameters that leads to a desired performance of that system. PISA is a general purpose Program for Inversion of Systems Analysis that does the optimization automatically. Combined with the appropriate analysis program, in this case FLOW, and given a starting configuration of the system, it varies a given set of parameters until a specified performance is attained within close

tolerances. PISA accomplishes this inversion of the forward program by finding that set of parameters that minimizes a weighted sum of the squares of the deviation between the desired and the accomplished performance of a system, subject to restraining conditions that require certain performance specifications to be met exactly³.

IV. B. 2. Formulation of the problem

A detailed mathematical formulation of the problem is presented in Appendix H.

PISA has successfully been applied to the design of certain magnets,³ to numerous examples in data analysis, and to the optimisation of the extractor structure used in the sample problem treated earlier. In the latter case, the FLOW calculations, all by themselves, were not realistic since the shape of the emitter surface was unknown. By making certain assumptions about the nature of the plasma behind the emitter surface¹ this shape was defined by requiring that everywhere on the emitter the electric field have a value that is related to the initial conditions and to other properties that are associated with the ions and electrons².

It is frequently desirable to optimize some properties of the beams at the exit of the extractor structure. The quantities considered exclusively so far are the angles θ between the tangents to the trajectories at the exit and a given direction. Both the variation of the electric field at the emitter surface and the divergence of the exiting ions are approximately optimized by the

minimization of the following function.

$$\sum W_i(E_i - E_0)^2 + \sum W_k(\phi_k - \phi_0)^2$$

with appropriately chosen weights W_i and W_k .

The parameters used for optimization are the shape of the emitter surface, positions, shapes and potentials of electrodes and possibly some other properties associated with the emitter forming plasma.

It must be noted that PISA is not an easy program to use and it is suggested that references^{1,3} be studied first to gain some idea of how optimization with PISA is approached.

As indicated in section IV A (vii), the input data for the PISA program are incorporated in the input data file for the WOLFCALC program. The details of this required input are discussed in Appendix G.

IV. C. WOLFCALC output files

A WOLFCALC run produces a set of five external output files. These files are used for the recording of results obtained during the calculation and also for recording data for use by subsequent runs or for analysis by WOLFLOT.

Table 2: WOLFCALC external files

UNIT	DEFAULT NAME
UNIT 3	OUTW
UNIT 4	OUTPUT
UNIT 20	TAPE20
UNIT 36	TAPE36
UNIT 40	TAPE40

The first two output files OUTW and OUTPUT are used for the recording of results obtained during various phases of the FLOW (and PISA) calculations. These two files will contain information indicating whether the user supplied problem has successfully been solved or not. TAPE20 and TAPE40 contain all the information needed by the WOLFPLOT program to produce graphical output of the WOLFCALC computations.

TAPE36 is an additional file made available to the user via the value of PRBCON(35). If a series of WOLFCALC runs are to be carried out, sequential dumps can be requested on TAPE36 and the calculation can then be restarted, if so desired, from any of these dumps. Also if the WOLFCALC program terminates by either exceeding the stipulated time limit, reaching the POISSON cycle limit of 600 cycles or exceeding the POISSON-electron-beamtrace cycle limit of 30 cycles a dump will be generated on TAPE36 provided that PRBCON(35) is 0. WOLFCALC can then be restarted from this dump. The default value of PRBCON(35) is 0.

V. The WOLFLOT Program

Included in the WOLF code package is the capability of providing plotted output of the results obtained by the WOLFCALC calculations. The user has control over this plotted output as indicated in this section. Plotter output is available for ion trajectories superimposed on any combination of problem geometry, mesh triangles and/or equipotential lines as specified in PRBCON(18).

V. A. Data input specifying selected output

There are elements of the PCON array which are reserved for the output program. This allows certain of the calculated ion trajectories to be omitted from the plotted output. The variable NBSKIP in PCON(19) gives the number of beams to be omitted from the plot of the trajectories. The integer numbers (NBSKIP of them) entered into PCON(20) to PCON(30) are then the beam numbers which will be skipped when plotting the trajectories.

The plot of the ion trajectories requires the user to specify the plot scale, and orientation of the plot. These data are listed after the above mentioned additions to the PCON array. The following format is used:

XMIN	XMAX	YMIN	YMAX	SCALE
IPSENT	NPH	YPMIN	YPMAX	IPLROT

where XMIN is the minimum value of x(in cm) to be plotted

XMAX is the maximum value of x(in cm) to be plotted

YMIN is the minimum value of y(in cm) to be plotted

YMAX is the maximum value of y(in cm) in be plotted

SCALE is the linear multiplication scale factor for the plot dimensions

The flag IPSENT specifies the type of plot desired, where the value

0: plot equipotentials

1: plot mesh

2: plot mesh and equipotentials

-1: no plot.

NPH indicates the number of equipotentials to be plotted

VPMIN is the minimum potential to be plotted

VPMAX is the maximum potential to be plotted

IPLROT indicates the rotation of the plot by

0: do not rotate

1: rotate the plot counterclockwise through 90° .

The plot dimensions mentioned above need not be the same as the maximum and minimum dimensions of the problem. These coordinates, together with SCALE, can be used to select any rectangular portion of the problem for plotting.

The constant PRBCON(18) specifies the combination of plots according to its values as:

- 0: no plot
- 1: plot beams on problem boundary
- 2: same as option $\approx$ 1, but also plot mesh and/or equipotentials as indicated by IPSENT
- 3: overlay beams on equipotentials
- 4: overlay beams on mesh
- 5: overlay equipotentials on mesh.

V. B. Plotter output selection

After this input data follow two groups of data specifying the details of the plot and phase space output. These groups are

- i) beam trajectory plots, maximum of 5 groups
- ii) phase space plots, maximum of 25

Each group of cards has the format

```

N
  ISTART  IEND  ISKIP
  ISTART  IEND  ISKIP
  -1     -1   -1

```

The user must supply N cards of the variables ISTART, IEND, ISKIP. If there are less than five of these cards e.g., if only two groups are required, then the -1 -1 -1 terminator card must be used.

Particles are tracked in ascending order of sub-beam grouping; consider four beams, each consisting of three sub-beams. The tracking order would then be as indicated in Figure 8.

ISTART, IEND and ISKIP specify the trajectory numbers to be plotted in each group. Beginning with sub-beam ISTART, the subsequent sub-beams are incremented by ISKIP and plotted until sub-beam IEND is finally plotted. If all of the beams in Figure 8 are to be plotted, these variables would be set to 1, 12, 1 respectively. If a plot of only the second sub-beam in each beam is required, the values 5,8,1 would be used. Finally, if a plot of the sub-beams in the second beam only, is required then this is indicated by using 2,10,4.

If N=0 for either the beam or the phase plot groups is used, then the default option of plotting all the trajectories, or phase plots of all the trajectories, on a single plot is used by the program.

V. C. Phase space plots

The WOLFLOT program can produce phase space plots of the angular deviation of the beamlet trajectory from the beam centreline plotted against the distance of each beamlet from this beam centreline.

These phase space plots are controlled by the value contained in PRBCON(44), which gives the number of phase space plots requested by the user. The locations of these phase space plots (PRBCON(44) = N) are then input to the elements PRBCON(125) to PRBCON(225). A maximum of 25 phase space plots is allowed.

The positions at which these phase space plots are required are

given by specifying the K,L values of the starting and end points of a line drawn normal to the beam direction e.g.

```
*125    4    112    36    112
```

specifies a phase plot at the point where the beam intersects the line drawn from the point (4, 112) to point (36, 112). This procedure is followed by WOLFPLLOT when a full sized problem is being run (PCON(14) = 2). At the above mentioned location of the phase space plot, WOLFPLLOT calculates for each beamlet, the angle which its trajectory makes with the direction of the beam centre-line. Also the x- and y-coordinates of each beamlet are retrieved. Once this has been done, an average value THAV, is found for these angles. Similary, a value XAV is calculated, for the average of the x-coordinates of the beamlets. The phase space plot is then centered around XAV and THAV and thus there values are printed with the plot. Also printed out is a value XW of the width of the beam (in the x-direction). A similar value THW is quoted for the angles. In addition to the problem identification heading, a value of $\sqrt{2} * \theta_{\text{RMS}}$ is also printed. All angles are given in milliradians. Examples of phase space plots from a WOLFPLLOT run of the test problem are given in Figures 4 and 5. For a half-sized problem with PCON(14) = 1, then PRBCON(44) = 4 is the maximum number of phase space plots allowed, and the user has to supply the four values of y in PRBCON(45) to PRBCON(48), indicating where the phase plots are desired.

All angles are given in milliradians.

The option of using the value 0 in PRBCON(44) indicates that phase plots are not required.

The PRBCON array contains elements which can be used for the control of plotted output. A list of the PRBCON array elements is given in Appendix F. Those most commonly used in WOLFLOT input are PRBCON(18) for control of beam trajectory output as described above, and PRBCON(44) for indication of the number of phase plots required. If PRBCON(44) > 0 then the phase plot positions must be entered into PRBCON(125) and subsequent locations. Elements 121 to 124 are used to set limits on the relative positions and relative angular deviations from the "centre-of-mass" described earlier for the beam phase plots. If the user does not specify these, the program by default will calculate these limits on the plots.

A sample of the input data to the WOLFLOT program together with reference to selected plotted output is presented in Appendix I.

VI. Access and Implementation of the WOLF code

The four component programs of the WOLF computer code are available for use on the National Magnetic Fusion Energy Computer Centre network as mentioned in the Introduction.

The programs LATTICE and TRIPLLOT, together with their respective input files may be compiled and executed without making any changes to the source coding, except where dimensions are to be increased from default values (see section II. G. part i). To implement WOLFCALC and WOLFLOT, however, certain parameters need to be set to define the limits of the problem. The following seven parameters need to be set by the user:

IVERS: 0 for the CDC 7600

1 for the CRAY

PNX: This value is determined from the product of $(KMAX + 2)$ and $(LMAX + 2)$ and due to strict storage requirements must satisfy the following formula:

$$PNX = \text{INTEGER} * 64 + 8 \quad \text{and where,}$$

$$PNX \geq (KMAX + 2)(LMAX + 2)$$

e.g., $KMAX = 39$, $LMAX = 315$ implies $(KMAX + 2)(LMAX + 2) = 12997$. Substituting into the above equation and solving yields $\text{INTEGER} = 202.95$. Thus set INTEGER to 203 which gives $PNX = 13000$.

PEG: number of emitters used in the problem

PB: number of beams (summed over all emitters)

PSB: number of sub-beams (summed over all emitters)

PBSB: beams*sub-beams (summed over all emitters)

PTJ: number of trajectory steps, usually about $3 * LMAX$.

The values of these parameters are input to the code immediately after all the initial commentary in the source listing has been completed. Once these data have been input, the programs can be compiled and executed using standard procedures available on either the CDC 7600 or CRAY machines.

Together with these four programs of the WOLF code the user will need the program called BRDATA. This program comprises the set of subroutines for reading the data from the user supplied input files. The programs WOLFCALC and WOLFLOT must be compiled with BRDATA. For example, to compile WOLFCALC on the CRAY-D machine the following lines are input

```
PRECOMP WOLFCALC, PWOLF [cr]
```

CRAY: ALL DONE

```
CFT I = PWOLF [cr]
```

CRAY: ALL DONE

```
LDR BIN = ($BLD, BRDATA), X=XCALC, LIB=(FORTLIB),
```

```
STSIZE = LONG. [cr]
```

CRAY: ALL DONE

This produces the program XCALC, a CFT compiled version of WOLFCALC which can now be run on the CRAY-D or CRAY-C machine. It must be remembered that WOLFPLLOT should be compiled with both the FORTLIB and TV80LIB libraries.

VI. A. Input/output Summary

The following is a brief input output summary of the external files used and produced by each of the four programs of the WOLF code:

a) LATTICE

A user supplied input file (Appendix E) called INPUT (by default) is needed. e.g., execute LATTICE with the input file TESTLAT. If XLAT is the controllee

```
XLAT INPUT = TESTLAT [cr]
```

Two output files, OUTPUT (by default) and RECORD (by default) are produced. The user checks OUTPUT for possible negative triangles and convergence information.

b) TRIPILOT

As input, TRIPILOT requires the RECORD file from the LATTICE run. If the user has renamed this file as TAPE35 by

```
SWITCH RECORD TAPE35
```

then TRIPILOT (if it needs to be run) is executed by the following instruction

```
XTRIPILOT RECORD = TAPE35 [cr],
```

where XTRIPILOT is the compiled form of TRIPILOT. This produces a file F3WOL20X which can be plotted using the NETPLOT option on a graphics terminal.

c) WOLFCALC

The WOLFCALC program requires two input files. The first is called INPUT (by default - see Appendix F and Appendix G) and the second is TAPE35 (originally RECORD) from LATTICE. With an input file called PICALC, WOLFCALC is executed as follows:

```
XCAL INPUT = PICALC [cr]
```

This run will produce five output files. OUTPUT (default name) and OUTW (default name) contain printed output of the program results. TAPE36 will contain further dumps if so requested by the user. TAPE20 contains the particle trajectories and velocities at certain stages of the calculation. TAPE40 contains relevant data for the WOLFPLT program.

d) WOLFPILOT

The WOLFPILOT program requires four input files. A user supplied INPUT file (see Appendix I). TAPE35 (and TAPE36, if it contains updated information in the form of dumps), TAPE20 and TAPE40 are all supplied as output files from the WOLFCALC program. With a user supplied input file called PIPILOT, the WOLFPILOT controllee XPLOT is run as follows:

```
XPLOT    INPUT    =    PIPILOT [cr]
```

All the other files needed by XPLOT have the default name needed by XPLOT and they will automatically be retrieved from the user's working file space. This program produces three output files. OUTPUT and OUTW (default names) list printed results of some of the program calculations. The graphics output file F3WOLFOX contains the user requested plots of the WOLFCALC calculations. This file may be examined using the NETPLOT option on a graphics terminal.

REFERENCES

1. W. S. Cooper, K. Halbach, and S. B. Magary, Proc. 2nd Symposium on Ion Sources and the Formation of Ion Beams, Berkeley, CA., October 22-25, 1974. Also published as Lawrence Berkeley Laboratory report LBL-3317, October 1974.
2. K. Halbach, "Mathematical Models and Computer Algorithms for the Computer Program 'WOLF'", Published as Lawrence Berkeley Laboratory Report LBL-4444, December 1975.
3. K. Halbach, Proc. 2nd International Conference on Magnet Technology, Oxford University, Oxford, England, July 10-14, 1967. Published as University of California Radiation Laboratory Report UCRL-17436, July, 1967.
4. A. M. Winslow, "An Irregular Triangular Mesh Generator". Published as University of California Radiation Laboratory Report UCRL-7880, 1964.
5. O. A. Anderson et al., "Efficient, Radiation-hardened, 400 and 800-keV Neutral Beam Injection Systems". Nuclear Technology/Fusion, 4, p. 1418 (1983).

FIGURE CAPTIONS

- Fig. 1 Structure of the LATTICE code
- Fig. 2 Problem outline - test example
- Fig. 3 Logical diagram for example
- Fig. 4 Phase plot at $L=14$
- Fig. 5 Phase plot at $L=19$
- Fig. 6 Allocation of regions in example
- Fig. 7 Logical diagram for PISA example
- Fig. 8 Beam and sub-beam structure
- Fig. 9 Beam emitter example
- Fig. 10 Trapezoidal bump
- Fig. 11 Rectangular bump
- Fig. 12 Triangular bump
- Fig. 13 Bell-shaped bump
- Fig. 14 Half-bell shaped bump
- Fig. 15 Half-bell shaped bump
- Fig. 16 Non-isolated and isolated electrodes
- Fig. 17 Output beam trajectories
- Fig. 18 Geometry and equipotentials
- Fig. 19 Triangular mesh generated by LATTICE
- Fig. 20 Identification of quantities in a triangle
- Fig. 21 Identification of quantities in a triangle
- Fig. 22 Identification of quantities in a triangle
- Fig. 23 Identification of quantities in a triangle
- Fig. 24 Identification of quantities in a triangle

APPENDIX A²The Difference Equations for Solving the Two-Dimensional Poisson Equation

In order to approximately solve the Poisson equation

$$\operatorname{div}(\epsilon \vec{E}) = \rho, \quad (A1)$$

with

$$\vec{E} = -\operatorname{grad} V \quad (A2)$$

representing the electric field, V the scalar potential, ϵ the relative dielectric constant multiplied by $\epsilon_0 = 8.85 \cdot 10^{-14} \text{ A sec V}^{-1} \text{ cm}^{-1}$, and ρ the charge density, we use the variational principle

$$I = \int (\epsilon E^2 - 2\rho V) da = \text{Min}. \quad (A3)$$

This integral is to be taken over the whole problem area, with the desired boundary conditions imposed at the boundaries.

The variational principle will be used in both directions, i.e., the fact will be used that if Eq. (A3) is satisfied it follows that (A1) is satisfied, and vice versa.

The values of V are introduced at the mesh points of an irregular triangular mesh (see Fig. 20) that has the property that each mesh point (except those on a boundary of the problem) is a vertex of six triangles. The x, y coordinates of the mesh points are computed by a mesh generator,

with mesh regions defined in such a way that one obtains approximately the desired mesh size in every such region of the problem geometry. ϵ is assumed to be constant throughout every triangle, but may

have different values in different triangles in the general case; since there is no extra computation associated with this generalization, we treat ϵ that way, even though for the present application ϵ will have the same value ϵ_0 throughout the whole problem. There are no restrictions on $\rho(x,y)$.

In order to be able to calculate, and then minimize, the integral in (A3), one has to make a rule that determines how V is to be calculated inside every triangle. A rule that leads to the simplest meaningful difference equation is the following:

The potential inside a triangle is a linear function of the values of the scalar potential at its vertices, depends in addition only on the charge density in the triangle, and is invariant to translation and rotation of the coordinate system.

This rule has the consequence, as will be seen in more detail below, that when optimizing T (Eq. A3) with respect to the potential at a particular mesh point, one obtains a linear relationship between that potential and the potentials of the other vertices of the six triangles surrounding the point under consideration.

To arrive at a description of V within a triangle, we discuss first the consequences of a discontinuity of V , as one goes from one triangle to another across a line connecting two mesh points. Clearly, a singularity would result for $\vec{E}$, and this singularity would not disappear upon calculating $\int \epsilon E^2 da$. To prevent this from happening, V has to vary along a line connecting two mesh points in a unique way, depending only on the values of V at these two mesh points. The only reasonable function that will do this is the linear function of x,y that assumes the proper V -values at the two mesh points.

Assuming for the moment $\rho = 0$ within a triangle, it is clear that the function V_0 that minimizes the contribution from that triangle to T must satisfy Eq. (A1) with $\rho = 0$ within the triangle. With the above stated boundary condition, V_0 must obviously be a linear function of x and y in the triangle, assuming the values V_1, V_2, V_3 at its vertices. For $\rho \neq 0$, we can set $V = V_0 + V_4$, with V_4 satisfying Eq. (A1) with non-zero ρ , subject to the boundary condition $V_4 = 0$ along the sides of the triangle. With $\vec{E}_0 = -\text{grad } V_0$, $\vec{E}_4 = -\text{grad } V_4$, the contribution to T resulting from the triangle under consideration becomes

$$U = \int [\epsilon(\vec{E}_0 + \vec{E}_4)^2 - 2\rho(V_0 + V_4)] da/2,$$

and its derivative with respect to the potential V_1 , at vertex 1 is

$$U_1' = \int \epsilon \vec{E}_0 \frac{\partial \vec{E}_0}{\partial V_1} da - \int \rho \frac{\partial V_0}{\partial V_1} da + \int \epsilon \frac{\partial \vec{E}_0}{\partial V_1} \vec{E}_4 da. \quad (\text{A4})$$

In taking this derivative we have used the fact that V_4 does not depend on V_1 . We will now proceed to show that the last integral on the right side of Eq. (A4) is zero, meaning that it is sufficient to assume that the potential within a triangle is a linear function of x and y . We then calculate $\partial V_0 / \partial V_1$ (Eq. (A7)), and then the first integral on the right side of Eq. (A4), (Eqs. (A10) and (A11)). Finally, $\int \rho \partial V_0 / \partial V_1 da$ is expressed in terms of the charge, and its "center of mass", within the triangle (Eq. (A12), (A13)), and the contributions from all triangles immediately surrounding point 1 are added up to give the difference equation (A14).

To show that

$$\int \epsilon \vec{E}_4 \frac{\partial \vec{E}_0}{\partial V_1} da = 0, \quad (\text{A5})$$

it is sufficient to prove that statement to be correct for the case where ρ is a δ -function. It is furthermore clear that since V_0 is a linear func-

tion of $x, y, \vec{E}_0$, and with it $\partial \vec{E}_0 / \partial V_1$, is not dependent on x or y , so that we have to consider only $\oint \vec{E}_4 da$. It is convenient to break this integral up into two parts: one integral over an infinitesimal disc surrounding the singularity, and the other over the rest of the triangle. Since $|\vec{E}_4| \sim 1/r$ for sufficiently small distance r from the singularity, the integral over the infinitesimal disc goes to zero like the linear dimension of the disc. The other integral is easily converted to a line integral:

$$\oint \vec{E}_4 da = - \iint \left(\vec{e}_x \frac{\partial V_4}{\partial x} + \vec{e}_y \frac{\partial V_4}{\partial y} \right) dx dy = \vec{e}_y \oint V_4 dx - \vec{e}_x \oint V_4 dy$$

$$\oint \vec{E}_4 da = (\vec{e}_x \times \vec{e}_y) \times \oint V_4 ds.$$

This line integral has to be taken over the boundary of the region under consideration, i.e. the sides of the triangle and the circumference of the infinitesimal disc. The former integral is zero because $V_4 = 0$ on the sides of the triangle, and the latter integral vanishes with r since V_4 behaves like $\ln r$, and $\oint |ds|$ like r .

To evaluate the remaining two integrals in Eq. (A4), we have to find

$$\partial V_0 / \partial V_1 = \lim_{dV_1 \rightarrow 0} [(V_0(x, y, V_1 + dV_1, V_2, V_3) - V_0(x, y, V_1, V_2, V_3)) / dV_1]$$

Since $V_0(x, y, V_1, V_2, V_3)$ is a linear function of x and y , assuming the values V_1, V_2, V_3 at the corners of the triangle, $\partial V_0 / \partial V_1$ becomes (see Fig. 2):

$$\partial V_0 / \partial V_1 = V_0(x, y, dV_1, 0, 0) / dV_1 = d_1 / h_1. \quad (A6)$$

Multiplying numerator and denominator of that fraction by b_1 gives us

alternate form

$$\partial V_0 / \partial V_1 = 2a_1 / 2a = (x(y_2 - y_3) + x_2(y_3 - y) + x_3(y - y_2)) / 2a, \quad (A7)$$

where a is the area of the total triangle, and a_1 is the area of the partial triangle indicated in Fig. A2.

Introducing now the unit vectors $\vec{e}_1, \vec{e}_2, \vec{e}_3$ as indicated in Fig. A3, it follows from Eq. (A6) that

$$\partial \vec{E}_0 / \partial V_1 = \vec{e}_1 / h_1, \quad (A8)$$

and consequently

$$\vec{E}_0 = \vec{e}_1 V_1 / h_1 + \vec{e}_2 V_2 / h_2 + \vec{e}_3 V_3 / h_3. \quad (A9)$$

Because $\vec{E}_0$ is independent of x, y , it follows that

$$\int 2\vec{E}_0 \partial \vec{E}_0 / \partial V_1 \cdot d\mathbf{u} = h_1 b_1 \cdot \frac{1}{h_1} \left(\frac{V_1}{h_1} - \frac{V_2 \cos \alpha_3}{h_2} - \frac{V_3 \cos \alpha_2}{h_3} \right),$$

$$\int 2\vec{E}_0 \partial \vec{E}_0 / \partial V_1 \cdot d\mathbf{a} = V_1 \cdot \frac{b_1}{h_1} - V_2 \cdot \frac{\cos \alpha_3}{h_2 / b_1} - V_3 \frac{\cos \alpha_2}{h_3 / b_1}.$$

From Fig. (A4) follows $h_2 / h_1 = \sin \alpha_3$, and from Fig. (A5) follows $b_1 / h_1 = \cot \alpha_2 + \cot \alpha_3$, giving

$$\int 2\vec{E}_0 \partial \vec{E}_0 / \partial V_1 \cdot d\mathbf{a} = \frac{1}{2} (V_1 (\cot \alpha_2 + \cot \alpha_3) - V_2 \cot \alpha_3 - V_3 \cot \alpha_2) \cdot c. \quad (A10)$$

While this form is very well suited for interpreting the individual contributions in terms of the geometry of the triangle, for a computer program one obviously has to express the cotangents by the coordinates of the corners of the triangle. One can clearly write (see Fig. 23)

$$\cot \alpha_3 = \frac{b_1 b_2 \cos \alpha_3}{b_1 b_2 \sin \alpha_3}.$$

Since the numerator equals the scalar product of the vectors connecting point 3 to points 1 and 2, and the denominator equals twice the area of the triangle, we get

$$\cot \alpha_3 = \frac{(x_1 - x_3)(x_2 - x_3) + (y_1 - y_3)(y_2 - y_3)}{2a}. \quad (A11)$$

To evaluate $\int \rho \partial V_0 / \partial V_1 da$, we use the explicit form for $\partial V_0 / \partial V_1$ given by Eq. (A7). It is also convenient to use the "center of mass" coordinates $\bar{x}, \bar{y}$ of the charge distribution ρ in the triangle:

$$\int \bar{x} = \int x \rho da / q_0, \quad \bar{y} = \int y \rho da / q_0; \quad q_0 = \int \rho da. \quad (A12)$$

To prevent the artificial "disaster" of the line charge q_0 being zero from occurring, one has to treat the effects from charges of different signs separately, which is done in the computer program.

Using Eq. (A12) in Eq. (A7) yields for the line charge q_1 associated with point 1 of the triangle under consideration:

$$q_1 = \int \rho \partial V_0 / \partial V_1 da = q_0 \frac{\bar{x}(y_2 - y_3) + x_2(y_3 - \bar{y}) + x_3(\bar{y} - y_2)}{2a}. \quad (A13)$$

From the geometrical interpretation of q_1/q_0 as the ratio of areas of triangles (see Fig. 21) it is clear that $q_1 + q_2 + q_3 = q_0$, where q_2 and q_3 are the line charges associated with points 2 and 3 when optimization of T with respect to V_2, V_3 is considered. It is furthermore clear that the "center of mass" of the line charges q_1, q_2, q_3 deposited at the corners of the triangle coincides with the "center of mass" of the ρ -distribution. To show this explicitly, we need to consider only

$$\int_3 x_1 q_1 / q_0 = \bar{x} \int_3 x_1 (y_2 - y_3) / 2a + \bar{y} \left(\int_3 x_1 x_3 - \int_3 x_1 x_2 \right) / 2a + \left(\int_3 x_1 x_2 y_3 - \int_3 x_1 x_3 y_2 \right) / 2a.$$

The factor multiplying $\bar{x}$ is clearly one, and the two differences of the

cyclical sums are obviously zero.

When optimizing T (Eq. A3) with respect to the potential at point 1, one obtains contributions according to Eqs. (A4, A5, A10, A11, A12, A13) from every triangle surrounding point 1. The potential of every point next to point 1 makes two contributions to $\partial T / \partial V_1$, one from each triangle that has the line connecting point 1 with its neighbor as a side. Instead of ordering the terms coming from Eq. (A10), (A13) according to triangles, it is more convenient to order them according to the points surrounding point 1.

Using a notation explained in Fig. 24, we get

$$\sum_6 (V_1 - V_n) (\epsilon_{n+} \cot \alpha_{n+} + \epsilon_{n-} \cot \alpha_{n-}) \cdot \frac{1}{2} = \sum_6 q_n.$$

One should notice that the dielectric constant ϵ associated with each triangle appears twice in this equation, once as ϵ_{n+} , and once as $\epsilon_{(n+1)-} = \epsilon_{n+}$. Defining the weights

$$W_n = (\epsilon_{n+} \cot \alpha_{n+} + \epsilon_{n-} \cot \alpha_{n-}) / 2,$$

we get:

$$V_1 \sum_6 W_n = \sum_6 V_n W_n + \sum_6 q_n \quad (A14)$$

The weights are, for the applications in this report, dependent only on geometry and need therefore to be computed only once. The total line charge $\sum_6 q_n$ associated with the point under discussion needs to be recalculated only, but always, when the charge distribution in the triangles changes.

Appendix B
Input and Output Characteristics
of the WOLF package

Field Free Input

Field Free Input is provided for all four programs of the WOLF package. All input is read in field free form as described below. As each card is read, its image is printed into the output stream. Input may be entered in any column of the card up to column 72. Furthermore,

- i) All entries bracketed by dollar ('\$') signs are considered comment and ignored by the computer.
- ii) Entries are separated by blanks and not by commas.
- iii) Numeric blocks consist of numbers or groups of numbers which constitute the data input. These numbers may be
 - a) integers (no specified decimal point)
 - b) floating-point (with decimal point given)
 - c) exponential
 - d) indices to arrays
- iv) Provision for repetition is allowed by the use of the 'R' (repeat) specification.
- v) Blank cards in the data stream are ignored.
- vi) Lists of data can be read on as many cards as necessary to specify the list or until an 'S' is encountered. This designates the end of data input for this list (array).

- vii) The index of an array can be inserted in the input by preceeding it with an asterisk '*'. Numbers following this index will be stored starting in that location. This facilitates input of the elements of an array.

Legitimate input is then some combination of the following forms:

INTEGERS	1	2	25
FLOATING POINT	3.14159	-2.54	.667
EXPONENTIALS	2.3E-6	1E8	2.54E+3
REPEATS	OR6	2.54R4	1.42E-3R8
INDICES	*66	*3	
ARRAY TERMINATION	S		

Consider the input of 2.54 into the 66th location and 2.178 into the 70th location of an array. The required data would be

```
*66    2.54
*70    2.178
S
```

The data is read by calls to the input routine DATA () of the following form:

```
CALL DATA (X1, N1, X2, N2, ..., X7, N7)
```

Thus N1 numbers will be read into array X1, N2 numbers read into array X2 and so forth. If any N is zero no entry is made in the corresponding array. If an 'S' is encountered in the data, the processing of the data into that array is terminated and scan of the card or next card is continued for the next array entry in the calling sequence.

Mesh Description

An irregular triangular mesh is used to cover the problem geometry. Two drawings of the problem should be made by the WOLF user. First a real X-Y sketch and then a logical K-L mesh diagram on isometric graph paper like AR-0828-BD. This paper is printed by Graphic Controls Corporation, Buffalo, New York. The boundaries of the problem are defined by the end points of line segments specified by their K-L and X-Y values. Legitimate line segments on the logical mesh can be straight horizontal lines, straight diagonal lines and zig-zag vertical lines matching the left edge of the paper. All line segments in the logical mesh are mapped into straight line segments in the X-Y space by LATTICE.

The origin of the logical mesh is at $(K,L) = (1,1)$; the origin of the real X-Y diagram can be arbitrarily chosen but should conventionally be chosen at $(0.0,0.0)$. All relevant details on the mesh generator appear in the section describing input to the LATTICE program.

TAPE35 Dumps

A dump is a set of data stored on the computer's disk file. For the WOLF package this disc file is called TAPE35 or TAPE36 by default. The program LATTICE writes its output as DUMP 0 on TAPE35. WOLFCALC retrieves DUMP 0, runs to completion (either achieving user specified goals or terminating due to error) and then writes it's

results on DUMP 1 on TAPE36. It must however be noted that no dump is written in the case of certain software errors such as attempting to divide by zero. The user can then restart WOLFCALC from DUMP 1 and continue the calculation if required.

TRIPLLOT can plot the mesh from any DUMP. For example, a plot from DUMP 0 will give the mesh generated by LATTICE. The dumps on TAPE35 or TAPE36 are the means by which the component programs of the WOLF code package communicate with each other. They may be saved by writing to magnetic tape or by cataloging as permanent files for subsequent use.

APPENDIX C²Evaluation of $\int e^V da$ and $\int e^V z da$ for electron deposition

It has been shown in Appendix A (see Eq. (A12)) that for proper deposition of charges in a triangle, we have to know the total charge, and its "center of mass", for every triangle. For Boltzmann electrons, with a charge density proportional to $\exp(\text{scalar potential}/V_e)$ this requires calculation of $\int e^V da$, $\int e^V z da$ if we introduce the argument of the exponential as new variable.

The derivation of the expressions for these integrals is simplified if we discuss the more general integral

$$I = \int F''(v) da. \quad (C1)$$

We assume that v is a linear function of x and y

$$v = \alpha x + \beta y + \delta, \quad (C2)$$

and the primes indicate differentiation with respect to v . Assuming temporarily that $\alpha \neq 0$, and that the values v_1, v_2, v_3 that v assumes at the three vertices of the triangle are all different from each other and not too small, the value for I can be obtained in the following way

$$I = \frac{1}{\alpha} \int \frac{\partial F'}{\partial x} \cdot dx dy = \frac{1}{\alpha} \oint F' dy = \frac{1}{\alpha} \int_3^2 \int_1^2 \frac{dy}{dv} \cdot F' dv.$$

Since $dy/dv = \text{constant}$ along every side of the triangle, we get, with

$$F(v_n) = F_n:$$

$$I = \frac{1}{\alpha} \int_3^2 \frac{y_2 - y_1}{v_2 - v_1} (F_2 - F_1) = \frac{1}{\alpha} \int_3^2 F_2 \left\{ \frac{y_2 - y_1}{v_2 - v_1} - \frac{y_3 - y_2}{v_3 - v_2} \right\} dv.$$

Bringing the two contributions of the bracket in the last sum onto the common denominator, the terms in the numerator proportional to β cancel each other, leaving only terms proportional to α . Their sum is easily identified as $-\alpha \cdot 2u$, where u equals the area of the triangle. We therefore get

$$I = 2\alpha \int_3 F_1 / [(v_2 - v_1)(v_3 - v_1)] . \quad (C3)$$

In view of the formulas for the first moments for the case $F = e^V$, it is convenient for evaluation of (C3) to introduce the two sets of cyclical quantities

$$\mu_i = 1/(v_2 - v_3) \quad (C4)$$

and

$$G_1 = F_1 \mu_2 \mu_3 , \quad (C5)$$

giving for Eq. (C3)

$$I = -2\alpha \int_3 G_1 . \quad (C6)$$

To get expressions for the moment of a function, we consider

$$J_x = \int F''' x du = \frac{\partial}{\partial u} \int F''' da = \frac{\partial}{\partial u} I ,$$

and obtain $J_x = \int_3 x_1 \partial I / \partial v_1$, or more generally, with $z = x + iy$,

$$J_z = \int_3 z_1 \frac{\partial}{\partial v_1} I . \quad (C7)$$

With Eqs. (C4, C5, C6) we obtain the explicit expression

$$J_z = -2\alpha \int_3 z_1 [F_1 (\mu_2 \mu_3 + G_1 (\mu_2 - \mu_3) - G_2 \mu_3 + G_3 \mu_2)] . \quad (C8)$$

For the specific case $F = e^V$, we get

$$J_z = -2u \int_3 z_1 [G_1 (1 + \mu_2 - \mu_3) - G_2 \mu_3 + G_3 \mu_2] . \quad (C9)$$

with

$$G_1 = e^{v_1} \mu_2 \mu_3 . \quad (C10)$$

It should be pointed out that if one wants the integral and moment of the same given function, the function F used in Eq. (C5) in conjunction with Eq. (C7) is the derivative of the function F used in Eq. (C5) in conjunction with Eq. (C6). The notation introduced in Eq. (C5) is therefore convenient only for the case of interest here, namely $F = e^V$. It is furthermore convenient to write

$$\exp(v) = \exp(\tilde{v}) \cdot \exp(v - \tilde{v}) , \quad (C11)$$

with

$$\tilde{v} = \frac{1}{3}(v_1 + v_2 + v_3)_{\text{original}} , \quad (C12)$$

and pull the factor $\exp(\tilde{v})$ outside all integrals. The function v then entering the integrands has the property

$$v_1 + v_2 + v_3 = 0 . \quad (C13)$$

This condition, which will be used throughout the rest of this Appendix, has to be used with some caution: for instance, Eq. (C13) can be applied after the operation on the right side of Eq. (C7) has been carried out, but not before.

Equations (C6, C9, C10) are not usable when two values of v entering these equations are identical, or when all values of v are zero. This means that when the difference between two values of v is sufficiently small, or when the absolute values of all v 's are sufficiently small, round-off errors become so large that one has to use special expressions suited for these two cases. We will start with the case where two values of v , which we assume to be v_1 and v_2 , satisfy

$$|v_2 - v_1| \ll 1 ,$$

but without at the same time satisfying $|v_1 + v_2| = |v_3| \ll 1$. To obtain the expressions for I and its derivatives, it is convenient to introduce the following abbreviations:

$$\gamma = (v_2 - v_1)/2 \quad (C14)$$

$$C = \cosh \gamma; S = \sinh \gamma \quad (C15)$$

$$v_4 = \frac{3}{2} v_3. \quad (C16)$$

With Eq. (C13), one then gets

$$v_2 = -v_3/2 + \gamma; v_1 = -v_3/2 - \gamma \quad (C17)$$

$$v_3 - v_2 = v_4 - \gamma; v_3 - v_1 = v_4 + \gamma \quad (C18)$$

$$\exp v_2 = e^{-v_3/2} \cdot (C + S); \exp v_1 = e^{-v_3/2} (C - S) \quad (C18)$$

Using this notation in Eqs. (C4), (C6), (C9), and (C10), one obtains after some manipulation

$$I = \frac{2a e^{-v_3/2}}{v_4^2 - \gamma^2} \left(e^{v_4} - C - v_4 \frac{S}{\gamma} \right) \quad (C19)$$

$$\partial I / \partial v_3 = \frac{2a e^{-v_3/2}}{(v_4^2 - \gamma^2)^2} \left[2v_4 C + (v_4^2 + \gamma^2) \frac{S}{\gamma} + (v_4^2 - 2v_4 - \gamma^2) e^{v_4} \right] \quad (C20)$$

To obtain an easily usable expression for $\partial I / \partial v_1$ is not as straightforward as it is to get Eq. (C19), (C20). To show how our particular form of the result, Eq. (C21), is derived, we reproduce some intermediate steps:

$$\partial I / \partial v_1 = \frac{a \cdot e^{-v_3/2}}{v_4^2 - \gamma^2} \left(\frac{2e^{v_4}}{v_4 + \gamma} + \frac{K}{2\gamma} \right)$$

$$K = (C - S) \cdot 2\gamma \cdot (v_4 - \gamma) \left(\frac{1}{v_4 + \gamma} + 1 + \frac{1}{2\gamma} \right) - (C + S)(v_4 + \gamma)$$

$$K = (C - S)(v_4 - \gamma) \left(\frac{1}{v_4 + \gamma} + 1 \right) + 2\gamma - 2\gamma C - 2\gamma v_4 \cdot S/\gamma$$

$$K/2\gamma = -C - v_4 S/\gamma + (C - S) \left(v_4 + \frac{v_4^2 + \gamma^2}{v_4^2 - \gamma^2} - \gamma \left(1 + \frac{2v_4}{v_4^2 - \gamma^2} \right) \right)$$

$$K/2\gamma = v_4 (C - S/\gamma) + C \cdot \frac{2\gamma^2}{v_4^2 - \gamma^2} - S \left(v_4 + \frac{v_4^2 + \gamma^2}{v_4^2 - \gamma^2} \right) - \gamma (C - S) \left(1 + \frac{2v_4}{v_4^2 - \gamma^2} \right)$$

$$\begin{aligned} \partial I / \partial v_1 = & \frac{S e^{-v_3/2}}{v_4^2 - \gamma^2} \left(\frac{2v_4}{v_4 + \gamma} + v_4 \frac{C - S/\gamma}{\gamma} + C \frac{2v_4}{v_4^2 - \gamma^2} - \frac{S}{\gamma} \left(v_4 + \frac{v_4^2 + \gamma^2}{v_4^2 - \gamma^2} \right) - \right. \\ & \left. (C - S) \left(1 + \frac{2v_4}{v_4^2 - \gamma^2} \right) \right). \end{aligned} \quad (C21)$$

Since $|\gamma| \ll 1$, it is clear that in Eqs. (C19-C21), one should use the first terms of the power series for S/γ , $(C - S/\gamma)/\gamma$, to express these functions and S, C .

If one displaces the origin of the coordinate system, it follows immediately from the definition of J_2 and Eq. (C7) that $\int_3 \partial I / \partial v_1 = 1$, and we use this relation to obtain $\partial I / \partial v_2$.

$$\partial I / \partial v_2 = 1 - \partial I / \partial v_1 - \partial I / \partial v_3. \quad (C22)$$

We now turn to the case where the largest of the absolute values of v_1, v_2, v_3 , is small compared to one. It is then convenient to expand the exponential in $\int_0^v da, \int_0^v z da$, into a power series, and then integrate the first few terms individually. We therefore have to evaluate

$$I_n = \int v^n da \quad (C23)$$

and

$$J_{n,z} = \int v^n z \, da \quad . \quad (C24)$$

for $n = 0, 1, 2, \dots$.

With Eqs. (C1) and (C3) we obtain

$$I_n = \frac{2a}{(n+1)(n+2)} \int_3^1 v_1^{n+2} / (v_2 - v_1)(v_3 - v_1) \, .$$

To make this expression suitable for evaluation, we rewrite this as

$$\begin{aligned} -(v_1 - v_2)(v_2 - v_3)(v_3 - v_1) \cdot (n+1)(n+2) I_n / 2a &= v_1^{n+2} (v_2 - v_3) + v_2^{n+2} (v_3 - v_1) + \\ &+ v_3^{n+2} (v_1 - v_2) \, . \end{aligned}$$

The right side of this equation is obviously zero for $v_1 - v_2 = 0$, $v_2 - v_3 = 0$, $v_3 - v_1 = 0$, i.e. it is possible to factor these terms out. We do this by using several times the explicit expression for the geometric sum, and start by rewriting the right side of the last equation as

$$v_3(v_2^{n+2} - v_1^{n+2}) + v_1 v_2 (v_1^{n+1} - v_2^{n+1}) + v_3^{n+2} (v_1 - v_2)$$

Pulling out $v_2 - v_1$, and then continuing the same process, we get

$$\begin{aligned} (v_2 - v_3)(v_3 - v_1)(n+1)(n+2) I_n / 2a &= v_3(v_2^{n+1} + \sum_0^n v_1^{n+1-\nu} v_2^\nu) - v_2 \sum_0^n v_1^{n+1-\nu} v_2^\nu - v_3^{n+2} \\ &= (v_3 - v_2) \sum_0^n v_1^{n+1-\nu} v_2^\nu - v_3(v_3^{n+1} - v_2^{n+1}) \end{aligned}$$

$$(v_1 - v_3)(n+1)(n+2) I_n / 2a = \sum_0^n v_1^{n+1-\nu} v_2^\nu - \sum_0^n v_3^{n+1-\nu} v_2^\nu = \sum_0^n (v_1^{n+1-\nu} - v_3^{n+1-\nu}) v_2^\nu$$

$$I_n = \frac{2a}{(n+1)(n+2)} H_{3,n}(v_1, v_2, v_3) = \int v^n du \quad . \quad (C25)$$

$$H_{3,n}(v_1, v_2, v_3) = \sum_{v=0}^n \sum_{\mu=0}^{n-v} v_1^{n-v-\mu} v_3^\mu v_2^v \quad (C26)$$

$H_{3,n}$ is the homogeneous polynomial of the three variables v_1, v_2, v_3 of order n with all coefficients equal to one. Despite the fact that this polynomial has $(n+1)(n+2)/2$ terms, it is easily evaluated with a method patterned after Horner's method for the evaluation of a polynomial of a single variable: by splitting $H_{3,n}$ into an addend $H_{2,n}$ that depends only on v_1, v_2 , and representing the remainder by v_3 times a factor that is obviously $H_{3,n-1}$, and then representing $H_{2,n}$ in the same manner, one arrives at the following algorithm:

$$\left. \begin{aligned} H_{1,n} &= v_1 H_{1,n-1}; \\ H_{2,n} &= H_{1,n} + v_2 H_{2,n-1} \\ H_{3,n} &= H_{2,n} + v_3 H_{3,n-1} \\ n &= 2, 3, \dots \end{aligned} \right\} \quad (C27)$$

With the following initial conditions:

$$H_{1,1} = v_1; H_{2,1} = H_{1,1} + v_2; H_{3,1} = H_{2,1} + v_3. \quad (C28)$$

This recursion formula requires only $3(n-1)$ multiplications and $2n$ additions to calculate $H_{3,n}$.

A convenient alternate method of evaluating $H_{3,n}$ is obtained by applying Eq. (C13) to the explicit expressions of $H_{3,n}$. This leads to

$$\left. \begin{aligned}
 H_{3,0} &= 1 \\
 H_{3,1} &= 0 \\
 H_{3,2} &= (v_1^2 + v_2^2 + v_3^2)/2 \\
 H_{3,3} &= v_1 v_2 v_3 \\
 H_{3,4} &= v_1^2 v_2^2 + v_2^2 v_3^2 + v_3^2 v_1^2
 \end{aligned} \right\} \quad (C29)$$

With the $H_{3,n}$ calculated, we finally get

$$\int e^v da = 2\pi \sum_n H_{3,n} / (n+2)! \quad (C30)$$

From Eqs. (C24, C7, C25) follows

$$\int v^n z da = \frac{2\pi}{(n+1)(n+2)(n+3)} \int z_1 \frac{\partial}{\partial v_1} H_{3,n+1} \quad (C31)$$

The derivative of H_3 with respect to v_3 can be obtained from the last of Eqs. (C27):

$$\partial H_{3,n} / \partial v_3 = H_{3,n-1} + v_3 \partial H_{3,n-1} / \partial v_3. \quad (C32)$$

Since $H_{3,n}$ is completely symmetrical in v_1, v_2, v_3 , the explicitly appearing variable v_3 in Eq. (C32) can be replaced by v_1 or v_2 , thus giving a general recursive method to calculate the needed derivatives.

As done above, one can also obtain explicit expressions for the derivatives by carrying out the derivatives and then using Eq. (C13).

Doing that gives

$$\begin{aligned}
 \partial H_{3,0} / \partial v_1 &= 0 \\
 \partial H_{3,1} / \partial v_1 &= 1 \\
 \partial H_{3,2} / \partial v_1 &= v_1 \\
 \partial H_{3,3} / \partial v_1 &= v_1^2 + v_2^2 + v_3^2 + v_2 v_3 \\
 \partial H_{3,4} / \partial v_1 &= 2v_1 v_2 v_3 + v_1 (v_1^2 + v_2^2 + v_3^2)
 \end{aligned} \quad (C33)$$

and one obtains

$$\int_0^V z \, da = 2u \sum_n \int_3 z_1 \partial H_{3,n} / \partial v_1 \cdot \frac{1}{(n+3)!} \quad (C34)$$

We can now summarize the procedure to calculate the desired integrals:

- 1) Carry through the normalization leading to Eq. (C13)
- 2) If the largest of the absolute values of v_1, v_2, v_3 is smaller than a predetermined number, go to 3, if not go to 4.
- 3) Calculate the $H_{3,n}$ and their derivatives with Eqs. (C27, C28, C32), or Eqs. (C29, C33), and use them in Eqs. (C30, C34).
- 4) If the smallest of the absolute values of $v_1 - v_2, v_2 - v_3, v_3 - v_1$ is smaller than a predetermined number, go to 5, if not go to 6.
- 5) Order v_1, v_2, v_3 , without changing the cyclical order, so that $|v_1 - v_2|$ is the smallest of the absolute values of the differences of the v 's. Calculate $I = \int e^V da$ and $J_z = \int e^V z da$ with Eqs. (C19, C20, C21, C22, C7).
- 6) Calculate I and J_z with Eqs. (C4, C10, C6, C9).

Appendix D

This Appendix provides a brief summary of the LATTICE program structure and outlines the functions of the main subroutines in the program. A diagram of the LATTICE structure is given in Figure 1.

The following subroutines constitute the LATTICE code⁴.

1) INPT

- Functions:
- a) Reads problem constants, checks for errors, stores internally and prints out.
 - b) Reads regional input, checks for errors, stores externally and prints out.

2) HSTAR

- Functions:
- a) Reads in the regional information written by INPT and constructs a list of region constants stored internally.
 - b) Sets up x,y and k blocks internally in which it:
 - i) interpolates and stores regional boundary points.
 - ii) labels region boundary points with boundary sentinels.
 - iii) labels triangle sides with boundary sentinels for use in distinguishing the insides of a region from the outside.
 - iv) assigns region numbers to triangles.

- v) assigns region numbers to non-boundary points.
- vi) stores zero for coordinates of non-boundary points.

The above operations are performed one region at a time.

3) SETTLE

Functions: a) x and y coordinates of non-boundary points are calculated by an iterative procedure using intersecting "equipotentials" for mesh lines.

4) GENOR

Functions: a) An output tape called RECORD is written containing, in addition to the coordinates, other quantities required as input by the triangle mesh code, arranged by K values.

- b) If triangles with negative areas have been created due to crossing of zone lines (this may occur even without input errors when a sharply indented boundary is zoned with too few triangles), an error printout is made giving the location of these triangles.
- c) Pictures of the mesh are made.

- 5) The subroutines DATA, DATA1, RNUM and RDBYTE are required to read in the data from the data stream.
- 6) The subroutine PRINTP prints a list of the problem constants and the dump number given by PRBCON(9).
- 7) The subroutine MAP is called by SETTLE and maps the x and y coordinates into the output file.

- 8) The subroutines AREU and AREL are called by GENOR to calculate triangle areas.
- 9) Program RUNLAT coordinates the execution of the whole code.

Appendix E

Input to the LATTICE Program

The input data to LATTICE start with a title card followed by the listing of certain problem variables via an array PRBCON. The data stream ends when each point (K,L) and (X,Y) for each region in the problem has been entered.

PRBCON Data

The PRBCON array contains constants and variables which are required by LATTICE and the other WOLF programs. This array is filled with fixed values, default values and mandatory user specified values. The only mandatory user specified input is the number of regions in the problem and the maximum values of K and L. This input is listed in field free format and the data stream terminated by the letter 'S'. The table below lists the PRBCON array elements used by LATTICE which can be altered as specified by the user, along with a default value assumed by the program if the variable is not altered by the user.

Table E1: Selected problem constants from LATTICE

ARRAY INDEX	VARIABLE NAME TITLE	DEFAULT VALUE	DESCRIPTION
1			
2	NREG		user specified
3	LMAX		
4	KMAX		
21	NBSUP	0	upper boundary sentinel
22	NBSLO	0	lower boundary sentinel
23	NBSRT	0	right boundary sentinel
24	NBSLF	0	left boundary sentinel
36	ILMAP	0	= 1 generates coordinate list
77		1.6	optimum over relaxation factor
89		1.0E-10	convergence criterion
91	NDUMP	0	Dump number to be written by LATTICE

On input the data may be referred to by the appropriate index within the PRBCON array. The value of the index is flagged by a preceeding asterisk '*'.

Consider the input of the number of regions, PRBCON(2) and the values of LMAX and KMAX, PRBCON(3) and PRBCON(4). The data could be given in either of the following ways:

*2	5	\$ NUMBER OF REGIONS \$
*3	15	\$ LMAX \$
*4	36	\$ KMAX \$

S

or *2 5 15 36 S \$ NREG, LMAX, KMAX \$

Since the problem title is always read into PRBCON(1) by a separate read statement, the number of regions is always preceded by the array index *2. Any input enclosed by dollar signs is treated as a comment and skipped by the subroutine reading the data.

A comprehensive listing of all the PRBCON elements used by both the WOLF and LATTICE programs, as well as those used by WOLFLOT, is given in Appendix F.

Region Data

After all the problem constants have been input, the data for each individual region of the problem is given. The first card for each region lists the values of the following four variables:

NREG NTYPE POTENL NTRI

where NREG is the region number, which is a sequential number starting from 1 and incremented by 1 for each additional region up to the number given by PRBCON(2). NTYPE specifies the region type as 0, 1 or 2 as described in section II C. POTENL is the potential (in volts) to be used for regions with NTYPE = 0. This potential must be set to 0.0 for all non-Type 0 regions.

Three types of triangles for the mesh are available from the mesh generator. The constant NTRI enables the user to specify the kind of triangle required. Values of NTRI could be 0 (equal weight triangles), 1 (equilateral triangles) or 2 (right triangles). The Type 1 triangles have generally proven to be the most useful.

For Type 0 regions the above card must be followed by a card giving the value of the special boundary sentinel (0, or -1).

Then follow the cards giving the values of (K,L) and (X,Y) of the end points defining the line segments in the problem, one set of (K,L) and (X,Y) values per card. At the end of the input of this coordinate data for the region, a card with the letter "C" must be added to indicate that the data for that region is complete. Thus the input format is as follows

K ₁	L ₁	X ₁	Y ₁
K ₂	L ₂	X ₂	Y ₂
⋮			
K _n	L _n	X _n	Y _n
C			

for a region in which n points are specified. The procedure outlined in this section is followed for each region in the problem.

Example

An example of the use of the LATTICE program is now illustrated. The device to be studied here is an emitter and electrode system for the extraction of ions from a source plasma and the initial electrostatic focusing of this beam of ions. No details of the device are presented here. These may be found in the literature⁵.

An outline of this system is presented in Figure 2. The LATTICE program is to be used to generate a mesh for this problem. The procedure followed is that outlined in sections II B, II C and II D.

The first six cards identify the problem and adjust some of the problem constants as follows:

TEST

This card is a title card whose image will identify all output from the computation.

```
*2 7 20 15 $NREG, LMAX, KMAX$
```

As indicated here, the problem is divided into 7 regions with LMAX and KMAX as 20 and 15 respectively.

```
*23 1 *24 1
```

The left and right boundaries are defined as Neumann boundaries.

```
*36 1 $WITH LATTICE MAP$
```

The value of 1 in PRBCON(36) indicates that a map of the lattice λ -Y coordinates is to be printed. This can be avoided by using the value 0.

S

The 'S' indicates that all the PRBCON data changes have been made. Region data now follows in this input stream, and is determined as follows:

- i) The drawing of the real problem is shown in Figure 2.
- ii) The logical diagram corresponding to Figure 2 is shown in Figure 3. It is usually recommended that some scale be adhered to in transferring the problem from rectangular to triangular coordinates. In this example, a 1 cm to 1 cm scale is adopted. In problems where the geometry is complicated, more space will be required in the logical mesh to adequately describe this geometry.

- iii) The division of the problem into different regions is shown in Figure 6. The first Type 2 region describes the outer boundaries of the problem. The problem boundary is usually traversed in a counter clockwise direction for this purpose. In describing this region, all the points where lines touch the outer boundaries must be included. This "universe" region is given by lines 6-17 of the listing at the end of this Appendix, for the problem outlined above.
- iv) Since there are no isolated regions in the problem, the second region is a closed Type 1 region in which Poisson's equation is to be solved. This region follows the outer boundaries of the problem and the inner boundaries around the emitter (region 7) and the electrodes (regions 4-5). The coordinates for this region are given in lines 19-32 of the listing.
- v) Finally, open Type 0 line regions are added. These have special boundary sentinels of -1 or 0 respectively to indicate fixed or floating potentials on these regions. The listing in this Appendix outlines these regions. The regions 4, 5 and 7 have been assigned potentials of 80000.0, 80000.0 and 84000.0 volts respectively while regions 3 and 6 are at potentials of 0.0 and 80000.0 volts.
- vi) Note that the letter 'C' is used to indicate the end of the input data for each specific region.

The listing of the input data to LATTICE for this problem is provided overleaf.

```

mlf4
#2 7 20 15
#23 1 #24 1
#36 1 $with lattice maps
s
1 2 0.0 1
1 1 -6.00 -1.50
15 1 6.00 -1.50
15 3 6.00 -1.00
15 7 6.00 0.00
15 13 6.00 1.21
15 20 6.00 0.00
1 20 -6.00 0.00
1 13 -6.00 1.21
1 7 -6.00 0.00
1 3 -6.00 -1.00
1 1 -6.00 -1.50
c
2 1 0.0 1
1 3 -6.00 -1.00
15 3 6.00 -1.00
15 7 6.00 0.00
11 7 2.50 0.00
14 13 5.42 1.21
15 13 6.00 1.21
15 20 6.00 0.00
1 20 -6.00 0.00
1 13 -6.00 1.21
2 13 -5.42 1.21
5 7 -2.50 0.00
1 7 -6.00 0.00
1 3 -6.00 -1.00
c
3 0 0000.0 :
-1
15 20 6.00 0.00
1 20 -6.00 0.00
c
4 0 0000.0 :
-1
1 13 -6.00 1.21
2 13 -5.42 1.21
5 7 -2.50 0.00
1 7 -6.00 0.00
c
5 0 0000.0 :

```

```

-1
15 7 6.00 0.00
11 7 2.50 0.00
14 13 5.42 1.21
15 13 6.00 1.21
c
6 0 00000.0 1
-1
5 7 -2.50 0.00
11 7 2.50 0.00
c
7 0 04000.0 1
-1
1 3 -6.00 -1.00
5 3 -2.50 -1.00
11 3 2.50 -1.00
15 3 6.00 -1.00
c

```

APPENDIX F

Input file for the WOLF code - WOLFCALC input

The WOLFCALC program incorporates the FLOW program and the optimization routine PISA. Input to both of these programs forms part of this input file for the WOLF code. In this Appendix, it will be assumed that no optimization is required, and the input format discussed is thus that for the FLOW program only. The input format for PISA is treated in Appendix G.

Note that the input to the FLOW and PISA programs is not introduced by a title card. The title card supplied with the LATTICE input lists the title format to be maintained throughout the entire computation.

PCON Array Input

PISA constants controlling the optimiser are supplied in an array designated PCON. Originally this array of constants was intended for use by the PISA part of WOLF only. However, when TFF (Transverse Field Focussed) geometries⁵ were being studied, additional input variables were required. Use was thus made for further WOLF input through the PCON array. The description of the elements PCON(1) to PCON(11) is provided in Appendix G. For general WOLF use the relevant additions to the array are the elements PCON(19) to PCON(30).

PCON(19): Number of beams to be skipped in the trajectory plots.

PCON(20) to PCON(30): Beams to be skipped.

It must be noted that the values in PCON(19) to PCON(30), if input, must be included in the data input to the WOLFLOT part of the code. It is not necessary that they be included with the WOLFCALC input.

Since the assumption made here is that no optimization is to be performed, the PCON array input will either contain some of the elements listed above, or not be referenced at all.

Pole Points

PISA optimizes the solution by changing the coordinates or the potential of a set of mesh points forming the "optimization boundary." These points, called the "pole points," are specified by their coordinates (K, L) on the logical mesh. If no optimization is required, this is indicated by entering a single zero on this card.

Flow Input

1. Particle Specification

Ions and electrons originate from imaginary lines called emitters. Up to 425 emitting lines can be specified. The trajectories emitted can be thought of as being comprised of a group

of beams localized in space and a group of sub-beams localized in transverse energy as shown in Fig. 8.

Beam data input consists of specification of an emitter surface from which ions of given velocity components and mass should emerge. A beam will be generated from each increment of the surface in the logical space. This beam may be divided into several beamlets called sub-beams. The emitting surface is specified by giving its starting and ending K-capital letter and L values in the logical space under the same restrictions which exist in the LATTICE program data.

The FLOW input data is started with a card giving the number of emitter groups to be read from the data following that card:

NEGRP

Each emitter group then consists of the following card types:

NTYGRP NBGRP NSBGRP EXGRP EYGRP AMUGRP QGRP CURGRP

KSTART() LSTART() KEND() LEND() \$NBGRP ENTRIES \$

FFIT() WFIT() \$ NBGRP entries \$

VXBEAM() VYBEAM() CBEAM() \$ NSBGRP ENTRIES \$

Each emitter is characterized by a divisional structure as determined by the variables listed above. These variables are defined as follows:

NTYGRP: emitter group type = 1 implies that the program is to calculate the electric field at the emitter: = 0 implies that no electric field calculations are done at the emitter.

NBGRP: number of emitter segments in this group (sometimes the words beams and emitter segments are used interchangeably.)

NSBGRP: number of sub-beams in the group.

EXGRP: Velocity in the x-direction (eV) for the group

EYGRP: Velocity in the y-direction (eV) for the group

AMUGRP: atomic mass of the beam in this group (amu)

QGRP: Charge of group in units of electron charge. A negative sign for negative charge will be carried through in the beam tracing.

CURGRP: Fraction of current of PRBCON(66) for this group

The total current for this emitter is thus

$$\text{PRBCON}(66) * \text{CURGRP}$$

This emitter card is followed by the K, L, logical points for the start and end of each beam segment

KSTART(	)	} NBGRP entries each
LSTART(	)	
KEND (	)	
LEND (	)	

As an example, consider the 3 beams as shown in Fig. 9. The required data for these K, L, values would then be

14	16	18	% KSTART %
12	10	8	% LSTART %
16	18	20	% KEND %
10	8	6	% LEND %

The actual X, Y coordinates (in cm) used by WOLF for these emitters will be taken from the lattice mesh according to these specified K, L, indices.

The desired electric field (in volts/cm) and a weight for the electric field for each beam segment of the emitter must be specified next.

FFIT(	)	} NBGRP entries each
WFIT(	)	

Next the sub-beam structure is defined by giving the fraction of the global VXGRP, VYGRP and CURGRP for the beams, which is to be used for each sub-beam.

VXBEAM(	)	} NSBGRP entries each
VYBEAM(	)	
CBEAM (	)	

where

VXBEAM: fraction of VXGRP for each sub-beam of the group
 VYBEAM: fraction of VYGRP for each sub-beam of the group
 CBEAM: fraction of the current (CURGRP) in the beam for
 each sub-beam of the group.

Lastly, for each emitter surface the desired value of the angles and their weights for each particle emitted must be given. Those are used by PISA optimization and may all be set to zero if beam tracing alone is to be done.

$$\left. \begin{array}{l} \text{ANGLE(} \quad) \\ \text{WEIGHT(} \quad) \end{array} \right\} \text{NBGRP} * \text{HSBGRP entries each}$$

ii) Electron specification

Electrons can be specified for up to 5 emitting surfaces. The electrons are deposited as a Boltzmann distribution specified by an electron temperature and an electron to ion density ratio. The deposition of electron charge on the mesh is performed until some cut-off value is reached when the deposited charge becomes negligible. The required parameters are

$$\left. \begin{array}{l} \text{IEGRP} \\ \text{IETYP} \quad \text{IEMGRP} \quad \text{VELECT} \quad \text{CUTOFF} \quad \text{ELECTI} \\ \vdots \\ \text{IETYP} \quad \text{IEMGRP} \quad \text{VELECT} \quad \text{CUTOFF} \quad \text{ELECTI} \end{array} \right\} \text{IEGRP cards}$$

where

- IEGRP: is the number of electron groups and also indicates the number of subsequent cards to be read.
- IETYP: specifies the direction of electron deposition: 1 for direction of increasing L and -1 for decreasing L deposition.
- IEMGRP: specifies which emitter (group number) the electrons are to be associated with
- VE'ECT: is the value of K times the electron temperature. K is the Boltzmann constant
- CUTOFF: is the cut off value such that the deposition of negative charge proceeds until the Boltzmann factor is less than $(\exp - \text{CUTOFF})$ for every point on each line of constant L.
- ELECTI: ELECTI specifies the electron to ion charge density ratio at the associated emitter.

It must also be noted that if the deposition of electrons is not required, this is achieved by use of the value 0 in PRBCON(7) instead of the value 1.

3. Sample Input

This section lists a sample input for the problem already outlined in the LATTICE section of this report. Here we consider a single emitter comprising 12 beams and 5 sub-beams. This input format is outlined below.

```
$TEST$
*1  0  $ NO PISA LOOP $
*4  1  $ NO MATRIX PRINT $
*12 0  $ NEUMANN EMITTER $
*13 1  $ START SUB-BEAMS AT SEGMENT MIDPOINT $
*14 2  $ FULL SIZED PROBLEM $
*15 0  $ L VALUE OF ION EMITTER WHERE RHO IS TURNED ON $
S
0 $ NUMBER OF POINTS TO BE VARIED $
1 $ NUMBER OF EMITTERS $
1 12 5
180.0R2 2.0 1.0 1.0
2 S
3 S
3 S
3 S
010.0R12
2.0E-0BR12
0.0R5
1.0R5
0.2R5
0.0R60 $ DESIRED THETAS $
0.0R60
1 $ NUMBER OF ELECTRON DEPOSITION REGIONS $
1 1
5.0 10.0
0.990
```

After this data follows that describing the printed output required. In addition the printed output generated during the program execution, the following input data requests details of the particle trajectories to be printed as well.

N				
ISTART	IEND	ISKIP	}	Maximum of 5 such groups
⋮				
ISTART	IEND	ISKIP		
-1	-1	-1		

For this printed output, N is less than or equal to 5. If N equals 0 then all the trajectories are printed by default. For N equals 1 see section V. B. for details on how to specify the parameters ISTART, IEND, ISKIP. For the sample problem, the following input is used:

1		
1	12	1
-1	-1	-1

Once this data has been input then the dump number and any changes required within the PRBCON array must be listed.

Dump Number

WOLFCALC may be instructed to start from the output of the LATTICE program (the output file TAPE35 DUMP 0), or from some previous WOLFCALC solution of the problem which may exist as some other dump on TAPE36. Input of the single number of the dump must accordingly be made.

NUMDMP

Normally, NUMDMP is 0 and the mesh and potentials as generated by the LATTICE program are used as the starting conditions for WOLFCALC.

Two points should be kept in mind when starting WOLFCALC from some dump other than DUMP 0:

- i) The PISA problem constants PCON are not written as part of a WOLFCALC output dump. One of these PCON(6), called RLX, is the relaxation parameter for optimization. RLX is modified by PISA during optimization. If WOLFCALC is to start from a different dump and continue as an optimization, for continuity, the value of RLX in the WOLFCALC input data should be changed to agree with the final value reached in the previous optimization cycle.
- ii) If the current density is specified as a PISA optimization parameter, the value of PRBCON(66) is continually updated, to show the present value. Normally PRBCON(66) will be set to some initial value in the WOLFCALC run from DUMP 0. When starting from another dump the entry for PRBCON(66) must be removed from the set of constants specified in the WOLFCALC input data (described below). The present value of the current density as determined by PISA will be used in the next run.

PRBCON Array

After the dump number has been input, the PRBCON array data should follow. This allows alteration of the problem constants as read from the TAPE35 dump written by LATTICE, or TAPE36 from previous WOLFCALC runs if NUMDMP is greater than or equal to 1. This array input is read until terminated by an 'S' in the data stream. A listing of all the problem constants used by both LATTICE and WOLFCALC is included at the end of this Appendix. The following problem constants are now used, following the dump number described above:

```

0    $ DUMP NUMBER $
*7   1  $ TURN OFF THE ELECTRONS $
*25  10 $ PRINT CONVERGENCE INFORMATION $
*32  124
*33  300 $ PE CYCLE LIMITS $
*35  1  $ 1 FOR NO DUMP ON TAPE36 $
*66  0.008 $ CURRENT DENSITY FROM CONVERTER $
*B6  1.0E-6 $ CONVERGENCE CRITERION $
S

```

A sample input and output generated by WOLFCALC is provided at the end of this Appendix.

Potential Array Input

WOLFCALC has the capability of being able to change the potentials on certain problem boundaries without having to reconstruct the lattice with the LATTICE program. Following the PRBCON array input, input of the new potentials along some boundary can be provided if PRBCON (20) is not set to 0. If PRBCON (20) = N.

then the input of N cards giving the K, L, and POTENL (the new value of the potential) values for each of the N points will be read. If PRBCON(20) = -N, then the input of an array of N K's, N L's and N values of POTENL will be read on as many cards as necessary.

Consider the example of changing the potentials of two electrodes with PRBCON(20) = -100. The input would then consist of the K values of the first electrode and the K values for the second electrode. Then the L values of the first electrode and the L values of the second electrode are given. This is followed by the 100 values of the new potentials. This input follows directly after the 'S' terminator card mentioned above. It is important to note here that each point on the boundary for which potential changes are indicated, must be specified as input. This includes all points input to the LATTICE program for these boundaries plus all intermediate points which LATTICE would normally interpolate.

The following is a copy of the input file PICALC to the WOLFCALC program. The output generated is then shown.

Thereafter follows a list of all the PRBCON array elements used by the WOLF code. All elements marked with an asterisk are those which may be altered by the user.

```

$tests$
#1 0 $size loops$
#4 1 $nn matrix: print$
#12 0 $1 ebsu for neumann emitters$
#13 1 $0 ebsu for start at the same mid points$
#14 2 $prz=2 for full size problem, 1 for half$
#15 0 $1 vde of the ion emitter where rho is turned on$
s
0 $number of points to be varied$
1 $number of emitters$
1 12 5
180.0-2 2.0 1.0 1.0
2 s
3 s
3 s
3 s
010.0-12
2.0e-00-12
0.0r5
1.0r5
0.2r5
0.0r61 $desired thetas$
0.0r60
1 $number of electron deposition regions$
1 1
5.0 10.0
0 990
1
1 60 !
-1 -1 -1
0 $dump$
#7 1 $turn on the electrons$
#25 0 $print convergence information$
#32 124 $0 for no map, 124 for e, ion and p maps$
#33 300 $pe cycle limit$
#35 1 $no dump on tape36$
#66 0.000 $j$
#66 1.0e-06 $conv. criterion$
s

```


wolf:2 13888 mesh points 05/03/84 13:46:58 d

```

$test$
*1 0 $size loops$
*4 1 $no matrix on n1$
*12 0 $1 absu for neumann emitter$
*13 1 $0 abspr for start at the same old point$
*14 2 $prsz=2 for full size problem, 1 for half$
*15 0 $1 vole of the ion emitter where rho is turned on$
$
0 $number of points to be varied$
1 $n number of emitter$
! 12 5
100.0r2 2.0 1.0 1.0
2 $
3 $
3 $
3 $
010.0r12
2.0e-08r12
0.0r5
1.0r5
0.2r5
0.0rE0 $desired theta$
0.0rE0
1 $number of electron deposition regions$
1 1
5.0 10.0
0.990
1
1 60 1
-1 -1 -1
0 $dump$

*7 1 $turn on the electrons$
*25 18 $print convergence information$
*32 124 $0 for no map, 124 for e, ion and p maps$
*33 300 $no cycle limits$
*35 1 $no dump on tape36$
*66 0.000 $js$
*66 1.0e-08 $conv. criterion$
$

```

the following execution is from dump number 0

```

poisson problem constants
prbcont( 1) = mtf4 , prbname
prbcont( 2) = 7, rregions
prbcont( 3) = 70, lmax
prbcont( 4) = 15, kmax
prbcont( 5) = 17, lmax
prbcont( 6) = 0,
prbcont( 7) = 1, selectn
prbcont( 8) = 0, b field
prbcont( 9) = 1.000e+00,
prbcont(10) = 0,
prbcont(11) = 100, mat1 pts
prbcont(12) = 40, mat2 pts
prbcont(13) = 0, intf pts
prbcont(14) = 225, tot pts
prbcont(15) = 0,
prbcont(16) = 15, bnd pts
prbcont(17) = 39, spl pts
prbcont(18) = 3, plot-opt
prbcont(19) = 0, inpute
prbcont(20) = 0, inpu pot
prbcont(21) = 0, lupp bry
prbcont(22) = 0, lupp bry
prbcont(23) = 1, iris bry
prbcont(24) = 1, ilef bry
prbcont(25) = 10, ipr opt
prbcont(26) = 0,
prbcont(27) = 0,
prbcont(28) = 0,
prbcont(29) = 0,
prbcont(30) = 600, maxpycs
prbcont(31) = 600, iprpycs
prbcont(32) = 124, map cpth
prbcont(33) = 300, pery lim
prbcont(34) = 0,
prbcont(35) = 1, dump opt
prbcont(36) = 1, lott map
prbcont(37) = -1.000e+50, symm fit
prbcont(38) = -6.000e+00, x-left
prbcont(39) = 6.000e+00, x-right
prbcont(40) = 0,
prbcont(41) = 0, nphsgr
prbcont(42) = 0,
prbcont(43) = 1, phase-pr
prbcont(44) = -1, phase-pl
prbcont(45) = 0, yphase1
prbcont(46) = 0, yphase2
prbcont(47) = 0, yphase3
prbcont(48) = 0, yphase4
prbcont(49) = 6.000e+00,

```

```

prbcon( 50) = 1.000e+02, etamx(1)
prbcon( 51) = 5.000e-01, beta(1)
prbcon( 52) = 5.000e-03, omega
prbcon( 53) = 25, irmax(1)
prbcon( 54) = 0.
prbcon( 55) = 0.

```

number of print groups= 1

group number	1	starting beam number	ending beam number	increment
		1	68	1

```

(1) loop = 0, max. no. of optimization cycles
(2) laplace = 0
(3) ipunch = 0
(4) mprint = 1
(5) muse = 1, no. of solutions with each coeff. matrix
(6) rlx = 0.5000000, starting relaxation factor
(7) rlxm = 0.0079125, minimum relaxation factor
(8) test1 = 0.1000000, relaxation factor test
(9) ssum = 1.000e+20, min. sum (wt. sq. of delta quantities)
(10) pcnt = 0.1000000, convergence test
(11) femax = 0.500, factor for parameter limits
(12) iebw = 0, emitter boundary condition
(13) istbpr = 1, subbeam spread condition
(14) iprsz = 2, problem size
(15) iemt = 0, 1 value to turn on the ion charge
(16) ndsp = 0, down stream plasma emitter option
(17) efey8 = 0.0001000, conv for init. b engyineumann)
(18) icsw = 0, oscar curca? formula option

```

the couplings have been calculated

the residuals for the charges at cycle 5 is 1.0028e+00
 the electron underrelaxation factor at cycle 5 is 2.200000e-01 total time was 0.000 seconds
 the residuals for the charges at cycle 10 is 4.5556e-03
 the residuals for the charges at cycle 20 is 2.1315e-05
 the residuals for the charges at cycle 32 is 1.4316e-07
 the residuals for the charges at cycle 44 is 5.8655e-10

delta e field	e field	percent deviation
delta e field (1) = 5.996e+00	4.00213	-59.979
delta e field (2) = 6.286e+00	3.71308	-62.862
delta e field (3) = 6.278e+00	3.72218	-62.779
delta e field (4) = 6.265e+00	3.73457	-62.654
delta e field (5) = 6.269e+00	3.73101	-62.690
delta e field (6) = 6.270e+00	3.72992	-62.701
delta e field (7) = 6.270e+00	3.72994	-62.701
delta e field (8) = 6.269e+00	3.73109	-62.689
delta e field (9) = 6.266e+00	3.73405	-62.659
delta e field (10) = 6.281e+00	3.71856	-62.814
delta e field (11) = 6.278e+00	3.72204	-62.700
delta e field (12) = 5.853e+00	4.14749	-58.525

rms value for the deviation of the electric fields (w.ne. 8.0) = 6.217e+00
 rms value for the deviation of the angles (status1=1 or 2) = 1.373e-02 mrad = 1.112 deg
 time used: cpu = 1.190 io = 1.090 sys calls = 0.039 sec
 sum total = 0.052 min

Listing of the PRBCON array elements used by the programs
LATTICE, WOLFCALC and WOLFPLLOT (1)

ELEMENT NO.	DEFAULT VALUE	NAME	DESCRIPTION
* 001			Problem name of up to 10 characters
* 002		NREG	Number of lattice regions
* 003		LMAX	Mesh size in Y-direction
* 004		KMAX	Mesh size in X-direction
* 005		IMAX	Value of (KMAX +2)
* 007	0	IELECT	1: electron space charge included 0: no electron space charge
* 008	0.0	BFIELD	Magnetic field perpendicular to plane of plot: Program will abort if $\neq 0.0$
011	0	NAIR	Number of Type 1 material points
012	0	NFE	Number of Type 2 material points
013	0	NINTER	Number of Type 1/ Type 2 boundary interface points.
014	0	NTOTAL	Sum of 011; 012 and 013
016	0	NBND	Number of Dirichlet boundary points
017	0	NSPL	Number of Type 2, interface or Dirichlet boundary points which are held at fixed potential.
* 018	3	IPLT	Beam and equipotential plot options: 0: no plot; 1: beam plotted inside problem boundary 2: same option for 1 but also plot mesh and/or equipotential as determined by IPSENT option (see WOLFPLLOT input). 3: plot equipotentials on top of beams. 4: plot mesh on top of beams 5: plot equipotentials or mesh only

Listing of the PRBCON array elements used by the programs
LATTICE, WOLFCALC and WOLFPLOT (2)

ELEMENT NO.	DEFAULT VALUE	NAME	DESCRIPTION
<u>BOUNDARY VALUES</u>			
* 020	0	INPUTA	H: N numbers of points to have new values of potential: the (K,L) and potential values are to be read in from the lattice data stream from cards following the S-card terminating input to the PRBCON array.
* 021	0	NBSUP	Upper boundary sertinel
* 022	0	NBSLO	Lower right sertinel
* 023	0	IRIGHT	Right boundary sertinel
* 024	0	ILEFT	Left boundary sertinel. For these 4 boundaries:
		NBSLF	0 non-symmetry boundary 1 symmetry boundary
<u>DUMPING AND PRINTING</u>			
* 025	10	IPRT	Beam print option, either 0,1,2,3,10,11,12 or 13. If a two digit number is chosen the first digit (=1) indicates that the electron residuals should not be printed at the end of every PE cycle. The second digit selects as follows: 0 no printing of final beam trajectories or Poisson convergence information 1 print Poisson convergence information 2 print final beam trajectories 3 print Poisson convergence information and final beam trajectories
* 030	600	MAXCY	maximum number of P iterations for Poisson solution
* 031	600	IPRFIN	Print frequency for the Poisson cycles

Listing of the PRBCON array elements used by the programs
LATTICE, WOLFCALC and WOLF-PLOT

(3)

ELEMENT NO.	DEFAULT VALUE	NAME	DESCRIPTION
* 032	1	IMAP	Map print option after WOLF has converged.
* 032	1	IMAP	0 no map point 1 print potentials 2 print positive charges 4 print negative charges 5 print X-coordinates of mesh 6 print Y-coordinates of mesh This option can include multiple choices e.g. 124
* 033	75	ICYLIM	PE cycle limit
* 035	0	NODUMP	0 write next dump in sequence on TAPE 36 1 no dump
* 036	0	ILMAP	0 no lattice map is printed 1 print lattice map

SPECIAL COORDINATE VALUES

037	-1.0E50	SB XSAVE	-fixed value: do not change-
* 038	0.0	XLEFT	X-value of left boundary, calculated by the program
* 039	0.0	XRIGHT	X-value of right boundary, calculated by the program
* 041	0.0		if = 1.0 no grid is placed on phase plots
* 043	1	NPHSPR	PRINT phase space information 0 no print required +1 print for every WOLF loop -1 print only for initial and final WOLF loops
* 044	0	NPHSPL	PLOT phase space information; gives the number of phase plots required

Listing of the PRBCON array elements used by the programs
LATTICE, WOLFCALC and WOLFPLT (4)

ELEMENT NO.	DEFAULT VALUE	NAME	DESCRIPTION
* 045 049	0.0	YPHASE YHIGH	Value of Y beyond which beams are counted in useful current, present to .75 * YMAX

RHODAIR OPTIMIZATION PARAMETERS

050	1.0		Used in optimization of RHOXY in LATTICE
051	0.5		Used in optimization of RHOXY in LATTICE
* 052	0.005	OMEGA	Used in optimization of RHOXY in LATTICE
053	25	IRMAX	It is the number of cycles at which the over relaxation factor is calculated.

CURRENTS AND CURRENT FACTOR

* 066	0.0	CURDEN	Current density (A/cm ²) assumes a depth of 1 cm
-------	-----	--------	--

OVERRELAXATION FACTORS

* 069	5	NPEMIM	Minimum number of PE cycles before beam tracing
* 070	15	JOLIM	Relaxation factor is calculated after first beam trace and after every N PE cycles
* 071	0	IPRINT	0 no printing of trial relaxation factor information 1 print the information
* 072	0.04	RLMIN	Minimum relaxation factor allowed
* 073	0.22	RELMAX	Maximum relaxation factor allowed
* 074	0.03	RELSTP	Step size of relaxation factor
* 075	2.7	RHODAIR	Optimum over-relaxation factor for P

Listing of the PRBCON array elements used by the programs
LATTICE, WOLFCALC and WOLFPLLOT (5)

ELEMENT NO.	DEFAULT VALUE	NAME	DESCRIPTION
* 076	1.0	CREL RHOINT	For over-relaxation factor. Initially 1.0 AND SET TO PRBCON (76) after ion residual is below PRBCON (88)
* 077	1.6		Optimum over-relaxation factor for V in LATTICE
* 078	0.3	RELE	Optimum electron under-relaxation factor, only for DUMP 0

CONVERGENCE CRITERIA

* 079	1.0E-1	FRESTR	Starting value of the electron convergence factor
* 080	1.0E-3	FREEND	The final electron convergence factor
* 081	0.9	FRESTP	Electron convergence factor
* 082	1.0E-9	FRSTP	Starting value of POISSON convergence factor
* 083	1.0E-14	FRPEND	The final POISSON convergence factors
* 084	0.5	FTSTP	POISSON convergence factor
	0.85	EPSILA	Convergence criterion for POISSON
* 086	1.0E-8	EPSILI	Convergence criterion for ions
	0.87	EPSILE	Convergence criterion for electrons
* 088	1.0E-3	RLXLIM	If ion residual is > 1.0E-3 use relaxation factor of 1.0. If below 1.0E-3 use the over-relaxation factor in PRBCON (76)
* 089	1.0E-10		LATTICE convergence criterion.

Listing of the PRBCON array elements used by the programs
LATTICE, WOLFCALC and WOLFPL0T (6)

ELEMENT NO.	DEFAULT VALUE	NAME	DESCRIPTION
<u>SOLUTION VARIABLES</u>			
090	0	ICOUNT CYCLE	Present iteration number for POISSON
091	0	NUMDMP NUMDMP	Dump number to start from
092	0.0	AMIN	Minimum potential, cycle N
093	0.0	AMAX	Maximum potential, cycle N
094	0.0	DAMAX	Maximum charge in potential, cycle N
096	0.0	SMOLDA	Sum of potentials in cycle N
098	0.0	SNOLDA	Air potential difference sum, cycle N-1
100	0.0	SMMJDA	Air potential difference sum, cycle N
102	0.0	SMDENA	Air potential length, cycle N
104	0.0	XNORMA	Air potential normalizer, cycle N
106	0.0	ETA ETAAIR	Rate of convergence in air, cycle N
108	0	ICYSEN	Sentinel for min/max potential calculation
109	0	1TOT	Total mesh points: (KMAX+2)* (LMAX+2)
111	0.0		Bump rotation (clockwise) in radians
112	0	JASTRT	Starting value for air relaxation
113	0	JAEND	End value for air relaxation
114	0	JFSTRT	Starting value for ion relaxation
115	0	JFEND	End value for ion relaxation

Listing of the PRBCON array elements used by the programs
LATTICE, WOLFALC and WOLFLOT (7)

ELEMENT NO.	DEFAULT VALUE	NAME	DESCRIPTION
116	0	JISTR	Starting value for interface relaxation
117	0	JIEND	End value for interface relaxation
118	0	JGSTR	Starting value for GAMINT relaxation
119	0	JGEN	End value for GAMINT relaxation
* 121	-1.0E20	PXMAX	Maximum x-scale for phase plot. If not reset, scaling will be done by the program.
* 122	-1.0E20	PYMAX	maximum theta scale for phase plot. If not reset, scaling will be done by the program.
* 123	1.0E20	PXMIN TNEGC	Minimum x-scale for the phase plots. If not reset, scaling will be done by the program.
* 124	1.0E20	PYMIN TPOSC	Minimum theta scale for phase plot. If not reset, scaling will be done by the program.
* 125			This element and all subsequent elements of the PRBCON array .. to a maximum of PRBCON (225) - can be used to indicate where the phase plots are to be taken.

*These values may be changed by the user.

Appendix G

PISA Input to the WOLFCALC Program

The PISA section of the WOLFCALC program can vary several parameters such as the current density, the position, shape and potential of electrodes to optimize the electric field at the emitter surface and/or the angular divergence of the emitted beam of ions. This requires the specification on the part of the user of the allowed parameters and the desired values of the constrained term for the problem being modeled. The allowed variables i.e., those which can be altered in the optimization, will be referred to as parameters, and each parameter will reference 'pole points' specified by giving the logical mesh K, L coordinate. A maximum of 10 parameters can be specified.

The input for PISA has the following input format to be discussed in detail here:

```
PCON ( )
NPOLE  § NUMBER OF POLE POINTS §
      KPOLE ( )
      LPOLE ( )
NPAR   § NUMBER OF PARAMETERS §
      FUAIR ( )
      JTYPE  INDX ( )  S  CUMIN  CUMAX
NGEN   § NUMBER OF REGENERATION REGIONS §
      IGEN  MINK MAXK MINL MAXL IREG
```

By referring to the input data format for the WOLFCALC program (Appendix F) it will be seen that this PISA input data follows immediately after the PCON constants (if any) for the FLOW routine have been input.

PCON Array Input

The PISA constants controlling the optimizer are supplied in the PCON array. The elements of this array are discussed in a table at the end of this Appendix. The user should thus input any changes required to the default values supplied.

Pole points

PISA optimizes the solution by changing the coordinates or the potential of a set of mesh points forming the 'optimization boundary'. These points, called the pole points, are specified by their coordinates (K,L) on the logical mesh.

NPOLE KPOLE () LPOLE ()

where

NPOLE is the number of points on the optimization boundary. NPOLE can be between 0 and 200. If NPOLE = 0 no more PISA input data is expected and the next data after NPOLE should be the beam data of the FLOW program (see Appendix F part i)

This feature is used when LOOP (PCON(1)) = 0 to avoid having to enter PISA data when no optimization is to be done.

KPOLE() is the set of K-values for the pole points. If NPOLE = 0 no values for KPOLE should be given. Otherwise, NPOLE values are entered. If a K-value is not specified for a pole point, the K-value for the previous pole point will be incremented by 1 and used.

LPOLE() is the set of L-values for the pole points. If NPOLE = 0 no values for LPOLE should be given. Otherwise, NPOLE values are entered. If an L-value is not specified for a pole point, the L-value for the previous pole point will be used.

It should be noted that the optimization boundary described by the NPOLE pole-points does not have to constitute a continuous boundary. For example, pole points 1 through 10 could be on one electrode, 11 through 17 on another, etc.

Parameters

The data following the definition of the pole points tell PISA which parameters it may vary in optimizing the solution. The input specifies the amount by which PISA is to perturb each optimization parameter to calculate the numerical derivatives required for optimization, the shapes and location of bumps on the optimization boundary, and the limits by which an optimization parameter may be changed by PISA.

The number of parameters which can be varied, NPAR, is first defined. If NPAR is zero, no further input is given and control passes to the reading of regeneration region input. The maximum number of parameters is 10. Data input is

NPAR

FUAIR ()

where the NPAR values of FUAIR(J) gives the size of the perturbation for the Jth parameter. PISA calculates a coefficient matrix $\delta F_i / \delta P_j$ which gives the effect on the ith fitted quantity (electric field or final beam angles for instance) caused by a differential change in the amplitude of the jth optimization parameter. This matrix is calculated numerically by perturbing the Jth optimization parameter by an amount specified by FUAIR(J), evaluating the perturbed solution, setting the Jth parameter back to its unperturbed value and then proceeding to the (J + 1)th parameter till all have been treated.

FUAIR(J) must be small enough so that change in the solution is linear in the change in the Jth parameter but must be large enough to avoid round-off errors. Calculated this way, the value of $\delta F_i / \delta P_j$ is a good approximation to the correct local value of the partial derivative. FUAIR(J) should also be chosen so that the $\delta F_i / \delta P_j$ are independent of the sign or magnitude of FUAIR(J).

If the Jth parameter deforms a surface, FUAIR(J) should be much smaller than a mesh interval in the direction of the deformation. The reason for this is that PISA does not regenerate the mesh when it evaluates the matrix elements $\delta F_i / \delta P_j$; it assumes that the

perturbations specified by the FUAIR(J) are so small that this is unnecessary.

WOLF provides a way to verify that the choice of FUAIR(J) is reasonable. If the first element of the PCON array, LOOP, is set to -1, WOLF will calculate 5 coefficient matrices, using values for the FUAIR(J) equal to 0.25, 0.5, 1.0, 2.0 and 4.0 times the values of FUAIR(J) given in the input data. The coefficients of the matrices are conveniently arranged so that the five values of a given $\delta F_i / \delta P_j$ are printed side-by-side so that it is easy to scan them and see which (if any) of the perturbation sizes is reasonable.

If this perturbation size test is made by setting LOOP = -1, be sure to allow enough time for calculating all the coefficient matrices; the results are only printed after all the coefficients are calculated, so if the run exceeds a time limit before completing the calculations, no answers are provided.

The meaning of FUAIR depends on the type of parameter being perturbed. This is indicated in the table below:

Table H 1: Definition of the variable FUAIR

JTYPE	PARAMETER	MEANING OF FUAIR
0	current density	fractional perturbation in the current density
1 or 10	trapezoidal or triangular bump	Change in amplitude of the bump (in cm)
5 or 50	bell shaped bump	Change in amplitude of the bump (in cm)
6	Potential	Fractional change in potential
60	Potential	Change in potential (in volts)

The perturbation size input is followed by the definition of the four different types of parameters that may be varied to optimize a device; current density, trapezoidal or triangular bump, bell shaped bump and the potential of a set of points on the optimization boundary. Bumps represent possible deformations or positional changes for surfaces. The amplitude changes of these bumps can be in one of two directions: the y-direction for JTYPE < 10 or in the direction of PRBCON(111) for JTYPE $\geq$ 10. A bump is specified by a type code (JTYPE), four indices (INDX(I), I=1,4) and the allowed maximum cumulative change of amplitude of the variable i.e., from CUMIN to CUMAX. The general form of the data for a parameter is

```
JTYPE  INDX(1) INDX(2) INDX(3) INDX(4)  S  CUMIN  CUMAX
where S is, as before, the field free data input array stopper
flag. Explicitly, for various allowed types, the data list is

0          S CUMIN  CUMAX    $ Current density $
1  I1 I2 I3 I4  S CUMIN  CUMAX    $ y trapezoidal bump $
5  I1 I2 I3 I4  S CUMIN  CUMAX    $ y bell bump $
6  I1 I2      S CUMIN  CUMAX    $ y potentials fractional change $
10 I1 I2 I3 I4  S CUMIN  CUMAX    $ x trapezoidal bump in direction
                                of PRBCON(111) $
50 I1 I2 I3 I4  S CUMIN  CUMAX    $ x bell shaped bump in direction
                                of PRBCON(111) $
60 I1 I2      S CUMIN  CUMAX    $ potential, change in volts $
```

If the current density is to be a parameter it must be given as the first parameter followed by any other which may be used. The indices of the bump pole points I1, I2, etc. refer to the ordered pairs of K,L mesh values specified as pole points. These indices

are used to describe the geometric properties of the Jth bump i.e., the end points and vertices of trapezoidal, rectangular or triangular bumps; the end points and points of inflection of bell-shaped bumps; and the end points of a set of points where the potential will be varied. They are the numbers of the pole points as they are ranked in the list of pole points that defines the optimization boundary, and should not be confused with either the logical or spatial coordinates of the same pole points. The limits of the cumulative changes made by PISA are individually specified for each variable parameter. CUMIN and CUMAX give the maximum cumulative change in the amplitude of the optimization parameter in both the positive and negative directions. The initial amplitude of all bumps is zero - PISA begins with an undeformed surface, of any shape, as specified in the input data for LATTICE.

The indices of a trapezoidal bump I1, I2, I3 and I4 are the pole point numbers for the parameter pointing to K,L mesh point pairs giving the value of X and Y for the point. The bump shape resulting is shown in Figures 10 to 15 for the normal bump orientation. As illustrated in Figures 10 to 15, by special choice of JTYPE and the pole point indices I1, I2, I3 and I4, the trapezoidal bump can be deformed to be rectangular or triangular. If the bump type is greater than or equal to 10 the amplitude of the bump is varied in a direction given by PRBCON(111). The axis marked DY is the perturbation amplitude in the Y direction as applied by interpolation along the X-direction.

The indices for a bell shaped bump I1, I2, I3 and I4 are the pole point numbers for the parameter pointing to K,L mesh point

pairs giving the value of X and Y for that point. The bump shape resulting is shown in Figure 13.

In Figures 13, 14 and 15 for the bell shaped and half bell shaped bumps, I1 and I4 specify the end points of the bell shaped bump and I2 and I3 give the pole point numbers for the inflection points. If an end point and a point of inflection are identical, then a "half-bell" bump is generated as shown.

By suitably overlapping a series of bell shaped bumps (JTYPE=5 or 50) one can generate a surface deformation as a continuous curve with continuous first derivative in the interval defined by the indices with zero amplitude and first derivative at the end points. In each interval, the function is second order (a section of a parabola) except in the two intervals adjacent to each end of the deformation.

The intervals defined by the indices (pole point numbers of the ends and points of inflection of a bell shaped bump) may contain any number of pole points (mesh intervals). The intervals need not be the same length. The bell shaped bump should be distributed among the pole points so that the intervals overlap, and are displaced in adjacent bumps by one interval.

A convenient means is provided for inserting several overlapping bell shaped bumps by means of a bump generator. The generator is called into operation when more than four indices are given. The data generator card has the following structure

```
JTYPE  I1 I2 I3 I4 I5 . . . I(N) S CUMIN CUMAX
```

and generates the input otherwise provided by

JTYPE	I2	I3	I4	I5	S	CUMIN	CUMAX
JTYPE	I3	I4	I5	I6	S	CUMIN	CUMAX
JTYPE	I(N-3)	I(N-2)	I(N-1)	IN	S.	CUMIN	CUMAX

Regeneration Regions

If PISA deforms a boundary as part of the optimization, the mesh must be 'regenerated' in the vicinity of the deformation, to make a new mesh that conforms with the deformed boundary. This guarantees that the matrix elements $\delta F_i / \delta P_j$ will be continuous, and will not suffer discontinuities as might occur if the boundary passed over the mesh points. It also guarantees that the problem as viewed on the logical mesh is not changed by any deformation. It is not necessary to regenerate the entire mesh; two general types of regeneration regions may be specified, lines or boxes, and PISA contains the necessary parts of LATTICE needed to regenerate the mesh in these regions.

Following the input of the parameter specification is a data set defining the regeneration regions. The number of regeneration regions NGEN can be zero, in which case no further data is read and control passes to the next data call which is for the beam input data for the FLOW routine. A maximum of 10 regeneration regions can be used.

The required data is

NGEN

IGENT MINK MAXK MINL MAXL IREG

⋮

IGENT MINK MAXK MINL MAXL IREG

}

NGEN CARDS

Three types of regeneration regions can be used as specified by the value of IGEN_T.

- IGEN_T = 1 The regeneration region is a rectangular box on the logical mesh with sides along lines of constant K and L.
- IGEN_T = 2 The regeneration region is a line of constant K or L on the logical mesh.
- IGEN_T = 3 The regeneration region is a special line, used only for regeneration of a common boundary between two box (IGEN_T = 1) regions bordering a LATTICE Type 2 region.

MINK, MAXK, MINL and MAXL specify the K,L coordinates of the regeneration region on the logical mesh. If all four numbers are different, the regeneration region is a rectangular box on the logical mesh (although it can have any shape in x,y geometry). If MINK = MAXL, the regeneration region is a line of constant K extending from MINL to MAXL. If MINL = MAXL the regeneration region is a line of constant L, extending from MINK to MAXK.

IREG gives the triangle type to be used in regenerating the region and it should be the same as was originally used by LATTICE (Appendix C, on the LATTICE program).

The purpose of regeneration is

- a) to reconstruct the mesh in any regions in which the boundary has been deformed by PISA, and
- b) to avoid negative triangles which will occur if PISA moves a boundary over mesh points in the (old) mesh.

The following comments and explanations apply to this section:

- a) Problem boundary points (those points between Type 1 and Type 2 regions in LATTICE) in a line generation region are not moved, even if PISA has changed the coordinates of one or more of them after an optimization cycle.
- b) All points on a line regeneration region containing no problem boundary points are moved, resulting in a new straight line in (x,y) coordinate space. The mesh points are evenly distributed along the line.
- c) No points on the boundary of a box regeneration region are moved. There is one exception: if one or both sides of the box are also symmetry planes (as defined in the LATTICE input data), and the emitter is in the box, any points between that where the emitter contacts the symmetry plane and the end of the box are regenerated as if they were on separate line regeneration regions.
- d) Problem boundary points inside a box are not moved during regeneration unless they have been moved by PISA in the previous optimization cycle.

It is difficult to use this regeneration feature of WOLF correctly.

Try to go by the following rules:

Rule 1: Specify line regions before box regions.

Rule 2: Regenerate any boundaries deformed by PISA with one or more boxes.

Rule 3: A line regeneration region should not include any pole points moved by PISA; split the line region into separate line regions with the pole points at the ends.

Rule 4: Use the same triangle type (IREG = NTRI) that was specified to LATTICE.

An Example

Figure 7 shows a position of the logical mesh for a sample problem. The two optimization boundaries are shown by the 29 pole points marked by heavy dots. The electrode has 13 pole points and the emitter has 16. The sides of the problem at K=1 and K=16 have been defined to be symmetry planes of the problem.

For this example, consider the PISA input section shown below:

```
THIS IS A TEST PROBLEM SHOWING PISA INPUT

*1  2  % LOOP.GT.0, USE PISA %
S      % ENO PCON ARRAY INPUT %
%      DEFINE POLE POINTS      %
29
% K FOR EMMITTER SURFACE
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
% K FOR CHANGING POTENTIAL ON ELECTRODE %
1 2 3 4 5R2 6 5R2 4 3 2 1
% L FOR EMITTER SURFACE AND ELECTRODE %
6R16
38R5 37 36 35 34R5
%      SPECIFY PISA PARAMETERS      %
5      % NPAR %
4.0E-4    2.0E-4R3    -50.    % PERTURBATION SIZES %
```



```

1 1 1 16 16 S -65. .01 $ RECTANGULAR BUMP $
5 3 5 7 10 S -.10 .015 $ BELL SHAPED BUMPS $
5 5 7 10 13 S -.10 .015
5 7 10 13 16 S -.10 .015
60 17 29 S -3.0E3 3.0E5 $ POTENTIALS $
$ DEFINE REGENERATION REGION $
1 $ NGEN $
1 1 16 1 11 1 $ BOX ENCLOSING EMITTER $

```

WOLF is called upon, through PISA, to optimize several parameters to minimize in a least squares sense the deviation of certain quantities from specified values. These fitted values are the electric field at the emitter surface E_1 and the final beam angle θ_1 . Defining a sum S which is to be minimized in the least squares sense,

$$S = S_E + S_\theta$$

we are required to minimize S . Define also N to be the total number of fitted points, equal to the sum of the number of electric fields (number of beams) plus the number of ion trajectories (number of beams times the number of sub beams) plus unity

$$N = 1 + N_b (1 + N_{sb})$$

where N_b and N_{sb} are the number of beams and sub-beams respectively, then

$$S_E = \sum_{i=1}^{N_b} \text{WFIT}_i (E_i - \text{FFIT}_i)^2$$

$$S_\theta = \sum_{i=1}^{N_b N_{sb}} \text{WEIGHT}_i (\theta_i - \text{ANGLE}_i)^2$$

where the quantities $WEIGHT_i$, $ANGLE_i$, $WFIT_i$ and $FFIT_i$ are defined in Appendix G (where the input to WOLFCALC program is discussed) and also in the next section of this Appendix.

Choosing the weights

The choice of weights requires some care and is not necessarily obvious. Since there are only two sets of weights in question we consider the ratio of the weights for the electric fields ($WFIT_i$) to those of the final beam angles ($WEIGHT_i$).

It is convenient to have the weights on the final beam angles either 0.0 or 1.0 so that they can conveniently be switched off or on respectively. The weight on the first value of the electric field should be 0.0, as this value is inaccurately calculated [The value of E in the first mesh interval is not accurately calculated because space charges in adjacent regions corresponding to smaller values of K are not taken into account. If the first interval is adjacent to the left-hand symmetry plane, space charge is not taken into account in the calculation of $E(1)$. In either case the corresponding weight should be set to zero]. The weights for the remaining values of the electric field cannot be chosen without some experience in running the problem.

If the weights on the final angles are zero, the weights on the electric fields are non-zero and the beam current density j is specified and constant (not a PISA variable), then the optimization

procedure consists of finding the self-consistent plasma boundary and seeing where the beams go.

If additionally PISA is permitted to vary parameters to try to minimize the final beam divergence, a reasonable choice of weights is such that after the optimization has converged, $S_E \cong S_\theta$. Of course, one does not know in advance what these final sums are likely to be, so a few iterations may be necessary. The final value of S_E is determined partly by the accuracy in calculating the E_i 's and partly by the number of parameters used to fit them; the final value of S_θ is determined partly by aberrations and partly by the initial volume of phase space available, through Liouville's theorem.

For example, we have found that with four parameters to vary the shape and position of the emitting surface and 15 mesh intervals across it, we obtain good optimization with the weights on the final angles either 0.0 or 1.0 and the weights on the electric fields $1.0E-5$.

Reasonable choice of weights is very important; by making an unrealistic choice, one can force PISA to produce physically unrealistic solutions.

PISA Convergence Array - PCON

<u>ARRAY INDEX</u>	<u>VARIABLE</u>	<u>DEFAULT VALUE</u>	<u>DESCRIPTION</u>
1	LOOP		If LOOP > 0 the maximum number of optimization cycles
2	IDUMMY	0	Not used
3	IPUNCH	0	If IPUNCH > 0 write on TAPE10 the values of (K,L,X,Y) for the final optimized boundary. Note the control card dispose (TAPE10,PUNCH) must be used after the execution card when using this option to obtain a punch card deck.
4	MPRINT	2	If MPRINT = 1, print the coefficient matrix. MPRINT > 1 print the (MT WM), MT matrices (Appendix 1).
5	MUSE	1	The number of solutions with each coefficient matrix. Do not change.
6	RLX	0.5	The starting PISA relaxation factor. Preset to 0.5 and described in Appendix 1. PISA calculates recommended changes in all parameters available to it for optimization, multiplies them by RLX then changes the parameters by that amount. PISA automatically varies RLX during the course of the optimization. ³

<u>ARRAY INDEX</u>	<u>VARIABLE</u>	<u>DEFAULT VALUE</u>	<u>DESCRIPTION</u>
7	RLXS	1/128	The minimum value of RLX. Preset to 0.0078125. If the new solution (after PISA has made changes in the optimization parameters) is worse than the previous solution, PISA reduces the relaxation factor RLX by a factor of 2 and tries again.
8	TEST1	0.1	<p>Relaxation factor change tests.</p> <p>(S) = sum of the weighted square of delta quantities at the fitted points. TEST = ratio of improvement in (S) to the 'as if' linear improvement in (S)-1.0.</p> <p>[TEST]=absolute value of TEST.</p> <p>If [TEST] $\geq$ (4.0*TEST1), halve RLX, and if TEST > 0.0, proceed, but if TEST < 0.0, reject the previous solution.</p> <p>If (4.0*TEST1)>[TEST] $\geq$ (2.0*TEST1) halve RLX and proceed.</p> <p>If (2.0*TEST1)>[TEST]>TEST1, proceed. If [TEST] $\leq$ TEST1, double RLX and proceed.</p>

<u>ARRAY INDEX</u>	<u>VARIABLE</u>	<u>DEFAULT VALUE</u>	<u>DESCRIPTION</u>
9	SRSUM	1.0E-20	Convergence test satisfied when $(S) \leq \text{SRSUM}$.
10	PCNT	0.1	Convergence test satisfied when the absolute value of $[(S) - (\text{anticipated final sum})]$ i.e., $(S) - (\text{AFS})$ is $\leq$ $(\text{AFS}) * \text{PCNT}$ and RLX is ≥ 0.25 .
14	IPRSZ	1	IPRSZ=1 for half size problem, i.e., the left or right boundary is a symmetry line for the beam. IPRSZ=2 for full size problem.

Appendix H

Numerical Solution of Least Squares Optimization by Linearization

To obtain a compact representation of the mathematical framework, matrix algebra is used, with column or row vectors represented by lower case letters and matrices by capital letters; the components of vectors or matrices are indexed in standard fashion. In particular, the following vectors are used repeatedly

p = column vector formed with parameters that will be used as primary variables

Δp = column vector of small changes in p

s = column vector formed with secondary variables that describe the performance of the system and which will be fitted to the desired values in the least squares sense.

$\underline{s}$ = desired s i.e., the components of $\underline{s}$ describe the desired values of those secondary variables of the system that will be approximated in the least squares sense.

s_{last} = s given by the system in the last (or starting) configuration.

s_{new} = s after last configuration has been changed by Δp .

Δs = $s_{new} - s_{last}$

$\underline{\Delta s}$ = $\underline{s} - s_{last}$

r = column vector formed of quantities that are to be fitted exactly i.e., restraints that have to be met exactly.

Δr and $\underline{\Delta r}$ have meanings equivalent to Δs and $\underline{\Delta s}$.

To approximate $\underline{s}$ in the least squares sense subject to restraints, one has to find the components Δp_m that minimize

$$S = \sum_n (\Delta s_n - \underline{\Delta s}_n)^2 W_{nn} \quad (H1a)$$

subject to the condition that $\Delta r - \underline{\Delta r} = 0$ (H1b)

The W_{nn} are weight factors that can be used advantageously when some components of S have to be fitted more closely to the desired values than others. The use of weight factors is obviously imperative if some of the components of S have different dimensions than others. Arranging the W_{nn} to form a diagonal matrix W , and denoting the transpose of a vector (and later of matrices) by the superscript T , S can be represented by

$$S = (\Delta s^T - \underline{\Delta s}^T) W (\Delta s - \underline{\Delta s}) \quad (H1c)$$

Using Lagrange's method to minimize S subject to eq. (H1b), the function L is formed by adding to S each of the individual equations expressed by (H1b) multiplied by a Lagrange multiplier n . Considering n as a component of the vectors, L can be expressed as

$$L = (\Delta s^T - \underline{\Delta s}^T) W (\Delta s - \underline{\Delta s}) + \Delta r^T (\Delta r - \underline{\Delta r}) \quad (H1d)$$

and S is optimized by fulfilling the equations

$$\frac{\delta L}{\delta \Delta p_m} = 0 \quad (H1e)$$

$$\Delta r = \underline{\Delta r} \quad (H1f)$$

It might seem that one could for all practical purposes fulfill the restraining conditions by including them in the least squares sum with appropriately chosen large weights. In general this is not

advisable since it can lead to a problem of inverting ill conditioned matrices (see eq. H4). In the procedure shown here, the number of components of Δr can at most equal the number of primary parameters.

Except in fairly rare cases, the components of Δs and Δr will be non-linear functions of the components of Δp and quite often even the explicit analytical relationships are unknown. This, combined with the fact that the number of variables is sometimes large makes the use of numerical methods (and thus digital computers) imperative for the solution of this problem. Linearisation of the relationship between the components of Δs and Δr and the components of Δp leads to a procedure that is both practical for numerical evaluation and conceptually simple.

The linearization of the relationship between the components of Δs and Δr and the components of Δp involves representing Δs and Δr by a Taylor series and retaining only the linear terms³:

$$\Delta s = M \Delta p \quad M_{nm} = \frac{\delta S_n}{\delta p_m} \quad (H2a)$$

$$\Delta r = N \Delta p \quad N_{nm} = \frac{\delta r_n}{\delta p_m} \quad (H2b)$$

Substituting this into Eq.(H1d), we get

$$L = (\Delta p^T M^T - \underline{\Delta s}^T) W (M \Delta p - \underline{\Delta s}) + \lambda^T (N \Delta p - \underline{\Delta r}) \quad (H2)$$

If we abbreviate the trivial derivatives $\frac{\delta \Delta p}{\delta \Delta p_m}$ by

$$a_m = \frac{\delta \Delta p}{\delta \Delta p_m} \quad (H3)$$

given by the Eq. (H4), but change the primary parameters by $\epsilon \Delta p$. ϵ is a positive number smaller than unity that has to be chosen and readjusted so that for the step $\epsilon \Delta p$, Eq. H2a, H2b holds reasonably well. The reason for reducing each of the components of Δp by the same factor ϵ is, qualitatively, that if one cannot make the full step Δp , one should at least go in the same direction in p-space as is recommended by Eq. (H4). Since for $\epsilon < 1$, one cannot reach the minimum in one step, one has to re-evaluate the system (i.e., Δs , Δr , M, N) after every step and repeat this procedure until the minimum has been reached. In order to keep ϵ small enough so that the linearization is justified, and large enough to get rapid convergence, the degree of non-linearity is tested after each step. We do this with the help of the test sum

$$T = \Delta s^T W \Delta s + \Delta r^T V \Delta r \quad (H5)$$

which is calculated after every step.

In this expression V is a diagonal weight matrix whose elements are chosen such that if all of the elements of Δr and Δs are of the order of their desired accuracy, all of the terms on the right side of Eq. (H5) contribute, on the average, about equally to T. To test the linearity, the actual change ΔT of T after every step is compared with the change ΔT_{lin} that would be expected if the problem were linear. According to the value of $R = 1 - \Delta T / \Delta T_{lin}$ an adjustment of ϵ is made. The initial ϵ is always chosen to be a negative integer power of 2, and ϵ is increased in the next step by a factor of 2 (unless ϵ is already 1) for $0 < R < 0.075$, ϵ remains unchanged for $0.075 < R < 0.30$, and ϵ is decreased by a factor 2 in the

next step for $R > 0.30$. If $R > 0.60$, the last evaluation of the system is rejected and repeated with the new reduced ϵ .

The choice to change ϵ by factors of 2, and of the numerical limits that determine when ϵ has to be changed are arbitrary. All tests prior to the release of this program indicated satisfactory convergence for the procedure described here. The starting value generally used for ϵ is 0.5. Using the Δp given by Eq. (H4), but actually applying $\epsilon \Delta p$ to the linearized system,

ΔT_{lin} becomes, with $\underline{\Delta S} = \underline{\Delta S}_{old} - M\epsilon \Delta p$ and

$$\underline{\Delta r} = \underline{\Delta r}_{old} - N\epsilon \Delta p,$$

$$\begin{aligned} \Delta T_{lin} = & (\underline{\Delta S}^T - \epsilon \Delta p^T M^T) W (\underline{\Delta S} - \epsilon M \Delta p) - \underline{\Delta S}^T W \underline{\Delta S} \\ & + (\underline{\Delta r}^T - \epsilon \Delta p^T N^T) V (\underline{\Delta r} - \epsilon N \Delta p) - \underline{\Delta r}^T V \underline{\Delta r} \end{aligned}$$

Replacing, according to Eq. (I4), $N \Delta p$ by Δr ,

$$\begin{aligned} \Delta T_{lin} = & -\epsilon \underline{\Delta S}^T W M \Delta p - \epsilon \Delta p^T M^T W \underline{\Delta S} + \epsilon^2 \Delta p^T M^T W M \Delta p \\ & - (2\epsilon - \epsilon^2) \underline{\Delta r}^T V \underline{\Delta r} \end{aligned} \quad (H6)$$

Since the first term is the transpose of the second term, and using Eq. (H4) again

$$\underline{\Delta S}^T W M \Delta p = \Delta p^T (M^T W \underline{\Delta S}) = \Delta p^T (M^T W M \Delta p + N^T \lambda)$$

so that finally, with $\Delta p^T N^T = \underline{\Delta r}^T$,

$$\begin{aligned} \Delta T_{lin} = & -(2\epsilon - \epsilon^2) \Delta p^T M^T W M \Delta p - 2\epsilon \underline{\Delta r}^T \lambda \\ & - (2\epsilon - \epsilon^2) \underline{\Delta r}^T V \underline{\Delta r} \end{aligned}$$

It should be noted that for a least squares fit without restraints, ΔT_{lin} , which is then identical with the change of S , is always negative unless the minimum has been reached. This means

that if M_{WM}^T is not singular, the real least squares sum will always be improved if ϵ has been chosen small enough so that the linearization is justified. The same also applies when $2\epsilon\Delta p^T\lambda$ is positive or small enough compared to the other terms.

If, with Eq. (H5) and Eq. (H6), $T + \Delta T_{lin}$ is calculated for $\epsilon = 1$, one obtains the final least squares sum that would be obtained if the problem were truly linear. In all the applications of the method so far, it has turned out that this anticipated sum, calculated from the initial system configuration, has agreed within a factor of 10 (most of the time the agreement was within a factor of 2) with the least squares sum that was ultimately obtained after the many iterations that were necessary because of the non-linearity of the problem. For reasons outlined³ the authors of PISA feel that it might be more economical in many cases to calculate the anticipated final sum for a number of different starting points, and then follow up the one that promises to be adequate for the purpose. Although it must be said that there is a possibility of being led astray by the use of this anticipated sum, it can be a useful tool to decide what kinds of variables, and how many, should be used to solve a particular problem. This applies especially when investigating the effectiveness of new or unusual parameters. The use of the anticipated sum can also save valuable time if the analysis program has some restrictions that allow, for instance, some parameter to only vary discontinuously. The anticipated sum can then be calculated, using the continuously varying parameters, for different values of the discontinuously variable parameter, and then follow up on the most promising starting configuration.

Termination of Iteration Procedure

The criteria that can be used to end the program run are many-fold and can be easily changed. The following procedure has been used successfully. The program execution is terminated when one of the three following conditions is fulfilled:

1. When S or T is smaller than a value that is given as input data: this means terminating the program when the system is good enough for the intended purpose.
2. When $\epsilon > 1/4$ and when there is less than 10% difference between the last S or T and the anticipated final least squares sum, calculated from the last configuration. Since the linear approximation is a good one when the relaxation criteria allow a value for ϵ that is equal to or larger than $1/4$, the anticipated least squares sum at this stage agrees very well with the finally obtainable sum. This criterion, therefore, means that one stops when one has reached the finally obtainable least square sum within about 10%.
3. When the relaxation factor criterion requires $\epsilon < 1/128$, indicating that the problem is extremely nonlinear and that some other choice of variables might be more appropriate.

Computation of M and N

When analytical expressions are available for the elements of the matrices M and N , defined by Eq. (H2a) and (H2b) they can be used directly to calculate M and N . Often, however, there expressions

are not known and the derivatives in Eq. H2a, H2b have to be replaced by difference quotients. This replacement can be preferable even if the analytical expressions are known, when the calculation of the derivatives is complicated. In the following discussion, the calculation of M is described. That for N follows an identical procedure.

When replacing $\delta S_n / \delta p_m$ by $\Delta S_n / \Delta p_m$, it is important that Δp_m be small enough so that higher order terms can be neglected. However, Δp_m should also be large enough so that unavoidable computational errors are negligible. When this method is applied to a new kind of system,³ a table is made of $\Delta S_n / \Delta p_m$ for a series of values of Δp_m that increase by a factor of 2 and that covers several orders of magnitude of Δp_m . To calculate $\delta S_n / \delta p_m$, a value of Δp_m is used that is the geometric mean of the values of Δp where $\Delta S_n / \Delta p_m$ changes systematically (effect of higher order terms) and the values of Δp where $\Delta S_n / \Delta p_m$ varies in a random fashion (computational errors). The authors have not automated nor incorporated this process to determine the value that should be used for the calculation of $\delta S_n / \delta p_m$ since, with the described process, one can usually find a simple "ground rule" for the magnitudes of the different δp_m that have to be used.

It should be noted that inaccuracies in calculating the elements of M and N show up most markedly when close to a minimum. This can easily be seen by assuming that a minimum has been reached and then asking the program to improve the system performance. If the matrices M and N are very accurate, the recommended Δp will be zero:

this however will not be the case if M and N are inaccurate and the linearity test will show a strong 'non-linearity' which is not caused by non-linearities but by inaccuracies in M and N. This is the reason that the program is terminated when the anticipated least squares sum has been reached to within 10%.

For completeness it is worth mentioning that there is a more general way to calculate M with the difference quotient method: if a system has some known intrinsic symmetries, it might be more advantageous to change more than one component of Δp at a time. If one calculates the vectors Δs that result from different vectors Δp , and arrange the vectors Δs and Δp in two matrices ΔS and ΔP such that column C of ΔS describes the changes resulting from column C of ΔP , then from Eq. (H2a)

$$\Delta S = M \Delta P$$

Introducing as many linearly independent vectors Δp as Δp has components, ΔP is a non-singular square matrix and M can be expressed by

$$M = \Delta S \times \Delta P^{-1} \quad (H7)$$

The method that was described first is the trivial case in which ΔP is a diagonal matrix. The general Eq. (H7) can be particularly useful when one uses PISA not in conjunction with a computer model, but a real physical system. In that case M is measured and PISA used to optimize the performance of the system. When that system has symmetries, they can often be used to simplify measurements and make them more accurately and Eq. (H7) makes the utilization of these symmetries possible.

When the relation between the primary parameters and the secondary variable is a complicated one, most of the time for the

optimization is spent calculating M and N. Substantial savings in computer time could be made by using every matrix one more time if the relaxation criteria after the first use of the matrices did not dictate a change of ϵ . Although more frequent use of the same matrix might save more time in some instances, indications are that in general the trend reverses.

Appendix I

Input and Output for the WOLFLOT Program

The typical input file for the WOLFLOT program would have some comments as the first few lines so as to identify the plotted output to be generated. Here after the changes (if any) to the PCON array are added:

```

$TEST$
S
$ PLOT DATA FOLLOWS - $
-6.00 6.00 -2.0 8.00 $ XMIN, XMAX, YMIN, YMAX $
1.0 $ SCALE FOR PLOT $
2 $ IPSNT $
24 $ NO OF EQUIPOTENTIALS $
0.0 800000.0 $VPMIN, VPMAX $
1 $ ROTATE THE PLOT $
1 $ PLOT GROUP ONE $
1 60 1
-1 -1 -1
1 $ PHASE PLOT BEAMS $
1 60 1 $ INCLUDE ALL BEAMS $
-1 -1 -1
0 $ DUMP NUMBER FROM WHICH DATA MUST BE READ $
*18 2
*41 1 $NO GRID ON PHASE PLOT $
*44 2 $ ONE PHASE PLOT $
*121 3.00 $XMAX FOR PHASE PLOT $

```

```

*122 100.0 $ THETA (MRAD)-MAX $
*123 -3.00 $ XMIN $
*124 -100.0 $ THETAMIN $
*125 1 14 15 14 $ POSITION OF PHASE PLOT $
1 19 15 19
S

```

It is very important to input to the WOLFPLLOT program the dump number from which the problem data required is to be read. This is the same dump number that has been used throughout the entire calculation.

The PIPLLOT input file to WOLFPLLOT is given on the next page. The output generated is then included after the PIPLLOT listing. Figures 4, 5, 17, 18 and 19 illustrate the plots obtained.

```

$test$
s
$plot data-- xmin,xmax,ymin,ymax$
-5.00 5.00 -2.0 0.00
200.0 $scale for plot, inactive$
2 $plot options: -1 for b, 0 for b&p, 2 for b,p&m$
20 $number of p intervals$
0.0 00000.0 $xmin,ymax for p plot$
1 $rotate the plot$
1
1 60 1
-1 -1 -1
1
1 60 1
-1 -1 -1
0 $dump$
*18 4 $plot b,p&m$
*41 1 $no grid in phase plot$
*44 2 $two phase plots$
*121 3.00 $xmax for phase plot$
*122 100.0 $theta max$
*123 -3.00
*124 -100.0
$K1 11 K2 12 for phase plots$
1 14 15 14
1 19 15 19
s

```

1ul14m 8968 mesh points 05/03/84 13:48:19 d

```

$test$
$
$plot data-- xmin,xmax,ymin,ymax$
-6.00 6.00 -2.0 0.00
200.0 $scale for plot, inactive$
2 $plot options: -1 for b, 0 for b&p, 2 for b,p&m$
20 $number of p intervals$
0.0 00000.0 $xmin,vmax for p plots$
1 $rotate the plots$
1
1 60 1
-1 -1 -1

```

number of plot groups= 1

group number	1	starting beam number	ending beam number	increment
1		1	60	1
1 60 1				
-1 -1 -1				

number of phase groups= 1

group number	1	starting beam number	ending beam number	increment
0 \$dumps\$		1	60	1

```

*18 4 $plot b.plt$
*41 1 $no grid in phase plot$
*44 2 $two phase plots$
*121 5.00 $xmax for phase plot$
*122 100.0 $theta max$
*123 -3.00
*124 -100.0
$*1 11 k2 12 for phase plots$
1 14 15 14
1 19 15 19
$

```

```

time used: cpu = 0.863  io = 1.790  sys calls = 0.040 sec
sum total = 0.845 min

```

APPENDIX J²Ion Trajectories

Since the scalar potential is computed at the mesh points of an irregular triangular mesh, computation of the ion trajectories is basically accomplished if one traces an ion from the intersection with one side of a triangle through the triangle to the next intersection with a side of the same triangle. Every traced ion is actually representative of a current (per unit length in the direction perpendicular to the x-y plane), and one needs therefore, in addition to "updating" information of initial conditions for the equations of motion, the total charge present in the triangle, and its "center of mass". To obtain the ion trajectory, we assume again that the electric field is constant within the triangle. After solving the trajectories first with only this electric field present, we then generalize that solution and take into account to first order a constant magnetic field in the direction perpendicular to the x-y plane. We use complex arithmetic to formulate and solve the trajectory equations.

If $2g$ represents the complex representation of the electric field multiplied by the charge to mass ratio of the ion, the equation of motion is:

$$\ddot{z} = 2g \quad (J1)$$

Indicating initial conditions by the subscript 0, we get from Eq. (J1):

$$\dot{z} = \dot{z}_0 + 2gt \quad (J2)$$

$$z = z_0 + \dot{z}_0 t + gt^2 \quad (J3)$$

The electric field is easily obtained from Eq. (A9):

$$\vec{E} = \int_3 \vec{c}_2 v_2 / h_2 = \int_3 \vec{c}_2 b_2 v_2 / (b_2 h_2) \quad \sim$$

Switching to complex representation, it is clear from Fig. 22 and 23 that

$$E = \int_3 i(z_3 - z_1) v_2 / (b_2 h_2) .$$

With $u = b_2 h_2 / 2$ representing the area of the triangle (Eq. (D3))* we get

$$E = \frac{1}{2u} \int_3 v_1 (z_2 - z_3) = 2g \text{ (m/e)} . \quad (J4)$$

To find the intersection of the trajectory, Eq. (J3), with one side of the triangle, say the one connecting point 1 to point 3, we set

$$z = z_0 + \dot{z}_0 t + g t^2 = z_1 + (z_3 - z_1) \cdot u .$$

The time of intersection follows from the condition that u is real:

$$u |z_3 - z_1|^2 = (z_3 - z_1)^* (z_0 - z_1 + \dot{z}_0 t + g t^2) .$$

With

$$b_0 = \text{Im}((z_0 - z_1)(z_3 - z_1)^*) ,$$

$$b_1 = \text{Im}(\dot{z}_0(z_3 - z_1)^*) ,$$

$$b_2 = \text{Im}(g(z_3 - z_1)^*) ,$$

we obtain for the time τ_4 of intersection of the trajectory, originating at z_0 on side 1-2, with side 1-3:

$$b_2 \tau_4^2 + b_1 \tau_4 + b_0 = 0$$

$$\tau_4 = -\frac{b_1}{2b_2} \pm \sqrt{\left(\frac{b_1}{2b_2}\right)^2 - \frac{b_0}{b_2}} .$$

$$* \quad 2a = \sum_3 x_1(y_2 - y_3) = -\sum_3 y_1(x_2 - x_3) \quad (D3)$$

When the expression under the square root is negative, an intersection does not exist. Among the existing intersections with all three sides of the triangle, one is obviously only interested in the one with the smallest positive time t_4 . To avoid tests for the signs of t_4 , it is easier to find the largest value of $1/t_4$, needing only to pick the largest of the three values

$$1/t_4 = -\frac{b_1}{2b_0} + \sqrt{\left(\frac{b_1}{2b_0}\right)^2 - \frac{b_2}{b_0}}. \quad (J5)$$

In the expression for the intersection with the side of the triangle from which the trajectory starts, $b_0 = 0$ and Eq. (J5) has to be replaced by

$$1/t_4 = -b_2/b_1. \quad (J6)$$

Having thus found the correct t_4 for the intersection, the new starting point and velocity are obtained by using t_4 for t in Eqs. (J2) and (J3).

In the following discussion of charge deposition, when talking about current charge, etc., it should be understood that these terms mean the mentioned quantity per unit length of the problem in the direction perpendicular to the x-y plane.

With each trajectory is associated a current I . The relation between I and the charge ρ per unit length of trajectory is obviously

$$I = \rho \cdot |z| = \rho v = \rho ds/dt.$$

The total charge residing in a triangle is then given by

$$q_0 = \int \rho ds = I \cdot \int dt = I \cdot t_4, \quad (J7)$$

and its "center of mass", $\bar{z}$, also needed for the solution of Poisson's

equation, is given by

$$\begin{aligned}\bar{z} + q_0 &= \int z \rho dt = 1 \int_0^{t_4} z dt, \\ \bar{z} &= \int_0^{t_4} z dt / t_4\end{aligned}\quad (J8)$$

With Eq. (J3), we obtain

$$\bar{z} = z_0 + \dot{z}_0 t_4/2 + g t_4^2/3. \quad (J9)$$

When a homogeneous magnetic field B in the direction perpendicular to the x - y plane is present, the equation of motion becomes

$$\ddot{z} = 2g - iC\dot{z}. \quad (J10)$$

C is the cyclotron frequency (B times the charge to mass ratio) of the ion species under consideration. First integration of Eq. (J10) gives

$$\dot{z} = \dot{z}_0 + 2gt - iC(z - z_0). \quad (J11)$$

While this equation can be integrated in closed form, this is not advisable since the solution involves trigonometric functions, and approximations have to be made anyway for the calculation of t_4 . We therefore carry from now on only terms up to first order in C . It turns out that that is a good approximation as long as

$$(Ct_4)^2/6 \ll 1 \quad (J12)$$

With this approximation, we get from Eq. (J11):

$$z = z_0 + \dot{z}_0 t + gt^2 - iC(\dot{z}_0 t^2/2 + gt^3/3). \quad (J13)$$

Defining, in addition to h_0 , h_1 , h_2 the quantities

$$h_3 = Re\{z_3 - z_1\}^2/3,$$

$$b_4 = \operatorname{Re} z_0 (\dot{z}_3 - z_1)^* / 2 ,$$

the condition for intersection of line 1-3 becomes

$$b_2 t^2 + b_1 t + b_0 - C(t^3 b_3 + t^2 b_4) = 0 .$$

Introducing $1/t_4 = \omega_4$, we get

$$b_0 \omega_4^2 + b_1 \omega_4 + b_2 - C(b_3 + b_4 \omega_4) = 0$$

With ω_{40} representing the right side of Eq. (J5) or (J6), we get to first order in C

$$1/t_4 = \omega_{40} + C/\omega_{40} + (b_3 + b_4 \omega_{40})/(b_1 + 2b_0 \omega_{40}) . \quad (J14)$$

With this value of t_4 , the endpoint of the trajectory in the triangle is given by Eq. (J13), and the endpoint velocity follows from Eq. (J11) and (J13). The total charge deposition is again given by Eq. (J7), and its "center of mass" is obtained from Eqs. (J8) and (J13):

$$\bar{z} = z_0 + z_0 t_4 / 2 + g t_4^2 / 3 + i C (2 \dot{z}_0 t_4^2 + g t_4^3) / 12 . \quad (J15)$$

XBL 8510-4466

Figure 1 : Structure of the LATTICE code

XBL 8510-4465

Figure 2: Problem outline - test example

XBL 8510-4464

Figure 3: Logical diagram for example

XBL 8510-4475

Figure 4: Phase plot at L = 14

XBL 8510-4474

Figure 5: Phase plot at L = 19

XBL 8510-4467

Figure 6: Allocation of regions in example

XBL 8510-4476

Figure 8: Beam and sub-beam structure

XBL 8510-4469

Figure 9: Beam emitter example

XBL 8510-4477

Figure 10: Trapezoidal bump

XBL 8510-4479

Figure 11: Rectangular bump

XBL 8510-4478

Figure 12: Triangular bump

XBL 8510-4480

Figure 13: Bell-shaped bump

XBL 8510-4481

Figure 14: Half-bell shaped bump

XBL 8510-4482

Figure 15: Half-bell shaped bump

XBL 8510-4470

Figure 16: Non-isolated and isolated electrodes

XBL 8510-4483

Figure 17: Output beam trajectories

XBL 8510-4484

Figure 18: Geometry and equipotentials

Figure 19: Triangular mesh generated by LATTICE

XBL 8510-4486

Figure 20: Identification of quantities in a triangle

XBL 8510-4487

Figure 21: Identification of quantities in a triangle

XBL 8510-4473

Figure 22: Identification of quantities in a triangle

XBL 8510-4472

Figure 23: Identification of quantities in a triangle

XBL 8510-4471

Figure 24: Identification of quantities in a triangle

APPENDIX K

Addenda and Errata

p.3 The electrons mentioned in this page are those described by a Boltzmann distribution in a plasma. However, it should be noted that in some other instances electrons are treated as individual particles with mass and charge and their trajectories are traced.

p.16,17 It should be noted that a general user has no control over the variables MAXCNT, NOMAX.

p.30 Some of the stated options abouts the variable IPSENT may not be true. The outcome due to the interaction of IPSENT and PRBCON(18) is quite complicated. Here are some examples: (B=Beam,P=Potential,and M=Mesh)

IPSENT	*1B	plots
-1	2	B in one plot
-1	3	BP in one plot
0	2	B,P in two separate plots
1	3	BP,M in two separate plots
2	4	BP,M,P in three separate plots

p.33 It should be noted that all those phase space statistical quantities are calculated only for those beams whose phase space quantities lie within the limits of the plots. Those limits are either set internally by the program or specified by the user.

p.37 WOLFCALC also needs the input file TAPE36 in case DUMP is greater than or equal to 1.

p.50 The difference between TAPE35 and TAPE36 should be noted. One of the output files of LATTICE is RECORD. It must be renamed as TAPE35 when used as one of the input files to WOLFCALC. The first time when WOLFCALC is used to calculate a particular problem, it picks up DUMPO from TAPE35, and if it runs successfully, it produces ,among other output files,DUMP1 of TAPE36. If the user wants to continue the problem, he can rerun WOLFCALC from DUMP1 of TAPE36, (TAPE35 is still needed), and then produces DUMP2. This process can be repeated.

Both TAPE35 and TAPE36 are written in binary format.

TAPE35 contains the following arrays:

PRBCON : the problem constants.
 X : the x-coordinates of all the mesh points.
 Y : the y-coordinates of all the mesh points.
 SOURCE : the potential values at the electrodes.
 INDEX : the array with bits containing various
 information about the mesh points

TAPE36 contains the following arrays in each dump:

PRBCON : the problem constants.
X : the x-coordinates of all the mesh points.
Y : the y-coordinates of all the mesh points.
A : the potential values of all the mesh points.
SOURCE : the ion charges deposited at the mesh
points.
ESAVE : the Boltzmann electron charges deposited
at the the mesh points.

p.65 The convention for boundary labels is illustrated with the following example:

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.