Precision ion optics of axisymmetric electric systems

by

Peter Varfalvy

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of
Master of Science

Foster Radiation Laboratory
Department of Physics
McGill University
Montreal, Canada

August, 1995

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

ISBN 0-612-12284-0

Canada

ABSTRACT

A comprehensive computer package for the calculation and simulation of charged-particle dynamics in electromagnetic fields has been developed and tested. The program provides a user-friendly and flexible interface for visualizing particle dynamics using phase space diagrams, which are essential for complete understanding of a beam optics system. The program performs an accurate finite difference computation of a user-defined boundary value problem (in the form of a grid) followed by a high-order Runge-Kutta numerical integration of the equations of motion to evaluate the particle dynamics within the field. The program is unique in its combination of these flexible finite calculation techniques with the parallel processing of particle ensembles in order to display phase space diagrams.

After extensive testing, the program has been used to design a low emittance ion source and an ion beam deceleration system for high-efficiency ion collection. The program has also been used to analyze a radiofrequency quadrupole collisional focusing system using ion mobility concepts.

RÉSUMÉ

Un logiciel compréhensif qui calcule et simule la dynamique des particules chargées dans des champs électromagnétiques a été développé et testé. Le programme est facile à utiliser et très utile pour évoquer les images de la dynamique des particules en utilisant des diagrammes d'*action*, qui sont essentiels pour la compréhension complète des systèmes d'optiques ioniques. Le programme calcule une différence finie précise d'un *problème aux valeurs limites* définies par l'utilisateur (sur une grille), suivi par une intégration numérique *Runge-Kutta*, d'ordre-élevé, des équations de mouvement pour évaluer la dynamique des particules dans le champ. Le programme est unique dans la manière dont il intègre ces techniques flexibles de calcul finies avec le calcul parallèle de toute les trajectoires d'un ensemble de particules pour visualiser les diagrammes d'*action*.

Après des tests consciencieux, le programme a été utilisé pour établir le plan d'une source d'ions à base émittance et d'un système de décélération de faisceau d'ions pour la collection d'ions avec haute efficacité. Le programme a aussi été utilisé pour faire l'analyse d'un système (avec un champ quadrupolaire oscillatoire à convergence par collision) qui utilise des concepts de mobilité d'ions.

To Helen, with all my love and soul.

TABLE OF CONTENTS

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to, first and foremost, thank Professor R. B. Moore and Dr. M. D. Lunney who have provided much insight for the content and organization of the thesis; their help was invaluable and essential for the completion of this thesis. Additional thanks go to Professor R. B. Moore for allowing the use of figures 2.1 and 2.2 in the thesis. Furthermore, I would like to thank Mr. A. M. Ghalambor Dezfuli for allowing the use of figure 4.11 as well as having helped considerably for the discussion on simulating beam profiles in section 4.2.1. Much time has been spent researching and writing this thesis, all of which would have not been possible without the kind encouragements and patience of many people, especially the entire staff of the physics department including Foster Radiation Laboratory. Special thanks go to two summer students: Rui Lopes and Debbie Reynolds, who have both helped with the implementation of a general boundary condition input and a user-friendly interface for the McGill FDC/RKI program. I would like to especially thank Helen Siourbas for translating most of the abstract and giving me her unconditional love and support: this thesis would have never been finished without her. Finally I would like to thank my parents for their emotional (and financial) support as well as for their undying love and understanding; To them, and everyone else, both mentioned and not mentioned here, who have encouraged and supported me, i owe my life.

1. Introduction

1.1 Charged Particle Beam Optics

Many studies and applications of fundamental physics require the use of electromagnetic fields for guiding charged particles. Perhaps the best known examples are particle accelerators and television picture tubes. Systems for guiding, or transporting, charged-particle beams can be quite complex, requiring detailed simulation for their realization. Consequently, techniques for such simulation have been significantly developed, generally falling into the category of "beam optics".

A major application of beam optics is found in the field of nuclear physics using isotope separator facilities. At such facilities, radioactive beams are produced by bombarding a target with a high energy projectile. The radioisotopes are then ionized and extracted from the target as a particle beam. As these facilities are quite rare and in very high demand, they include a network of paths so that the beam may be transported to a number of experimental stations. An example of an isotope separator facility, in this case ISOLDE at CERN, is shown in figure 1.1. The area after the mass separator itself is called the experimental switchyard. Various experimental programs (users) install apparatus that is customized for particular types of measurements. The central beam line is split into smaller beam lines using chambers containing electrostatic parallel plate deflectors. At any give time, the beam is deflected along a certain path and transported to only one installation. In the straight sections of the beam lines there are quadrupole focusing elements to keep the beam from diverging and being lost.

The positioning of these standard beam line elements is chosen using a certain type of beam optics calculation methods, most of which fall under the category of transfer matrix techniques. These are applied in cases where the effects of certain electrode geometries on ion trajectories can be approximated analytically. The beam-line elements have characteristic transfer matrices which are multiplied together with a beam profile

matrix. The desired focal point is specified and the overall matrix is inverted to determine the element positions. An important figure of merit for a beam transport system is the *emittance* of the beam which characterizes the beam quality using the spot size and the divergence. Emittance is derived from general phase space arguments which will be developed in the following chapter. All transfer matrix beam transport calculations are based on phase space considerations.

In the individual experimental areas, usually the electromagnetic optics are more complicated as they are tailored to the particular experimental technique and it becomes impossible to define characteristic and analytical elements as such. In these cases, the particle dynamics must be evaluated by direct, numerical methods. In general, such methods are comprised of two main features: an accurate determination of the electromagnetic field, followed by an evaluation of the resultant dynamics of the particles subjected to these fields. Again, several methods exist, generally known as finite calculation techniques. Unlike transfer matrix methods, finite calculation methods tend to treat dynamics on a particle-by-particle basis rather than using phase space considerations. Thus, it becomes difficult to gauge the performance of apparatus in conjunction with the particle beam and information that can be critical, is not displayed.

In any field of endeavour, it makes no sense to "re-invent the wheel". When a charged particle optics problem needs to be solved, one naturally looks for the tools that are already available. In the next two sections, a brief review is given concerning both transfer matrix and finite calculation techniques. From these should emerge the fact that there is a lack of a tool providing the advantages of finite calculations with the formulations and especially, display, of phase space considerations that come out of transfer matrix approaches. The last section of this chapter will therefore outline the contribution of this thesis in this regard.

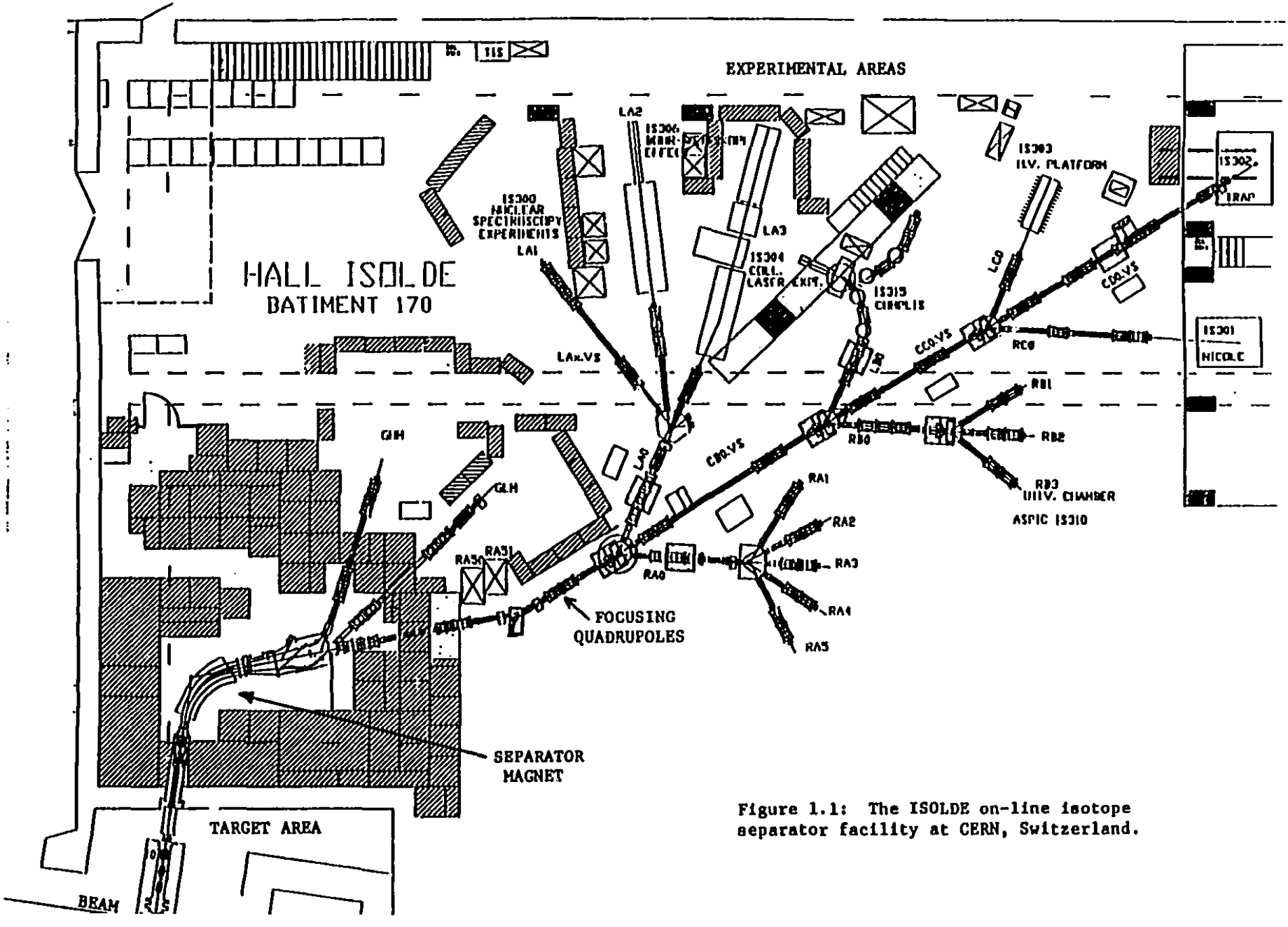


Figure 1.1: The ISOLDE on-line isotope separator facility at CERN, Switzerland.

1.2 Transfer Matrix (Algebraic) Methods

There are many available programs that use transfer matrix methods for ion trajectory simulations of accelerators and beam transport facilities. One of the first such programs was Brown's TRANSPORT program [BRCI]. It was developed through the efforts of many people working at various laboratories around the world (CERN, SLAC, and NAL) in the 1960's and early 70's. The program was first written in a language called BALGOL and was later translated into FORTRAN. The TRANSPORT program was primarily intended for the design of static-magnetic beam transport systems. Therefore, in its simplest form, the geometries that could be handled by this program were limited to a sequence of paraxial magnetic elements and the spaces separating them. However, the TRANSPORT program also allowed users to specify their own calculations as elements in order to simulate other effects such as the presence of electric fields.

The TRANSPORT program is referred to as a first and second order matrix multiplication program. For the first order calculation, each element along the beam line has a first order transfer matrix assigned to it. The product of all those matrices yields the mapping of a collection of particles from the beginning to the end of a beam line in six-dimensional phase space. (For each of the three directions, the corresponding phase space coordinate is represented by the two components of momentum versus displacement. Chapter 2 provides details on phase space derivation.) Thus, the first order transfer matrices are all 6 by 6. The second order correction is obtained by including cross terms of the 6 dimensions of phase space in the analytical approximations. Overall, the mapping can be represented mathematically as follows:

$$X_i' = \sum_j R_{ij} X_j + \sum_{jk} T_{ijk} X_j X_k \,, \qquad (1.1)$$

where **X** is the vector representing the 6D point in phase space, **R** is the first order transfer matrix, and **T** is the second order transfer matrix that behaves like a tensor.

The TRANSPORT program was very popular in accelerator simulations because it allowed for an entire phase space ellipse (see chapter 2) to be specified and mapped

through a system of electrodes and magnets. The program was also able to fit the transfer matrices (and hence the physical parameters related to them) in order to obtain a desired beam. Unfortunately, TRANSPORT had a few drawbacks. It was not always very accurate because it was limited to second order (the order of the approximation depends on, and is limited by, the analytical solutions). The authors of the program even suggested that if greater precision than what TRANSPORT calculations permit is required, then ray-tracing programs that integrate the equations of motion should be used [BRCI]. Furthermore, TRANSPORT's input and output was entirely text and not always easily understood. Finally, TRANSPORT was usually only available on mainframes, making it almost impossible to be used anywhere, anytime, and interactively.

Because of the popularity and usefulness of TRANSPORT, higher-order programs such as MAFIA [KRWE], PARMELA [MCDO], MARYLIE and COSY INFINITY were created. All these codes calculate the motion of charged particles in phase space by using transfer maps combined with other techniques to render them more generally applicable. MARYLIE [DFBW] and COSY INFINITY [BERZ] use special techniques which make use of Lie polynomials or differential algebras.

MARYLIE is a third-order mapping program that was developed at the University of Maryland in the early 1980's for the design of accelerators and storage rings [DFBW]. Like TRANSPORT, MARYLIE is specifically designed for magnetic beam line elements. On the other hand, COSY INFINITY is a very general program (allowing electrostatic sector fields and multipoles to be specified directly as elements in a beam line) which can compute to an arbitrary order (as high as the user wishes) the maps for standard beam line elements [BERZ]. In order to be able to compute maps to arbitrary order, COSY INFINITY uses a numerical integration technique (a seventh order Runge-Kutta with adaptive step size) to integrate the equations that arise from the differential algebraic description [BERZ]. Given all their power, these codes are very computationally extensive and not readily available on simple platforms such as personal computers.

A transfer matrix program very similar to TRANSPORT that can calculate maps up to third-order was developed for personal computers (IBM compatibles) in the 1980's: GIOSP, which stands for General Ion Optical Systems on PC's [PRZE]. The

program was based on an earlier version, known simply as GIOS, from the 1970's that ran on mainframes. Both programs were written in FORTRAN; GIOSP was more specifically compiled on a PC with the Lahey FORTRAN-77 compiler.

GIOSP is particularly useful as it can be run on a PC. It also has an excellent capability for displaying graphics of the resulting calculations. Not only is the beam envelope displayed as it traverses the optics system, but the beam phase space profiles can be displayed at any position. This gives the user critical information on how the beam enters the apparatus (that is, the beam spot size and the energy spread).

All the transfer matrix programs mentioned make use of phase space concepts that are essential for the accurate design of beam optics systems. The most recent codes also provide graphical outputs for the computed beam profiles. Unfortunately, these programs remain inflexible when electrode systems stray too far from the standard geometries for which these programs were written. For such situations, finite calculation methods become a valid alternative to transfer matrix maps.

1.3 Finite Calculation Methods

Finite calculation methods essentially solve the boundary value problem for the potential which is defined by the electrode system geometry. The equations of motion of the charged particle subjected to the electric field are then directly integrated by a numerical algorithm to obtain the ion trajectories. Numerical methods approximating the continuous solution of a potential distribution may be of integral or differential form depending on the initial format of the equations used. A good review of solution methods for electromagnetic problems is found in Lowther & Silvester [LOSI].

In the integral approach, the potential at a point inside a volume may be determined directly from a knowledge of the source distribution by the application of Green's theorem. By dividing a region into small surface elements over which the potential and its normal derivative are known, a matrix of linear equations can be

constructed to calculate the potential at each of a set of specified points within the volume. The potential at any point requires knowledge of only the boundary conditions and the accuracy at any point is determined by the accuracy of the integration. Integral, or charge density methods, can be very accurate but tend to become quite computationally intensive for general cases. ELECTRA [ISMA] is an example of a program that uses such techniques. MAFIA, which was mentioned earlier, also uses integration techniques.

Differential techniques, on the other hand, calculate the potential at a point via a connection matrix relating all the unknown potentials. There are two main differential methods in widespread use at present: *finite differences* and *finite elements*. With these methods, the problem area is sub-divided into a grid and the potential is determined for the various grid points. The electric field can then be derived from the potential distribution. The grids can be two- or three-dimensional, but for systems with axisymmetric symmetry or systems that have a constant cross section, 2D grids are all that are needed.

With the finite element method (FEM), the problem area is arbitrarily sub-divided into small areas, or elements. The potential is then calculated on node points forming the elements (see figure 1.2). The FEM does not approximate the differential equation directly but rather minimizes a global function of the field and attempts to provide a best fit to the entire field region. Elements can be made of 3 edges (triangles) or more, with 3 and 4 edged elements being the most common. By far the greatest advantage of FEM is the ease with which a geometry may be discretized, or "meshed". For this reason it is often employed in designing magnetic or electronic devices and machines. Furthermore, the global minimization procedure makes calculation of global quantities, for example unit capacitance, quite straightforward. On the other hand, in axisymmetric cases, there are often problems with the potential along the axis of symmetry which deteriorate accuracy in this important region. Computing packages available using FEM include MagNet, Tosca, PE2D, Ansoft and MSC. While they are targeted for design applications in magnetics, some provide options for computing trajectories but only on a particle by particle basis.

The finite difference method (FDM) is a computer implementation of an earlier manual method known as "relaxation". The approach is to produce a discrete form of the potential equations in their differential form. To do this, the area of interest enclosed by the electrode geometry and other boundary conditions is divided into a grid of successive nodes that is usually evenly spaced (see figure 1.3). The problem is solved by iteratively calculating the potential at each node from its nearest neighbours until the residual after each calculation becomes small enough. One widely used computer package using FDM is SIMION which is a very accurate and user-friendly program nicely suited for custom electrode geometries. SIMION uses a fourth-order finite difference technique to solve Laplace's equation for arbitrary planar symmetric and axisymmetric geometries. It performs a bi-linear interpolation of the grid potential values to within half a gridstep in all directions about the current particle position. Furthermore, it performs trajectory calculations on a particle by particle basis. The only problem is that with trajectories alone rather than full phase space diagrams, important information for detailed analysis of the beam dynamics is missing.

While the FDM method can be somewhat inflexible for arbitrary geometries (especially compared to FEM), it is extremely accurate and can compute highly continuous potentials along the axis. This feature is extremely important for calculating particle trajectories since the potentials are only available at the node points whereas particle trajectories will always pass through arbitrary points between nodes. Interpolating between grid points to compute the field must be done very carefully and the potential map must be as continuous as possible to avoid errors. A way of avoiding this is to derive a field value for a point off axis using a high-order multipole expansion of the potential values along the axis. The idea for using this technique came from the SLAC electron trajectory program EGUN [HERR] which was designed for the calculation of electron and ion trajectories in electron guns and lenses.

EGUN uses a variable (non-evenly spaced) grid to obtain highly continuous potentials even near electrodes. However, the high-order multipole expansion technique is used only for calculating the magnetic field. The high-order multipole expansion technique as applied to an electric field on an evenly spaced mesh is described in

chapter 2. Some comparisons between the interpolation and the expansion techniques are also given in chapter 4.

One feature of the FDM is that it remains open to wide variety of numerical integration methods to calculate the particle trajectories. The three basic methods are: Runge-Kutta, Richardson extrapolation, and predictor-corrector methods. Runge-Kutta methods are the most widely used since their algorithm is relatively simple. Furthermore, the Runge-Kutta method can be applied to wide variety of problems, even those where the other two methods usually fail [PFTV]. The authors of *Numerical recipes in C* [PFTV], recommend the Runge-Kutta integration method whenever it is not certain that one of the other two methods will work better. In cases where the FDM is applied to arbitrary problems, the Runge-Kutta numerical integration method is clearly the wisest choice.

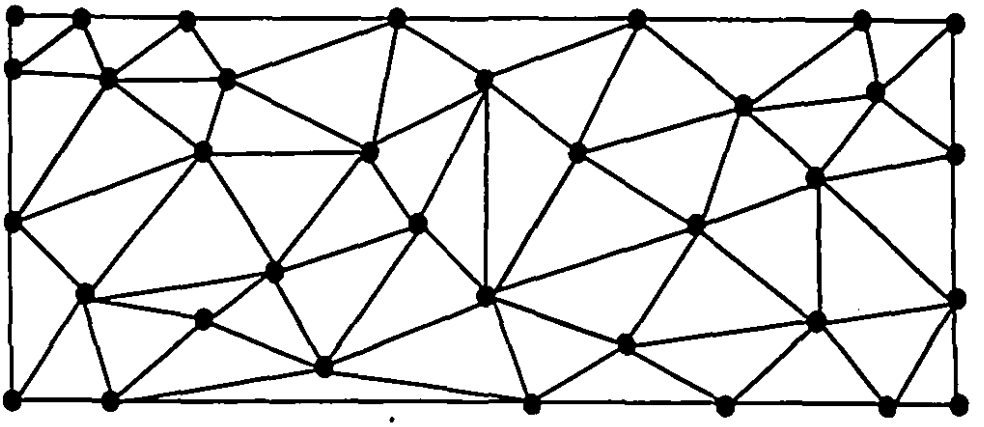


Figure 1.2: Example of a triangular finite element mesh.

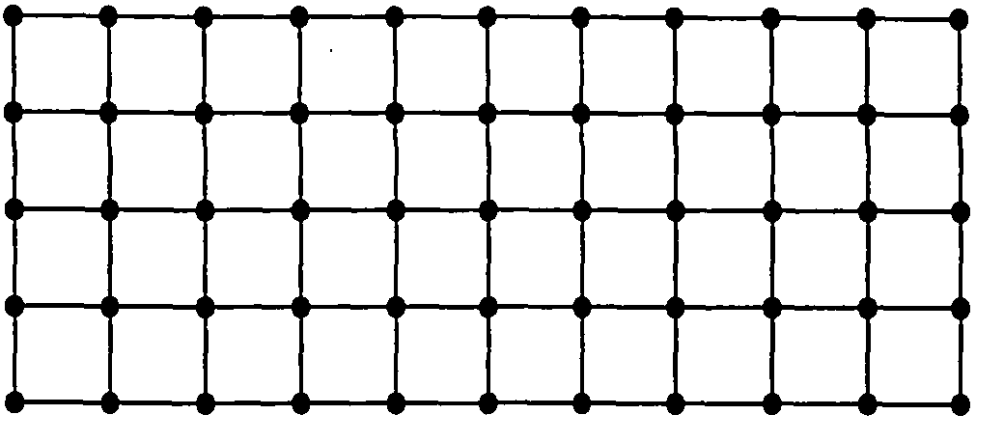


Figure 1.3: Example of an evenly spaced finite difference mesh.

1.4 Contribution of Thesis

It is clear that in the evolution of tools for the evaluation of charged particle optics and dynamics, a dichotomy persists between those designing particle accelerators and transport systems, and those designing more arbitrary and "custom" beam optics for individual applications. The main problem is in the omission of phase space concepts in the design and analysis processes using the more flexible and direct, finite calculation schemes. By simply looking at particle trajectories, important information is missing concerning the velocity components of the particle motion. On the other hand, transfer matrix methods which give a complete phase space analysis of the particle dynamics, are not suitable for arbitrary electrode geometries.

The Foster Laboratory at McGill University is involved in two particular collaborations for making precise determinations of atomic masses using the ISOLDE radioactive beam facility at CERN. From this experimental program, the use of quadrupole ion traps [MLRS] and mass filters [MGVZ] has become of paramount importance for improving the apparatus performance and continuing in this field. When this auxiliary development work was started, it became apparent that existing computer packages were grossly inadequate for the detailed analysis needed. Not only was very high accuracy required, but also a comprehensive phase space picture of the ion dynamics and other features related to superimposed, time-varying fields and pulsed beams [LWMO, LBMO].

The contribution of this thesis is therefore the creation and extensive testing of a comprehensive computer package combining the power and flexibility of finite calculation techniques with necessary calculation and display capabilities for phase space diagrams, with a modular design and user-friendly interface, to perform high accuracy charge-particle optics simulations. This computer package has the following general capabilities:

- User specification (or importation from another application) of an arbitrary electrode geometry with corresponding boundary conditions.

- An optimized finite difference calculation of the above geometry with user-specified target residual (accuracy) that generates a grid of electric potential values.
- Accurate and continuous computation of the resultant electric fields using either (1) a seventh-order multipole expansion of the axial potentials or (2) a bi-linear interpolation of the off-axis grid points.
- User specification of an arbitrary initial phase space distribution describing an ensemble of charged particles (or initial characteristics of single particles).
- A fifth-order Runge-Kutta numerical integration with adaptive step size control to completely determine the particle trajectories in phase space (or geometric space) using parallel evaluation of all particles in the ensemble.
- Specification of time-varying functions for the fields in question.
- Superposition of multiple potential maps.
- Superposition of magnetic fields.
- A fully interactive, graphic display capability for the phase space diagrams and/or single particle trajectories.
- Output of results to a file for subsequent analysis.

No other program, at least documented in the literature, exists with these capabilities. It has been used extensively in the McGill Foster group for analysis of ion sources [GDMV], a quadrupole ion trap collection system [MGDP], an ion beam collisional focusing studies using quadrupole rods at high pressure [KIMT], and a deceleration system for ion collection in a Penning Trap [ZHAO].

Chapter 2 presents the theory relevant for a description of ion dynamics using phase space considerations and brief mathematical descriptions of the finite calculation techniques as well as the high-order multipole expansion. Chapter 3 gives a detailed description of the computer package and chapter 4, a description of the program testing as well as the results obtained for some of the above applications.

2. Theory

2.1 Phase Space

The state of motion of a particle at time, t is completely specified if two quantities: displacement, $x(t)$ and velocity, $\dot{x}(t)$ are known. These quantities are considered to be the two coordinates of a point in *phase space*, $(x,\dot{x})$. For an ion beam with three degrees of freedom, the phase space is, therefore, six dimensional. For different initial conditions, the motion is described in time by different phase paths. The totality of all possible phase paths constitutes the *phase space volume*.

The concept of phase space is nicely derived from the equations of Hamiltonian mechanics (shown here in canonical form):

$$\dot{q}_i = \frac{\partial H}{\partial p_i} , \tag{2.1}$$

$$\dot{p}_i = -\frac{\partial H}{\partial q_i} , \tag{2.2}$$

where q is the *generalized position*, p the *generalized momentum* and H the *Hamiltonian*, or total energy of the collection of particles, i = 1,2,3.... If, at a given instant of time, the initial positions and momentums of all the particles in a collection are known, then the subsequent motion of the collection is completely determined (e.g., Goldstein) [GOLD].

Taking as an example a harmonic oscillator, we can define (as functions of time) the position $x(t)$ and momentum $p_x(t)$ of a mass m, oscillating with amplitude A, at frequency ω, as:

$$x(t) = A\cos\omega t , \tag{2.3}$$

$$p_x(t) = m\dot{x}(t) = -m\omega A \sin\omega t \ . \tag{2.4}$$

Eliminating t from these equations gives:

$$\frac{x^2}{A^2} + \frac{p_x^2}{m^2\omega^2 A^2} = 1 \ . \tag{2.5}$$

This equation represents a family of ellipses which constitute the phase space of the oscillator as shown in figure 2.1 [MOOR]. This area is the projection of the entire phase space volume in one direction. The angular motion around each ellipse, at frequency ω, is clockwise because when $x < 0$ the velocity is always increasing (and vice versa) as shown by equations (2.3) and (2.4). No two phase paths of the diagram can cross as the solution of the governing differential equation is unique for given initial conditions. The combined generalized coordinates of position and momentum can be considered to have the dimension of *action*. For this reason, phase space diagrams (projections of the phase space volume in one direction) are sometimes called *action diagrams*.

The total phase space volume S can be defined as the product of the phase space area in each dimension:

$$S = S_x S_y S_z = \int dx dp_x \int dy dp_y \int dz dp_z \ . \tag{2.6}$$

From beam optics comes the important quantity of *emittance*, ξ_u which is defined as:

$$\xi_u = \int \theta du \ , \tag{2.7}$$

where θ is the angle of the beam with the corresponding axis, or beam *divergence* and u is the corresponding spatial coordinate. Emittance is usually described by three components: two transverse and one longitudinal. An *emittance diagram* is generally used to characterize the components of the beam. The emittance is plotted as $\dot{u}$ versus u so the emittance is the area represented by this phase space diagram. The emittance is related to the transverse phase space components by:

$$S_x = p_o \xi_x \ ; \ S_y = p_o \xi_y \ , \tag{2.8}$$

where p_o is the central momentum of the beam.

For complex collections of numerous particles, such as an ion beam, it is clearly a practical impossibility to determine the initial conditions for each constituent ion. Since we cannot identify any particular point in phase space as representing the actual conditions at any given time, we must turn to the field of *statistical mechanics* and introduce the representation of the ion collection by an *ensemble of systems* (*e.g.*, Landau and Lifshitz) [LALI].

The phase space is a collection of points where each point represents a particular system of the ensemble. Each system in an ensemble is comprised of any number of constituents *e.g.*, particles. The ensemble has a *phase space density* that determines the number of points inside an infinitesimal hypervolume element of the six-dimensional phase space. An important result from Hamiltonian mechanics is that the phase space density of the ensemble remains constant as it moves through phase space in time. This result is known as *Liouville's theorem* [GOLD, LALI].

The geometrical consequence of Liouville's theorem is that the phase space volume will behave like an incompressible liquid drop that, squeezed one way, will bulge out the other conserving both the density and the volume. Furthermore, for linear forces (forces which do not couple the different motions) acting on the ensemble, each projection of the phase space volume, (p_x vs x, p_y vs y, p_z vs z) will also behave as an incompressible area. (For non-linear forces, coupling can occur between dimensions resulting in the projections changing area.) It is interesting to note that a similar theorem does *not* exist for three-dimensional configuration space.

An example of Liouville's theorem at work is illustrated by figure 2.2 [MOOR]. A phase space diagram representing a beam in the transverse direction is shown at position 1. As the beam drifts to position 2 at a constant momentum spread, the displacement spread increases but the overall phase space area is the same. When the beam goes through the lens, the phase space diagram is transformed at position 3 resulting

in an increased momentum spread for the same displacement spread but the area is still the same. When the beam reaches the focus at position 4, it has the same momentum spread as position 3 but the beam width is dramatically reduced so as to conserve the overall area. This figure illustrates how a tight geometric focus can only come at the expense of a large spread in momentum because of Liouville's theorem.

As previously mentioned, the product of generalized position and momentum, coordinates said to be *canonically conjugate*, has the dimension of *action*. Such a dimension is also described by energy × time. Using these coordinates, the phase space area can be rewritten as:

$$S_x = \Delta p\, \Delta x = m\Delta v\, v\Delta t = \Delta E\, \Delta t \ . \qquad (2.9)$$

Using these coordinates, a particularly useful unit for phase space is the eV-μs which equals about 1.6×10^{-25} kg-m^2/s. The eV-μs is more manageable and better lends itself to phase space representation. Lenses and voltage pedestals are always controlled by *voltage* and pulsed system operation is always controlled by *time* so the eV-μs is a more intuitive unit. The corresponding unit for momentum in this system is the eV-μs/mm from the relation $p = 2E/\omega a$.

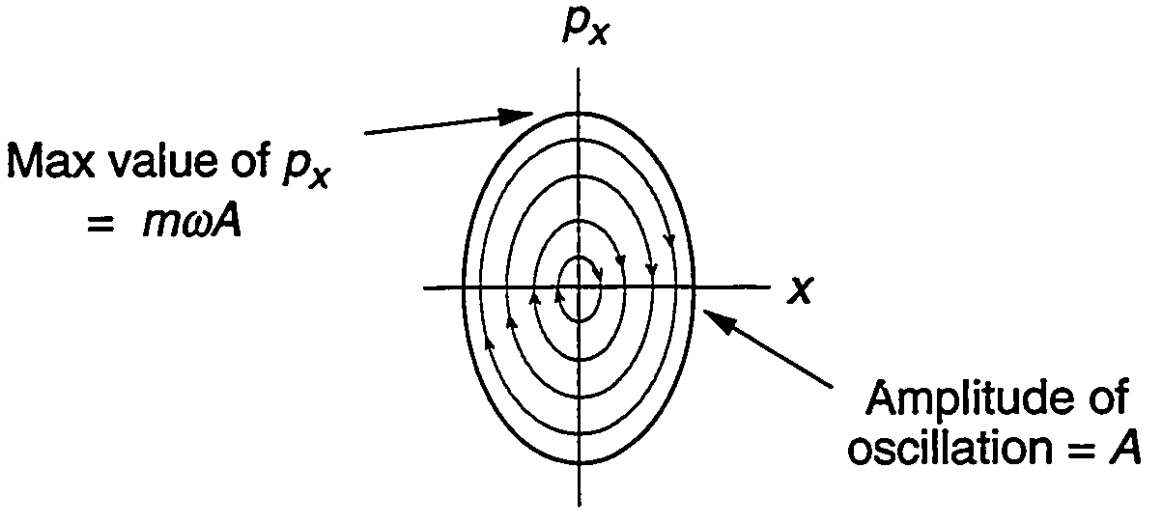


Figure 2.1: Family of ellipses constituting the phase space of a simple harmonic oscillator.

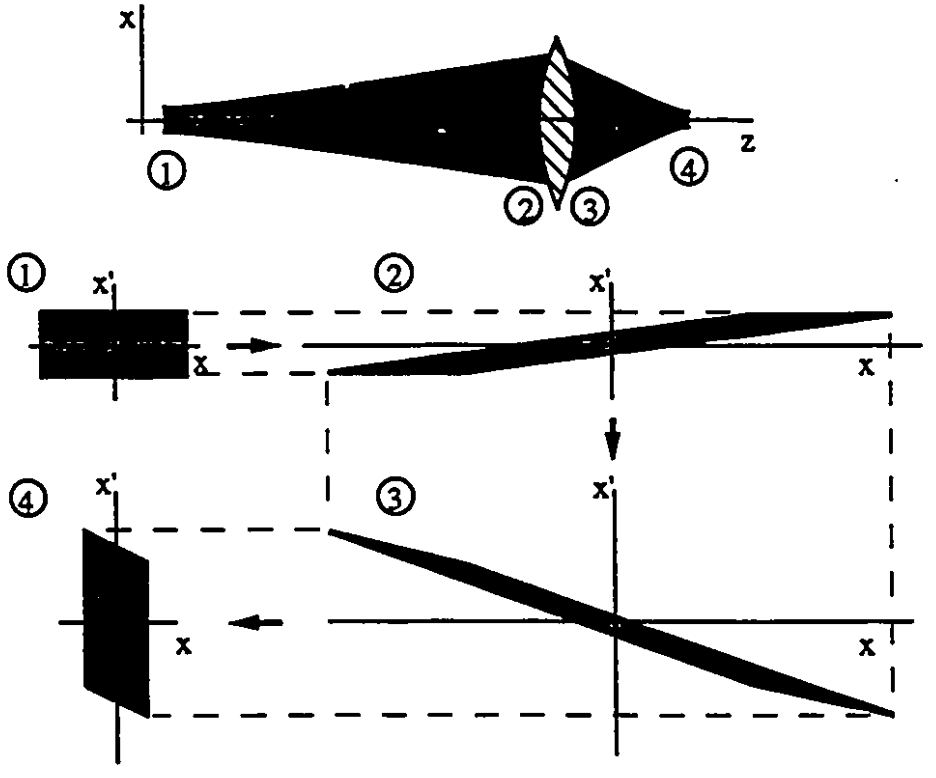


Figure 2.2: Geometric and transverse phase space representation of the focusing of a beam by a lens. A detailed description of each step is given the text.

2.2 The Finite Difference Method

The second order partial derivatives of the differential form of Laplace's equation ($\nabla^2 u = 0$) may be approximated by using the first few terms of a Taylor series expansion of u. In most problems, u will only depend on two variables, such as the cylindrical coordinate variables r and z in axisymmetric geometries which will reduce Laplace's equation to:

$$\frac{\partial^2 u}{\partial z^2} + \frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} = 0 \ . \tag{2.10}$$

Using the evenly spaced grid shown in figure 2.3 with a grid step size of h and the first three terms of the Taylor series expansion of u about u_o, the partial derivatives of u from equation (2.10) can be rewritten to give:

$$\frac{u_2 - 2u_o + u_3}{h^2} + \frac{u_4 - 2u_o + u_1}{h^2} + \frac{u_1 - u_4}{2rh} = 0 \ , \tag{2.11}$$

where $r = jh$. Equation (2.11) can be simplified to give:

$$4u_o = u_1 + u_2 + u_3 + u_4 + (u_1 - u_4)/(2j) \ . \tag{2.12}$$

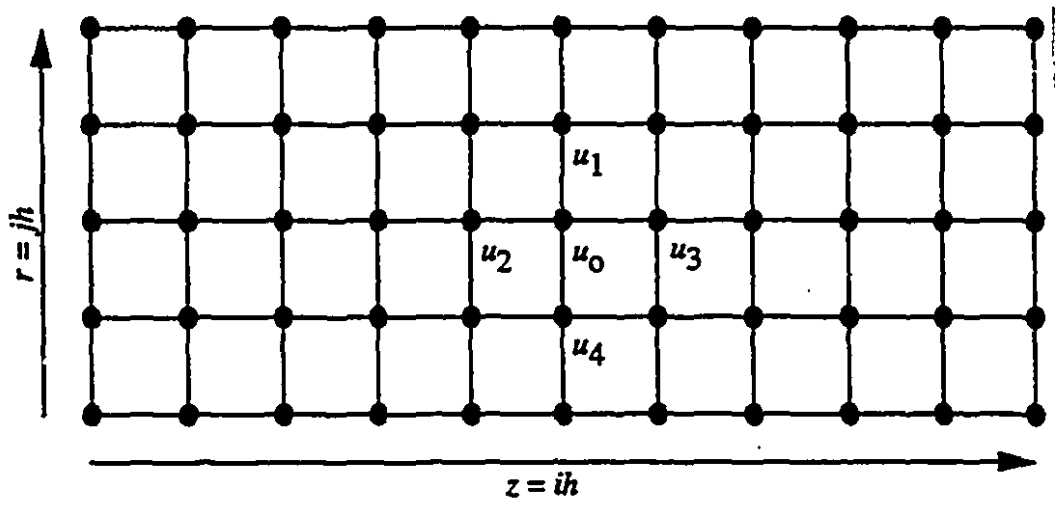


Figure 2.3: The evenly spaced grid used by finite difference methods.

Equation (2.12) forms the basis of the *four-point* relaxation method for axisymmetric systems. The value of u at any grid node can be approximated by using the values of its four nearest neighbours. Those grid points which correspond to boundary conditions are fixed to an initial value if they represent Dirichlet boundaries or set to some function of the nearest grid points if they represent Neumann boundaries. In axisymmetric geometries, the axis of symmetry (usually chosen to be along z) is special kind of Neumann boundary for which equation (2.12) needs to be modified. Due to the symmetry about the z-axis, the point below the axis must have the same value as the point above (*i.e.*, $u_4 = u_1$). By taking the limit of equation (2.10) as r approaches 0 and by using *Hôpital*'s rule on the first partial derivative term, equation (2.10) becomes:

$$\left(\frac{\partial^2 u}{\partial z^2}\right)_{r=0} + 2\left(\frac{\partial^2 u}{\partial r^2}\right)_{r=0} = 0 \; . \tag{2.13}$$

By using $u_4 = u_1$ at $r = 0$, equation (2.13) is further reduced to:

$$6u_o = 4u_1 + u_2 + u_3 \; . \tag{2.14}$$

The basic algorithm of the finite difference method (FDM), as applied to axisymmetric systems, is therefore to assign all the Dirichlet boundary conditions to the appropriate grid nodes and then approximate the value of u at each grid node by using either equation (2.12) or (2.14). In the simplest form of the FDM, this is done by continuously iterating through the 2D array of values for u and replacing them by the approximations of the current iteration. Eventually the difference between new and old values of u—referred to as the *residual*, ξ—will become very small and a solution for Laplace's equation will be obtained. The residual is actually defined as follows:

$$\begin{aligned} \xi &= u_1 + u_2 + u_3 + u_4 + (u_1 - u_4)/(2j) - 4u_o \; ; \; j \neq 0 \; , \\ \xi &= 4u_1 + u_2 + u_3 - 6u_o \; ; \; j = 0 \; . \end{aligned} \tag{2.15}$$

The finite difference method achieves much better results when modified to include *Successive Over-Relaxation* (SOR) combined with *Chebyshev acceleration*

[PFTV]. With the SOR algorithm, an *over-relaxation parameter*, ϖ is introduced to make an *over-correction* to the new grid node values of u. If the new value of u is given by u^{new} and the old value of u is given by u^{old} then the SOR algorithm's approximation is given by:

$$\begin{aligned} u^{new} &= u^{old} + \varpi\frac{\xi}{4} \;;\; j \neq 0 \,, \\ u^{new} &= u^{old} + \varpi\frac{\xi}{6} \;;\; j = 0 \,. \end{aligned} \tag{2.16}$$

The SOR algorithm converges to a solution only if the over-relaxation parameter satisfies the condition: $0 < \varpi < 2$. In its simplest form, the value for the over-relaxation parameter is determined from a value known as the *spectral radius*, ρ_s and remains static throughout the relaxation. It suffices to say that the spectral radius is a value between 0 and 1 (exclusively) which depends on the density of the grid and gives a measure as to how fast the relaxation algorithm will converge to a solution. The greater the grid node density, the closer ρ_s gets to 1 and the longer the relaxation algorithm takes to converge.

The Cebyshev acceleration algorithm sets the initial value for ϖ to 1 and then recalculates the value of ϖ at the end of each sweep through the grid. Furthermore, each sweep through the grid is actually taken as a half-sweep, once for the odd points ($i+j$ odd) then once for the even points. The result is that at the end of each half step, ϖ gets updated and closer to the optimal value. If $\varpi^{(0)}$ represents the initial value of ϖ and $\varpi^{(1/2)}$ represents the value of ϖ at the end of the first half-sweep, then the Cebyshev acceleration algorithm can be written as:

$$\begin{aligned} \varpi^{(0)} &= 1, \\ \varpi^{(1/2)} &= 1/(1 - \rho_s^2/2), \\ \varpi^{(n+1/2)} &= 1/(1 - \rho_s^2\varpi^{(n)}/4), \qquad n = 1/2, 1, 3/2, ... \end{aligned} \tag{2.17}$$

The only parameter left is the spectral radius which for most applications will involve fine grids and therefore have a value usually very close to 1.

2.3 The Runge-Kutta Numerical Integration

Trajectories are computed using successive numerical integrations of the Lorentz equation for a mass m of charge e, in an electric field **E** and/or magnetic field **B**:

$$\mathbf{F} = m\frac{d^2\mathbf{r}}{dt^2} = e[\mathbf{v}\times\mathbf{B} + \mathbf{E}] \ , \tag{2.18}$$

in order to obtain a new particle position, **r**. Such a second order ordinary differential equation can be rewritten as two first order differential equations for each of the three cartesian coordinate variables, x, y, and z. Each of these sets of two equations will have the form:

$$\begin{aligned} \frac{dx}{dt} &= u(t) \ , \\ \frac{du}{dt} &= f(t) - g(t)u(t) \ , \end{aligned} \tag{2.19}$$

where u, f, and g are some function of the vector components of **v**, **B**, and **E**. The problem of integrating equation (2.18) is therefore reduced to integrating 6 first order differential equations.

The most commonly used algorithm for integrating a set of N coupled first order differential equations of the form:

$$\frac{dy_i}{dx} = f_i'(x,y_1,...,y_N), \qquad i = 1,...,N \ . \tag{2.20}$$

(which includes equations (2.19)) is the *fourth-order Runge-Kutta* [PFTV]. The fourth-order Runge-Kutta is based on the very simple Euler method. To improve on the Euler method, the fourth-order Runge-Kutta takes four intermediate steps within the full integration step, h to correct itself. If y_n is the current value of y at x_n and y_{n+1} is the value of y to be calculated at $x_n + h$, then the fourth-order Runge-Kutta is given by the following set of equations:

$$
\left.
\begin{aligned}
k_1 &= hf'(x_n, y_n), \\
k_2 &= hf'(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}), \\
k_3 &= hf'(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}), \\
k_4 &= hf'(x_n + h, y_n + k_3), \\
y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5),
\end{aligned}
\right\} \qquad (2.21)
$$

where $O(h^5)$ implies that the procedure is fourth-order.

The fourth-order Runge-Kutta can be made accurate to *fifth-order* by using a technique known as *adaptive stepsize control.* This technique evaluates y_{n+1} from equations (2.21) twice by first taking the full integration step h, and then by taking two half-steps to x_{n+1}. If y_1 is the solution for the full integration step and y_2 is the solution for the two half-steps, then the fifth-order approximation to y_{n+1} is given by:

$$
y_{n+1} = y_2 + \frac{\Delta}{15} + O(h^6), \qquad (2.22)
$$

where $\Delta = y_2 - y_1$. What gives this technique its name is that the quantity, Δ can be used to control the stepsize, h. When the integration is having no problems due to smooth derivatives, the quantity Δ will be very small and the stepsize can be increased until some threshold value of Δ is reached. The effect of this technique is to speed up calculations in areas of the geometry where very little is happening. Likewise, if Δ goes beyond a certain threshold at some point in the integration, the stepsize can be decreased and that same integration step redone until Δ goes below the desired threshold. Overall, adaptive stepsize control allows the integration process to be self-checking.

2.4 High-Order Multipole Expansion of Axial Potentials

In order to express the multipole expansion of the electric field in terms of partial derivatives along the axis of symmetry, some properties of the electric field and its potential must be used. In axisymmetric systems the potential has no azimuthal dependence—that is, in the spherical coordinates r, φ, θ, the potential is independent of φ. The potential, ϕ also satisfies Laplace's equation, $\nabla^2\phi = 0$, when charge densities are neglected. The complete axisymmetric solution in spherical coordinates for Laplace's equation is given by

$$\phi(r,\theta) = \sum_{l=0}^{\infty}\left(A_l r^l + \frac{B_l}{r^{l+1}}\right)P_l(\cos\theta)\ , \tag{2.23}$$

where the P_l are the Legendre polynomials and the A_l and B_l are constants. Since source charges have been neglected, the main constraint to equation (2.23) is that ϕ be finite at the origin ($r = 0$). To satisfy that constraint, the B_l must all be equal to zero, reducing equation (2.23) to

$$\phi(r,\theta) = \sum_{l=0}^{\infty} A_l r^l P_l(\cos\theta). \tag{2.24}$$

Next, the z-axis is defined to be the axis of symmetry and the first partial derivative of the potential with respect to z can be derived. The relationships between cylindrical and spherical coordinates with no azimuthal component are expressed as follows:

$$\begin{aligned} z &= r\cos\theta\ , \qquad \rho = r\sin\theta\ , \\ r &= (\rho^2 + z^2)^{1/2}\ , \qquad \theta = \tan^{-1}\frac{\rho}{z}\ , \end{aligned} \tag{2.25}$$

from which the following partial derivatives are obtained:

$$\begin{aligned} \frac{\partial r}{\partial z} &= \frac{z}{r} = \cos\theta \ , \\ \frac{\partial \theta}{\partial z} &= -\frac{\rho}{r^2} = -\frac{\sin\theta}{r} \ . \end{aligned} \tag{2.26}$$

By the chain rule, the following equation for the first partial derivative of the potential with respect to z is obtained:

$$\frac{\partial \phi}{\partial z} = \cos\theta \frac{\partial \phi}{\partial r} - \frac{\sin\theta}{r} \frac{\partial \phi}{\partial \theta} \ . \tag{2.27}$$

Since ϕ is given by equation (2.24), the first part of equation (2.27) simply gives

$$\cos\theta \frac{\partial \phi}{\partial r} = \sum_{l=1}^{\infty} l A_l r^{l-1} \cos\theta P_l(\cos\theta) \ . \tag{2.28}$$

In order to obtain the second part of equation (2.27), two properties of the Legendre polynomials have to be used, notably

$$\left.\begin{aligned} P_0(u) &= 1 \ , \\ (1 - u^2)\frac{dP_n(u)}{du} &= nP_{n-1}(u) - nuP_n(u) \ . \end{aligned}\right\} \tag{2.29}$$

Furthermore, the following relationship is also needed:

$$\frac{\partial P_l(\cos\theta)}{\partial \theta} = -\sin\theta \frac{\partial P_l(\cos\theta)}{\partial(\cos\theta)} \ . \tag{2.30}$$

By combining equations (2.29) and (2.30), the second part of equation (2.27) gives

$$\begin{aligned} -\frac{\sin\theta}{r}\frac{\partial \phi}{\partial \theta} &= \sum_{l=1}^{\infty} A_l r^{l-1} \sin^2\theta \frac{\partial P_l(\cos\theta)}{\partial(\cos\theta)} \\ &= \sum_{l=1}^{\infty} l A_l r^{l-1} \left(P_{l-1}(\cos\theta) - \cos\theta P_l(\cos\theta)\right) \ . \end{aligned} \tag{2.31}$$

Finally, by inserting equations (2.28) and (2.31) into equation (2.27), the following is obtained:

$$\begin{aligned}\frac{\partial \phi}{\partial z} &= \sum_{l=1}^{\infty} l A_l r^{l-1} P_{l-1}(\cos\theta) \\ &= \sum_{l=0}^{\infty} \left[(l+1)A_{l+1}\right] r^l P_l(\cos\theta) \ .\end{aligned} \tag{2.32}$$

Since equation (2.32) is of the same form as equation (2.24), it is possible to derive an expression for partial derivatives of the potential with respect to z to any order. First, the following notation is introduced:

$$\begin{aligned} A_l^{(1)} &= (l+1)A_{l+1} \ , \\ A_l^{(2)} &= (l+1)(l+2)A_{l+2} \ , \\ &\vdots \\ A_l^{(n)} &= (l+1)\cdots(l+n)A_{l+n} \ , \end{aligned} \tag{2.33}$$

where l and n are integers such that $l \geq 0$ and $n \geq 1$. By repeatedly taking the partial derivative of equation (2.24) with respect to z and by using the above notation, the following useful result is derived:

$$\frac{\partial^n \phi}{\partial z^n} = \sum_{l=0}^{\infty} A_l^{(n)} r^l P_l(\cos\theta) \ . \tag{2.34}$$

It follows directly from the above equation that

$$\begin{aligned} \left(\frac{\partial^n \phi}{\partial z^n}\right)_{r=0} &= A_0^{(n)} = n! A_n \ , \\ \phi_0 &= A_0 \ . \end{aligned} \tag{2.35}$$

Equations (2.35) allow for the potential to be expressed entirely in terms of partial derivatives with respect to z evaluated along the axis of symmetry. For the program developed at McGill, the Taylor series expansion is always expressed in the local coordinates about the axial grid point nearest to a given particle. That means that a

solution for the potential of the exact form of equation (2.24) is used for every axial grid point, in local coordinates. Therefore, each axial grid point will have its own set of coefficients, A_i and hence, its own set of partial derivatives with respect to z. By noting that E_z is simply the negative of the first partial derivative of the potential with respect to z and by introducing the following notation:

$$D_n = \left(\frac{\partial^n E_z}{\partial z^n} \right)_{z=0}, \tag{2.36}$$

equations (2.35) give

$$\begin{aligned} A_n &= -\frac{D_{n-1}}{n!}, \\ A_1 &= E_{z_0}. \end{aligned} \tag{2.37}$$

Equation (2.24) can now be expanded in local cylindrical coordinates to provide the desired results. In the McGill program, the expansion of the potential is done up to the seventh order. First, the Legendre polynomials from P_0 to P_7 are given here as follows:

$$\begin{aligned} P_0(u) &= 1, & P_4(u) &= \tfrac{1}{8}(35u^4 - 30u^2 + 3), \\ P_1(u) &= u, & P_5(u) &= \tfrac{1}{8}(63u^5 - 70u^3 + 15u), \\ P_2(u) &= \tfrac{1}{2}(3u^2 - 1), & P_6(u) &= \tfrac{1}{16}(231u^6 - 315u^4 + 105u^2 - 5), \\ P_3(u) &= \tfrac{1}{2}(5u^3 - 3u), & P_7(u) &= \tfrac{1}{16}(429u^7 - 693u^5 + 315u^3 - 35u). \end{aligned} \tag{2.38}$$

By using equations (2.37) and (2.38) in equation (2.24), the expansion of the potential in cylindrical coordinates gives equation (2.39) below.

The next step is to take the appropriate partial derivatives of the potential to obtain the expansions of E_z and E_ρ. Those expansions are also given below as equations (2.40) and (2.41) respectively. With the change of variable $\rho \rightarrow r$ and with the terms in the parenthesis fully expanded, equations (2.40) and (2.41) are identical to the multipole expansions given in section 3.5.

$$\left.\begin{aligned}\phi(\rho,z) = \;& \phi_0 - E_{z_0} z - \frac{D_1}{4}(2z^2-\rho^2) - \frac{D_2}{12}(2z^3-3z\rho^2) \\ & - \frac{D_3}{192}(8z^4-24z^2\rho^2+3\rho^4) - \frac{D_4}{960}(8z^5-40z^3\rho^2+15z\rho^4) \\ & - \frac{D_5}{11520}(16z^6-120z^4\rho^2+90z^2\rho^4-5\rho^6) \\ & - \frac{D_6}{80640}(16z^7-168z^5\rho^2+210z^3\rho^4-35z\rho^6) - \ldots.\end{aligned}\right\} \quad (2.39)$$

$$\left.\begin{aligned}E_z = -\frac{\partial\phi}{\partial z} = \;& E_{z_0} + D_1 z + \frac{D_2}{4}(2z^2-\rho^2) + \frac{D_3}{12}(2z^3-3z\rho^2) \\ & + \frac{D_4}{192}(8z^4-24z^2\rho^2+3\rho^4) \\ & + \frac{D_5}{960}(8z^5-40z^3\rho^2+15z\rho^4) \\ & + \frac{D_6}{11520}(16z^6-120z^4\rho^2+90z^2\rho^4-5\rho^6) + \ldots.\end{aligned}\right\} \quad (2.40)$$

$$\left.\begin{aligned}E_\rho = -\frac{\partial\phi}{\partial\rho} = \;& -\frac{D_1}{2}\rho - \frac{D_2}{2}z\rho - \frac{D_3}{16}(4z^2\rho-\rho^3) \\ & - \frac{D_4}{48}(4z^3\rho-3z\rho^3) - \frac{D_5}{384}(8z^4\rho-12z^2\rho^3+\rho^5) \\ & - \frac{D_6}{1920}(8z^5\rho-20z^3\rho^3+5z\rho^5) - \ldots.\end{aligned}\right\} \quad (2.41)$$

Finally, the partial derivatives, D_n are calculated directly from the potentials of each axial grid point. The next section describes in detail how those partial derivatives are extracted from the axial potential.

2.5 Extraction of Partial Derivatives from the Axial Potentials

In order to use the multipole expansions of the electric field (equations (2.40) and (2.41)) for the Runge-Kutta numerical integration, the partial derivatives, D_n need to be extracted from the axial potentials. The axial potentials are obtained from the finite difference algorithm on an equally-spaced grid. To evaluate the D_n at a specific grid point, Taylor series expansions about that point can be used [KOME]. If the notation $f_n = f(x_0+nh)$ is used (where h is the grid's stepsize, n is any integer, and x_0 is the grid point about which a Taylor series is expanded), then the Taylor series expansions can be written as:

$$f_n = f_0 + \sum_{l=1}^{\infty} \frac{(nh)^l}{l!} \left. \frac{df(x)}{dx} \right|_{x=x_0} . \tag{2.42}$$

In order to calculate numerically derivatives up to the m^{th} order from the Taylor series, m+1 grid points (including x_0) are needed—that is, m Taylor series expansions to the m^{th} order, to m grid points about x_0.

Since equations (2.40) and (2.41) need to have all the D_n up to $n = 6$ calculated, the partial derivatives must be calculated numerically up to the seventh order. At least 8 grid points are needed to do so. However, the McGill program always uses 9 grid points, usually 4 on either side of x_0 (plus x_0). That allows the accuracy to be improved slightly and a certain symmetry to be maintained when calculating derivatives about most points. (The program always makes sure that the relaxation grid has at least 9 grid points on axis.)

When x_0 is near the ends of the axis, more grid points must be used on one side than on the other. For example, if x_0 is 2 grid steps away from the beginning of the axis, then the only 2 grid points available to the left of x_0 plus the 6 grid points to the right of x_0 will be used. If x_0 is the very first or last grid point on the axis, then all 8 grid points used will be on one side. That implies that the Taylor series expansions required can be

up to 8 grid steps away from x_0. All 16 possible Taylor series expansions up the eighth order are given here:

$$
\left.
\begin{aligned}
f_{\pm 1} &= f_0 \pm hf' + \frac{h^2}{2}f'' \pm \frac{h^3}{6}f''' + \frac{h^4}{24}f^{(iv)} \pm \frac{h^5}{120}f^{(v)} \\
&\quad + \frac{h^6}{720}f^{(vi)} \pm \frac{h^7}{5040}f^{(vii)} + \frac{h^8}{40320}f^{(viii)} + \dots , \\
f_{\pm 2} &= f_0 \pm 2hf' + 2h^2f'' \pm \frac{4h^3}{3}f''' + \frac{2h^4}{3}f^{(iv)} \pm \frac{4h^5}{15}f^{(v)} \\
&\quad + \frac{4h^6}{45}f^{(vi)} \pm \frac{8h^7}{315}f^{(vii)} + \frac{2h^8}{315}f^{(viii)} + \dots , \\
f_{\pm 3} &= f_0 \pm 3hf' + \frac{9h^2}{2}f'' \pm \frac{9h^3}{2}f''' + \frac{27h^4}{8}f^{(iv)} \pm \frac{81h^5}{40}f^{(v)} \\
&\quad + \frac{81h^6}{80}f^{(vi)} \pm \frac{243h^7}{560}f^{(vii)} + \frac{729h^8}{4480}f^{(viii)} + \dots , \\
f_{\pm 4} &= f_0 \pm 4hf' + 8h^2f'' \pm \frac{32h^3}{3}f''' + \frac{32h^4}{3}f^{(iv)} \pm \frac{128h^5}{15}f^{(v)} \\
&\quad + \frac{256h^6}{45}f^{(vi)} \pm \frac{1024h^7}{315}f^{(vii)} + \frac{512h^8}{315}f^{(viii)} + \dots , \\
f_{\pm 5} &= f_0 \pm 5hf' + \frac{25h^2}{2}f'' \pm \frac{125h^3}{6}f''' + \frac{625h^4}{24}f^{(iv)} \pm \frac{625h^5}{24}f^{(v)} \\
&\quad + \frac{3125h^6}{144}f^{(vi)} \pm \frac{5^6h^7}{1008}f^{(vii)} + \frac{5^7h^8}{8064}f^{(viii)} + \dots , \\
f_{\pm 6} &= f_0 \pm 6hf' + 18h^2f'' \pm 36h^3f''' + 54h^4f^{(iv)} \pm \frac{324h^5}{5}f^{(v)} \\
&\quad + \frac{324h^6}{5}f^{(vi)} \pm \frac{1944h^7}{35}f^{(vii)} + \frac{1458h^8}{35}f^{(viii)} + \dots , \\
f_{\pm 7} &= f_0 \pm 7hf' + \frac{49h^2}{2}f'' \pm \frac{343h^3}{6}f''' + \frac{2401h^4}{24}f^{(iv)} \pm \frac{7^5h^5}{120}f^{(v)} \\
&\quad + \frac{7^6h^6}{720}f^{(vi)} \pm \frac{7^6h^7}{720}f^{(vii)} + \frac{7^7h^8}{5760}f^{(viii)} + \dots , \\
f_{\pm 8} &= f_0 \pm 8hf' + 32h^2f'' \pm \frac{256h^3}{3}f''' + \frac{512h^4}{3}f^{(iv)} \pm \frac{4096h^5}{15}f^{(v)} \\
&\quad + \frac{2^{14}h^6}{45}f^{(vi)} \pm \frac{2^{17}h^7}{315}f^{(vii)} + \frac{2^{17}h^8}{315}f^{(viii)} + \dots ,
\end{aligned}
\right\} \qquad (2.43)
$$

where all the derivatives are evaluated at $x = x_0$. By taking various linear combinations of the above expansions to the desired grid points, any derivative (up to the eighth) of the

potential can be extracted. The McGill program has all the possible formulae—for the first to seventh derivatives of the potential—that can be extracted from the above expansions *coded-in*; that is, the program does not invert any matrix that arises from the linear combinations of the expansions (2.43).

For example, the formulae for the first to seventh derivatives of the axial potentials for grid points that have four neighbours on either side are coded-in as follows:

$$
\begin{aligned}
hf' &= \tfrac{1}{280}(f_{-4}-f_4) - \tfrac{4}{105}(f_{-3}-f_3) + \tfrac{1}{5}(f_{-2}-f_2) - \tfrac{4}{5}(f_{-1}-f_1) \ , \\
h^2f'' &= -\tfrac{1}{560}(f_{-4}+f_4) + \tfrac{8}{315}(f_{-3}+f_3) - \tfrac{1}{5}(f_{-2}+f_2) + \tfrac{8}{5}(f_{-1}+f_1) - \tfrac{205}{72}f_0 \ , \\
h^3f''' &= -\tfrac{7}{240}(f_{-4}-f_4) + \tfrac{3}{10}(f_{-3}-f_3) - \tfrac{169}{60}(f_{-2}-f_2) + \tfrac{61}{30}(f_{-1}-f_1) \ , \\
h^4f^{(iv)} &= \tfrac{7}{240}(f_{-4}+f_4) - \tfrac{2}{5}(f_{-3}+f_3) + \tfrac{169}{60}(f_{-2}+f_2) - \tfrac{122}{15}(f_{-1}+f_1) + \tfrac{91}{8}f_0 \ , \\
h^5f^{(v)} &= \tfrac{1}{6}(f_{-4}-f_4) - \tfrac{3}{2}(f_{-3}-f_3) + \tfrac{13}{3}(f_{-2}-f_2) - \tfrac{29}{6}(f_{-1}-f_1) \ , \\
h^6f^{(vi)} &= -\tfrac{1}{4}(f_{-4}+f_4) + 3(f_{-3}+f_3) - 13(f_{-2}+f_2) + 29(f_{-1}+f_1) - \tfrac{75}{2}f_0 \ , \\
h^7f^{(vii)} &= -\tfrac{1}{2}(f_{-4}-f_4) + 3(f_{-3}-f_3) - 7(f_{-2}-f_2) + 7(f_{-1}-f_1) \ .
\end{aligned}
\qquad (2.44)
$$

All the other appropriate formulae, such as for two neighbours on one side and six on the other, are also coded-in.

3. The Code

3.1 Choice of System and Language

Having done an overview of most of the common programs available, it became evident that a new program would have to be written in order to display action diagrams dynamically. For this new implementation, two questions had to be asked: what programming language to use, and what computer system to use. Most of the programs available were coded in FORTRAN for use on mainframes or Intel™ based personal computers (IBM™ PCs and compatibles). When implementation of the code was started over five years ago, it was chosen to go against the above standard.

For the computer system it was necessary to have readily available and accessible systems for personal computing—at an affordable price—to allow use of the program by anyone who has limited knowledge of computers. This ruled out the use of mainframes at the time. As for IBM PCs and compatibles, the 640 kilobyte memory barrier, under DOS, was considered a serious limitation—as mentioned earlier for SIMION. The use of so-called DOS-extenders, or even Microsoft™ Windows™, would merely add to the cost and complexity of implementing the code. Apple Computer's line of Macintosh™ computers seemed like the appropriate choice due to its user-friendly interface and extensive programming support. Furthermore the Macintosh operating system is not limited to a 640 kilobyte memory barrier, which made the Macintosh™ line of computers the ideal choice for implementing large user-friendly projects.

For the choice of programming language, it was necessary to have a language that was at once structured, had an extensive library of mathematical functions, and was popular enough to be easily learned and used on any system. These were necessary considerations since the code was expected to evolve with time and become increasingly complex. This limited our choice—at the time the programming began—to either FORTRAN-77, PASCAL, BASIC, or C (all other languages being less popular).

FORTRAN-77 is a very cumbersome language due to certain limiting characteristics such as identifier (variable name) lengths of no more than eight characters, and poor support for structured programming. Furthermore, with FORTRAN-77, as a program evolves and its code becomes increasingly more complex, it becomes more difficult for programmers to see the general flow of the code and be able to debug it. This makes upgrades to existing FORTRAN-77 source code very time-consuming. In other words, FORTRAN-77 lacks the necessary modularity of a truly structured language. Even with the recent arrival of FORTRAN-90 (which includes C-like functions), the language remains impractical for implementing large projects. On the other hand, PASCAL supports structured programming but is primarily intended for instructional purposes and, in most cases, has a poor library of mathematical functions. Finally, BASIC lacks the modularity and the extensive support for various data types that PASCAL has. Despite its popularity, these short-comings of BASIC make it impractical for the implementation of very large programs. The only language at the time that seemed to meet all the above criteria was C, and therefore was initially chosen for the implementation of the code.

The C language has several additional virtues. It allows for high level programming with its extensive mathematical and input/output libraries, as well as for low level assembly-like programming for machine-dependent code. In fact, the C language can make the amazing claim of having had the bulk of its first compiler compiled in its own language [BOOC]. Furthermore, the C language eventually expanded to include object-oriented extensions and evolved into a new language: C++, of which standard C remained a subset.

It is claimed that the object-oriented programming approach—as opposed to the structured programming approach—allows for better organization of the inherent complexity of large programs [BOOC]. An object-oriented approach would ease upgrades of the code since it would cut down on the time necessary for programmers to become familiar with the entire structure and purpose of a program. On the other hand, structured programs tend to become ever more complicated and difficult to decipher as they grew. Furthermore, it is claimed that structured programming "appears to fall apart" when

source code exceeds 100 000 lines [BOOC]. Since our program was expected to become increasingly more complex, an object-oriented language seemed a more logical choice.

Many object-oriented languages were available but C (with object-oriented extensions and now also C++) remained a valid choice. Both C with objects and C++ had the advantage of allowing for a smooth transition from a structured design to an object-oriented design, since both design approaches are supported in the same language (C being a subset of C++ and its other object-oriented extended languages). Therefore, for the final implementation of the code on the Macintosh, a C compiler with object-oriented extensions was chosen: SYMANTEC™ THINK C (versions 4 and 5) since they provided the most affordable C compilers (with object-oriented support) for the Macintosh at the time.

Finally, in the Summer of 1993, a newer version of the above compiler (version 6) that includes a C++ compiler was purchased. It is hoped that eventually the current object-oriented program will be fully implemented in an X-Windows based system using the C++ language.

3.2 Brief History of the Code

Implementation of the McGill FDC/RKI program began over five years ago by Professor R. B. Moore and Dr. M. David N. Lunney, then a doctoral student of Professor Moore's. As described, a standard four-point relaxation algorithm was used to calculate the scalar potential from Laplace's equation and a fifth order Runge-Kutta with adaptive step-size control to calculate the equations of motion. The actual electric field used by the Runge-Kutta was calculated using a local multipole expansion of the potential about the axis (see later sections for details).

In its very first form, the program was only able to solve for a single geometry at a time that had to be coded into the program. Furthermore, the finite differences and Runge-Kutta were actually separate programs with the latter using as input the output of

the former. This meant that any user of the programs would have to modify both, each time a new geometry would have to be solved. The program was however, unique in its ability to display action diagrams dynamically. This allowed to save considerable time during the early designing stages of the ion trap systems and later for determining the properties of built systems.

Unfortunately, the code did not have as yet the necessary user-friendly interface for boundary condition input and display. Furthermore, certain errors had to be corrected, especially in the Runge-Kutta, to render the code both more accurate and more effective. Finally, few actual tests were conducted to determine the accuracy of the program in its use of local multipole expansions. Hence there was a dire need for the code to be upgraded.

In the late Spring of 1992, when this thesis work began, it was decided to upgrade the C code to include object-oriented extensions. The developers of THINK C, at that time, had not come up with a true C++ compiler for the Macintosh but had implemented object-oriented extensions into their compiler. The inherent complexity of a graphical, user-friendly interface with its pop-up menus, dialogue boxes, and multiple windows seemed to require an object-oriented approach. Furthermore, the developers of the THINK C package had implemented several such objects which would greatly facilitate the implementation of a user-friendly interface. That would also greatly facilitate the implementation of an interface for general boundary condition inputs.

In this new implementation, the relaxation and the Runge-Kutta each became separate objects of the same program rather than remaining two separate programs. Furthermore, with the help of Rui Lopes, an undergraduate summer student in 1992, the program truly became capable of general boundary inputs with the boundary conditions being a single object in itself. By the time actual calculations were carried out with the new code, the author of this thesis had corrected the problems with the Runge-Kutta's use of a local multipole expansion; implemented a user-friendly interface for boundary condition input (heavily based on one of the demonstration object-oriented codes that were available with the THINK C development environment); and carried out several tests of accuracy in comparison with SIMION on single geometries.

Finally, during the summer of 1993, another summer student, Debbie Reynolds, helped on making the program more object oriented by using the latest release of THINK C (version 6), which includes a C++ translator. Further modifications include the capability of using multiple overlapping geometries for ion trajectory calculations as well as an enhanced user-friendly interface. From this point on, it is hoped that the code will be fully implemented in C++ and made to be portable to the now far more affordable work-stations.

3.3 Boundary Conditions

As mentioned earlier, the original programs lacked a necessary feature: the ability to interpret general boundary conditions. When work for this thesis was started, the first step was to implement a user-friendly interface for boundary condition input. Some programs, like SIMION, relied on a graphical interface for specifying boundary conditions. This had the advantage that the same grid could be used for the relaxation. However, in the case of SIMION, the user is forced to enter the boundary conditions directly on a grid. This had the disadvantage of forcing the user to do all the mapping calculations from the actual geometry of the electrodes to the grid. It was felt that being able to enter the actual dimensions of boundaries and have the program interpret the input would have far more advantages. Two such advantages would be having a reduced time required for boundary inputs and having a computer record of the actual geometry.

In order to allow the user to enter the actual dimensions of boundaries, a spreadsheet-like interface was implemented. The SYMANTEC THINK C version 5 package contained a demonstration program that had all the necessary objects set up for a dummy spreadsheet (one whose table cells did not have any memory associated with them to allow for the storage of information). All the necessary functions associated with a spreadsheet (such as insertion and deletion of rows and columns) were included as part of those table objects.

For the purposes of our implementation, each row of the table is a segment of the boundary and each column refers to a specific property of a boundary segment. Each row is stored in memory as a record (or structure in the C syntax). Each record contains six fields (each associated with a table column) which together fully define a boundary segment. The six fields are as follows: a description or comment field ignored by the program and included solely for the user's convenience, a functional form field used to describe the shape of a boundary segment (line or curve), a Δz field for the projection of the length of the segment along the axis of symmetry, a Δr field for the projection of the length of the segment perpendicular to the axis of symmetry, a boundary type field to describe Dirichlet or Neumann boundary conditions, and finally a potential field to assign a fixed value to a Dirichlet boundary. To access a field a user needs only to double-click on the associated table cell.

The entire boundary itself must form a completely enclosed internal contour of the geometry with each segment defined in order, starting from the first point on the axis of symmetry (taken as the coordinate z=0, r=0). The axis of symmetry itself must be part of the contour. Since only axisymmetric geometries are allowed, this contour is exactly half the internal outline of the longitudinal cross-sectional cut of the geometry including the axis of symmetry. Figure 3.1 shows a cross-sectional view of a typical geometry used to test our program—that of an ion gun—as well as the associated internal outline (which is used for the boundary conditions) while figure 3.2 shows a screen capture of how the actual boundary condition table would look like.

The description field allows the user to input a string of up to thirty-two characters to describe the boundary segment. This field has no effect on any calculations and is ignored by the rest of the program. Its sole purpose is to serve as a comment for users.

The functional form field is used to tell the program whether the boundary segment is a line or a curve. Lines can be either be parallel, perpendicular or diagonal at any angle to the axis of symmetry. As for curves, only 90 degree arcs of circles (the quarter of a circles curve) are allowed. Figure 3.3 shows the only four such arcs allowed. Since most geometrical shapes can be approximated by a series of lines, the arc functions are sufficient for our needs for the time being. However, provisions have been made in

the code to allow for the definition of special functional forms by users who are willing to code.

The length of a boundary segment is stored in the two fields Δz and Δr. While reading in the boundary segments, the program assumes that each segment comes immediately after the previous one, with the first segment starting at the coordinate z=0, r=0. This permits the use of differentials for lengths. For example, Δz is the difference between the z value of the end point and the z value of the starting point. Since the differentials can be either negative or positive, the program can easily tell whether the end point of a segment is located above or below and before or after the starting point. To appropriately define a 90 degree arc boundary, the magnitudes of Δz and Δr are set equal to each other. Furthermore, the use of differentials facilitates later modifications to the geometry such as changing the axial length of an electrode. If absolute coordinates had been used (specifying the starting and ending coordinates of every segment instead of using differentials), this later modification would have required a change of the coordinates of the endpoints of every segment. With the use of differentials only the Δz of the electrode in question and of the z-axis would need to be changed. As the program scans through the boundary segment definitions, it sums up the differentials to keep track of the coordinates (z, r), ensure that $r \geq 0$, and to verify that the boundary is enclosed. Figure 3.4 shows all the eight possible combinations for Δz and Δr and how they would be interpreted.

The fifth field, that of the boundary condition type, is used to specify whether the boundary segment is to be considered a Dirichlet or a Neumann boundary condition. Two types of Dirichlet boundary conditions can be defined. The first, labelled as *Constant*, fixes the potential of the boundary segment to the value specified in the sixth field. This type is usually used on electrode surfaces and can be combined with any functional form specified in the second field. The second, labelled as *Dirichlet*, fixes the potential along the boundary segment by doing a linear interpolation between two *Constant* boundary segments. This type ignores the sixth field and is limited to lines perpendicular or parallel to the z-axis (that is, no diagonals or arcs). The *Dirichlet* condition can be used

instead of a Neumann boundary condition in a place where the potential is expected to vary linearly between electrodes. Likewise, a Neumann boundary condition is defined by a single label, *Neumann*, and is limited to lines perpendicular or parallel to the z-axis. This type also ignores the sixth field, setting $\partial\phi/\partial z = 0$ or $\partial\phi/\partial r = 0$ for lines perpendicular or parallel to the axis of symmetry respectively (where ϕ is the scalar potential). Finally, since the z-axis is also the axis of symmetry, a separate Neumann boundary condition, labelled *z-axis*, is used to emphasize that fact.

The description of the boundary is complete only when the last segment's end points coincide with r=0, z=0—the initial starting coordinates. The boundary file can then be saved and will be stored on disk as a binary file to save space. The relaxation segment will automatically quit the entire program if forced to read a boundary file that is not completely enclosed.

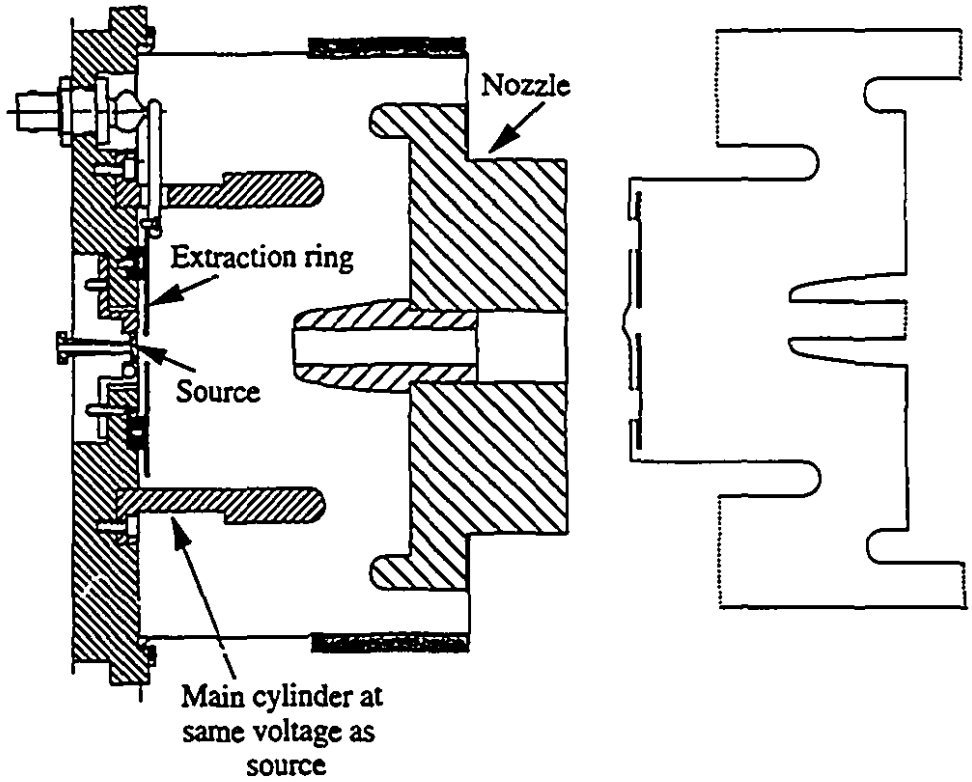


Figure 3.1: Cross-sectional view of the ion gun (left) and its internal outline used for the setting of boundary conditions. The dotted lines in the outline represent Dirichlet boundaries with a potential that varies linearly between electrodes. The solid lines are electrodes with a constant potential.

IonGun.Real

	Description	Function	Delta Z	Delta R	Bound. Cond	Const. Pot
1	ionizing plate	LINE	+0.00000e+00	+1.00000e+00	Constant	+6.00000e+04
2	cone plate	LINE	+1.50000e+00	+3.00000e+00	Constant	+6.00000e+04
3	electrode A	LINE	+0.00000e+00	+1.40000e+01	Constant	+6.00000e+04
4	insulator for disk	LINE	+2.00000e+00	+0.00000e+00	Dirichlet	+0.00000e+00
5	disk left face	LINE	+0.00000e+00	-1.40000e+01	Constant	+5.90000e+04
6	disk width	LINE	+1.00000e+00	+0.00000e+00	Constant	+5.90000e+04
7	disk right face	LINE	+0.00000e+00	+3.05000e+01	Constant	+5.90000e+04
8	disk width	LINE	-1.00000e+00	+0.00000e+00	Constant	+5.90000e+04
9	disk left face	LINE	+0.00000e+00	-8.00000e+00	Constant	+5.90000e+04
10	insulator for disk	LINE	-2.00000e+00	+0.00000e+00	Dirichlet	+0.00000e+00
11	electrode A	LINE	+0.00000e+00	+1.10000e+01	Constant	+6.00000e+04
12	electrode A	LINE	+4.70000e+01	+0.00000e+00	Constant	+6.00000e+04
13	electrode A	ARC+	+4.50000e+00	+4.50000e+00	Constant	+6.00000e+04
14	electrode A	ARC-	-4.50000e+00	+4.50000e+00	Constant	+6.00000e+04
15	electrode A	LINE	-2.30000e+01	+0.00000e+00	Constant	+6.00000e+04
16	gap	LINE	+0.00000e+00	+3.10000e+01	Dirichlet	+0.00000e+00
17	grounded cylinder	LINE	+6.60000e+01	+0.00000e+00	Constant	+0.00000e+00
18	gap (grounded cylinder)	LINE	+0.00000e+00	-1.30000e+01	Dirichlet	+0.00000e+00
19	end cap	LINE	-2.15000e+01	+0.00000e+00	Constant	+0.00000e+00
20	end cap	ARC-	-4.50000e+00	-4.50000e+00	Constant	+0.00000e+00
21	end cap	ARC+	+4.50000e+00	-4.50000e+00	Constant	+0.00000e+00
22	end cap	LINE	+6.00000e+00	+0.00000e+00	Constant	+0.00000e+00
23	end cap	LINE	+0.00000e+00	-4.35000e+01	Constant	+0.00000e+00
24	end cap	LINE	-3.15000e+01	-5.00000e+00	Constant	+0.00000e+00
25	end cap	LINE	+0.00000e+00	-2.00000e+00	Constant	+0.00000e+00
26	end cap	LINE	+3.15000e+01	+0.00000e+00	Constant	+0.00000e+00
27	end cap center (about)	LINE	+0.00000e+00	-5.00000e+00	Constant	+0.00000e+00
28	axis	LINE	-7.60000e+01	+0.00000e+00	z-Axis	+0.00000e+00

Figure 3.2: An example of a boundary condition table for the finite difference calculations.

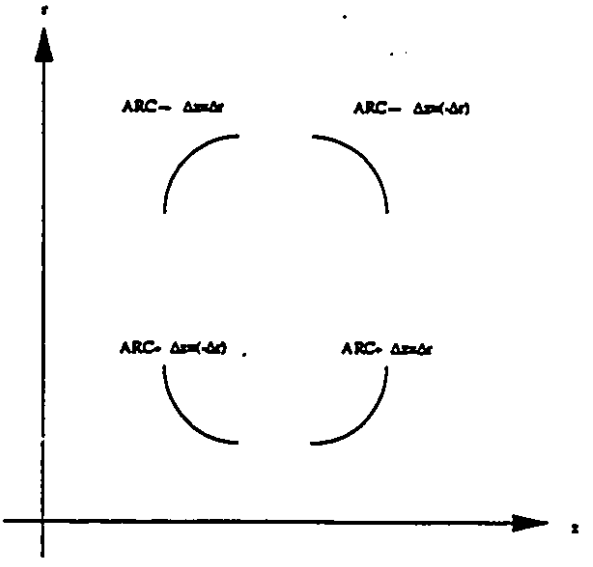


Figure 3.3: The four types of arc segments allowed .

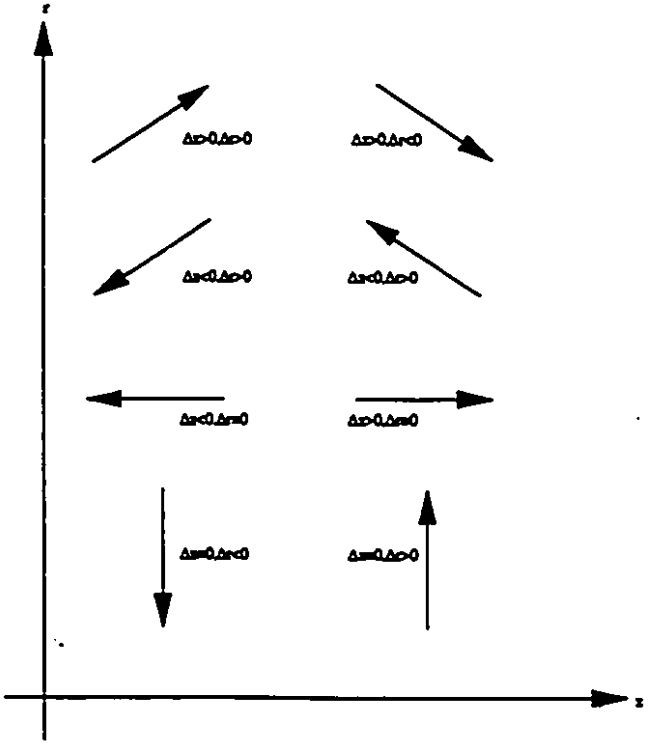


Figure 3.4: The eight possible combinations for linear segments.

3.4 Finite Difference Calculation

Once the boundary conditions have been defined, the program calculates the potential field at the internal points. Our program uses the Successive Over-Relaxation (SOR) method in cylindrical coordinates to calculate this field. The algorithm is presented in chapter 2, section 2 and is based upon the algorithm given in *Numerical Recipes in C, Section 17.5* [PFTV] with the necessary modifications.

The relaxation segment of the program starts by prompting the user for two scaling factors: one for scaling the actual geometry onto a finite grid in mm/(gridstep); and one for the scaling of the potentials. The scaling of the geometry is limited by the amount of memory required to store the grid—which can be set at compile time. To reduce memory requirements and to increase the speed of the calculations, the relaxation is done using long integers instead of double precision reals, hence the need to scale potentials as well.

For the scaling on the grid, the program reads in the boundary file which contains the description of the internal outline of the original geometry. The internal outline forms the boundaries on the grid on which the field will be determined, scaled according to the value given by the user. The program then determines which points on the grid are internal and which are external to the boundary. Only the points marked as internal will be used to calculate the field, all external points being ignored. The algorithms used to do the scaling of the geometry onto a grid are similar to those used by the Macintosh toolbox routines to map lines and circles on the screen—with the exception that the mapping is done on the two dimensional array which is used for the relaxation and is not displayed on screen.

The program then scales the potentials on the boundary points. It takes the largest potential it read from the boundary file (which is stored as a real number) and scales this potential to the long integer value specified by the user, with all other boundary potentials scaled accordingly. This scaling value can be any integer from 10 000 to 1 000 000 000, giving a one-part-per-billion accuracy at best (i.e., for a maximum potential of 1 volt the

best possible error if a scaling of 1×10^9 is used is $\pm\ 1\times10^{-9}$). The reasons for using long integers are simple. Long integers take up only 4 bytes of memory while double precision reals take up 10 bytes on a Macintosh. This is more than twice the amount of memory required by long integers and this translates into more than twice the computational time required for long integers. On the other hand, single precision reals which are also only 4 bytes would not provide the same degree of precision as long integers since 4 byte reals have only 6 significant figures.

Once all the scaling is done the program sets all internal points to 0 or some other potential chosen by the user (like the average of all the potentials for example). The relaxation is then carried out and can only be terminated by the user. To aid the user in deciding on termination, the program displays the *residuals* of the calculations on the axis of symmetry. Figure 3.5 shows a screen display of the residuals. These residuals are displayed in scaled values (long integers from 0 to 1×10^9) with the maximum axial residual written on the bottom right. The user would stop the calculations when the displayed residuals no longer have a pattern to them (they appear to behave as noise) and the maximum residual is suitably low (usually below 10 units). The final maximum residual gives the best possible error in scaled units. Figure 3.6 shows what is meant by the residuals appearing to behave like noise.

Once the calculations are stopped the potentials can be displayed along any horizontal (parallel to z-axis, fixed r value) or vertical (perpendicular to z-axis, fixed z value) line. This allows the user to determine if the potentials are smooth enough, if the calculations should be resumed, and if a different scaling should be used. When selecting a line along which to plot the potentials, the program requires the user to input all values in actual grid locations to enable the user to test the scaling algorithms as well. Once satisfied with the calculations, the potential field map can be saved.

The potential field map is saved as a binary file to save space. The file has the potentials saved as 4 byte long integers and also contains all the scaling information necessary to convert the potentials to their real values. The program first stores the maximum real potential and the maximum scaled potential in the file. These are followed by the grid step size in mm/(gridstep), the first boundary point on the z-axis in grid steps

(which can be non-zero for geometries like an ion trap), the last boundary point on the z-axis in grid steps (which can be less that the total length for geometries like the trap), and the total map length (parallel to z-axis) in grid steps. The program then stores the potentials on the axis alone which range from the first boundary point on the axis to the last boundary point on the axis for each grid point. This is immediately followed by the total map height (perpendicular to z-axis) in grid steps and the full potential map (from z=0 to z=maxz, for r=0 to r=maxr). This may seem redundant but will be made clear in the Runge-Kutta where the algorithm we use requires only the potentials on axis. The full potential map is saved only for those rare cases where a different algorithm is desired or to enable to store an incomplete relaxation for later completion.

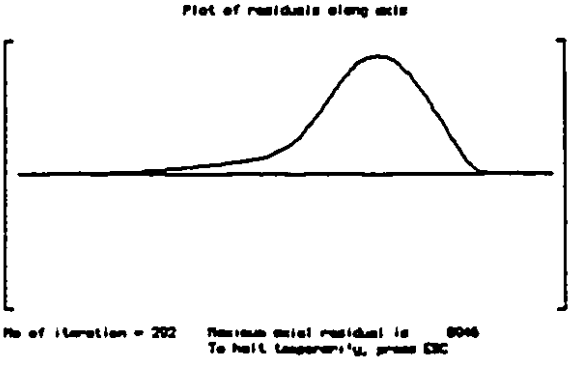


Figure 3.5: Typical display of residuals.

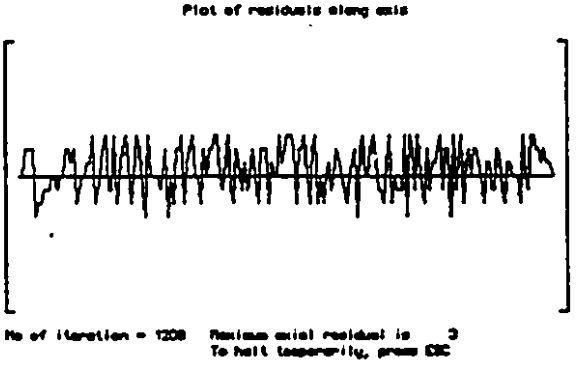


Figure 3.6: Display of residual "noise".

3.5 Runge-Kutta Integration

The third and final segment of the program actually calculates the trajectories of ions, and displays the phase space projections of the ion beam as the calculations are being carried out. The fourth-order Runge-Kutta algorithm with fifth order extrapolation by adaptive step size control (as shown in chapter 2 section 3) is used to calculate the velocities and accelerations of the ions. This is fairly common, but what makes our program unique is the way the electric field is extracted from the potential map. Programs that use finite difference methods usually calculate the field at a grid point by taking two or three point derivatives of the potential at that point. Our program uses only the axial potentials to extract the higher order partial derivatives, with respect to z, of the electric field on the axis of symmetry.

These partial derivatives can then be used to calculate the electric field anywhere within the boundaries of a given geometry. This is achieved by a local Taylor series expansion of the electric field about any point along the axis of symmetry. By using the inherent symmetries of the electric field due to axisymmetric geometries, all the partial derivatives in the Taylor series expansion can be expressed solely in terms of the extracted partial derivatives on the axis. If we let D_n be the n^{th} partial derivative, with respect to z, of the axial component of the electric field, **E**, along the axis of symmetry, then this Taylor series expansion gives, to sixth order, the formulae on the next page (where E_r is the radial field, E_z is the axial field, r is the distance form the z-axis, and z is the displacement from the grid point at which the derivatives were calculated—see chapter 2 section 4 for the derivation of equation (3.1)). Each grid point on the axis has its own set of the derivatives D_n. These derivatives are extracted from the axial potential by a "nine-point derivative method" which uses the potential values at the axial grid point itself and at the eight nearest axial grid points (usually four on either side). The procedure (described in detail in chapter 2 section 5) involves doing a Taylor series expansion up to the 8^{th} derivative of the potential about a given point with intervals of 1 gridstep, 2 gridsteps, and so on. Since the potentials at each grid point are known, the

$$
\left.
\begin{aligned}
E_z = \quad & E_{z_0} && - \frac{D_2}{4} r^2 && + \frac{D_4}{64} r^4 && - \frac{D_6}{2304} r^6 \\
& + D_1 z && - \frac{D_3}{4} z r^2 && + \frac{D_5}{64} z r^4 \\
& + \frac{D_2}{2} z^2 && - \frac{D_4}{8} z^2 r^2 && + \frac{D_6}{128} z^2 r^4 \\
& + \frac{D_3}{6} z^3 && - \frac{D_5}{24} z^3 r^2 \\
& + \frac{D_4}{24} z^4 && - \frac{D_6}{96} z^4 r^2 \\
& + \frac{D_5}{120} z^5 \\
& + \frac{D_6}{720} z^6
\end{aligned}
\right\} \qquad (3.1)
$$

$$
\left.
\begin{aligned}
E_r = & - \frac{D_1}{2} r && + \frac{D_3}{16} r^3 && - \frac{D_5}{384} r^5 \\
& - \frac{D_2}{2} z r && + \frac{D_4}{16} z r^3 && - \frac{D_6}{384} z r^5 \\
& - \frac{D_3}{4} z^2 r && + \frac{D_5}{32} z^2 r^3 \\
& - \frac{D_4}{12} z^3 r && + \frac{D_6}{96} z^3 r^3 \\
& - \frac{D_5}{48} z^4 r \\
& - \frac{D_6}{240} z^5 r
\end{aligned}
\right\}
$$

result is 8 equations with 8 unknowns (the 1st to 8th derivatives of the potential). From these equations it then becomes possible to generate formulae to extract up to the 8th derivative of the potential—that is up to the 7th derivative of electric field.

The derivatives D_n for each axial grid point are calculated only once at the time the Runge-Kutta is initialized. During initialization the program reads in the axial potentials only. To reduce disk access time, the potential map files are saved by the relaxation segment so that all the information needed by the Runge-Kutta is at the beginning of the file. Once the required information (scaling factors, potentials) is read in and the required derivatives calculated, the program reads in a parameter file which can be created by the user or the program itself. The parameter file contains the initial conditions for the particles which can have variable masses, charges, starting positions, and energies. It also contains other information related to the display functions and the maximum allowed time step. During the final step of the initialization procedure, the user is allowed to change the information that was read in from the parameter file. When the trajectory calculation is started the new parameters are used.

As the program calculates the ion trajectories, it finds for each particle—at each Runge-Kutta step—the nearest axial grid point. The program then uses the derivatives at each of these grid points to calculate the electric fields at the respective particle positions. Since the equation of motion that is integrated by the Runge-Kutta—for each particle—is simply given by $\mathbf{F}(z,r) = q\mathbf{E}(z,r) = m\mathbf{a}(z,r)$, the program has to simultaneously integrate the accelerations to get the velocities, and the velocities to get the displacements. For each particle the position, velocity and acceleration is stored in arrays. This allows the program to access the necessary information to display updated trajectory profiles, phase space projections, or energy diagrams. The maximum allowed time step serves as the plot time interval for these displays. The program can be stopped at any time by the user to change the plot time interval or even terminate the calculations.

An additional feature of the Runge-Kutta integration segment is that it can terminate trajectory calculations at a specific point along the axis of symmetry (the z-axis). The program includes an algorithm which uses simple kinematics to interpolate a particle's position back to the xy plane chosen by the user. This has to be done since the integration in time does not guarantee that a particle will land on a surface exactly within an integration step. If no plane is chosen, the end of the geometry is used. At this point the program saves in a file the positions, momentums, energies, and time of flights

for every particle. This data can then be used to display any of the action diagrams desired by the user at the specified distance z.

Furthermore, the Runge-Kutta segment of the program also includes the ability to overlap multiple potential field maps (up to 8). It further allows for these field maps to be specified as time varying. Therefore, it is possible to solve for the scalar potential of a geometry by solving for each electrode of the geometry separately and then overlapping them. Each of these electrodes could then be specified as either time varying or static.

Two basic types of time varying functions are implemented: sinusoidal or radio-frequency; and pulsed ramps. For the radiofrequency time variation the user can specify the amplitude, frequency, and phase shift of a sinusoidal wave as well as a DC offset voltage. In the case of the pulsed ramp, the user can specify the time when the first pulse arrives, the maximum amplitude, the rise time, the decay time, the duration time of the pulse, and the interval between pulses. For the last two cases, it is possible to specify a single pulse that remains turned on indefinitely, or that is turned on initially and gets turned off for certain time intervals that can be indefinite.

Finally, the program allows for a very simple ion mobility calculation where the user can specify the homogeneous temperature and pressure of the system as well as the *reduced mobility*, K_o value for the ions being tracked. If ion mobility calculations are turned on, an additional drag term is added to the equation of motion which acquires the form:

$$\mathbf{F} = q\mathbf{E} - b\mathbf{v} \,, \tag{3.2}$$

where b is some function of the reduced mobility, temperature, pressure and electric charge of the particle.

3.6 List of Program Capabilities

Below is a comprehensive list of the program's capabilities. First, is a list of all of the program's capabilities that does not require recompiling the source code:

- Set boundary conditions using actual dimensions independent of the grid size.
- Set two types of Dirichlet boundary conditions (constant or linear).
- Set the grid step size desired for the relaxation array (up to 1000 by 200 points).
- Set the scaling factor (up to 1e9 for max. potential) for 32-bit integer relaxation.
- Set other relaxation parameters (spectral radius and the initial grid values).
- Reset scaling factor at will.
- Reset grid step size at will.
- Plot potentials along an axial or radial cut at any time during the relaxation.
- Display residuals from the relaxation calculations.
- Save the full potential map when the relaxation is completed to user's liking.
- Read in the axial potentials alone for up to 8 field maps.
- Calculate all the derivatives required for the local multipole expansion algorithm for each field map.
- Set the desired order of the local multipole expansion (from the zeroth to sixth derivative of the electric field).
- Read in all the potentials for up to 8 field maps.
- Calculate the radial and axial components of the electric field for a bi-linear interpolation algorithm.
- Overlap the desired fields for up to 8 field maps.
- Save all the derivatives for the local multipole expansion in a special file.
- Save the off-axis potentials predicted from the above derivatives.
- Save the electric field components predicted by the above derivatives.
- Simultaneously track up to 441 particles.
- Perform all tracking with 80-bit floating point numbers.

- Specify initial coordinates in phase space for all particles using momentums or divergencies.
- Specify the masses and charges of all particles, as well as initial times.
- Interpolate positions of particles to the point where they left the field region.
- Save all the interpolated particle positions.
- Save all particle trajectories in phase space at selected time steps.
- Display phase space projections in (p_x,x), (p_y,y), (p_z,z), (p_x,p_y), and (x,y) in normalized units (eV-μs/mm × mm) at selected time steps.
- Display phase space projections in (p_x,x), (p_y,y), (p_x,p_y) and using beam divergences at selected time steps.
- Display transverse beam profiles in (x,z), (y,z), and (r,z) at selected time steps.
- Display transverse plots for (r,t), (z,t), and (E,t).
- Turn on a constant magnetic field in the axial direction for the entire map.
- Turn on ion mobility calculations (simplistic drag force term) for tracking.
- Save any graphical display on screen to a PICT file.

Furthermore, here is a list of all of the program's capabilities that does require recompiling the source code:

- Change the maximum grid size for the relaxation up to any value limited by the amount of memory available (change only two constants in one header file).
- Change the maximum number field maps allowed (change only one constant in one header file).
- Change the maximum number of particles allowed (change only one constant in one header file).
- Calculate the derivatives for the local multipole expansion using two-point derivatives instead of nine-point derivatives (Set to 0 one compiler directive in one file).

4. Program Evaluation and Results

Several computations and comparisons have been performed in order to test the program and to evaluate its performance. A preliminary test problem, consisting of a simple set a parallel plates, was run to ensure that both the finite difference calculation (FDC) of the electric potentials and the Runge-Kutta integration (RKI) of the particle trajectories were correctly performed. Next, the results of the FDC and RKI algorithms were compared to the results of the SIMION program. For this test, the phase space diagrams of ions traversing a simple Einzel lens were compared. The program was also tested for conservation of particle kinetic energy. Finally, designs for a ion deceleration system and an ion source were evaluated using the McGill FDC/RKI program and using SIMION as a check. **In addition to the following section describing these tests and their results, a second section of this chapter presents the results of some further analysis using the program with added options for thermodynamic distributions and ion mobility.**

SIMION was run on an IBM PS/2 model 55 SX (a 386SX 16MHz chip) and a 486DX 33MHz clone. The McGill program ran on a Macintosh Quadra 700 (68040 with a 20MHz clock speed). Overall speed comparisons would be unfair due to the different machines involved but in most cases the Quadra was only slightly faster than the 486 clone for all calculations. Another aside is the ambiguity with which SIMION defines its error checking parameter, called "accuracy level". Its actual implementation in the SIMION code is not discussed in the documentation. The results for the conservation of energy are given in terms of the best possible "accuracy level" that was found on a trial and error basis.

4.1 Tests

4.1.1 Parallel Plate Test Problem

The axisymmetric geometry used for this problem is shown in figure 4.1. Two trivial tests were made using this model: (1) a comparison of the calculated potentials to the analytic linear potentials across the centre at $r = 0$; and (2) a comparison of the calculated final position of a particle trajectory to the analytic case. In the first case, the potentials obtained from the finite difference calculation along the axis were all within 10^{-7} of the analytical value for a grid spacing of 0.5mm/gridstep (a 20 × 200 array). When the axial component of the electric field was compared with the analytical value of 10 000 V/m, a difference of at most 0.83% below the analytical value was obtained 15 gridsteps off-axis (this error drops to 0.06% for 10 gridsteps off-axis and to 10^{-5} percent along the axis)

For the second test, a single $^{133}Cs^{+}$ ion was positioned 5mm (10 gridsteps) off-axis and 1mm (2 gridsteps) away from the 50 Volts electrode. For such an ion, their will be a resulting acceleration along z due to the electric field of 10 V/mm (10 000 V/m). It is easy to show that the theoretical time of flight for the ion to travel the 9mm distance is, to 9 significant digits, given by 1.57518947μs. The reason for so many significant figures is that the program predicts a value of 1.57518945μs, again a difference of less than 10^{-5} percent. Furthermore, if the ion is given on initial velocity towards the axis of 20mm/μs it will hit the right plate about 26.504mm below the axis. Again, the program agrees within 10^{-5} percent.

The real test of the high-order multipole expansion is when the ion is placed far off-axis, such as 50mm off-axis (100 gridsteps). The full high-order multipole expansion (up to D_6, see equations (2.40) and (2.41), section 2.4) fails completely in this case. The order of the multipole expansion has to be lowered to 3rd order in the electric field (up to D_3) before meaningful results can be obtained. For this stringent case, the 3rd order

multipole expansion gives errors within 10^{-2} percent. At higher orders the errors range from 5% to 600%.

These later results do not invalidate the multipole expansion since most problems deal with paraxial beams (near axis). These results only indicate that the order of the multipole expansion can and has to be controlled depending on the problem at hand. For the following comparison tests with SIMION, all particles were usually within 20 gridsteps from the axis. Furthermore, the multipole expansion was set to 5^{th} order in the electric field (up to D_5).

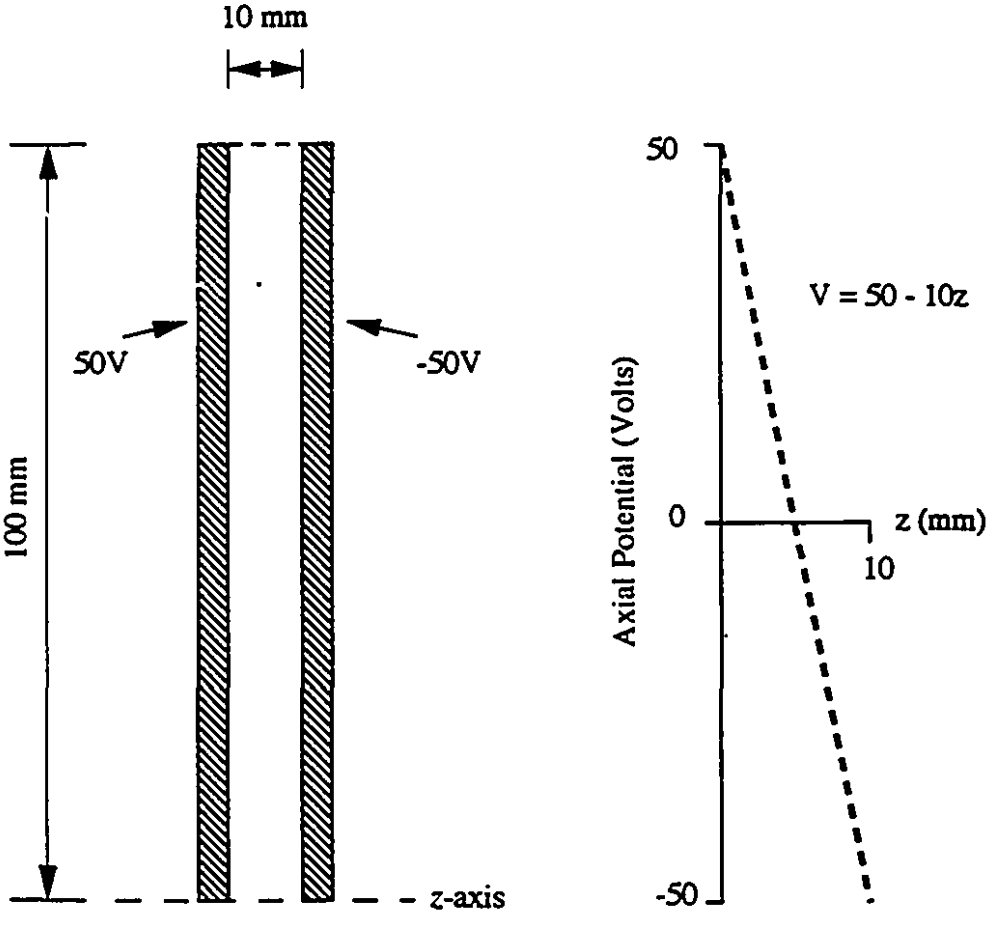


Figure 4.1: The parallel plate test problem with its axial potentials.

4.1.2 Einzel Lens Trajectories

The Einzel lens geometry used for these tests is shown in figure 4.2. For both SIMION and the McGill program, the grid step size chosen was 1mm/gridstep. The whole geometry was 398mm long along z, and 37mm wide along r. Before looking at the trajectories and making comparisons with SIMION, an evaluation of the off-axis multipole expansion algorithm was made. Shown in figure 4.2 are the off-axis potentials generated directly by the FDC and those generated by the high order multipole expansion scheme (outlined in chapter 2) at (a) $r = 5$ grid steps and (b) $r = 10$ grid steps as well as the percentage differences between the two methods. Very close to the edges, the differences can be greater than 10%. However this is also a region where the potential is zero. In the central electrode region, the differences are quite small (less than 0.001%), especially close to the z-axis which is usually the area of interest for trajectory calculations in lenses.

A conservation of energy test was performed and compared with the SIMION program which uses a straightforward planar interpolation of potentials to calculate the fields. When a particle comes into a region with a certain kinetic energy, it must exit the region with the same energy added to (or subtracted from) the total potential difference along the trajectory. This is a good test of whether or not the fields have been properly evaluated as well as the trajectories accurately integrated. For the particular case of this problem, all the ions were $^{133}Cs^{+}$. The ions all had 60 000eV of energy (about 9.6×10^{-15} J) with no radial spread and were evenly spaced every 0.1mm from 0 to 1mm off-axis. Since the Einzel lens is symmetrical, the final energies should all equal 60 000eV. The results of the two calculations (McGill's multipole expansion and SIMION) are summarized in table 4.1 below, with the exact same number of significant figures given by the McGill program and SIMION (the McGill program can provide up to 20). It can be seen that the final particle energies agree to within one part in 100 000 and the time-of-flight values to about the same precision. Results obtained by the McGill program when using the bi-linear interpolation technique to calculate the electric field, are almost identical to those given by SIMION in table 4.1.

The trajectory calculations were verified by a comparison of phase space (action) diagrams generated by the program and those extracted from SIMION. This is where the advantages of our program are clear since it provides dynamic phase space plots along the trajectory while SIMION does not. To extract phase space plots from SIMION, the trajectories were saved at given time intervals and the positions and velocities were extracted particle by particle to construct the phase space plots at those times. This is the major disadvantage of SIMION as all the calculations must be done before one may take a look at the phase space. Furthermore the process itself of extracting the phase space plot from the saved trajectories is very time consuming. Again, the same set of initial conditions were used for both programs. In this case, the initial phase space at time $t = 0$ was a rectangle of 9×9 $^{133}Cs^{+}$ ions with a transverse momentum range of ±5.25eV-μs/mm, a displacement range from the z-axis of ±1mm, and an axial energy of 60 000eV. Figure 4.4 illustrates the results of this operation for two instants of time (0.7 and 1.4 μs). Three evaluations of the action diagram for the x-component are shown: that extracted from SIMION (SIM), from the McGill program using the local (high-order) multipole expansion (LME) and a bi-linear interpolation (similar to what SIMION performs) of the potentials generated from the finite difference calculation (BLI). It can be seen that the three action diagrams for each time are identical indicating that there is little doubt concerning the integrity of the McGill program's performance. Furthermore, the calculations using LME were observed to be almost two times faster than those using BLI in this case. For 81 particles, the above LME calculations took approximately 8 minutes while the BLI calculations took approximately 15 minutes.

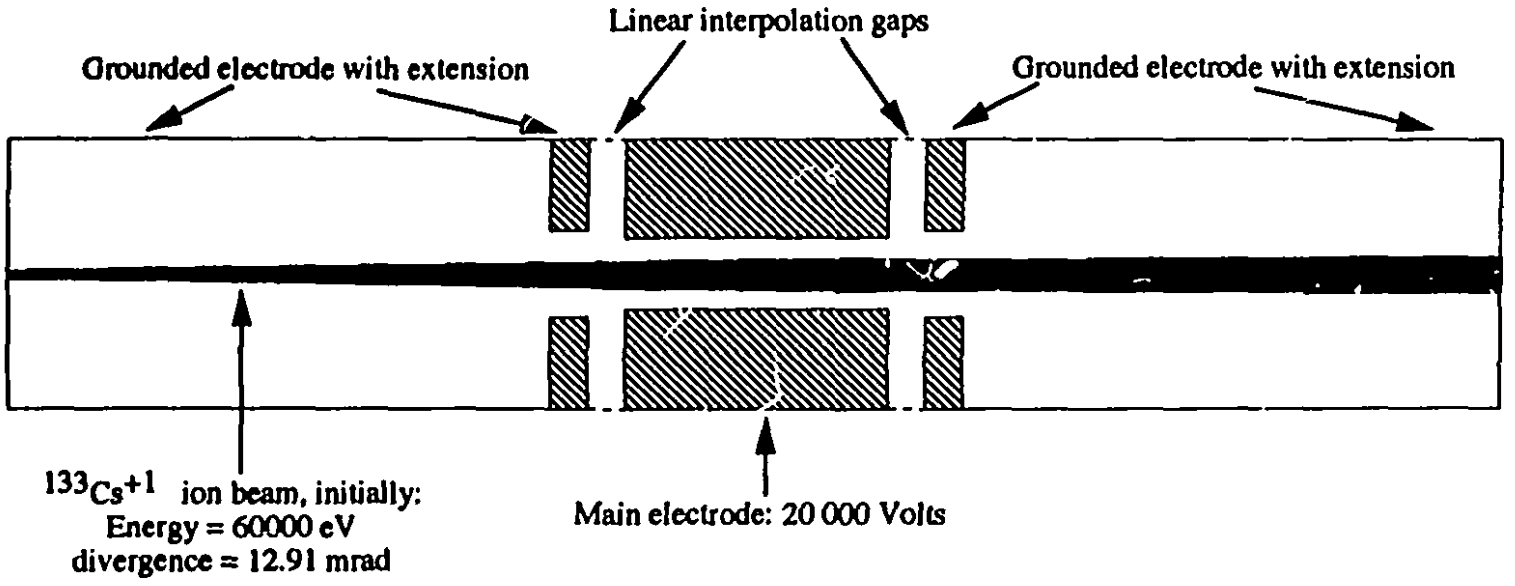


Figure 4.2: Einzel Lens outline with superimposed ion trajectories.

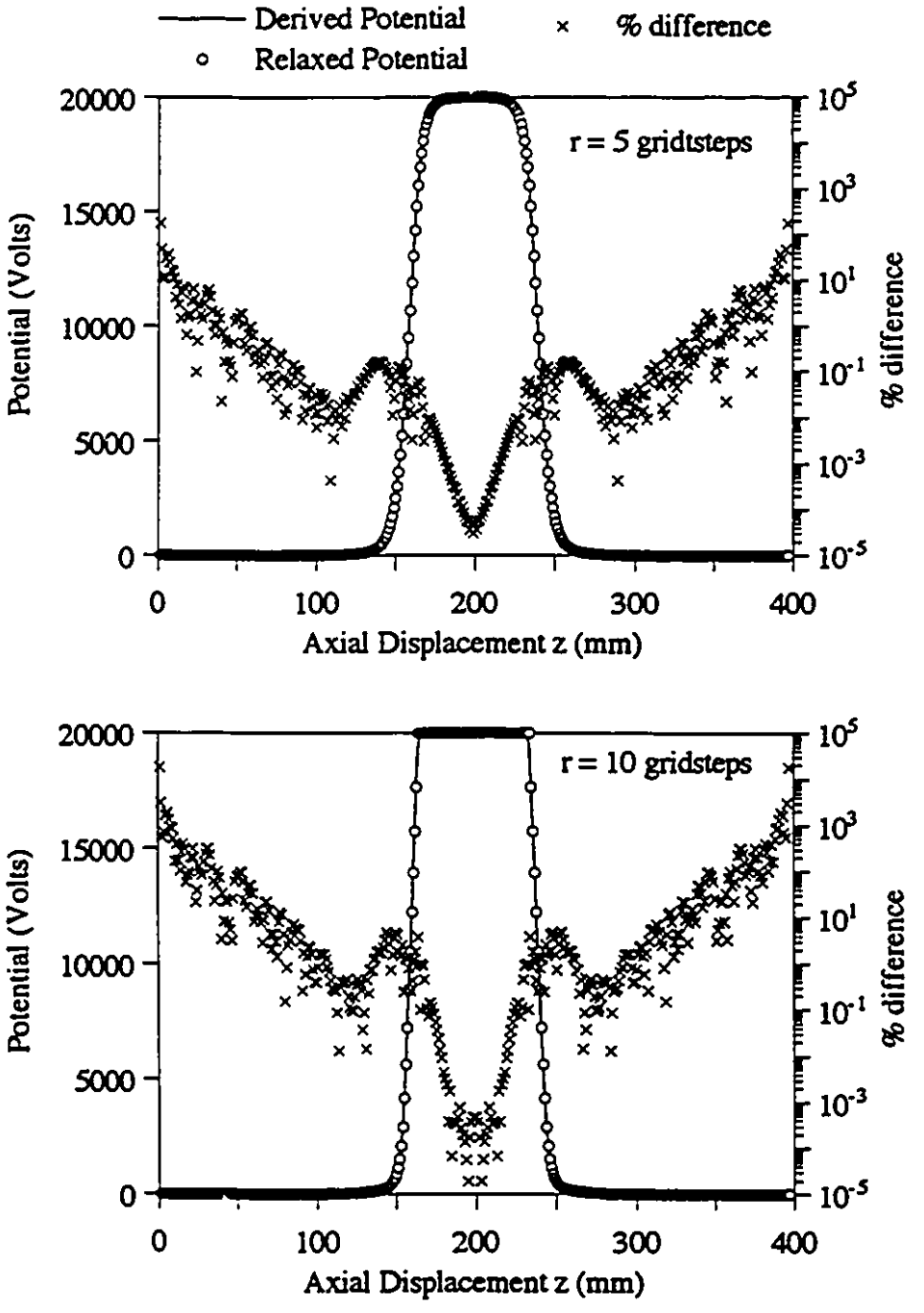


Figure 4.3: Comparison of the Einzel Lens's potentials parallel to the z-axis at r = 5 gridsteps and r = 10 gridsteps (1mm/gridstep).

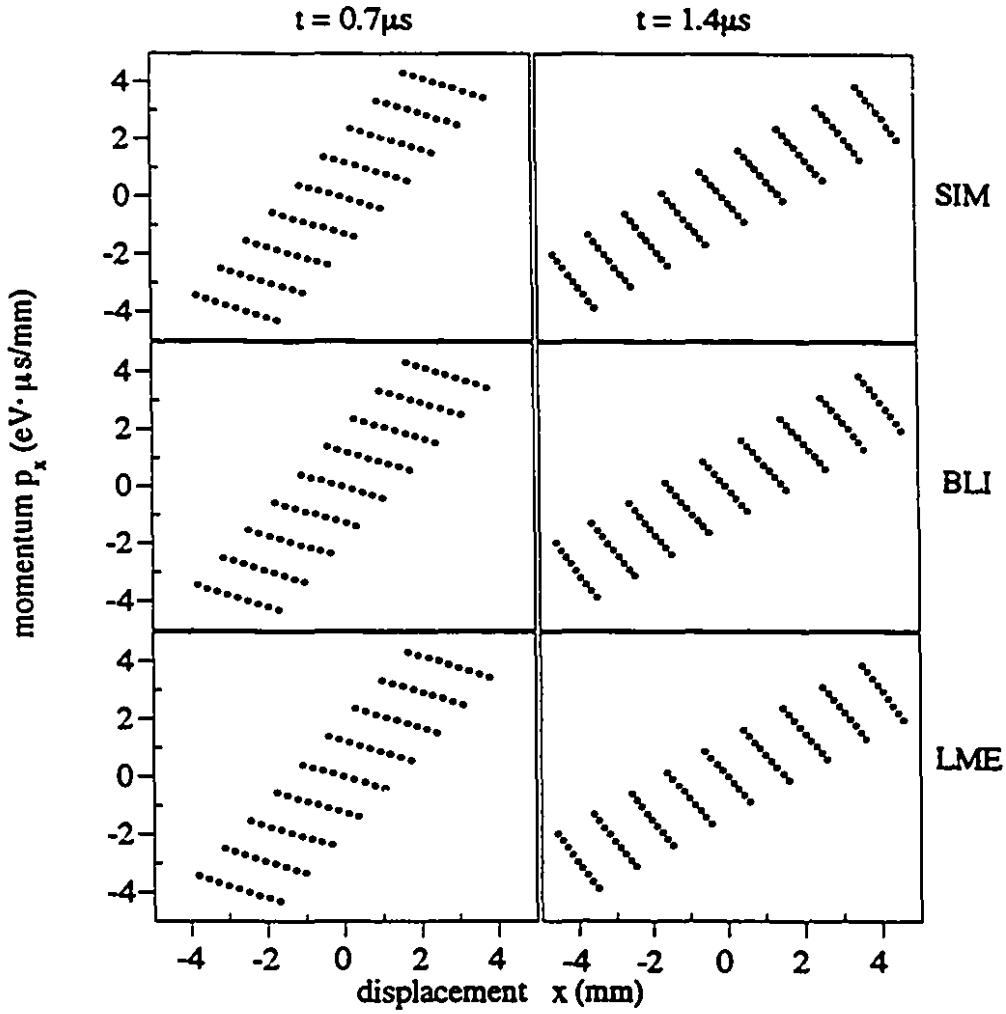


Figure 4.4: Action diagrams at 2 different times for the Einzel Lens as predicted by SIMION (SIM), and the McGill program using a Local Multipole Expansion (LME) and a Bi-Linear Interpolation (BLI) of the potentials.

r_i (mm)	E_M (eV)	E_S (eV)	t_M (μs)	t_S (μs)
0.0	60000.00006	60000.000	1.409262467	1.40931
0.1	60000.00006	59999.997	1.409262616	1.40931
0.2	60000.00006	59999.998	1.409263064	1.40931
0.3	60000.00007	60000.005	1.409263810	1.40932
0.4	60000.00008	60000.017	1.409264855	1.40932
0.5	60000.00009	60000.036	1.409266199	1.40932
0.6	60000.00010	60000.058	1.409267842	1.40932
0.7	60000.00011	60000.088	1.409269785	1.40932
0.8	60000.00013	60000.123	1.409272029	1.40932
0.9	60000.00016	60000.163	1.409274574	1.40932
1.0	60000.00019	59999.216	1.409277421	1.40933

Table 4.1: Final energy values and time of flights for the Einzel Lens problem as calculated by the McGill program (E_M,t_M) and SIMION (E_S,t_S). The theoretical final energy value is 60000eV.

4.1.3 Ion Beam Deceleration System

The deceleration system is used in the McGill physics program at ISOLDE to slow the mass separated beam from its nominal transport energy of 60 keV to some tens of eV for injection into a Paul trap collector. This is a delicate operation that requires high accuracy to simulate. The decelerator geometry is shown in figure 4.5. This problem provides a more stringent test of the FDC/RKI program than the Einzel lens from the previous section because of the different scales involved. The injection electrode near the end of the decelerator is very close to the trap end cap constraining the FDC to small stepsizes. The actual discretization of that critical area used by the program is shown in figure 4.6. For this problem the decelerator was 198mm long along z and 64mm wide along r with a grid step size of 1mm/gridstep. The ion trap was replaced by an equipotential volume as can be seen in figure 4.5.

The first test was a conservation of energy, as performed for the Einzel lens, for the McGill program and SIMION. This time however, the $^{133}Cs^{+}$ ions were evenly spaced every 1mm from 0 to 4mm off-axis. The ions all had 60 310eV of energy with no initial transverse energy. In this case, all the ions that enter the equipotential volume (as they all did) will have a theoretical final energy of 310eV. The results are summarized in table 4.2 below. As can be seen, the results are quite agreeable except for the last two cases. The problem here is the same as for the parallel plate test problem: the particles are several gridsteps off-axis as can be seen from the trajectories in figure 4.5. and from the additional information in table 4.2. This indicates that the fifth-order local multipole expansion scheme is still sufficiently accurate slightly past 11 gridsteps off-axis.

Phase space diagrams were also computed for three different times and only for the LME in the case of the McGill program. For this test, the initial phase space at time $t = 0$ was a rectangle of 9×9 $^{133}Cs^{+}$ ions with a transverse momentum range of ±5.25eV-μs/mm, a displacement range from the z-axis of ±2mm, and an axial energy of 60 310eV. These results, again identical for the two programs, are shown in figure 4.7.

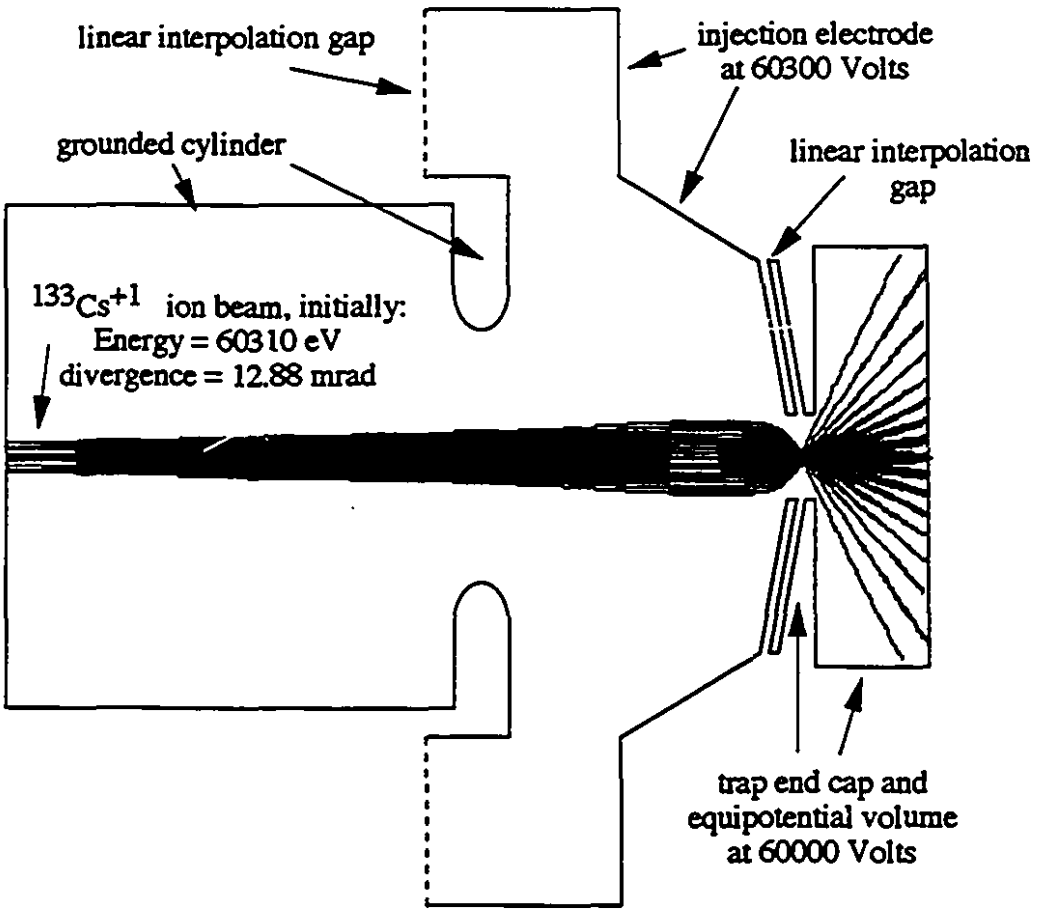
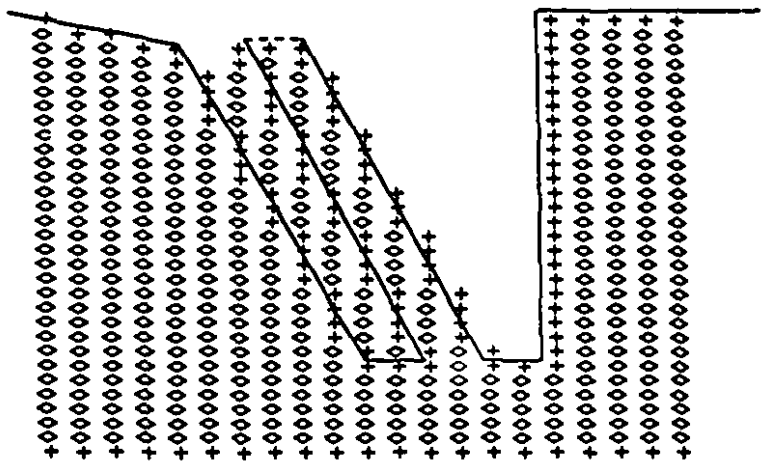


Figure 4.5: Decelerator outline with superimposed ion trajectories.



◇ free grid point

\+ boundary point, including z-axis

Figure 4.6: Blown-up outline of the decelerator at the trap end cap with grid superimposed.

r_i (mm)	E_M (eV)	E_S (eV)	t_M (μs)	t_S (μs)	r_{fM} (mm)	r_{fS} (mm)
0.0	310.000	310.000	2.34328	2.34160	0.000	0.000
1.0	310.045	309.816	2.37135	2.37035	5.227	5.202
2.0	310.042	308.486	2.46598	2.46823	11.222	11.200
3.0	290.987	307.813	2.67120	2.67377	19.174	19.238
4.0	251.751	307.841	3.07863	2.97078	30.000	30.000

Table 4.2: Final radial positions, energy values, and time of flights for the Decelerator problem as calculated by the McGill program (r_{fM},E_M,t_M) and SIMION (r_{fS},E_S,t_S). The theoretical final energy value is 310eV.

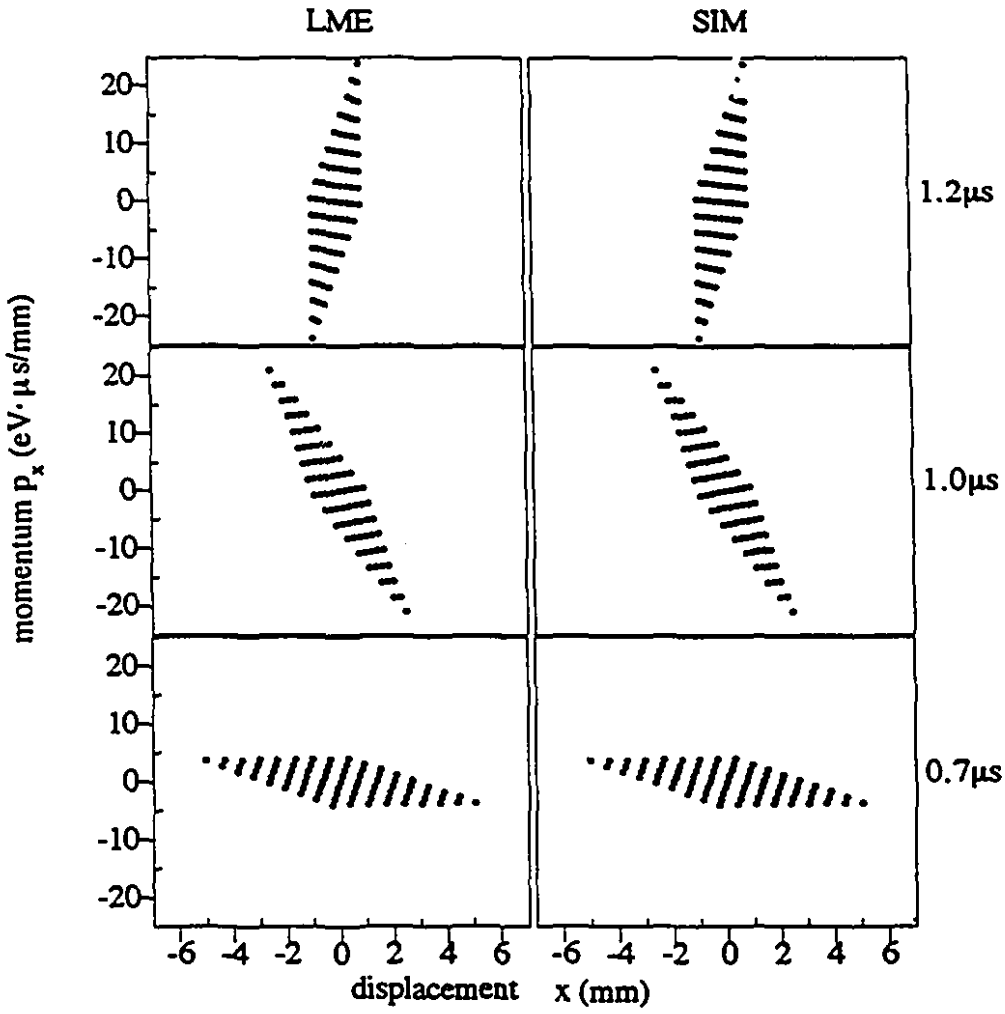


Figure 4.7: Action diagrams at 3 different times for the Decelerator as predicted by SIMION (SIM), and the McGill program using a Local Multipole Expansion (LME) of the potentials.

4.1.4 Ion Gun

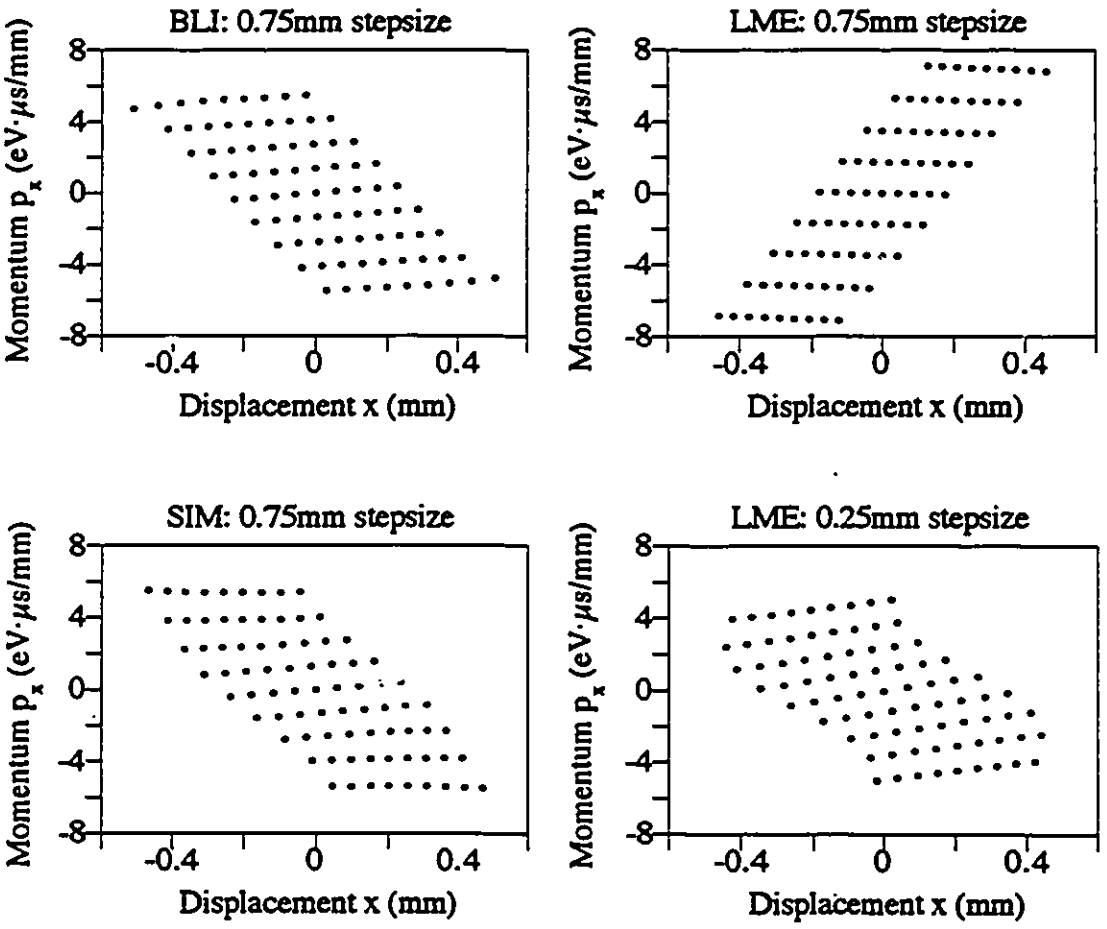
The ion source problem provides the most stringent test of the program due to the very high field gradients over small grid regions. The geometry is illustrated in figure 3.1 schematically and as exactly used by the FDC algorithm (and SIMION). Figure 3.2 also shows a screen capture of boundary definition table for the geometry which lists these boundary conditions explicitly.

What makes the ion gun different from all the other geometries used to test the program is the uncertainty in what initial conditions to use. All that is known is that ions will come off the hot plate, but there is no knowledge as to their energies or angular distributions. It was assumed that across this surface the ions would have anywhere from 0 to 20eV in total energy and would have angular spread of over 45 degrees. The later was required in order to facilitate comparisons between SIMION and our program since both programs differ greatly in the way initial conditions are specified. (In SIMION particles are defined in terms of grid units for position, and total energy and angle for direction whereas the McGill program allows the input of momentums in all dimensions.) Furthermore, the geometry was 91.5mm long along z and 77.5mm wide along r, with a 2mm diameter plate as the ionizing surface. The ideal stepsize for this problem would have been 0.5mm/gridstep but this would have required an array of 28 365 grid points. A limiting factor here is that the version of SIMION used required that the potential arrays contain no more that 16 000 points whereas this is not a problem on a Macintosh where it is straightforward to use all the memory available. Here again, the difference in ease of use between our program and SIMION is evident: the McGill program allows the geometry to be re-scaled at will since the actual dimensions are input whereas with SIMION, the geometry has to be input directly onto a grid.

In order to compare results from both programs, a grid stepsize of 0.75mm/gridstep was chosen (to avoid array size limitations for SIMION as well as to minimize round-off in the dimensions of the ion gun). For such a gridstep, the ionizing plate will be rounded down to 1.5mm in diameter. For the purposes of this test, the ionizing plate was set to 60 000 Volts and the extraction ring to 100 Volts below

providing a potential gradient of 100 Volts across only 4 gridsteps. The *nozzle* at the end of the geometry was kept at 0 Volts. For the phase space calculations, the original phase space distribution was fixed at ±1.66eV-μs/mm of momentum (1eV of energy) and ±0.75mm of displacement in the transverse plane, with an energy of 1eV along the axis of symmetry for every $^{133}Cs^{+}$ ion (this corresponds to an angular spread of 45 degrees).

Figure 4.8 shows three action diagrams for SIMION, compared to the LME and BLI of the McGill program. In this case, the resulting diagrams are not the same. Those determined via SIMION and the BLI are similar but the LME diagram has a different orientation. Furthermore, in the case of the LME the energy was not conserved within the same degree of accuracy as the others. This is an indication that the problem is not sufficiently discretized which means that the array size must be increased in order for the problem to be accurately treated. It was possible, in the case of the McGill program, to triple the array in size and recalculate the phase space diagrams (change the grid stepsize to 0.25mm/gridstep). In this case, with the ion source region better discretized, the phase space diagram has the same orientation as the others and moreover, the trajectories conserve energy. Overall, this would indicate that the LME algorithm is quite sensitive to large gradients. However, due to the ease of grid scaling, the problem can be avoided by improving discretization. Table 4.3 shows the energy conservation comparisons between SIMION and the McGill program at a 0.75mm/gridstep and for the LME only at 0.25mm/gridstep. In each case, a single $^{133}Cs^{+}$ ion was used with initially 1eV of energy with no transverse energy at 0.75mm off-axis.



(All at t = 0.28μs)

Figure 4.8: Action diagrams for the Ion-Gun as predicted by SIMION (SIM), and our program using a Local Multipole Expansion (LME) and a Bi-Linear Interpolation (BLI) of the potentials.

mm/gridstep	E_{LME} (eV)	E_{BLI} (eV)	E_S (eV)
0.75	60011.286	60001.990	60001.721
0.25	60000.999	--	--

Table 4.3: Final, energy values for the Ion Gun problem as calculated by the McGill program for the local multipole expansion (E_{LME}), bi-linear interpolation (E_{BLI}) and SIMION (E_S). The theoretical final energy value is 60001eV.

4.2 Analyses

4.2.1 Ion source beam profile

Having analyzed the ion source optics, an attempt was made to simulate the ion beam distribution in phase space using the local (high-order) multipole expansion on a 0.25mm/gridstep array for the ion gun. To do this, a thermodynamic model based on the Gibbs distribution was used to determine the temperature-determined weighting functions on the phase space diagram [LUNP, GDMV]. The phase space diagrams were calculated in the way already described and the corresponding distribution was applied to reconstruct the beam profile at the detector position.

The basic result of statistical mechanics (see, for example, *Statistical Physics* by Landau and Lifshitz) of importance in particle action diagrams is that the most probable state of a particle collection sharing a total energy E is that in which the density in six-dimensional *phase space*, made up of the spatial dimensions x, y, z and the momentum dimensions p_x, p_y and p_z, is given by

$$\frac{d^6n}{dxdydzdp_xdp_ydp_z} = Ae^{-\frac{E}{kT}} \quad . \tag{4.1}$$

where E is the energy of particles in the phase space volume element $dxdydzdp_xdp_ydp_z$, kT is a constant that characterizes the distribution of particle energies and is usually expressed as the Boltzman constant k multiplied by a temperature T and A is a normalization constant that results in the integration of the density function over all of the coordinates being simply the total number of particles [LALI].

The simplest cases are those in which E is only a function of the momentum coordinates. The integration over the spatial coordinates is then trivial resulting in

$$\frac{d^3n}{dp_xdp_ydp_z} = AVe^{-\frac{E}{kT}} \quad . \tag{4.2}$$

where V is the volume of the particle collection. For a collection in which the total energy is conserved, an underlying assumption in arriving at (4.1), E becomes simply the kinetic energy of the particles, which can be expressed as

$$\frac{d^3n}{dp_x dp_y dp_z} = AVe^{-\frac{p_x^2+p_y^2+p_z^2}{2mkT}} \quad . \tag{4.3}$$

Here the independence of the distribution in the three coordinates of motion is explicit as:

$$\frac{d^3n}{dp_x dp_y dp_z} = AVe^{-\frac{p_x^2}{2mkT}} e^{-\frac{p_y^2}{2mkT}} e^{-\frac{p_z^2}{2mkT}} \quad . \tag{4.4}$$

and the integration over any two of the coordinates to obtain the overall distribution in a third becomes simply a double integration over the Normal distribution

$$\frac{dn}{dp_x} = 2\pi AV\, mkT\, e^{-\frac{p_x^2}{2mkT}} \quad . \tag{4.5}$$

For an action diagram one needs the particle density as a function of both the momentum and the displacement coordinates;

$$\frac{d^2n}{dx dp_x} = 2\pi A \frac{dV}{dx} mkT\, e^{-\frac{p_x^2}{2mkT}} \quad . \tag{4.6}$$

Thus the variation throughout the volume must be taken into account. If the volume of the particles is rectangular on all faces then dV/dx is just the area of the particle collection in the p_y-p_z plane. However, a more usual particle source—like it is the case for the ion gun geometry—is a disk, usually expressed by an infinitesimal thickness in the z coordinate and a radius r in the x-y plane. Such a source has a total volume of πr^2 and yields the following expression for dV/dx:

$$\frac{dV}{dx} = 2\sqrt{r^2 - x^2} \quad . \tag{4.7}$$

Substituting equation (4.7) into (4.6), yields the action diagram density for such a source;

$$\frac{d^2n}{dx\,dp_x} = 4\pi A\, mkT\sqrt{r^2 - x^2}\; e^{-\frac{p_x^2}{2mkT}} \quad .$$

Taking into account that this density integrated over the area of the action diagram must be the total number of particles N, the distribution can be expressed as

$$\frac{d^2n}{dx\,dp_x} = \frac{2N}{\pi r^2\sqrt{2\pi mkT}}\sqrt{r^2 - x^2}\; e^{-\frac{p_x^2}{2mkT}} \quad . \tag{4.9}$$

For the numerical calculation of beam profiles, a $^{133}Cs^+$ beam with the same initial conditions as in the ion gun test problem in section 4.1.4 was used. However, the potential on the extraction ring was set to 0 Volts. The phase space diagram at the detector position (set at 540mm from the ionizing plate) is shown in figure 4.9. The resulting ion density distribution, after applying the weighting function given by equation (4.9), is shown in figure 4.10.

Measurements of the ion distribution on the detector using a moveable faraday cup were made independently to measure the intensity as a function of transverse position [MGDP]. (These measurements were done by A. M. Ghalambor Dezfuli as part of his thesis work.) The measured distributions were integrated and reconstructed as shown in figure 4.11. This indicates that their is a good agreement between the measured and simulated profiles. These results are very significant as they allow the design and optimization of ion source parameters with a direct simulation of the beam emittance. Such a procedure is not possible simply by simulated ion trajectories.

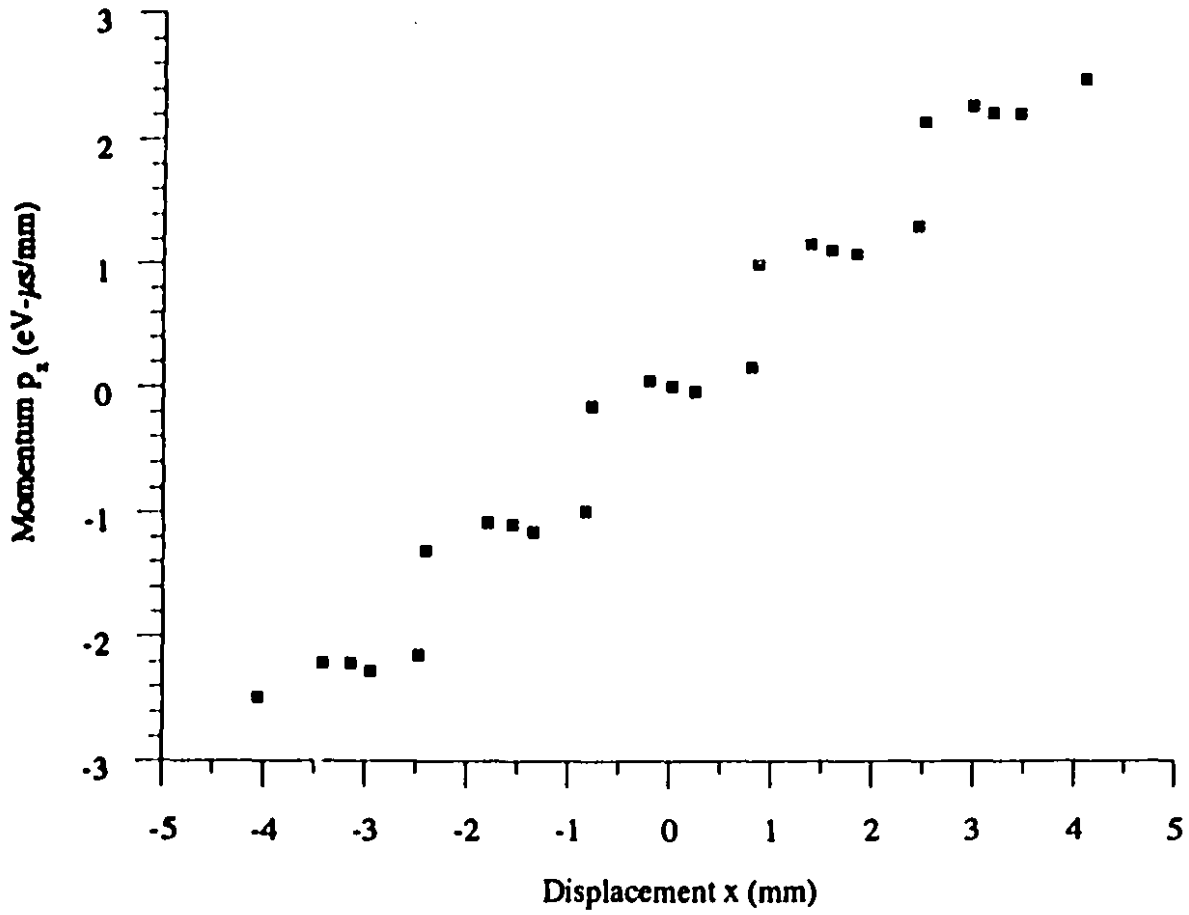


Figure 4.9: Action diagram for the beam at the detector with zero extraction voltage.

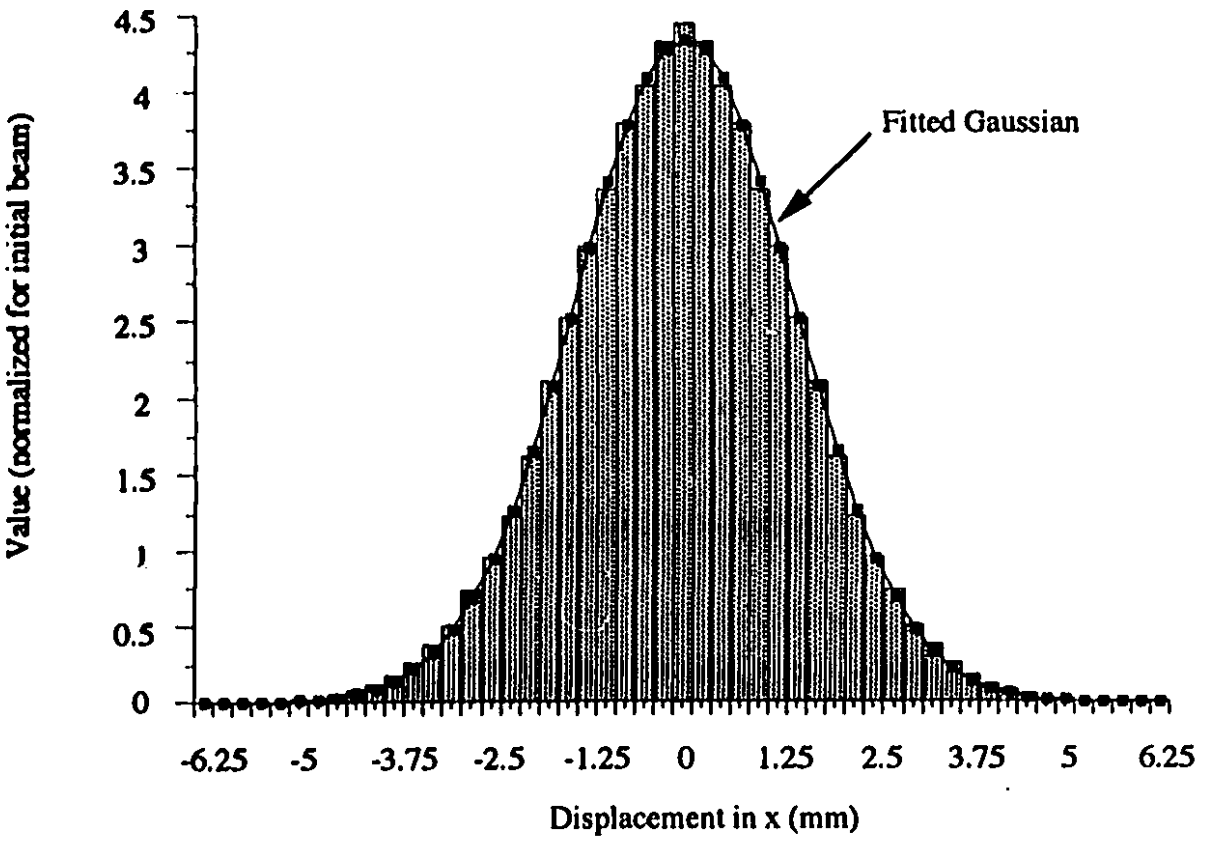


Figure 4.10: Simulated beam profile.

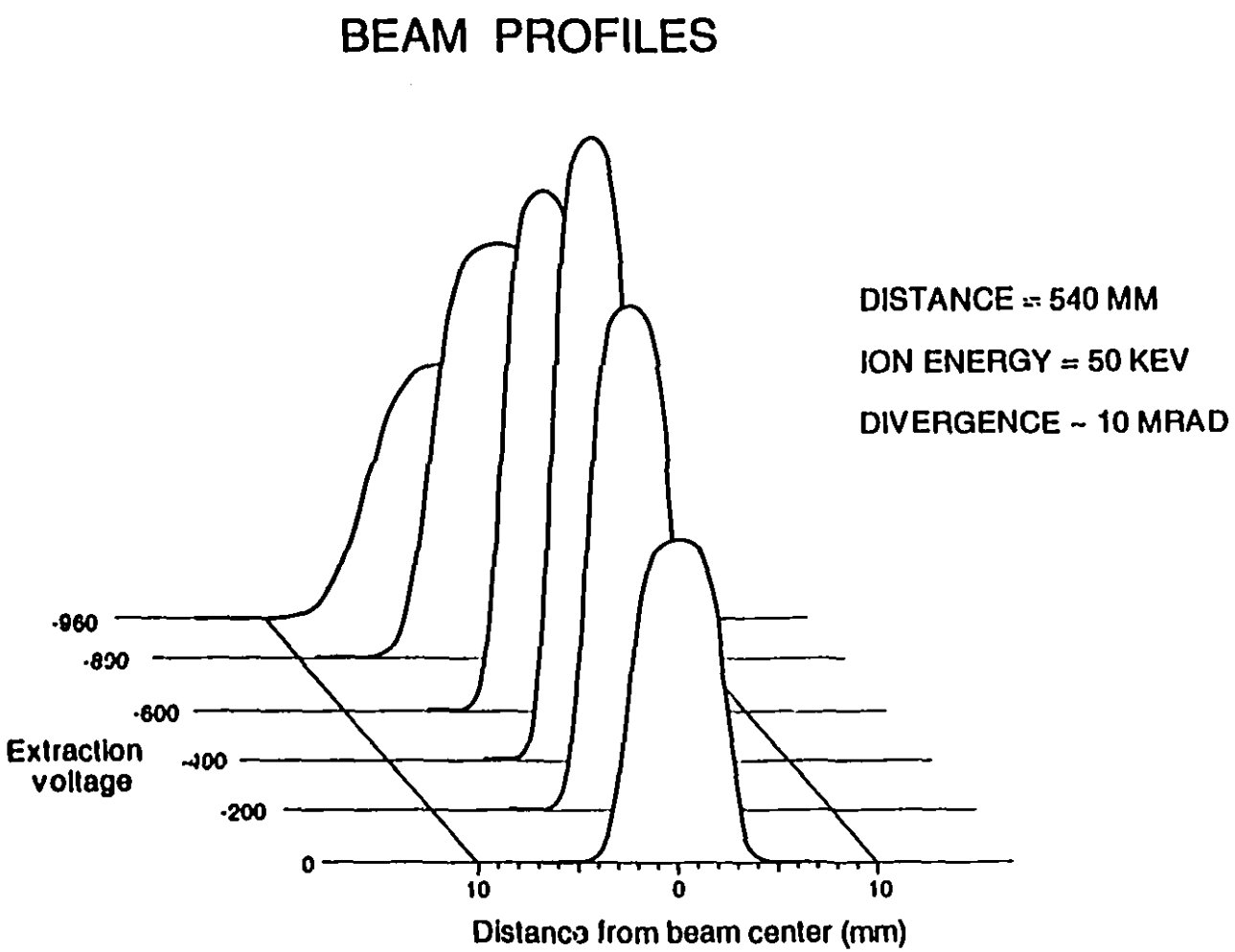


Figure 4.11: Measured beam profiles for the ion gun.

4.2.2 Ion mobility calculations

The damping of the motion of particles in radiofrequency (Paul) traps due to background gas was first demonstrated by Wuerker et al. in 1959 [WUEA] for metallic particles in air. The ease with which background gas can be used to cool ion motion in electromagnetic traps led to the study by Douglas and French [DOFR] of the effect of background gas on the transverse motion of ions in a radiofrequency quadrupole (RFQ) rod structure (shown in figure 4.14). Indeed it was shown that background gas at moderate pressures, up to about 1 Pa, did result in a significant increase in the transmission of ions through a 1 mm diameter orifice following the quadrupole rods. However, at pressures higher than this the transmission decreased, until at about 10 Pa it was essentially zero.

This result was very interesting for the purpose of preparing a beam of ions for collection in a trap or simply to improve the transverse emittance of an ion beam for subsequent use by sensitive apparatus. To investigate these possibilities, the McGill program was modified to study the dynamics of ions in an *ideal* radiofrequency quadrupole rod system at high gas pressures (*i.e.*, the Runge-Kutta integration was used on the equations of motion at the centre of an RFQ rod system with no fringe field effects). For this purpose, a short digression is necessary for the explanation of the concept of ion mobility.

To obtain an approximation to the behaviour of ions under buffer gas collisions, the effect of the collisions can be modeled as a viscous force. This is indicated by the results of ion mobility experiments where it is well known that the drift velocity of an ion through a gas due to an electric field is proportional to the electric field strength, at least at low values of the electric field. In fact, the proportionality constant K in the relation between the ion drift velocity v_d and the electric field E:

$$v_d = KE \ , \tag{4.10}$$

is defined as the "ion mobility". Thus the effect of the gas on the ion motion is to present a drag force which is proportional to the velocity, the proportionality constant being simply the ionic charge divided by the ion mobility:

$$F_d = \frac{q}{K} v_d \ . \tag{4.11}$$

Ion mobility measurements are made by observing the drift of an ion over a distance that involves many ion-molecule collisions. Thus the fluctuations that occur due to the collisions are averaged out over many collisions to give a drift velocity at which the average drag of the molecules is exactly balanced by the electric field force. The situation that is to be investigated is that of the transient condition when the velocity of the ion through the gas is different from that which would have an average drag force which would exactly balance the applied electric field. From the point of view of the molecules presenting a viscous drag it is tempting to consider the equation of motion of such an ion to be

$$m\ddot{x} = F_{applied} - \frac{q}{K}\dot{x} \ . \tag{4.12}$$

For the specific case of an ion suddenly created at zero velocity in an electric field in a gas, the above equation can be easily integrated to give

$$v = v_d\left(1 - e^{-\frac{q}{Km}t}\right) \ , \tag{4.13}$$

where v_d is the equilibrium drift velocity after a long time. Thus the velocity difference with the equilibrium drift velocity *relaxes* with a decay rate of *q/Km*.

The success of the simple equation of motion (4.13) in accounting for the equilibration of the drift velocity of ions in gases with an electric field, led to an attempt to use the equation to simulate ion motion in a radiofrequency quadrupole field with background gas. For this, the Runge-Kutta integrations were carried out for such time-

varying fields and some typical results are shown in figure 4.12. The calculations shown were for $^{23}Na^{+}$ ions in helium at radiofrequencies and field strengths appropriate for typical quadrupole rod operation.

The results for a helium pressure of 0.01 Torr (1.33 Pa) show the usual pattern of an RF oscillation superimposed on a slower macro-oscillation, of about 250 kHz. However, a decay of the overall motion is evident in the 40 μs of the ion motion for which the calculations were carried out. When the pressure is increased by 10 the decay rate increases accordingly , to a value of about 10^{4} s^{-1}. This continues until at 1 Torr (133 Pa) the motion seems to take on the characteristic of a critically damped simple harmonic oscillator, with a damping time constant of the order of one microsecond.

At pressures above 1 Torr the only motion remaining is seen to be the RF motion and this is seen to cause only a slow progression of the ion toward the axis. At 10 Torr (1333 Pa), the motion is seen to take an inordinately long time to proceed to the axis.

The variation of the damping time constants for three elements (for a radio-frequency of 2000 kHz) are shown in figure 4.13. From this it appears that the optimum pressure of helium for cooling the motion of these ions in a radiofrequency field is about 100 Pa.

It appears that the guiding and thermalizing of ions using RFQ fields in high background gas pressures is quite feasible and could have a broad range of applications in trace-beam technology. This is the first time that detailed numerical integration has been performed for ions in high electric fields at high pressures where ion mobility dominates the dynamics. This particular version of the program promises to have considerable impact in the future. Staff at Foster Radiation Lab are currently studying not only its possible use for the loading of such beams into Paul traps but also for the improvement of the emittance of high velocity ion beams for conventional magnetic sector mass spectrometers.

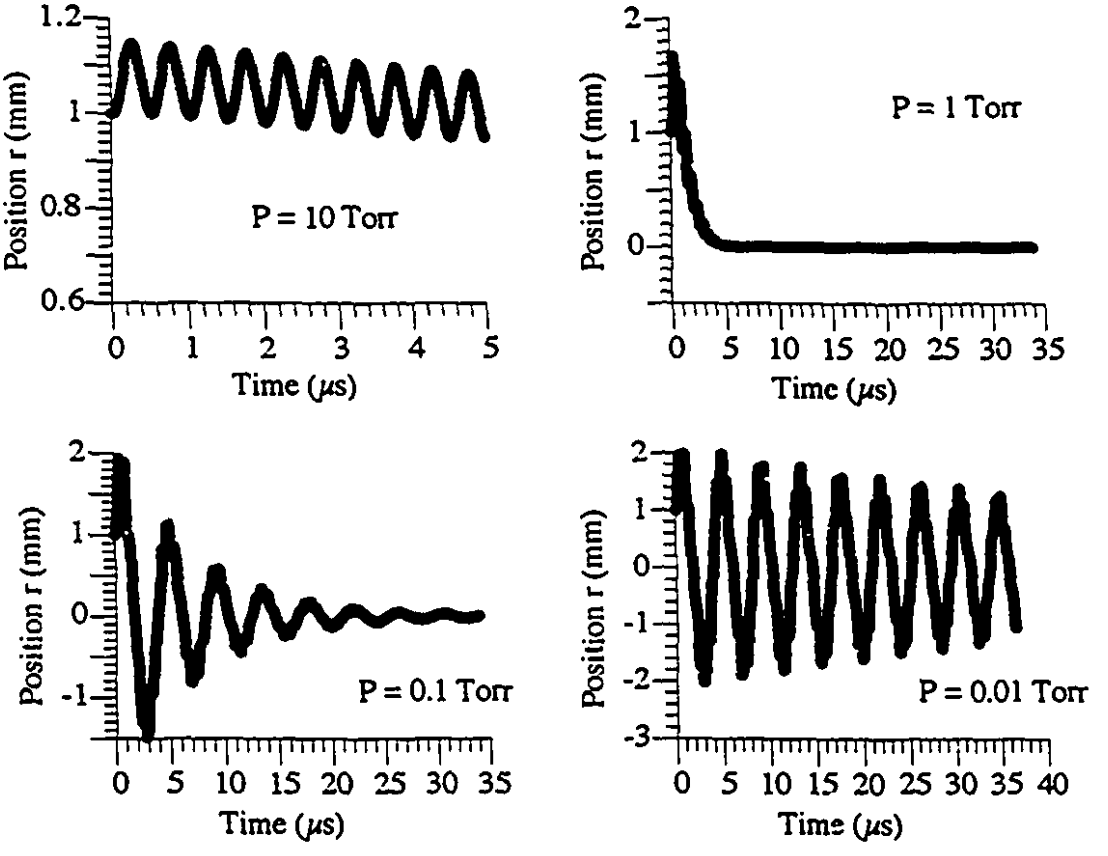


Figure 4.12: Examples of damped motion for $^{23}Na^{+}$ ions in Helium.

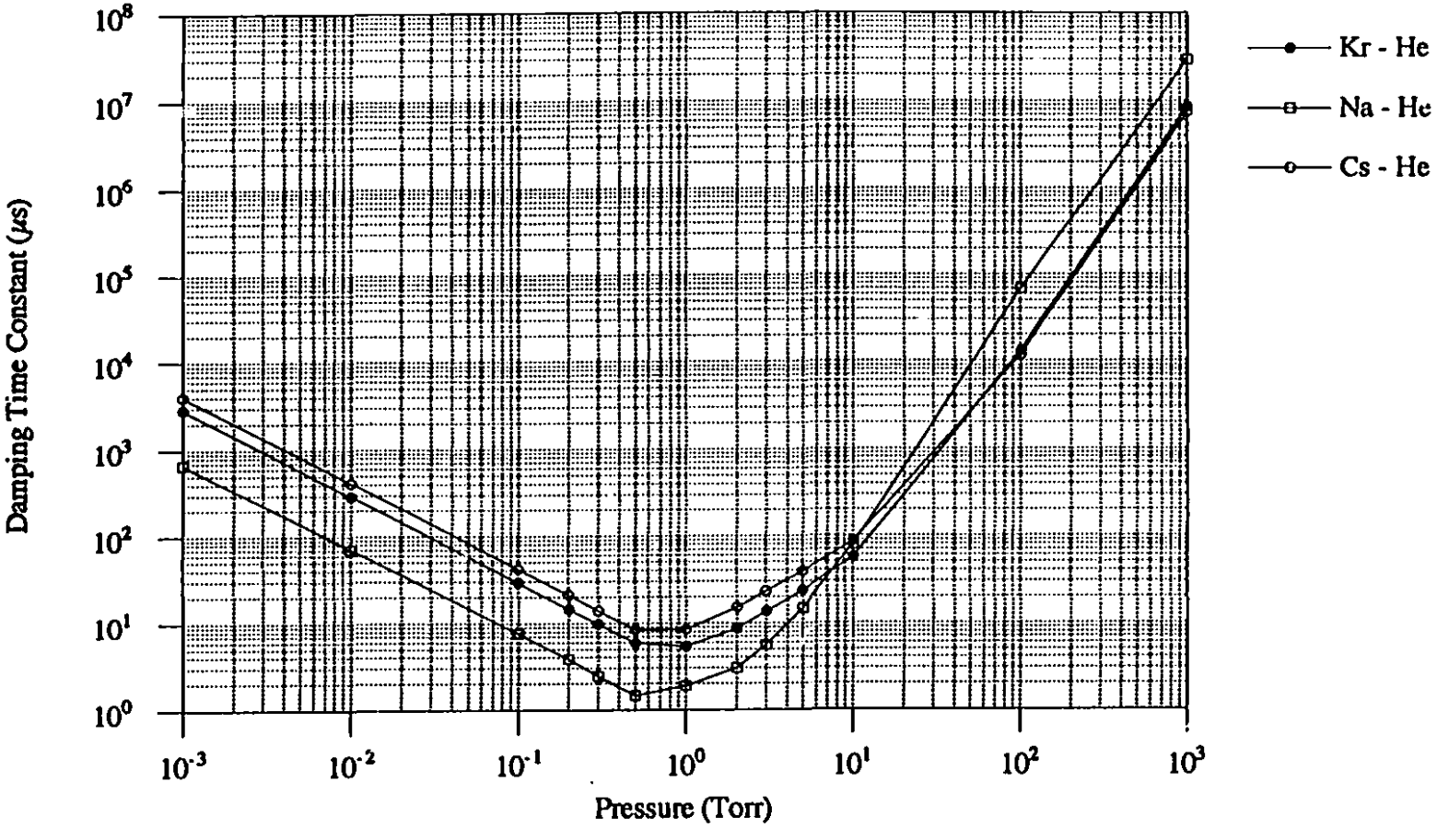


Figure 4.13: Numerical results for the damping time constants for three types of singly charged ions in Helium.

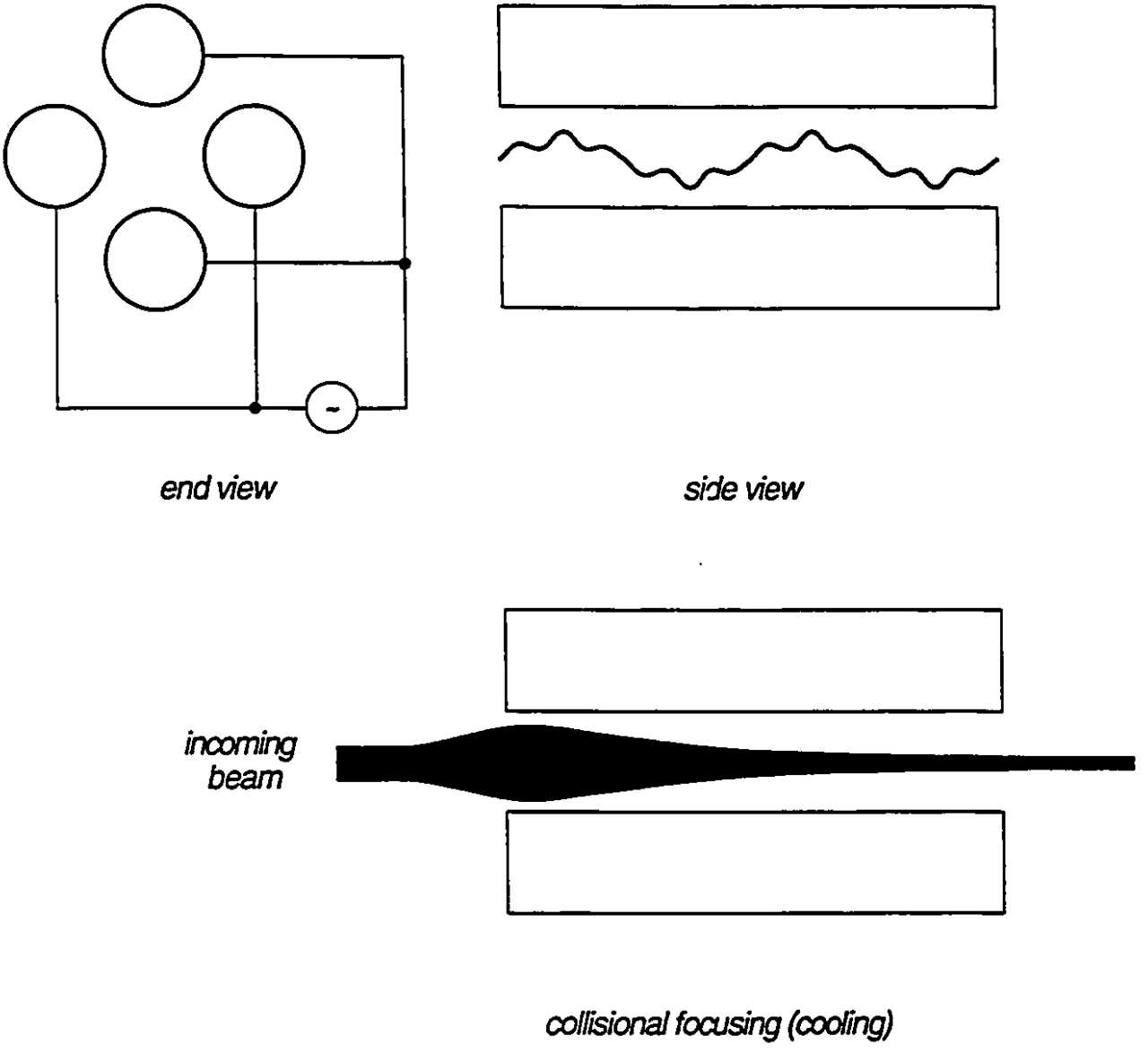


Figure 4.14: The schematics of a radiofrequency quadrupole rod system.

5. Conclusion

This thesis describes the creation and extensive testing of a unique computer package that performs high accuracy charge-particle optics simulations. It is unique in combining the power and flexibility of finite calculation techniques with calculation and display capabilities for phase space diagrams. The program features a modular object-oriented design and user-friendly interface. The major capabilities include:

- User specification of arbitrary electrode geometries, boundary conditions and initial phase space distribution for an ensemble of charged particles.
- An optimized finite difference calculation to generate a grid of electric potentials.
- Accurate and continuous calculation of electric fields using either (1) a sixth-order multipole expansion of the axial electric field or (2) a bi-linear interpolation of the off-axis grid points.
- A fifth-order Runge-Kutta numerical integration with adaptive step size control to completely determine the particle trajectories in phase space (or geometric space).
- Specification of time-varying functions, multiple potential maps, magnetic fields.
- A fully interactive, graphic display and output capability for the phase space diagrams and/or single particle trajectories.

After extensive testing, the program was used for the analysis of an ion source geometry, an ion beam decelerator, and an ion beam collisional focusing system using quadrupole rods at high pressure. The program further demonstrates that for paraxial beams, a multipole expansion of axial potentials can be used to speed up calculations while maintaining accuracy.

Using a thermodynamic weighting of the initial particle ensemble in phase space, the ion beam profile was simulated with the program and compared to intensity measurements.

Also, by incorporating relations for ion mobility and diffusion in gases, an extremely important simulation was performed to study the damping times and diffusion cross sections of ions cooled by background gas at elevated pressure, under the focusing

of a quadrupole rod structure. This work is very important for efforts to improve ion beam emittances at isotope separators as well as improving transmission in quadrupole mass filter instruments for trace element detection and biochemistry analysis. Moreover, it is the first time that detailed numerical integration of particle dynamics in gases has been performed using ion mobility concepts.

The program has thus proved to be extremely versatile, making it an essential part of on-going research efforts in nuclear physics, ion mobility and mass spectrometry.

References

[BERZ] Martin Berz. "Computational aspects of optics design and simulation: COSY INFINITY." Nuclear Instruments and Methods in Physics Research, A298 (1990), 473-479.

[BOOC] Grady Booch. "Object-Oriented Design with Applications." The Benjamin/Cummings Publishing Company, Inc., Redwood City, 1991.

[BRCI] K. L. Brown, F. Rothacker, D. C. Carey, CH. Iselin. "TRANSPORT: A Computer Program for Designing Charged Particle Beam Transport Systems." CERN 73-16, (1973), [also as SLAC 91 and NAL 91].

[DFBW] Alex J. Dragt, Etienne Forest, Kurt Bernardo Wolf. "Foundations of a Lie algebraic theory of geometrical optics." The CIFMO-CIO workshop on Lie Methods in Optics, Springer-Verlag, (1986), ISBN 0-387-16471-5.

[DOFR] D. J. Douglas and J. B. French. J. Am. Soc. Mass Spectrom. 3, 398 (1992).

[GDMV] A. M. Ghalambor Dezfuli, R. B. Moore, P. Varfalvy. "A Compact 65 KeV Stable Ion Gun for Radioactive Beam Experiments." Submitted to Nuclear Instruments and Methods, 1995.

[GOLD] Herbert Goldstein. "Classical Mechanics." Addison-Wesley Publishing Company, 1980.

[HERR] W. B. Herrmannsfeldt. "EGUN—An Electron Optics and Gun Design Program." SLAC-Report-331, (1988).

[ISMA] Morio Ishihara and Takekiyo Matsuo. "A new ray tracing code 'ELECTRA'." Nuclear Instruments and Methods in Physics Research, B70 (1992), 445-450.

[KIMT] Taemen Kim. Doctoral thesis to be submitted, 1996.

[KOME] S. E. Koonin and D. C. Meredith. "Computational Physics: FORTRAN Version." Addison-Wesley Publishing Company Inc., 1990

[KRWE] F. Krawczyk and T. Weiland. "A New Static Field Solver with Open Boundary Conditions in the 3D-CAD-System MAFIA." IEEE Transactions on Magntics. 24, 1 (1988), 55-58.

[LALI] L.D. Landau and E.M. Lifshitz. "Statistical Physics," Pergamon Press, Oxford, 1980.

[LBMO] M. David N. Lunney, F. Buchinger, R. B. Moore. "The temperature of buffer-gas cooled ions in a Paul trap." Journal of Modern Optics. 39, 2 (1992), 349-360.

[LOSI] D. A. Lowther and P. P. Silvester. "Computer Aided Design in Magnetics." Springer-Verlag, 1985.

[LUNP] M. David N. Lunney. "The phase space volume of ion clouds in Paul traps." Mcgill University, 1992.

[LWMO] M. David N. Lunney, J. P. Webb, R. B. Moore. "Finite-element analysis of radio-frequency quadrupole traps." Journal of Applied Physics. 65 (1989), 2883.

[MCDO] Kirk T. McDonald. "Design of the Laser-Driven RF Electron Gun for the BNL Accelerator Test Facility." IEEE Transactions on Elctron Devices. 35, 11 (1988), 2052-2059.

[MGDP] A. M. Ghalambor Dezfuli. Doctoral thesis to be submitted, 1996.

[MGVZ] R. B. Moore, A. M. Ghalambor Dezfuli, P. Varfalvy, H. Zhao. "Production, transfer and injection of charged particles in traps and storge rings." (Nobel symposium of trapped ions and related fundamental physics, Lysekil, Sweden, 1994), [to apear in Physica Scripta 1995].

[MLRS] R. B. Moore, M. D. N. Lunney, G. Rouleau, G. Savard. "The Manipulation of Ions using Electromagnetic Traps." Physica Scripta. 46 (1992), 569-574.

[MOOR] R. B. Moore. "The manipulation of charged particles into and out of electromagnetic traps." Hyperfine Interactions, 81 (1993), 45-70.

[PFTV] William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling. "Numerical Recipies in C: The Art of Scientific Computing." Cambridge University Press, (1988), ISBN 0-521-35465-X.

[PRZE] Andreas Przewloka. "GIOSP: A Program for the Design of "General Ion Optical Systems on PC's." User Manual Version 1.1.

[WUSL] R. F. Wuerker, H. Shelton, R. V. Langmuir. "Electrodynamic Containment of Charged Particles." Journal of Applied Physics. 30 (1959), 342-349.

[ZHAO] Hongyan Zhao. Doctoral thesis to be submitted, 1996.