MASTER

Numerical calculations on the internal beam of an isochronous low energy cyclotron

Op de Beek, P.

Award date:
1995

Eindhoven University of Technology
Department of Applied Physics
Group Particle Physics

Numerical calculations on the internal beam of an Isochronous Low Energy Cyclotron

Peter Op de Beek

Graduation report, november 1994
VDF-94-42

Supervisors:
dr. ir. J.A. van der Heide
prof. dr. ir. H.L. Hagedoorn

To Desirée

Abstract

Since January 1982 the small isochronous cyclotron ILEC is under construction at the Eindhoven University of Technology. In this report we describe the numerical calculations that have been performed on the internal beam of this cyclotron.

By linking the energy and radius of an accelerated particle to it s initial phase, we calculated the differential current distribution of the internal beam. The calculated current distribution can immediately be compared with the results we obtain from measurements with a moving differential target. By calculating the beam properties at extraction radius, we determined the conditions for single-turn extraction, and estimated an upper limit for the extraction efficiency. Also an estimation of the energy spread in the extraction region, and the effect of an internal aperture on the beam quality becomes possible. Finally, we discuss the results of the calculations of the orbit centre motion in ILEC under various conditions.

The results of the calculations indicate that ILEC should be able to produce a fairly intense proton beam of high quality. By limiting the initial phase width of the internal beam by an appropriately placed aperture, single-turn extraction is feasible, still leaving 30 % of the initial source-current. With only second harmonic acceleration, an extraction efficiency close to 100 % can be achieved, with an energy spread in the extracted beam of about 1 %. By applying the flattop system, this energy spread can be reduced to less than 0.1 %.

Contents

Chapter 1

Introduction

1.1 Introduction

The Isochronous Low Energy Cyclotron (ILEC) is a small fixed-energy cyclotron for a 3 MeV proton beam, which has been under construction at the Cyclotron Applications Group over the last decade. In fig. 1.1 a layout of the main components of ILEC is shown. In this chapter we first give a summary of the objectives of the minicyclotron project ILEC, and the forthcoming design-characteristics.

At the moment the cyclotron is fully in operation in 2^{nd} harmonic mode. Internal beam currents up to a modest 10 μA are possible. In the coming time, we expect the 6^{th} harmonic system to be in operation as well, so it remains to be seen if and when the external beam will meet the desired specifications.

1.2 Objectives of the ILEC project

The objectives which form the basis for the ILEC project are :

- the design and construction should require a low financial investment
- the cyclotron must be fit to produce a 3 MeV proton beam with low energy-spread (< 0.1 %) for micro-beam element analysis
- the possibility to produce high internal beam currents (up to 100 μA) for experimental study of space-charge effects

The main difficulty evolving from these three objectives, the construction of a small machine that can fulfill these rather stringent demands on beam quality, has led to a rather complicated design.

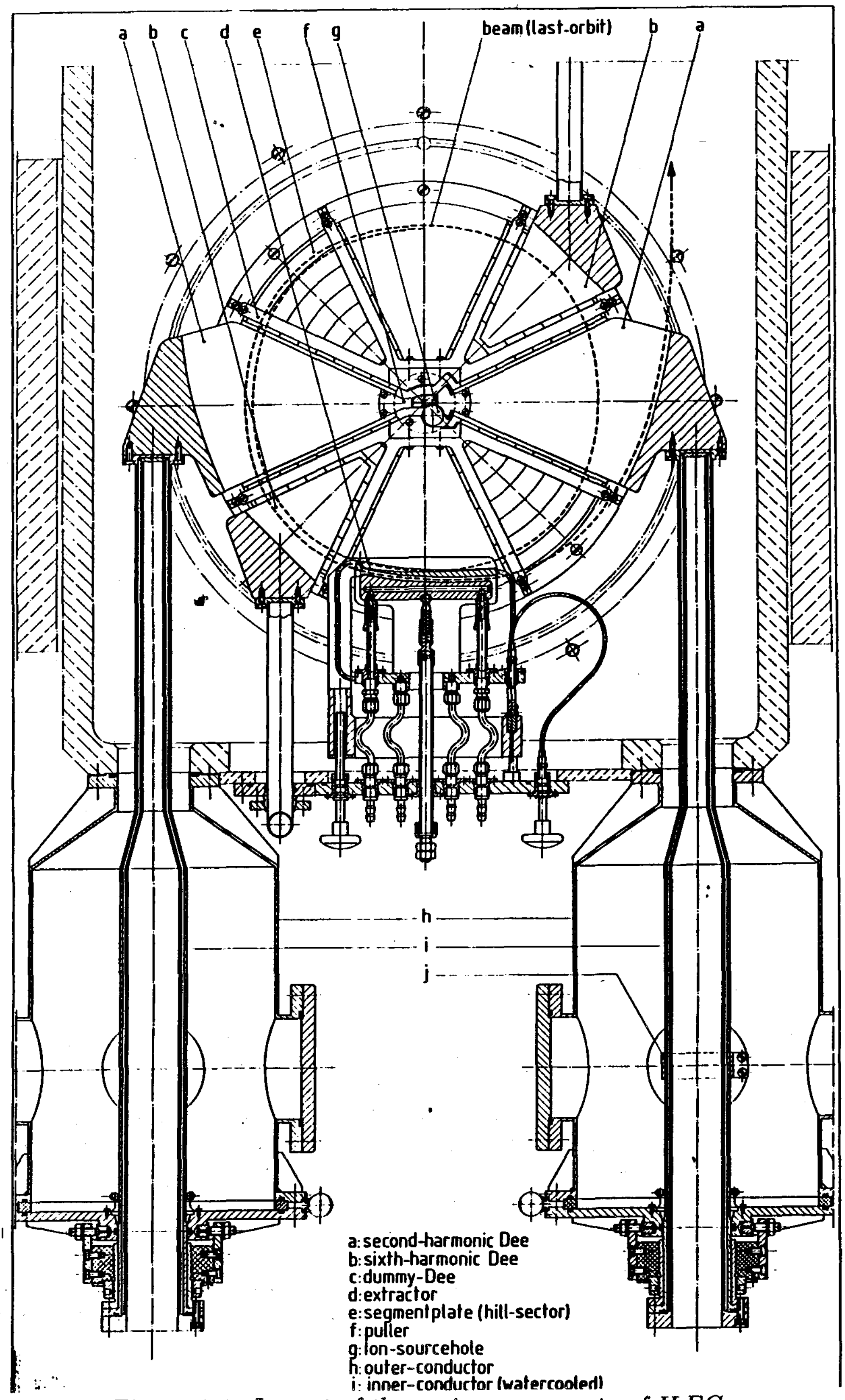


Figure 1.1: Layout of the main components of ILEC.

1.3 Design characteristics

ILEC is an isochronous cyclotron. In this type of cyclotrons the average angular velocity of the accelerated particles remains constant during the accelerating process, so the frequency of the RF-signal on the accelerating dee-system can be kept constant. Because of the relativistic mass increase of the particles during acceleration, the isochronous magnetic field should increase with the radius. In order to obtain good vertical stability of the beam in a radially increasing magnetic field, the main magnetic field is modulated azimuthally by a proper shape of the pole faces. Therefore this type of cyclotron is also called *azimuthally varying field* (AVF) cyclotron. For ILEC, the design of the main magnet is based on an azimuthally varying field with four-fold rotational symmetry : four sector-shaped hills with an azimuthal width of 40°, and four valleys with an azimuthal width of 50°. The radial growth of the magnetic field is realized by increasing the height of the hills.

The average magnetic field strength of approximately 1.42 T is generated by two relatively small (hence cheap) main coils of 140×192 ampère turns each. This implies the construction of a very small gap between the magnet poles, which should provide just enough space for the RF accelerating structure and the correction coils.

The RF accelerating structure used in ILEC consists of two pairs of dees. One pair, the *second harmonic* dee system, is placed in two opposite valleys of the poles, to keep the magnet gap as small as possible. Because the azimuthal extent of these dees is limited by the width of the valleys, we have two dees with an azimuthal width of 50°, operated in second harmonic push-push mode. This means that both dees are excitated in phase, at two times the revolution frequency of the accelerated protons.

In order to achieve the lowest possible energy spread at high beam currents a second pair of smaller dees, the *sixth harmonic* dee system, is placed between the hills of the upper and lower magnet poles. When these dees are operated in sixth harmonic mode, with proper phase and amplitude in respect to the second harmonic dee voltage, the sinusoidal shape of the energy gain as a function of the phase is altered into a more block-shaped fashion [11]. This technique is called *flat-topping*: adding odd higher harmonics to the basic accelerating voltage makes the energy gain per turn less dependent of the particle phase, thus resulting in wider phase acceptance and lower energy-spread.

Because the sixth harmonic dee system only effects beam quality, the basic operation mode of ILEC involves only second harmonic acceleration.

The internal ion source used in ILEC is positioned near the center, trough a 2 cm hole in the yoke. It is a self-heating PIG-source, described by Bennet [2], which has been scaled down to the desired proportions.

After the last revolution (at a radius of approximately 16 cm), the particles are extracted from their orbits by an electrostatic deflector, the *extractor*. The extractor is constructed according to a classical design [14], already used in former small cyclotrons.

In ILEC, the narrow magnet gap produces a strong negative gradient of the magnetic field towards the edge. This causes the extracted beam to diverge excessively in the horizontal direction. To compensate this defocussing influence of the fringing field, a *magnetic channel* is fixed between the upper and lower conductors of the second harmonic dees. The magnetic focussing channel used in ILEC was designed by de Regt [16], and is of the passive type. The small iron bars of which it is constructed adapt the local main magnetic field such that the extracted particles encounter a positive field gradient, providing the necessary refocussing.

1.4 Software for orbit calculations

In the past, various programs for the calculation of particle orbits in ILEC [1, 11] have been used. To gain insight in the influence of specific parameters on the particle motion, these calculations were principally based on numerical integration of the equations of motion following from application of the Hamilton formalism [6, 11]. In most cases [21, 16] integration of the initial value problem involved a classical fixed-step fourth-order Runge-Kutta method. Combined with the inevitably loss of higher-order terms in the Hamilton treatment this limits the precision and efficiency of the calculations. Calculations considering the current distribution of the internal beam by Theeuwen [23] clearly reveal the poor achievements of these codes when calculations demand high accuracy.

The wish to achieve the desired accuracy resulted in a new code for orbit calculations, based on the program CENTRUM [11]. The new program avoids the drawbacks of the former, by rigorously integrating the basic equations of motion, not only in the central region of the cyclotron, but for the complete orbit from source to extractor.

The use of an efficient and robust integration routine, introduced by Shampine and Watts [19], turned out to be a great improvement on both precision and speed. This integration routine is based on a 5^{th} order Runge-Kutta algorithm, developed by Fehlberg. Shampine and Watts [20] have moulded this scheme into a well-behaved variable-stepsize integration code, using the characteristic properties of the Fehlberg-algorithm to produce an estimate of the global truncation error.

A thorough revision of the original program to further enhance the efficiency, and the fast floating-point arithmetic of the new ALPHA workstations in our group, have drastically decreased the execution time.

We now have the possibility to simulate the influence of various parameters on the behaviour of the beam almost interactively. In the new program also the newly designed central region geometry is implemented. An additional program IDIF is written, to calculate the differential current distribution, as measured with a moving differential target.

Chapter 2

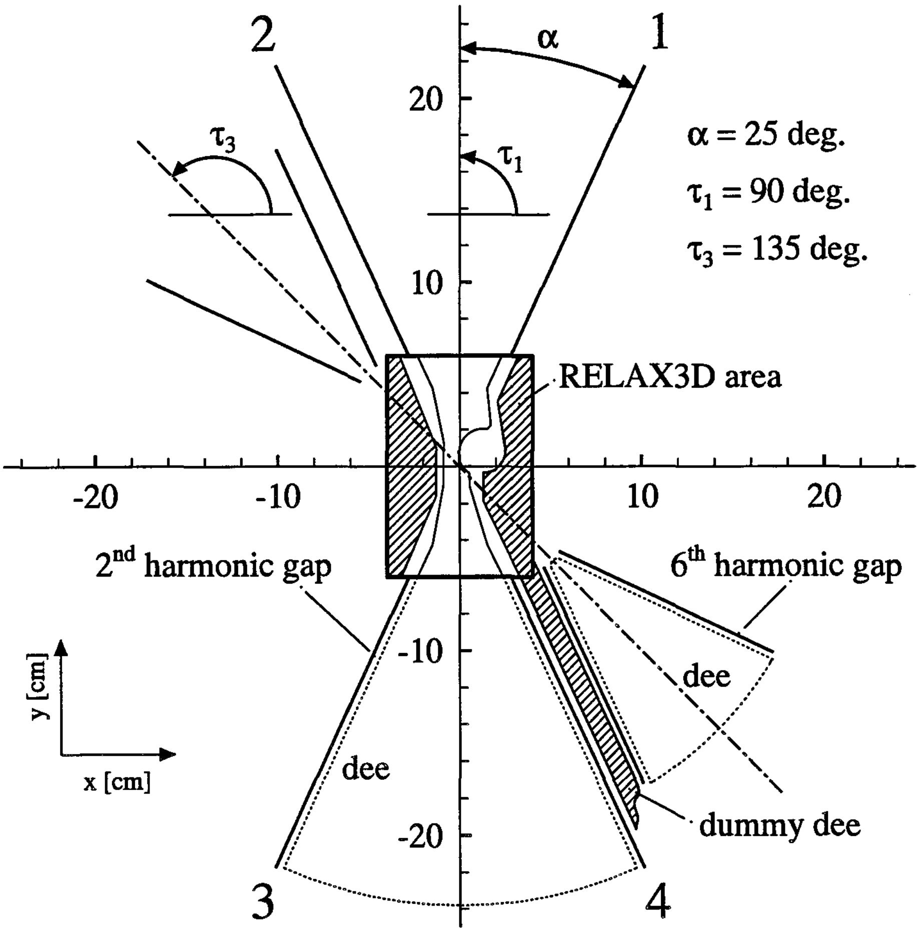
Representation of the magnetic and electric fields

2.1 Introduction

In the following sections we will discuss how the electric and magnetic fields in ILEC are represented in the numerical program. The electric field is evaluated in two different ways. For the central region, where the electric field structure is quite complex, we use a field map based upon the potential distribution calculated with RELAX3D. Outside the central region the field components in the accelerating gaps are calculated from an analytical expression known as the Hazewindus approach. From the measured main magnetic field a set of Fourier coefficients were calculated, which are evaluated with cubic-spline interpolation. This enables us to add lower harmonics to the (ideal) symmetrical field, and examine their specific influence on the internal beam. The magnetic field caused by the various correction coils and the magnetic focussing channel are treated as perturbations of the main field, acting in certain areas of the 'magnetic field map'. Eventually we derive the equations of motion we use in the computer program to calculate the orbit of an accelerated particle in ILEC.

2.2 The electric field

As a consequence of the numerical representation of the electric fields in ILEC, the RF accelerating structure can be divided in two areas: the *central region* and the *dee/dummy-dee gaps*. In the central region the field is calculated numerically, in the vicinity of the accelerating gaps we use the Hazewindus approach. A schematic overview of these regions is exhibited in figure 2.1. The field from the electrostatic deflector is incorporated as a perturbation of the magnetic field. We will therefore describe it in 2.3.4.



Figuur 2.1: Geometrical representation of the two regions of the electric field in ILEC. In the rectangular area the field is calculated with RELAX3D. We can distinguish the centerpart connecting the two 2^{nd} harmonic dees (which contains the puller), and the circular hole for the ion source. The shaded areas are part of the dummy-dee structure, which is kept at ground potential. The drawn lines represent the symmetry lines of the 2^{nd} and 6^{th} harmonic gaps, used in the Hazewindus approach. The 2^{nd} harmonic gaps are numbered from 1 to 4. For completeness, the position of the lower 6^{th} and 2^{nd} harmonic dees are indicated by the dotted lines.

2.2.1 The electric field in the central region

In the central region a detailed knowledge of the electric field is necessary to calculate the important first few turns with the desired accuracy. In the past the ILEC central electric field has been determined both by measurements on a 2:1 scale magnetic analogue model of the central region [7, 9] and numerical calculations with the program RELAX3D [11]. In 1993 a newly designed central region geometry was installed, which should possess better focussing properties, and a somewhat wider phase acceptance than the original. For this new central region geometry, the electric field was calculated with RELAX3D too. RELAX3D is a FORTRAN code from TRIUMF, which numerically evaluates the Poisson equation in three dimensions for a user-defined set of boundary conditions. The program is based on a finite-element scheme applied on a regular three-dimensional mesh, using successive over-relaxation to increase the speed. The boundary conditions have to be specified by a user-defined subroutine [12]. A fine grid with dimensions $n_x \times n_y \times n_z =$ $160 \times 240 \times 20$ is used with a grid spacing of 0.5 mm. Inside this region of $8 \times 12 \times 1$ cm^3 the electric field is calculated directly from the potentials at the grid points with a three-point differentiation routine. The electric field between grid points is found by linear interpolation between adjacent grid points.

2.2.2 The electric field in the gaps

Outside the central region, electric fields are only present in the vicinity of the dee/dummy-dee gaps. The shape of the electric field at the 2^{nd} and 6^{th} harmonic gap-crossing can be approximated by the Gaussian distribution function shown in figure 2.2, as proposed by Hazewindus et.al. [7]. The Hazewindus approach represents the distribution of the field component $E_{y'}$ perpendicular to the gap in a straight dee/dummy-dee system as a Gaussian distribution

$$E_{y'} = E_0 e^{-\frac{1}{2}\left(\frac{y'}{\sigma}\right)^2} \qquad (2.1)$$

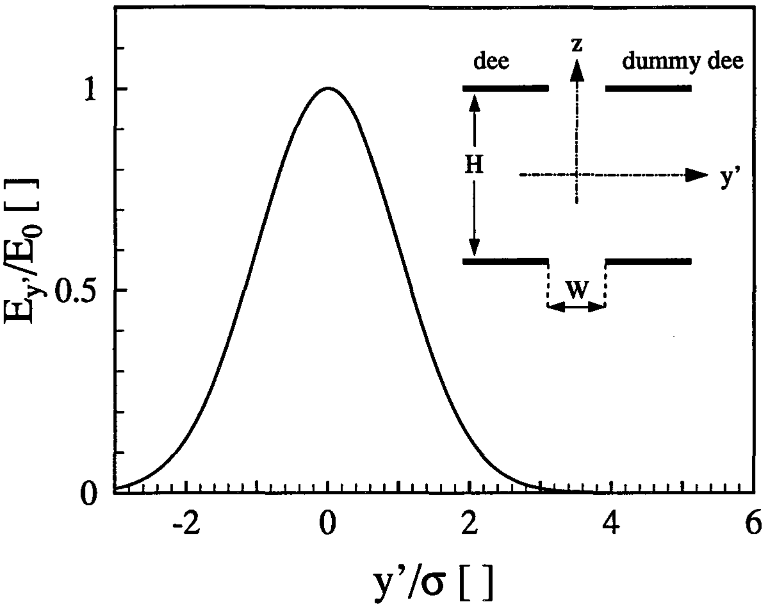
with the standard deviation σ related to the gap width W_{gap} and the dee-aperture H_{gap} according to the empirical relation

$$\sigma = \tfrac{1}{5}H_{gap} + \tfrac{2}{5}W_{gap} \qquad (2.2)$$

The value E_0 follows from the normalisation

$$\int_{-\infty}^{\infty} E_0 e^{-\frac{1}{2}\left(\frac{y'}{\sigma}\right)^2} dy' = V_{dee} \Rightarrow E_0 = \frac{1}{\sqrt{2\pi}}\frac{V_{dee}}{\sigma} \qquad (2.3)$$

For $0.3 \leq W/H \leq 3$ expression (2.1) is very accurate. Within the standard deviation ($|y'| \leq \sigma$) the fit with numerically calculated field distributions is nearly perfect. For $|y'| > \sigma$ the deviation is less than 10%.



Figuur 2.2: Representation of the electric field at a dee/dummy-dee gap. In ILEC we have $\sigma_1 = 6.2$ *mm for the* 2^{nd} *harmonic, and* $\sigma_3 = 5.4$ *mm for the* 6^{th} *harmonic gaps.*

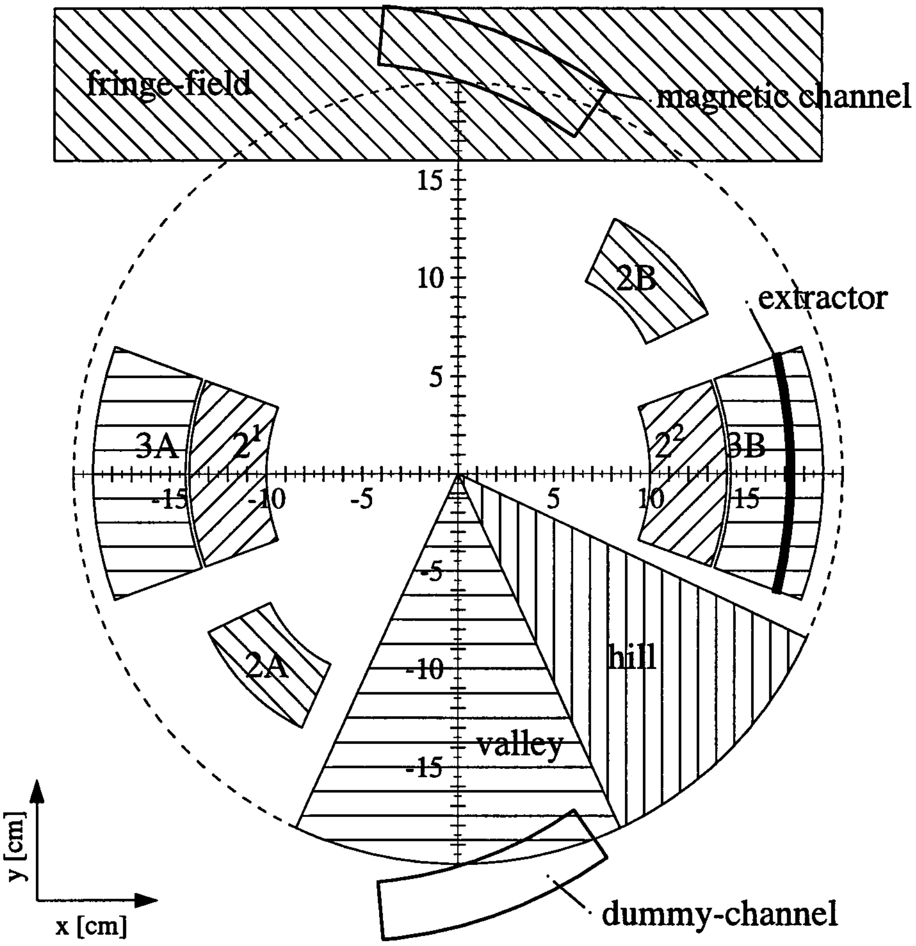
2.3 The magnetic field

In the orbit calculations the magnetic field in the median plane is regarded as a superposition of contributions from specific magnetic elements on the main magnetic field. The sphere of influence of these elements is restricted to a certain area on the field map, as is depicted in fig. 2.3.

2.3.1 Main magnetic field

ILEC is an isochronous cyclotron. In this type of cyclotrons the main magnetic field is modulated azimuthally by a proper shape of the pole faces. Therefore this type of cyclotron is also called *azimuthally varying field* (AVF) cyclotron. Because the average angular velocity of the accelerated particles in an AVF cyclotron remains constant during the accelerating process, the frequency of the RF-signal on the accelerating dee-system can be kept constant. Another advantage of the AVF field is that a good vertical stability of the beam can be obtained. For ILEC a four-fold symmetric AVF has been chosen : four sector-shaped hills with an azimuthal width of 40°, and four valleys with an azimuthal width of 50°. Between the hills the magnet gap decreases towards larger radii from 36 to 33 mm. The valleys are flat, and leave a gap width of 50 mm, which results in a field modulation of approximately 20 %.

The ILEC magnetic field has been measured with a Hall-probe, in a fully computerized



Figuur 2.3: The orientation of the various elements that determine the magnetic field in ILEC. The azimuthal modulation of the main magnetic field is achieved by four sector-shaped hills with an azimuthal width of 40°, *and four valleys with an azimuthal width of* 50°. *The valleys along the x axis contain two concentric pairs of correction coils:* $2^{1,2}$ *and* $3^{A,B}$. *The correction coil* $2^{A,B}$ *is placed between the hills. The magnetic focussing channel and the dummy channel are fixed inside the* 2^{nd} *harmonic dees, which lie in the valleys along the y axis.*

measuring machine [17, 21]. The z-component B_z of the magnetic induction in the median plane was mapped along concentric circles about the geometrical cyclotron centre. The accuracy in the position was 0.01 mm, the accuracy in the measured induction should be better than 0.05%. From symmetry considerations it follows that in the median plane ($z = 0$) $B_r(r,\theta) = B_\theta(r,\theta) = 0$, hence $B(r,\theta) = B_z(r,\theta)$. Therefore from now on $B(r,\theta)$ denotes the z-component of the magnetic induction in the median plane.

The measured AVF can be expressed in terms of an *average field* $\bar{B}(r)$ and a *flutter profile* $F(r,\theta)$

$$\begin{aligned}\bar{B}(r) &= \frac{1}{2\pi}\oint B(r,\theta)d\theta & (2.4)\\ F(r,\theta) &= \frac{B(r,\theta)}{\bar{B}(r)} - 1 & (2.5)\end{aligned}$$

which we can expand in a Fourier series

$$F(r,\theta) = \sum_n [A_n(r)\cos n\theta + B_n(r)\sin n\theta] \qquad (2.6)$$

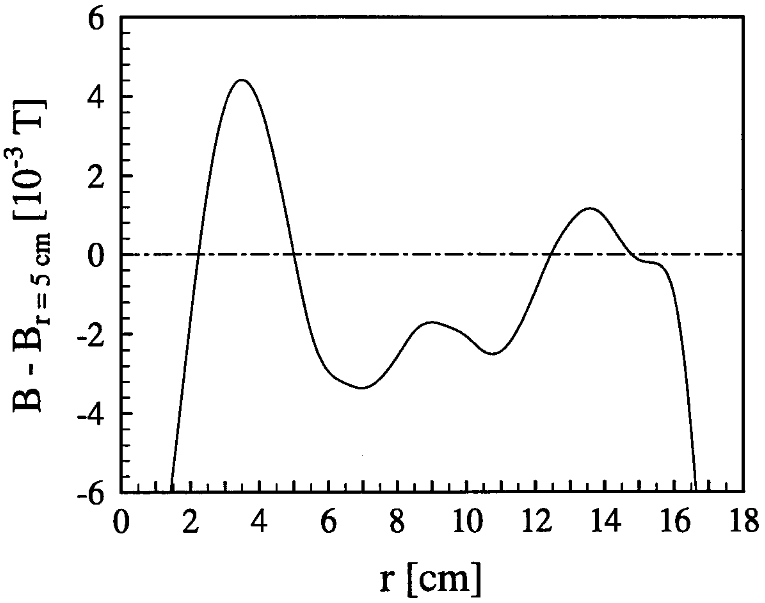
For the calculations we used the (smoothed) Fourier coefficients available from [21]. Since the AVF has fourfold symmetry, the coefficients with $n = 4k$ for $k = 1, 2, \ldots$ will be strongly dominant. Because the x-axis is also a symmetry-line, only cosine terms will be present. In the cyclotron centre the symmetry is disturbed by the eccentric placement of the 22 mm hole for the ion source. Therefore we used both the sine and cosine coefficients $A_n(r)$ and $B_n(r)$ for $n = 1, 2, 3$, and only cosine coefficients $A_n(r)$ for $n = 4, 8, 12, 16$. In the program the values for $A_n(r)$, $B_n(r)$ and $\bar{B}(r)$ at radial steps of 5 mm serve as internal nodes for the standard NAG-routine E02BAF, which determines a set of cubic-spline coefficients from a least-squares fit. The values for $A_n(r)$, $B_n(r)$ and $\bar{B}(r)$ are found by evaluating these splines with routines E02BBF or E02BCF (E02BCF also gives the first three derivatives in radial direction). Subsequently the magnetic induction in a point (r,θ) is calculated from (2.4) and (2.6). Figures 2.4 and 2.5 show the average field and the Fourier coefficients $A_{4k}(r)$ of the symmetrical field. We can use the analytical relations deduced in [6] for non-accelerated particles to calculate the radial and axial oscillating frequency ν_r and ν_z from the Fourier terms $A_{4k}(r)$ (see figures 2.6, 2.7). Further we can split the average induction $\bar{B}(r)$ in a constant part B_0 and a radius dependent part $\mu(r)$

$$\bar{B}(r) = B_0[1 + \mu(r)] \qquad (2.7)$$

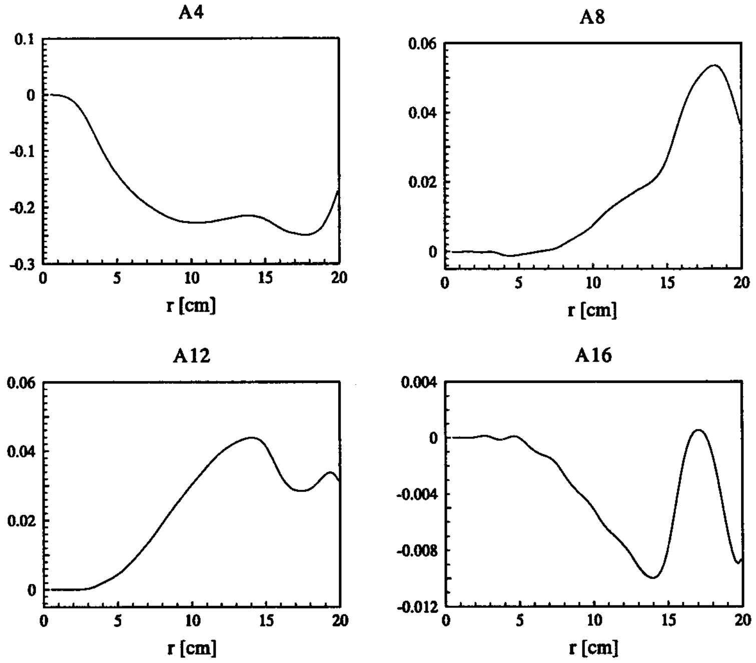
We define the *central field* B_0 as the (non-relativistic) homogeneous field belonging to the RF frequency ω_{RF}.

$$B_0 = \omega_0 \frac{m_0}{q} = \frac{\omega_{RF}}{h}\frac{m_0}{q} \qquad (2.8)$$

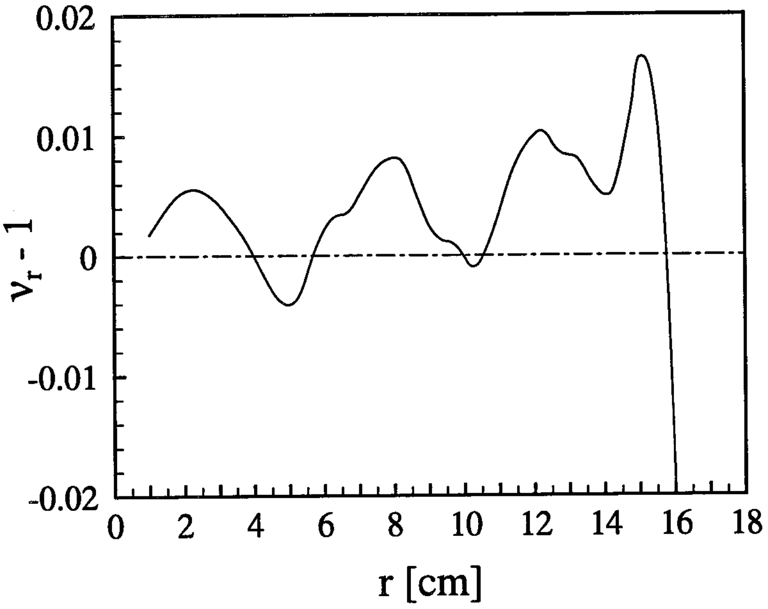
where ω_{RF} is the frequency of the RF-oscillator, and h the harmonic number of the acceleration. From [6] also an expression for $\mu(r)$ under the condition of isochronism is found. Figure 2.8 shows the deviation of the measured field $\bar{B}(r)$ from the isochronous field B_{iso} under slight variation of ω_0.



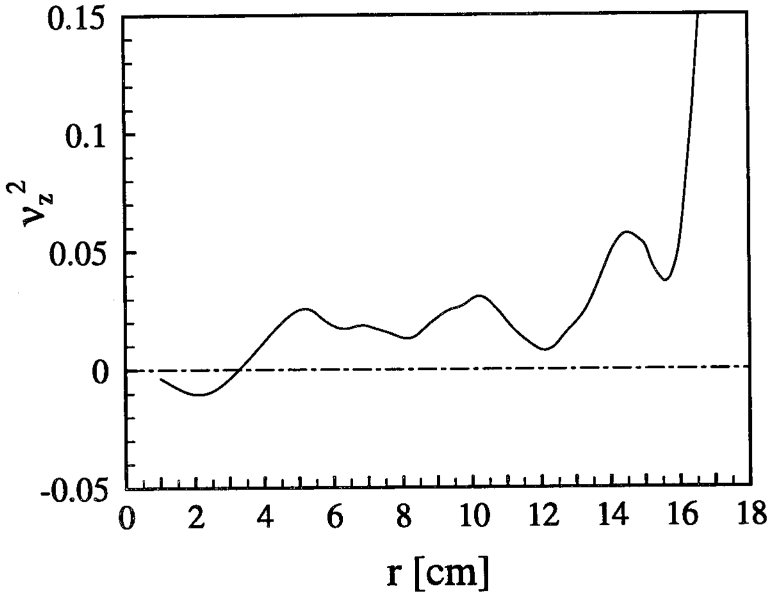
Figuur 2.4: The average magnetic field $\bar{B}(r)$ relative to the average field at $r = 5$ cm, which is 1.4252 T.



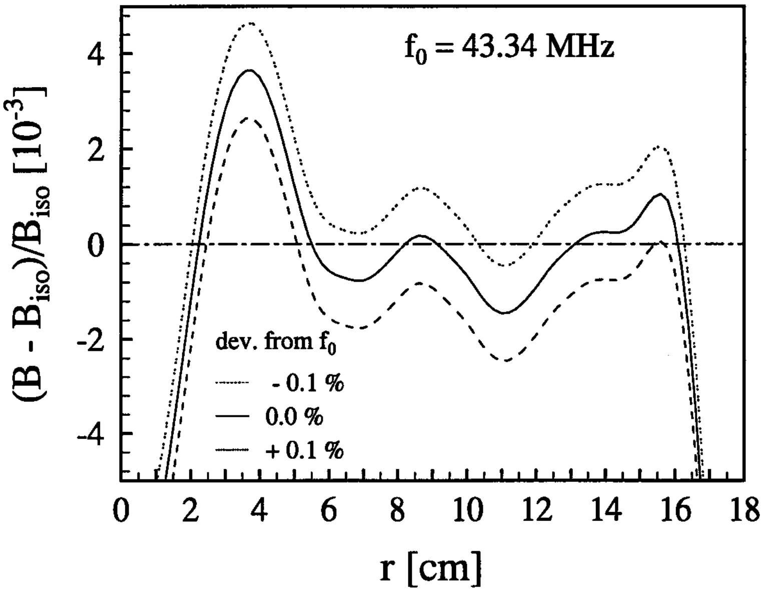
Figuur 2.5: The (smoothed) first four Fourier coefficients $A_{4k}(r)$ of the symmetrical field.



Figuur 2.6: The radial oscillation frequency of the ILEC magnetic field.



Figuur 2.7: The vertical oscillation frequency (squared) as a function of radius. The dip which occurs at $r \approx 12$ cm will cause an enlarged oscillation amplitude. Measurements with a moving target in this region showed a widening of the internal beam in the vertical direction.



Figuur 2.8: The relative deviation of the average magnetic field from the isochronous field, with the RF-frequency f_0 as parameter.

2.3.2 Correction coils

To enable trimming of the magnetic field and to compensate small unwanted first and second harmonic field components, three pairs of radially opposite correction coils are mounted at different radii between the pole faces of the main magnet, as depicted in figure 2.3. A first-harmonic field component is produced when the current trough two opposite coils is reversed in direction. We obtain a second harmonic when the currents trough each pair have the same direction. The small first and second harmonic perturbations caused by these correction coils are incorporated in exactly the same way as the various harmonic components of the main field. The radial dependence of the harmonics $C_c(r)$ are supposed to have a Gaussian profile

$$C_c(r) = e^{-\frac{1}{2}\left(\frac{r-r_c}{\sigma_c}\right)^2} \tag{2.9}$$

with r_c the radius of the coil centre and σ_c the characteristic coil width. For the resulting first and second harmonic field we then find

$$B_{1,c}(r,\theta) = \hat{B}_{1,c}C_c(r)\cos(\theta-\theta_c) \tag{2.10}$$

$$B_{2,c}(r,\theta) = \hat{B}_{2,c}C_c(r)\cos 2(\theta-\theta_c) \tag{2.11}$$

where θ_c is the azimuthal position of the coil centre. Both $\hat{B}_c$ and σ_c depend on the dimensions of the considered pair of coils and the gapheight. The maximum $\hat{B}_c$ is also determined by the number of ampère turns of the considered coils. Figure 2.3 shows the position and dimensions of the correction coils in ILEC.

2.3.3 Passive magnetic focussing channel

The magnetic focussing channel used in ILEC was designed by de Regt [16], who also implemented a routine for the calculation of the field perturbation the channel brings about. In ILEC, the narrow magnet gap produces a strong negative gradient of the magnetic field towards the edge. This causes the extracted beam to diverge excessively in the horizontal direction. To compensate this defocussing influence of the fringing field, a gradient of approximately 60 Tm^{-1} over a path length of 10 cm is necessary. Because the focussing field should best become effective directly behind the extractor exit, the magnetic channel is placed between the upper and lower conductors which form the second harmonic dees. As a consequence of the restricted physical dimensions (the dee aperture is only 15 mm), the channel used in ILEC is of the passive type. Due to the gradient in the main magnetic field, a mechanical force towards the cyclotron centre will act on the magnetic channel, and on the dee-structure to which it is fixed. To counteract this force and to avoid the introduction of a first harmonic perturbation, an identical *dummy channel* has been placed in the opposite dee.

2.3.4 The extractor

The electric field inside the extractor for a particle with velocity v is represented by lowering the main magnetic field with an amount E_{extr}/v, with

$$E_{extr}(\theta) = \frac{V_{extr}}{w_{extr}(\theta)} \tag{2.12}$$

w_{extr} is the radial distance between electrode and septum at an azimuth θ.

2.3.5 The fringe field

The main magnetic field behind the extractor (without the magnetic focussing channel) has been measured in a rectangular area (see fig 2.3) with a grid spacing of 3 mm. The magnetic field in a particular point is calculated by linear interpolation between four surrounding gridpoints.

2.4 Equations of motion

To find the equations of motion we write down the Lorentz force acting on a particle with charge q, which is moving in an electric field $\vec{E}$ and a magnetic field $\vec{B}$:

$$\vec{F} = q\left[\vec{E} + \vec{v} \times \vec{B}\right] \tag{2.13}$$

Because of the cross product $\vec{v} \times \vec{B}$ we have in general coupling between vertical and radial motion, which leads to rather complex equations of motion. Since the electric and magnetic fields are symmetric with respect to the median plane, we have for $z = 0$ $B_x = B_y = 0$.

This means that if we only consider the particle motion in the median plane, and neglect any influence from the vertical motion by putting $v_z = 0$, we find:

$$\begin{aligned}\frac{d^2x}{dt^2} &= \frac{q}{m}\left\{E_x(x,y) - \frac{dy}{dt}B_z(x,y)\right\}\\ \frac{d^2y}{dt^2} &= \frac{q}{m}\left\{E_y(x,y) + \frac{dx}{dt}B_z(x,y)\right\}\end{aligned}$$

For a particle with positive charge q these equations represent a counterclockwise rotation when viewed in the direction of $\vec{B}$. For ILEC this means that the induction of the main magnetic field is pointing in the direction of the negative z axis. When we introduce the dimensionless independent variable $\tau = \omega_0 t$ we get

$$\frac{ds}{dt} = \frac{ds}{d\tau}\frac{d\tau}{dt} = \omega_0 \dot{s} \tag{2.14}$$

Combining this with the central field B_0 defined in (2.8) yields

$$\begin{aligned}\ddot{x} &= \frac{m_0}{m}\left\{\frac{m_0}{qB_0^2}E_x(x,y) - \dot{y}\frac{B_z(x,y)}{B_0}\right\}\\ \ddot{y} &= \frac{m_0}{m}\left\{\frac{m_0}{qB_0^2}E_y(x,y) + \dot{x}\frac{B_z(x,y)}{B_0}\right\}\end{aligned}$$

Now the derivatives to the independent variable τ all have the dimension of length, where the independent variable itself can be regarded as an angle. When we finally insert the time-dependent electrical field, resulting from the 2^{nd} and 6^{th} harmonic acceleration, the complete equations are

$$\begin{aligned}\ddot{x} &= \frac{m_0}{m}\left\{\tfrac{1}{2}R_1^2\tilde{E}_{x,1}(x,y)\sin\left[h(\tau-\tau_1)\right] - \tfrac{1}{2}R_3^2\tilde{E}_{x,3}(x,y)\sin\left[3h(\tau-\tau_3)\right] - \dot{y}\frac{B_z(x,y)}{B_0}\right\}\\ \ddot{y} &= \frac{m_0}{m}\left\{\tfrac{1}{2}R_1^2\tilde{E}_{y,1}(x,y)\sin\left[h(\tau-\tau_1)\right] - \tfrac{1}{2}R_3^2\tilde{E}_{y,3}(x,y)\sin\left[3h(\tau-\tau_3)\right] + \dot{x}\frac{B_z(x,y)}{B_0}\right\}\end{aligned} \tag{2.15}$$

In these equations all parameters concerning the main (2^{nd} harmonic) dee system are provided with an index 1, the parameters concerning the flat top (6^{th} harmonic) dee system have index 3.

$$\begin{array}{rcl}\tau & = & \omega_0 t\\ & : & \text{the independent variable}\\ h & : & \text{harmonic number}\\ \dfrac{m_0}{m} & = & \sqrt{1 - \dfrac{\omega_0^2}{c^2}\left(\dot{x}^2 + \dot{y}^2\right)}\end{array}$$

$$
\begin{array}{rcl}
 & : & \text{relativistic mass correction} \\
R_1^2 & = & 2\dfrac{m_0 \hat{V}_{dee,1}}{qB_0^2} \\
R_3^2 & = & 2\dfrac{m_0 \hat{V}_{dee,3}}{qB_0^2} \\
B_0 & = & \dfrac{m_0\omega_0}{q} = \dfrac{m_0\omega_{RF}}{hq} \\
 & : & \text{homogeneous field belonging to } \omega_0 \\
\tilde{E}_1(x,y) & : & \text{scaled electric field for } V_{dee,1} = 1 \text{ V} \\
\tilde{E}_3(x,y) & : & \text{scaled electric field for } V_{dee,3} = 1 \text{ V} \\
\hat{V}_{dee,1} & : & \text{amplitude } 2^{nd} \text{ harmonic accelerating voltage} \\
\hat{V}_{dee,3} & : & \text{amplitude } 6^{th} \text{ harmonic accelerating voltage} \\
\tau_1 & : & \text{azimuth of } 2^{nd} \text{ harmonic dee-system} \\
\tau_3 & : & \text{azimuth of } 6^{th} \text{ harmonic dee-system}
\end{array}
$$

The Physical interpretation of constants R_1^2 and R_3^2 is that they represent the squared radii of particles with kinetic energy $q\hat{V}_1$ and $q\hat{V}_3$ in the central field B_0.

The equations of motion form a set of two coupled ordinary differential equations (ODE's) which we shall try to solve numerically. Because most numerical methods can only solve first order ODE's, we must reduce the order. This will result in the simultaneous integration of an array of four coupled equations $x(1)$ to $x(4)$, of which there are two trivial:

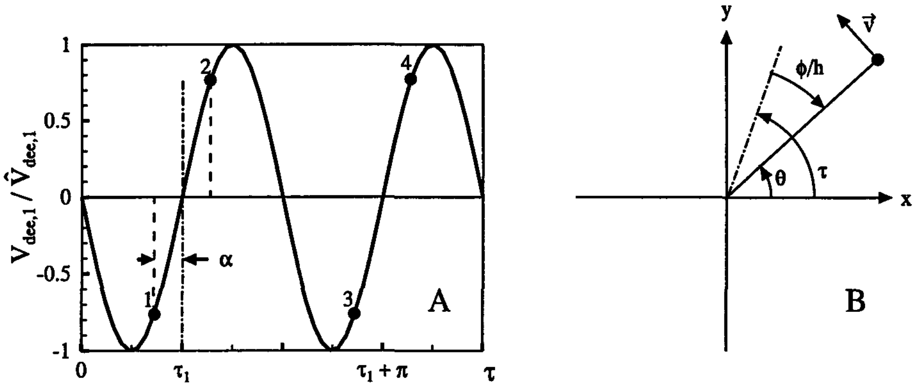
$$
\begin{array}{lcl}
x(1) = x & \Rightarrow & \dot{x}(1) = \dot{x} \\
x(2) = \dot{x} & \Rightarrow & \dot{x}(2) = \ddot{x} \text{ from eq. (2.15)} \\
x(3) = y & \Rightarrow & \dot{x}(3) = \dot{y} \\
x(4) = \dot{y} & \Rightarrow & \dot{x}(4) = \ddot{y} \text{ also from eq. (2.15)}
\end{array}
$$

The use of the independent variable τ instead of the time t enables the definition of the *high frequency phase* of a particle:

$$\varphi_{hf} = h(\tau - \theta(\tau)) \tag{2.16}$$

In figure 2.9 [B] we see that φ_{hf} gives the phase slip between the independent variable τ and the actual azimuthal position of the particle in terms of the accelerating voltage. For the second harmonic accelerating voltage $V_{dee,1}$ as a function of τ we have

$$
\begin{array}{rcll}
V_{dee,1} & = & \hat{V}_{dee,1} \sin\left[h(\tau - \tau_1)\right] & (2.17) \\
 & = & \hat{V}_{dee,1} \sin\left[\varphi_{hf} + h(\theta - \tau_1)\right] & (2.18)
\end{array}
$$



Figuur 2.9: [A] Particles with $\varphi_{hf} = 0$ will cross the dee-center when $V_{dee} = 0$, resulting in maximum energy gain per turn. The numbers 1 to 4 correspond with the numbering of the gaps in fig. 2.1. [B] The relation between the independent variable τ, the actual azimuthal position of the particle θ, and the high frequency phase φ_{hf}. The independent variable can be regarded as a phasor, rotating with angular velocity ω_0.

which means that $V_{dee,1} = 0$ when a particle with $\varphi_{hf} = 0$ is at azimuthal position τ_1. The azimuth τ_1 (see figure 2.1) coincides width the mid-dee line. This implies that a particle with $\varphi_{hf} = 0$ will experience maximum energy gain per turn (see figure 2.9).

The definition of the high frequency phase also links the inital conditions for the equations of motion to the initial azimuth θ_{in} at which the particle leaves the ion source, and the value φ_{in} of the high frequency phase at the start :

$$\tau_{in} = \frac{\varphi_{in}}{h} + \theta_{in}$$

Chapter 3

The numerical solution of ordinary differential equations

3.1 Introduction

In this chapter we first treat some mathematical aspects and general properties of discrete methods, from which we derive a method to estimate the error in the calculated solution. Our goal was to solve the equations of motion with an accuracy that was high enough to simulate the radial current distribution, as measured in practice with a moving differential probe. In order to investigate the effect of various parameters on the internal beam structure in a comfortable way, the efficiency of the calculation must be high.

In the past, numerically solving the equations of motion which are derived using the Hamilton formalism has proved to be very efficient [18, 11], but is not the best solution to gain high accuracy. The equations of motion following from application of the Hamilton formalism on accelerated particles in an AVF [18] turn out to become quite complicated. Moreover, in the derivation of these Hamilton functions higher-order terms in various power series expansions and cross-products have been neglected to keep the results comprehensible. Especially in the central region of the cyclotron, where the electric field shape is quite complicated, we may not expect to achieve high enough accuracy for the detailed numerical analysis we want to perform on the calculated orbits. Therefore we choose to directly solve the basic equations of motion in cartesian coordinates. Both the desired efficiency and error control of the orbit integration are achieved by the use of a powerful variable-step algorithm. The influence of discontinuities on convergence and error control is treated more elaborately, to point out that reasoning from a pure physical point of view can lead to numerical problems.

On the basis of some common computational problems we encountered, we choose to use the code GERK, which is implemented by Shampine and Watts [19].

3.2 The mathematical problem

The typical initial value problem for a system of ordinary differential equations is, in vector form:

$$\begin{aligned} \frac{d\vec{x}}{dt} &= \vec{f}(t,\vec{x}) \text{ for } t \in [t_0, t_1] && (3.1) \\ \vec{x}(t_0) &= \vec{x}_0 && (3.2) \end{aligned}$$

Here $\vec{f}$ is a continuous function for $t \in [t_0, t_1]$. A solution of (3.1) is a vector function $\vec{x}(t)$ defined on the interval $[t_0, t_1]$ which has a continuous derivative satisfying (3.1) as an identity and which for $t = t_0$ has the initial value $\vec{x}_0$ specified by (3.2). Because higher order differential equations[1] which can be expressed in the form:

$$\frac{d^n\vec{x}}{dt^n} = \vec{f}(t, \frac{d^{n-1}\vec{x}}{dt^{n-1}}, \frac{d^{n-2}\vec{x}}{dt^{n-2}}, \ldots, \frac{d\vec{x}}{dt}, \vec{x}) \tag{3.3}$$

can be converted to a system of first-order ODE's by a standard change of variable, a valid theory for the solution of (3.1), (3.2) can also be applied to these higher-order ODE's.

It is typically assumed that $\vec{f}$ satisfies a *Lipschitz condition* with constant L, which means that for all $\vec{u}$, $\vec{v}$:

$$||\vec{f}(t,\vec{u}) - \vec{f}(t,\vec{v})|| \leq L||\vec{u} - \vec{v}|| \text{ for } t \in [t_0, t_1] \tag{3.4}$$

A more comprehensible condition which implies (3.4), is that all the partial derivatives of $\vec{f}$ have an upper bound K:

$$\left|\frac{\partial f_i}{\partial x_j}(t,\vec{x})\right|_{t\in[t_0,t_1]} \leq K \tag{3.5}$$

which implies that there is exactly one solution of (3.1), (3.2) when $\vec{f}$ *is a continuous function.* When a point of discontinuity in the derivative function is determined solely by the independent variable t, it is possible to divide the interval $[t_0, t_1]$ in appropriate pieces on which (3.5) still holds. However, if the dependent variable $\vec{x}$ defines the discontinuity, we can not expect all of the theory to be valid, as is illustrated later in this chapter. Most of the theory and the numerical algorithms for a single ODE carry over to a system of ODE's. Therefore we will omit the explicit vector notation from here on.

3.3 Properties of discrete methods for ODE's

All numerical methods for solving ODE's have in common that they *discretize* the problem. The typical code steps trough the interval $[t_0, t_1]$ producing approximate solutions at certain

[1] Some authorities [3] have developed methods for the direct integration of equations of higher order, arguing that reduction to a first order system increases both the error and the necessary number of operations. The treatment of these matters by Henrichi [8] clearly indicates that the truncation error is generally not increased and the round-off error is frequently substantially decreased when an equation of higher order is first reduced to a first order system, and solved by an equivalent method for such systems.

mesh points. Proceeding from an approximation x_n to the true solution $x(t_n)$ at the meshpoint t_n, we try to compute an approximation x_{n+1} at the next mesh point $t_{n+1} = t_n + h_{n+1}$. The increase h_{n+1} of the independent variable is called the *stepsize*. Often the stepsize is chosen to have a constant value h over the entire interval. We define the *local solution* $u_n(t)$ as the solution of

$$\begin{aligned} \frac{du_n}{dt} &= f(t_n, u_n) \qquad (3.6) \\ u_n(t_n) &= x_n \qquad (3.7) \end{aligned}$$

which is the exact solution of the ODE, with the last computed value x_n as its initial value. Now we try to approximate this local solution over the step h_{n+1}. To do this, most numerical methods use no other information except for t_n, h_{n+1}, x_n, and the ability to evaluate $f(t, x)$. The error of this approximation is called the *local error*

$$\epsilon_l(t_{n+1}) = u_n(t_{n+1}) - x_{n+1} \qquad (3.8)$$

The most common situation in practice, especially for physical systems, is that the solution curves of (3.1) are more or less parallel. When a local error is committed, the code moves from one solution to a solution with a different initial condition. Because the curves do not diverge strongly, the effect of repeated local errors will be that the calculated solution slowly drifts away from the true solution, with initial condition (3.2).

An other way of looking at local errors is in terms of a truncated Taylor series. If we expand both the exact local solution $u_n(t_{n+1})$ and its approximation x_{n+1} over a step with length h in a Taylor series based on the point t_n

$$\begin{aligned} u_n(t_{n+1}) &= u_n(t_n) + \frac{h}{1!}\frac{du_n(t_n)}{dt} + \frac{h^2}{2!}\frac{d^2u_n(t_n)}{dt^2} + \ldots \\ x_{n+1} &= u_n(t_n) + \frac{h}{1!}c_1 + \frac{h^2}{2!}c_2 + \ldots \end{aligned}$$

we can find a last term whose coefficient is the same in both series, and remains constant over the step. The corresponding exponent of the stepsize h is called the *order* k of the method;

$$\begin{aligned} c_1 &= \frac{du_n(t_n)}{dt}, c_2 = \frac{d^2u_n(t_n)}{dt^2}, \ldots, c_k = \frac{d^ku_n(t_n)}{dt^k} \\ c_{k+1} &\neq \frac{d^{k+1}u_n(t_n)}{dt^{k+1}} \end{aligned}$$

which means that

$$u_n(t_{n+1}) = x_{n+1} + \mathcal{O}(h^{k+1}) \qquad (3.9)$$

For a method of k^{th} order we can now refer to the local error as the *local truncation error* being $\mathcal{O}(h^{k+1})$.

This means that when we evaluate the solution twice, either with a different stepsize or with a method of different order, it is possible to estimate the local truncation error over a single step, as described in [15, 10, 19]. By relating the stepsize to the estimated error, it is possible to keep the truncation error within a specified tolerance. This technique is called *local stepsize adjustment* or *adaptive stepsize control.* The gains in efficiency resulting from local stepsize adjustment can be tremendous. Another benefit of local error control is that the beginning of *numerical* instability also results in a growing local error, so the stepsize will be automatically reduced to keep the calculation stable.

What we are really interested in is the *true* or *global error*, which is

$$\begin{aligned} \epsilon_g(t_{n+1}) &= x(t_{n+1}) - x_{n+1} && (3.10) \\ &= \epsilon_l(t_{n+1}) + \{x(t_{n+1}) - u_n(t_{n+1})\} && (3.11) \end{aligned}$$

The second term in braces in equation (3.11) depends on the stability of the differential equation: the degree of divergence the family of solutions exhibits, and thus the way errors propagate. For a small stepsize h_{n+1} this term can be expressed in terms of the Jacobian J_n in $(t_n, x(t_n))$ of the ODE, which leads to the relation

$$\epsilon_g(t_{n+1}) = \epsilon_l(t_{n+1}) + \{1 + J_n\}\epsilon_g(t_n) \tag{3.12}$$

The global error at t_n is multiplied by $\{1 + J_n\}$, which is called the *amplification factor.* The stability of a system turns out to be related to the *eigenvalues* of the Jacobian. In general, these eigenvalues are complex numbers, wose values are not constant but depend on t. It can be shown that stability is gouverned by the eigenvalue with the largest real part. For most problems a stable solution corresponds with all eigenvalues having negative real parts.

Because the stability of the ODE is determined by the eigenvalues of J_{n+1}, and thus is likely to change as the evaluation of the solution proceeds, it is not generally possible to control the global error by local stepsize adjustment alone. However, if the ODE satisfies certain conditions, for a fixed order, one-step code it is possible to calculate a reliable estimate of the global error by means of *global extrapolation.* This method involves parallel integration, one carried out with half the step size of the other, using the same basic method. By using global extrapolation on the parallel solutions we can estimate the global error of the *more accurate result*, as is justified in [8].

3.4 Global error estimation

Let $x_n(t_1, h)$ denote an approximate solution obtained at the point t_1 using a variable step size h with $h = \zeta(t)H$ for H constant and $0 < \zeta(t) \leq 1$. Let $x_{2n}(t_1, h/2)$ be the approximate solution at t_1, calculated with step size $\zeta(t)H/2$. Then the global errors for a method of order k are

$$\begin{aligned} x_n(t_1, h) - x(t_1) &= (H)^k p(t_1) + \mathcal{O}\left(H^{k+1}\right) \\ x_{2n}(t_1, h/2) - x(t_1) &= (H/2)^k p(t_1) + \mathcal{O}\left(H^{k+1}\right) \end{aligned}$$

where $p(t_1)$ denotes the *magnified error function*, which in itself is the solution of an initial value problem, defined by the ODE and the method. If the solution has continuous derivatives of order $\leq k+1$, global extrapolation becomes valid and for sufficiently small values of H we get

$$x_n(t_1,h) - x_{2n}(t_1,h/2) = \left(2^k - 1\right)\left(x_{2n}(t_1,h/2) - x(t_1)\right) + \mathcal{O}\left(H^{k+1}\right) \tag{3.13}$$

which yields for the estimate of the global error in $x_{2n}(t_1,h/2)$ [2]

$$\begin{aligned} \epsilon_g(t_1) &= x_{2n}(t_1,h/2) - x(t_1) \\ &\approx \frac{x_n(t_1,h) - x_{2n}(t_1,h/2)}{2^k - 1} \end{aligned} \tag{3.14}$$

3.5 Convergence and stability

When the process of step halving or step doubling is proceeded, we obtain from eq. (3.14) an expression which is of basic importance for the validity of the solution we obtain:

$$C_n(t_1) = \frac{x_n(t_1,h) - x_{2n}(t_1,h/2)}{x_{2n}(t_1,h/2) - x_{4n}(t_1,h/4)} \approx 2^k \tag{3.15}$$

The calculation of C_n during the integration process enables us to monitor if the conditions for the global error estimation via eq. (3.14) are met. When we use an integration method of order k, and find the value of C_n to be 2^k, we can be certain that the equation we integrate is convergent, and that we have numerical stability. Hence global extrapolation is allowed, and the global error estimate will be reliable. When on the other hand C_n is considerable lower than 2^k over a particular integration interval, there are several possibilities:

1. The ODE, or system of ODE's is mathematically unstable and it is possible we can not solve the problem by numerical integration on this interval.

2. The ODE itself is mathematically stable, but the stepsize is chosen too large, resulting in numerical instability. Reducing the stepsize (or decreasing the local error tolerance for variable step codes) will solve the problem.

3. When the derivative function contains discontinuities, we can be sure that it does not satisfy (3.5), and the theory will simply not be valid. An example of the misleading results we obtained when trying to integrate over discontinuities will be given in the next section.

In either of these three cases, we have no other choice than to reject both the estimated error and the outcome of the integration, and investigate the reason for the lack of convergence.

[2] The relation (3.14) can also be obtained via a less strict argumentation [4], known as *Richardson's deferred approach to the limit.*

3.6 Some computational problems

The discretization error, which (theoretically) is a property of the method we use to solve an ODE, is not the only source of error in the numerical solution. The approximation of real numbers using floating-point numbers with finite word-lengths will generally lead to *roundoff error*. Roundoff error is a typical property of the computer and the program we use. An obvious requirement resulting from limited machine word-length is, that when the code tries to proceed the solution from t_n to t_{n+1}, operations must result in *different* machine numbers, or the possibility exist that the endpoint of the integration interval will never be reached. It is usually not the case that codes indicate the request of impossible accuracies [20]. A low order method is either incapable of achieving accuracies near the unit roundoff level of long word lengths, or becomes very inefficient, which means that we must select an appropriate code for the desired accuracy.

The use of pure relative error control on a solution that becomes zero can also cause serious trouble, and clearly the global error estimates calculated with eq. (3.14) becomes unreliable when we approach the limits of precision. The accuracy of floating-point arithmetic can be characterized by *machine epsilon*, the smallest floating-point number ϵ such that $1 \oplus \epsilon > 1$. A convenient algorithm to estimate the available precision for a program at execution time[3] is given in [5], which enables automatic adjustment of the lower bounds for the absolute and relative local error on different machines. In GERK the impact of roundoff error on the calculation is effectively suppressed by an appropriate scaling of the variables in the integration algorithm for each integration step, according to the value of ϵ, and the demanded relative local truncation error ϵ_l.

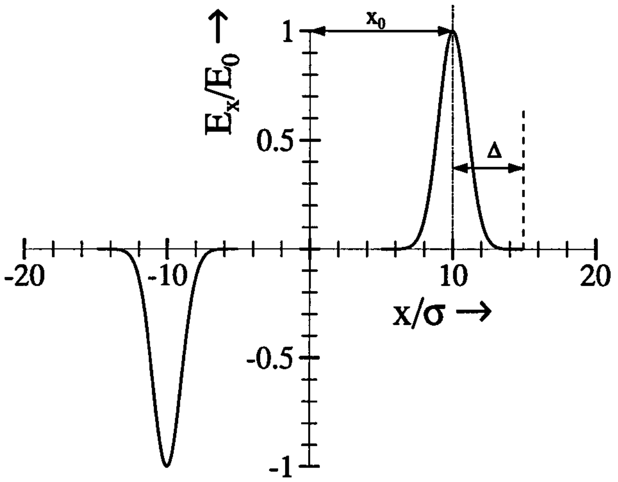
3.7 The impact of discontinuities on error estimates

To get an idea whether the global error estimate according to equation (3.14) is correct for our orbit calculations, we developed a simple test case that allows us to compare the real global error e_g with the estimated value ϵ_g.

We integrate the equations of motion for a particle with an initial kinetic energy E_{k0} in a homogeneous magnetic field B_0. The particle is alternately accelerated and decelerated by a constant electrical field, which is generated by two (somewhat artificially placed) parallel gaps. The non-relativistic equations of motion for this particle are

$$\begin{aligned}
\ddot{x} &= \tfrac{1}{2}R_0^2\tilde{E}_x(x,y) - \dot{y} \\
\ddot{y} &= \tfrac{1}{2}R_0^2\tilde{E}_y(x,y) + \dot{x} \\
R_0^2 &= 2\frac{m_0\hat{V}}{qB_0^2}
\end{aligned}$$

[3] The difference in ϵ we find when running the same FORTRAN code on different machines, with different compilers is striking. For instance on a DEC ALPHA system $\epsilon(REAL * 8) \approx 6.10^{-17}$, on a common PC, with the Salford FTN77 compiler $\epsilon(REAL * 8) \approx 5.10^{-20}$.



Figuur 3.1: The electric field from equation (3.16). In our test case we have $\sigma = 6.2$ *mm and* $x_0 = 5$ *cm. Because the direction of the electric field in the two gaps is opposite, there will be no change of the kinetic energy over one half revolution. This enables us to compare the real error in the calculated energy to the error estimation we get from applying (3.14).*

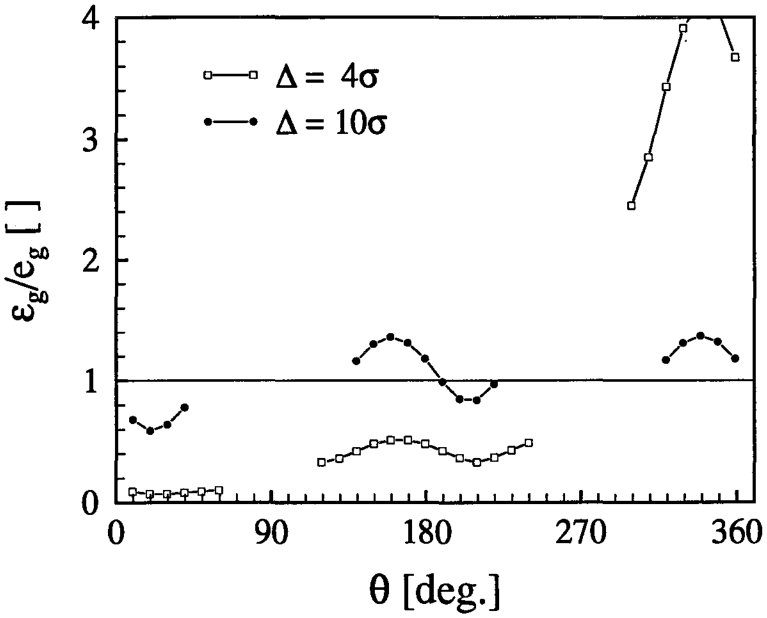
Because the electrical field is time-independent, we can start the integration at $\tau = 0$ with initial conditions

$$
\begin{aligned}
x_0 &= r_0 \\
\dot{x}_0 &= 0 \\
y_0 &= 0 \\
\dot{y}_0 &= r_0 \\
r_0 &= \frac{\sqrt{2mE_{k0}}}{qB_0}
\end{aligned}
$$

We take $B_0 = 1.4$ T and $E_{k0} = 2$ MeV, which in our case are realistic 'average' values. The particle then will start at the x-axis, at radius $r_0 \approx 14.6$ cm. For the scaled electric field $\tilde{E}_x(x,y)$ we have

$$
\begin{aligned}
\tilde{E}_x(x,y) &= E_0 \frac{|x|}{x} e^{-\frac{1}{2}\left(\frac{|x|-x_0}{\sigma}\right)^2} \text{ for } ||x| - x_0| < \Delta \qquad (3.16) \\
\tilde{E}_y(x,y) &= 0 \text{ otherwize}
\end{aligned}
$$

In figure 3.1 we show the electric field according to eq. (3.16). The electric field is limited to a region with width Δ around x_0. In our test case we took $\hat{V} = 30$ kV, $x_0 = 5$ cm and $\sigma = 6.2$ mm, which are typical values for the ILEC 2^{nd} harmonic system. Because the polarity of the (constant) accelerating voltage $\hat{V}$ is reversed according to the sign of the x coordinate, the kinetic energy of the particle will not be changed after each half revolution.

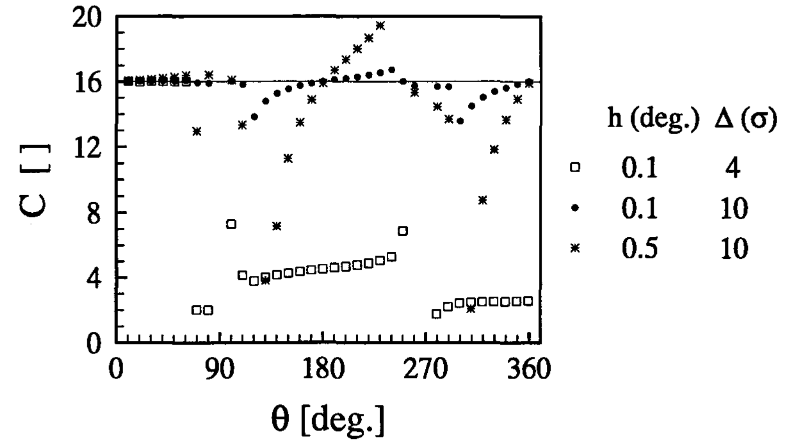


Figuur 3.2: The quotient ϵ_g/e_g along one turn of the particle orbit in the test program gives an immediate impression of the accuracy of the estimated error. The integration was performed with the code GERK, which is also used in the actual program.

This means that for $|x| > x_0 + \Delta$ the kinetic energy must be E_{k0}, so in this area we can compare the estimated global error in the energy ϵ_g, calculated according to eq. (3.14), with the real error e_g. Figure 3.2 displays the quotient ϵ_g/e_g for $\Delta = 4\sigma$ over one turn. We find no agreement between the real and the estimated error. From a physical point of view, limiting the extend of the field to $\Delta = 4\sigma$ gives at each gap crossing an error in the energy increase of

$$\begin{aligned} dE_k &= 2\int_{4\sigma}^{\infty} E_x(x-x_0)dx \\ &= 2\frac{\hat{V}}{\sqrt{2\pi}}\int_{4}^{\infty} e^{-\frac{1}{2}x^2}dx \\ &\ll 2\frac{\hat{V}}{\sqrt{2\pi}}\int_{4}^{\infty} e^{-2x}dx \\ &= \frac{\hat{V}}{\sqrt{2\pi}}e^{-8} \end{aligned}$$

For $\hat{V} = 30$ kV this comes down to an error lower than 4 eV, which is negligible. Moreover, because the reversed polarity of the gaps in our test case, the error in the energy gain per turn must be zero, so there must be another reason for the error in ϵ_g.



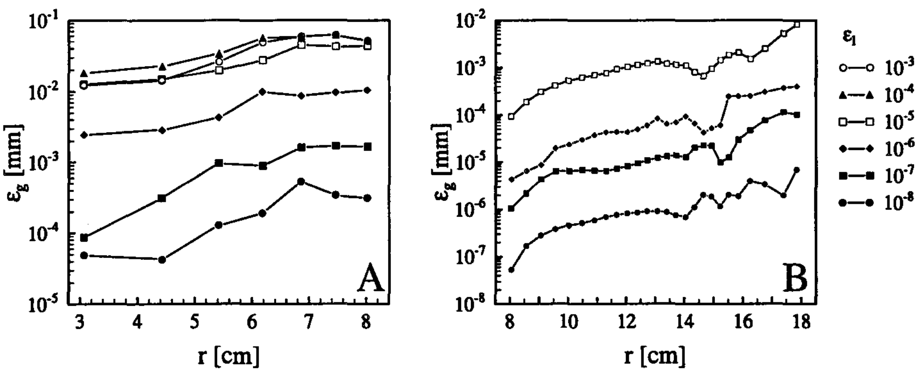
Figuur 3.3: Convergence factor over one complete turn, calculated with a fixed-step, fourth order Runge-Kutta method. When the integration passes a discontinuity in the electric field which is too big, the convergence is lost, and error estimation based on global extrapolation will be no longer valid.

In figure 3.3 we show the convergence factor C over one complete turn. We applied eq. (3.15) on a fixed-step 4^{th} order Runge-Kutta method. Clearly we can see the sudden loss of convergence which occurs at $\vartheta = 60°$, where $x = x_0 + \Delta$. The reason why the estimated error does not follow the real error is, that we introduced small discontinuities in the electric field at $||x| - x_0| = \Delta$. However small these discontinuities may be, they ruin the convergence of the integration, resulting in wrong error estimates. When we expand the field region to $\Delta = 10\sigma$ we see that the calculation remains convergent over the complete turn, and the estimated error is in good agreement with the real error.

Figure 3.3 also demonstrates the effects of numerical instability. For $\Delta = 10\sigma$ we see that if we enlarge the stepsize h from 0.1° to 0.5° the convergence in the regions where the particle is accelerated is lost. The integration code GERK, which applies local stepsize adjustment, does not exhibit this problem.

3.8 Estimation of the true error

In the previous section we saw the influence of discontinuities in the electrical field on the estimated global error. If the electrical field is not continuous, the estimated error will in general be lower than the true error. The reason for this is, that if we want to apply global extrapolation on a given ODE, solved with a method of order k, at least the derivatives of order $\leq k$ must be continuous over the considered integration interval. In other words, if we integrate a function for which the derivatives of order $\leq n$ are continuous, we must estimate the global error as if we applied an integration method of order n, even when we use a method of much higher order. This means that we must adapt our error estimate to



Figuur 3.4: The estimated global error ϵ_g along a particle orbit as a function of the radius, monitored at a fixed azimuthal position $\theta = -20°$. In figure [A] we calculated the error according to a first-order method, until the particle was well out of the central region (turn number 1–7). Figure [B] gives the global error for a particle which has been started outside the central region (turn number 7), where the method really is of fifth order. The legend at the right gives the upper limit for the relative local error ϵ_l we demanded during the calculation.

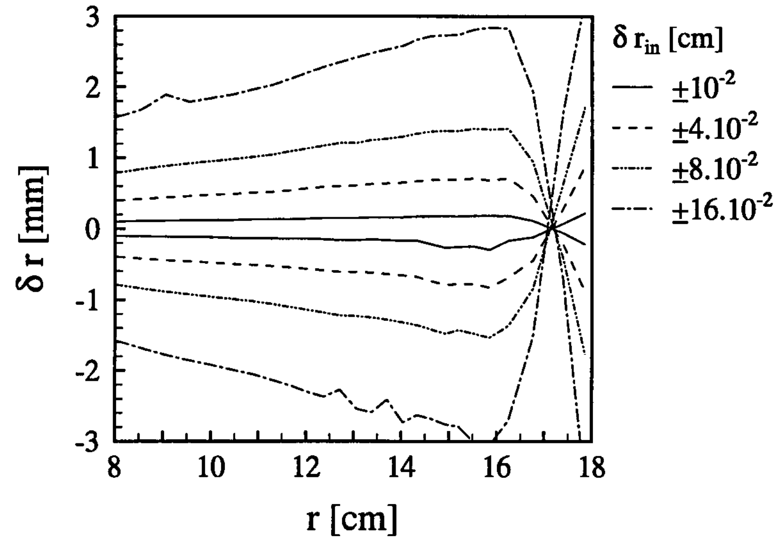
the representation of the electric and magnetic fields we use in the orbit calculations:

- The main magnetic field and the field resulting from the correction coils are constructed from (smoothed) Fourier coefficients. Because this will always result in continuous derivatives, we don't expect difficulties from the magnetic field.

- The electrical field in the central region is found by linear interpolation between mesh points in a rectangular grid. Therefore we calculate the global error in this area as if using a first-order method.

- By extending the effective gap width we use in the Hazewindus approximation of the electric field, we can estimate the global error as it is originally implemented in GERK, which uses a fifth-order Runge-Kutta scheme.

In figure 3.4 we show the growth of the estimated global error during acceleration in the central region [A], and for a particle which has been started at turn number seven [B]. We see, that if we assume the integration in the central region to collapse into first order, further acceleration of the particle outside the central region does not contribute substantially to the total error. By calculating how a small deviation from a central orbit propagates through the further solution, we can also estimate the growth of the global error which originates in the central region. Figure 3.5 shows how a small deviation δr_{in} from the central orbit at the beginning of the seventh turn develops as a function of the radius. As long as the particle does not enter the fringe field, we see that the global error behaves quite regular, and does not exceed twice it's initial value. From the calculated



Figuur 3.5: The deviation δr from the central orbit for various values of the initial deviation δr_{in}, as a function of the radius. The particle was started at the beginning of the seventh turn. We see that an error which originates at a radius of 8 cm propagates symmetrical and regular, until the particle reaches the fringe field, at a radius of approximately 16 cm.

error at a radius of 8 cm in figure 3.4 [A] we can now estimate, that if we calculate the complete orbit with a relative local error of 10^{-6}, the global error will be lower than $2 \cdot 10^{-2}$ mm, which is accurate enough to perform calculate the differential current distribution for a differential target of width 0.1 mm.

Chapter 4

Differential current calculation

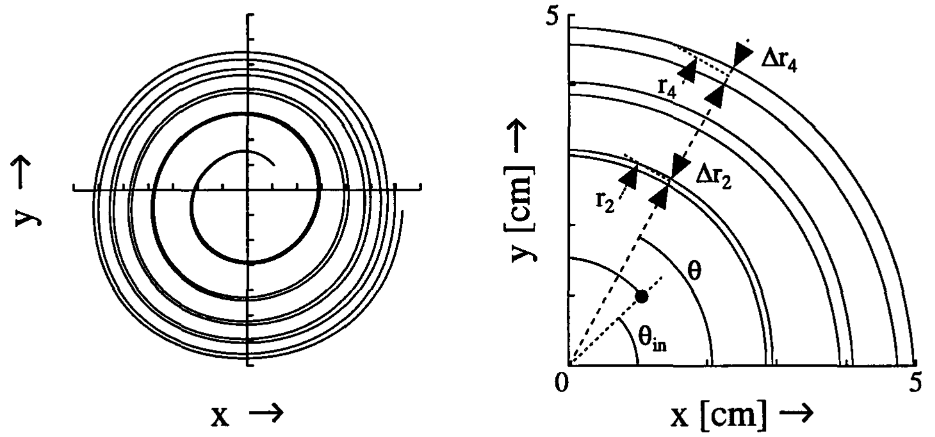
4.1 Introduction

In this chapter the calculation of the *differential current*, which follows from the numerical orbit calculations is discussed. The differential current is the radial current distribution in the median plane, as measured with a target with width Δr, moving in radial direction at a fixed azimuth. Naturally, the demand for sufficient resolution gives that the width of the target Δr must be much smaller than the beam width. The calculated differential current distribution can immediately be compared with the results we obtain from the measurements with a moving differential target. This enables us to get a good impression of the correctness of the outcomes of the orbit calculations. By calculating the differential current distribution in the extraction region, we can see if single-turn extraction is possible, and determine an upper limit for the extraction efficiency. Also an estimation of the energy spread in the extraction region, and the effect of an aperture on the beam quality at extraction radius becomes possible.

4.2 Differential current

To be able to simulate differential current measurements from the numerically calculated orbits, we first must understand what exactly we measure if a differential probe, consisting of a thin vertical wire, is moved in radial direction through the cyclotron at a fixed azimuth.

Consider a bunch of particles with charge q, departing from the ion source at initial azimuth θ_{in}, over the interval $[\tau_0, \tau_1]$. The innermost and outermost orbits over the first five revolutions are drawn in fig. 4.1. Because the particles departed over a small time interval, during acceleration their orbits tend to diverge in radial direction. If we look at a particular azimuth θ, we see that at the n^{th} revolution the bunch crosses this azimuth at a radius r_n, and has spread out over width Δr_n. All particle trajectories will only depend on the accelerating voltage at the start. Because the accelerating voltage is periodic with period $2\pi/h\omega_0$ (see eq. (2.18)), this means that all particles leaving at time intervals $2\pi/h$ will encounter exactly the same initial conditions, and thus propagate along the same orbits.



Figuur 4.1: Orbit spread over the first five turns for a bunch of particles departing from the source over the interval $[\tau_0, \tau_1]$.

Hence, if we want to calculate the time-average current we measure with a differential target with width Δr_n at position r_n, we only have to consider one period of the dee voltage, because the current will also be periodic with period p:

$$\begin{aligned} \bar{I}(r_n, \Delta r_n) &= \frac{q}{p} \\ &= \frac{h}{2\pi}\int_{\tau_0}^{\tau_1} I_s(\tau)d\tau \end{aligned} \tag{4.1}$$

Writing the source current I_s as a function of the initial phase $\varphi_{in} = h(\tau - \theta_{in})$, we find for the bunch current

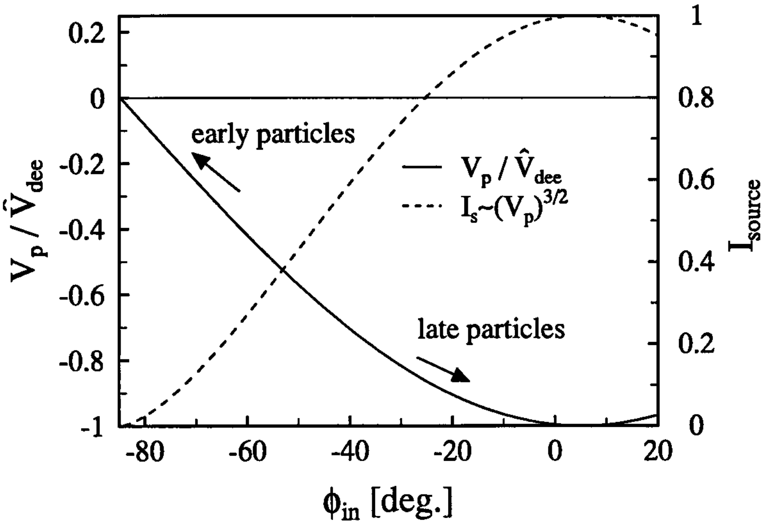
$$\bar{I}_n = \frac{1}{2\pi}\int_{\varphi_0}^{\varphi_1} I_s(\varphi_{in})d\varphi_{in} \tag{4.2}$$

For the calculations, we presume the source current to be limited by the space-charge effects which occur between ion source and puller. According to [22] the source current as a function of the puller voltage V_p is for positively charged particles:

$$I_s = A\,|V_p|^{3/2} \text{ for } V_p < 0 \tag{4.3}$$

with A a constant. When we want to write I_s as a function of φ_{in}, we have to take into account the azimuth θ_{in} at which the particles leave the source, and the definition of $V(\tau)$ in eq. (2.18). The theoretical interval for the initial phase φ_{in} follows from the demand that the puller voltage V_p must be negative in order to extract the positively charged protons from the ion source, which is at ground potential.

$$V_p(\varphi_{in}) = \hat{V}_{dee}\sin(\varphi_{in} + h\theta_{in} - h\tau_1) < 0 \tag{4.4}$$



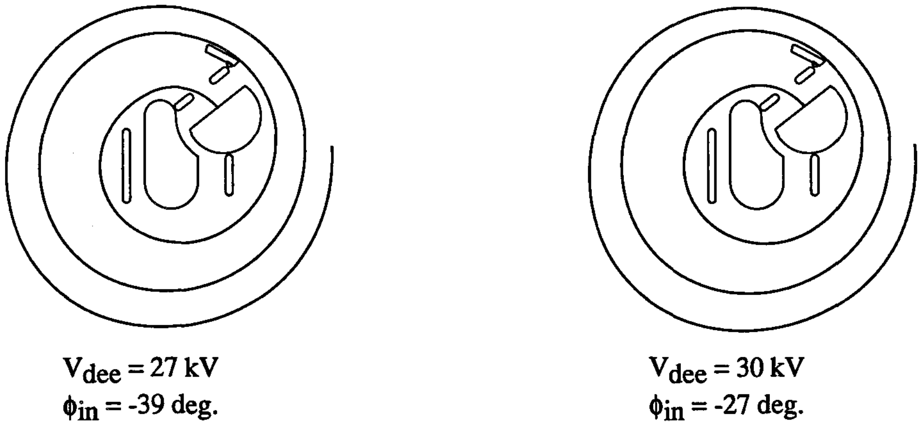
Figuur 4.2: The puller voltage V_p and the source current I_s as a function of φ_{in}, for $\theta_{in} =$ 42.4°. Early particles represent low values of the source current, whereas late particles, which yield higher current values, have lower energy gain.

The azimuth θ_{in} is the angular position of the hole through which the particles are extracted from the ion source, and is 42.4 °. For harmonic number $h = 2$, and $\tau_1 = 90°$, this gives a theoretical initial phase interval $-84.8° < \varphi_{in} < 95.2°$. In figure 4.2 we show the puller voltage V_p and the source current I_s as a function of φ_{in}.

The complete radial distribution of the internal current is determined by extending the interval $[\varphi_0, \varphi_1]$ to the full half-period in which the source emits particles, and calculate the particle orbits for every initial phase. In order to find the stationary distribution over the full radius of the cyclotron, every particle orbit should be carried through until the particle is either lost in the fringe field, or intercepted by an obstacle. The differential current at a target with width Δr at radius r is then found to be the summation of the currents from every initial phase interval for which particles impinge on the target:

$$I_{diff} = \frac{1}{2\pi} \sum_n \int\limits_{\varphi_{in}(r-\frac{1}{2}\Delta r)}^{\varphi_{in}(r+\frac{1}{2}\Delta r)} I_s(\varphi_{in}) d\varphi_{in} \tag{4.5}$$

Because the particles with high initial phase represent a relatively large part of the internal beam current, the width of the total initial phase region we consider is an important parameter. In the next section we shall discuss how we estimate the boundaries for the initial phase region.



Figuur 4.3: If φ_{in} is too high (the particle starts 'too late'), particles will be intercepted by the right side of the puller, and do not contribute to the internal beam current. The resulting initial phase interval depends on the dee voltage.

4.3 Estimation of the dee voltage

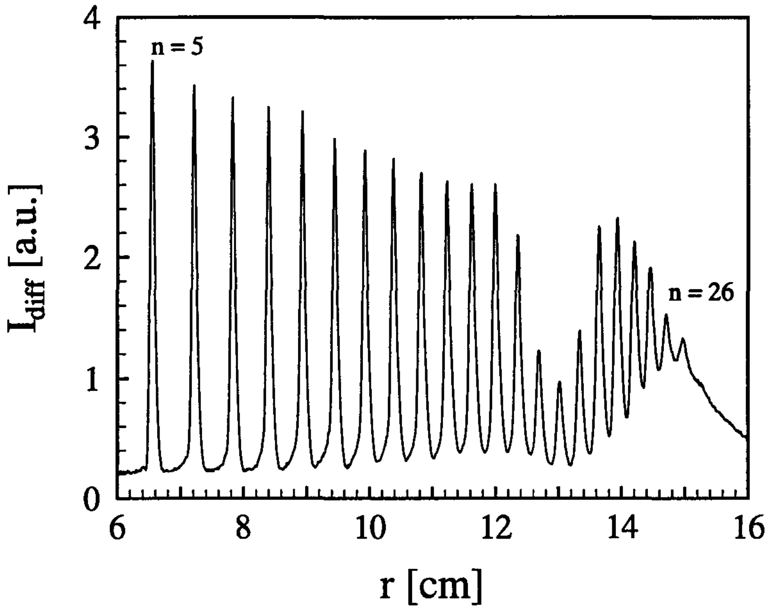
If we want the calculation of the differential beam current to be realistic, the dee voltage $\hat{V}_{dee}$ has to be estimated from measurements. The exact dee voltage is an important parameter for the numerical calculations, not only because it affects orbit separation and centre motion, but also the phase-acceptance of the center geometry.

Particles with different initial phase φ_{in} will traverse the central region along different trajectories. If φ_{in} is too high, the acceleration between source and puller will be poor, and chances arise that the particles will be intercepted by obstacles in the central region during the first or second revolution. This implies that the initial phase interval for particles that can contribute to the internal beam current has a fixed upper bound, which varies with the dee voltage. From figure 4.3 it shows for instance, that when $\hat{V}_{dee} = 30$ kV only particles with $\varphi_{in} < -33°$ should be considered. Figure 4.4 depicts a typical measurement of differential beam current versus radius in ILEC. Every peak corresponds with one turn number of the internal beam; the most left-hand peak corresponds with turn number 5. From this measurement we will try to estimate a typical value for $\hat{V}_{dee}$.

A particle which is rotating at radius r about the origin in a magnetic field with cylindrical symmetry has kinetic energy

$$E(r) = \frac{q^2 r^2 B^2(r)}{2m_0} \tag{4.6}$$

If we determine the radius $\bar{r}(n)$ at which the maximum differential current in each turn number n occurs, we can expect this radius to correspond with a particle which has an



Figuur 4.4: Internal beam current measured with a moving differential probe of 0.1 mm. The dip between 12 and 14 cm is due to vertical defocussing of the beam. After turn 26 the beam was intercepted by a fixed target.

average energy for that turn:

$$\bar{E}(n) = \frac{q^2\bar{r}^2(n)\bar{B}^2(\bar{r}(n))}{2m_0} \tag{4.7}$$

with $\bar{B}^2(\bar{r}(n))$ the average magnetic induction at $\bar{r}(n)$. Because we do not aim at a precision better than 1 %, the relativistic mass-increase of the particles can be neglected for our purpose.

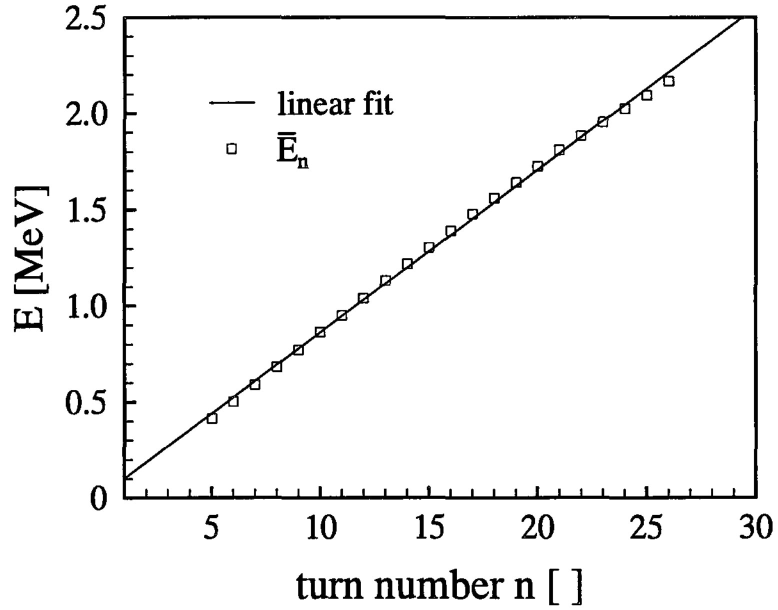
In applying (4.7), we presume that beam shift resulting from drift of the orbit centre is negligible. If the kinetic energy of a particle is high enough, we can also neglect the variation of the accelerating voltage during the gap crossing, and relate the energy increase of the particle to the amplitude of the accelerating voltage at the moment the particle arrives at the gap.

$$\Delta E_g = qV(\tau_g) \tag{4.8}$$

with τ_g the moment the particle arrives at the gap. If we define the gap number g as depicted in fig. 2.1, we find for the energy increase of protons in eV

$$\Delta E_g = \hat{V}_{dee}\sin[2\alpha + (-1)^g\varphi_g] \tag{4.9}$$

where α denotes the half dee-angle (which is 25° for the ILEC second harmonic dees), and φ_g has the value of φ_{hf} at the moment of gapcrossing. For a complete turn, in which four



Figuur 4.5: Average kinetic energy belonging to the maximum differential current for each turn n. With a linear fit we determine the average energy increase per turn.

gapcrossings occur, we can write for the energy increase in eV

$$\Delta E_n = \sum_{g=1}^{4} \hat{V}_{dee} \sin[2\alpha + (-1)^g \varphi_g] \tag{4.10}$$

which leads to

$$\Delta E_n = 4\hat{V}_{dee} \sin 2\alpha \cos \varphi_g \tag{4.11}$$

for φ_g constant at all four gapcrossings.

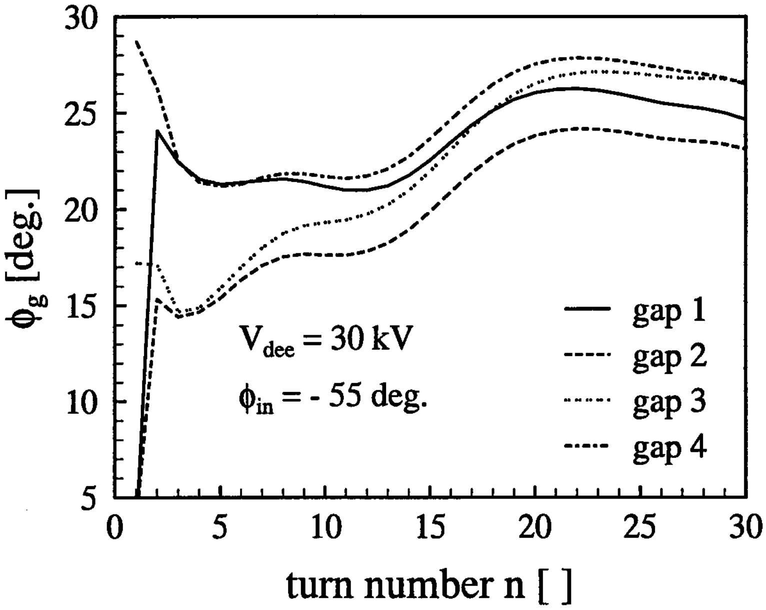
From the slope of the linear fit of $\bar{E}(n)$ to the turn number in fig. 4.5 we can derive the average energy increase per turn for turn 5 to turn 26

$$\overline{\Delta E}(n = 5 \rightarrow 26) = 84.46 \text{ keV} \tag{4.12}$$

If we assume $\varphi_g = 0$ we find for the dee voltage

$$\hat{V}_{dee} = \frac{\overline{\Delta E}(n = 5 \rightarrow 26)}{4 sin 2\alpha} = 27.56 \text{ kV} \tag{4.13}$$

However, from fig. 4.6, which shows the course of the HF-phase for an accelerated particle at each gap crossing, it is clear that we cannot expect φ_g to be constant for consecutive turns. So the value for $\hat{V}_{dee}$ in (4.13), is likely to be an underestimation of the real dee voltage.

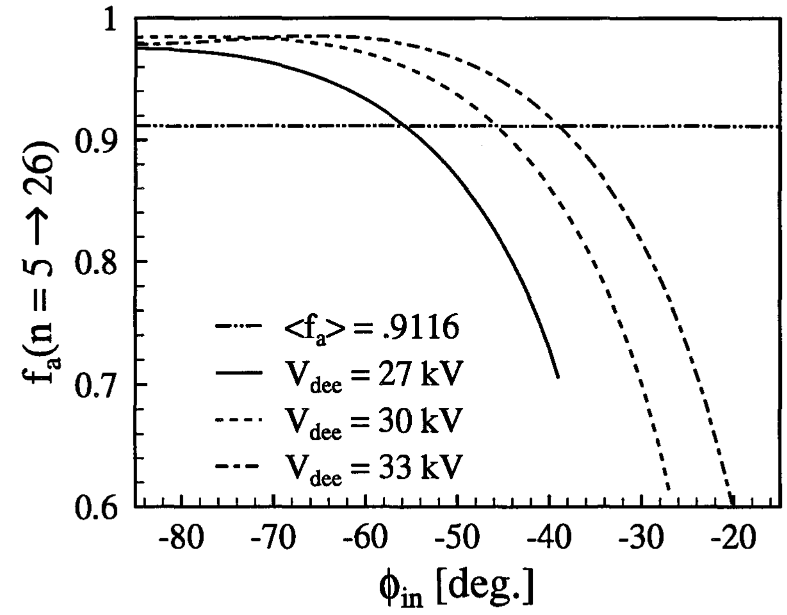


Figuur 4.6: Behaviour of the HF-phase at the four 2^{nd} harmonic gaps. Clearly we observe the phase shift during the first turns, and the slow coherent oscillation resulting from the centre motion.

To get a more realistic estimate for $\hat{V}_{dee}$, we may introduce an efficiency factor f_a for the acceleration of a particle. The factor f_a gives the real energy increase over a number of turns in proportion to the maximum achievable energy increase

$$f_a(\varphi_{in}, n = n_1 \rightarrow n_2) = \frac{E(n_2) - E(n_1)}{(n_2 - n_1)} \frac{1}{4\hat{V}_{dee} \sin \alpha} \tag{4.14}$$

For a given magnetic field $f_a(\varphi_{in}, n = n_1 \rightarrow n_2)$ will depend on the initial phase of the particle and the considered turn numbers. Figure 4.7 presents the results for $f_a(\varphi_{in}, n = 5 \rightarrow 26)$ for three different values of $\hat{V}_{dee}$. The kinetic energy at turns 5 and 26 are found from numerical orbit calculations. If we combine $\bar{f}_a$ – averaged over φ_{in} and $\hat{V}_{dee}$ – with the measured average energy increase (4.12), we find $\hat{V}_{dee}$ = 30 kV, which seems to be a realistic value. The numerical calculation of the differential current with $\hat{V}_{dee}$ = 30 kV in fig. 4.8 is in good agreement with the measurement for turns 5 – 17. At higher turn numbers we see a gradually increasing deviation in the position of the current maxima. This is a result of centre drift of the measured beam due to a first-harmonic component in the magnetic field, which was not present in the numerical calculation.

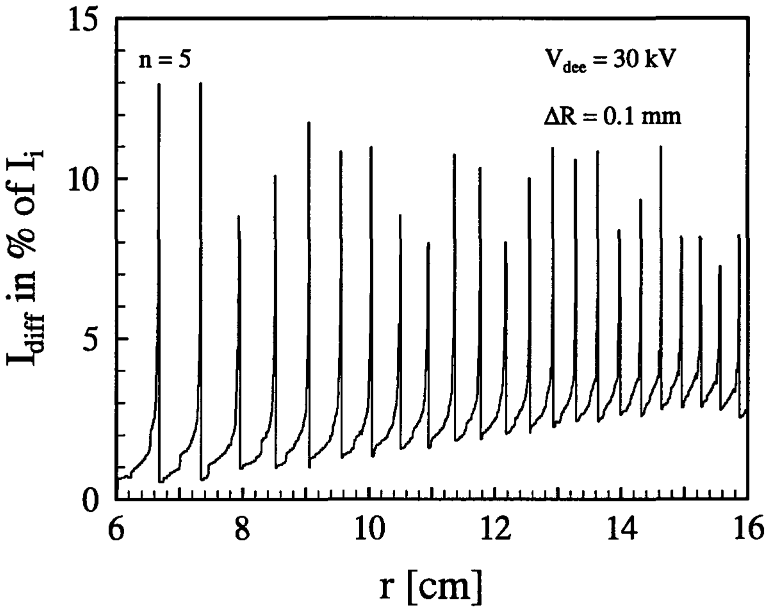


Figuur 4.7: The accelerating factor $f_a(\varphi_{in}, n = 5 \rightarrow 26)$ for $\hat{V}_{dee}$ = 27, 30 and 33 kV.

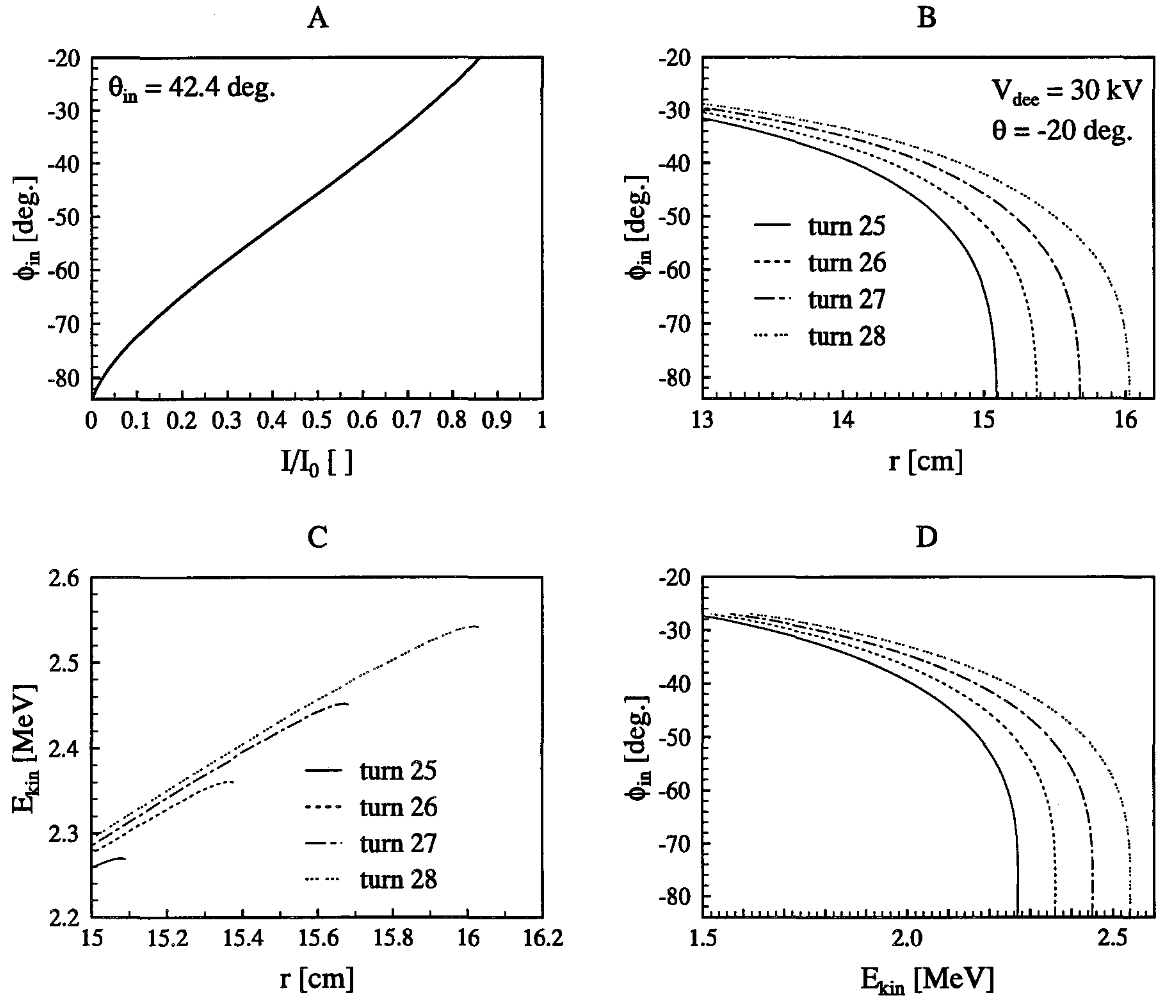
4.4 The program IDIF

From numerical orbit calculations we can obtain the radius and energy of a particle at a certain azimuth as a function of the initial phase. In figure 4.9 [B] and [D], we show the radius and energy of a particle as a function of the initial phase for turns 25 to 28. For the azimuth we took the azimuthal position θ_{ex} of the extractor entrance, which is $-20°$ (see figure 2.3). The step in the initial phase $\Delta\varphi_{in}$ is 1°, which means that for each integer number of φ_{in} between $-27°$ and $-84°$ we calculated the complete orbit to find the radius and energy each time the particle crosses the azimuth $\theta_{ex} = -20°$. On the basis of figure 4.9 we shall describe the algorithm we use in the program IDIF to calculate the differential current distribution and the energy-current distribution.

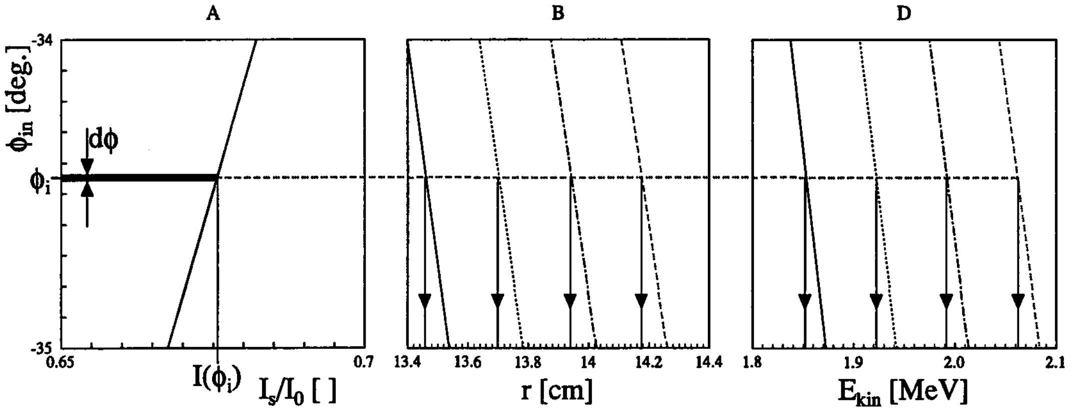
Suppose we want to calculate the differential current distribution for turn numbers 25 to 28, for a differential target with a width Δr. First, we split up the radial interval over which the particles cross θ_{ex} in intervals with length Δr. Then we scan each separate phase interval $\Delta\varphi_{in}$ with a small step $d\varphi$. For each initial phase value φ_i we find the radius for the considered turns by linear interpolation between two calculated values. In this way we determine to which interval Δr the source current $\bar{I}_s(\varphi_i)d\varphi$ will contribute. The integration step $d\varphi$ in a phase interval is limited by two demands: for each turn the radial step $dr(d\varphi)$ may not exceed a preset radial resolution δr, with $\delta r \ll \Delta r$, and $d\varphi$



Figuur 4.8: Calculated differential beam current at the azimuth of the differential probe in the fourfold-symmetrical field. In this case the RF frequency f_0 was 43.34 MHz; the frequency for which the measured magnetic field is closest to the isochronous field. Later on we shall see that a small deviation from f_0 can increase the turn separation at higher radii, and thus reduce the 'background' in the differential current.



Figuur 4.9: The radius [B] and energy [D] for turns 25 to 28 at the azimuth of the extractor entrance as a function of the initial phase. Combination of the data in [B] and [D] yields the energy as a function of the radius [C]. Subfig. [A] depicts the source current calculated according to equations (4.3) and (4.4).



Figuur 4.10: Calculation of the differential current distribution and the energy-current distribution from the data in figure 4.9 [A], [B] and [D]. Each initial phase interval between two values for which we calculated the radius and energy as a function of the turn number is evaluated with a small step $d\varphi$. The radius and energy for each turn, corresponding with current $I(\varphi_i)d\varphi$, is found by linear interpolation.

must be small enough to obtain an accurate integration[1] of the source current.

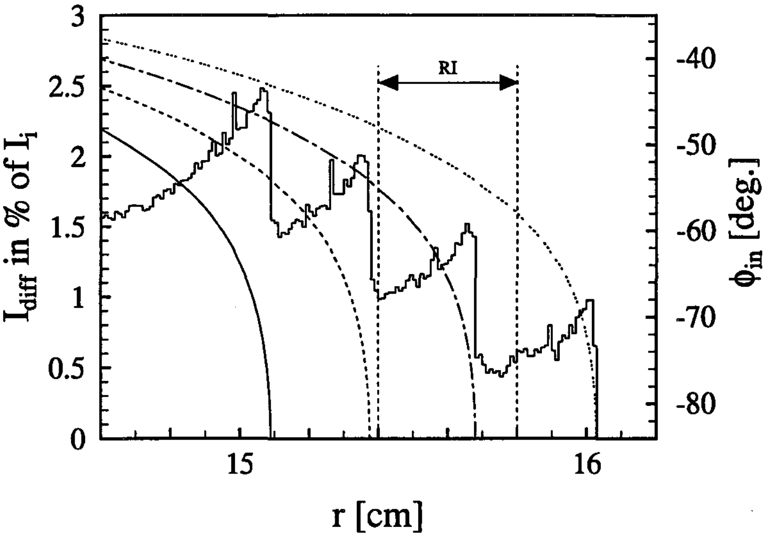
Figure 4.10 gives an impression of the construction of the differential current for one step in the initial phase from −35° to −34°. The result of the calculation is a staircase function of the radial current distribution, as depicted in figure 4.11. Each bar represents the differential current over a radial interval Δr, scaled to the total internal current I_i.

A simultaneous evaluation of the particle energy as a function of the initial phase (fig. 4.10 D) gives the possibility to calculate the energy-current distribution in a preset radial interval (RI). This RI can be the extractor entrance, or a particular aperture in the cyclotron. In this case we must only take into account the lowest turn for which a phase value φ_i contributes to the current in the RI, to avoid double-counting of the current. Now the integration step $d\varphi$ in the calculation is also limited by the demand that for each turn the energy step $dE(d\varphi)$ may not exceed a preset resolution δE. Figure 4.12 shows the energy-current distribution for a RI from 15.4 to 15.8 cm. This is a possible position for the extractor entrance, set for extraction of turn 27. By keeping track of the total current that enters the RI, we find that only 35 % of the total internal current I_i enters the extractor. This illustrates the remark in the preceding section: because we

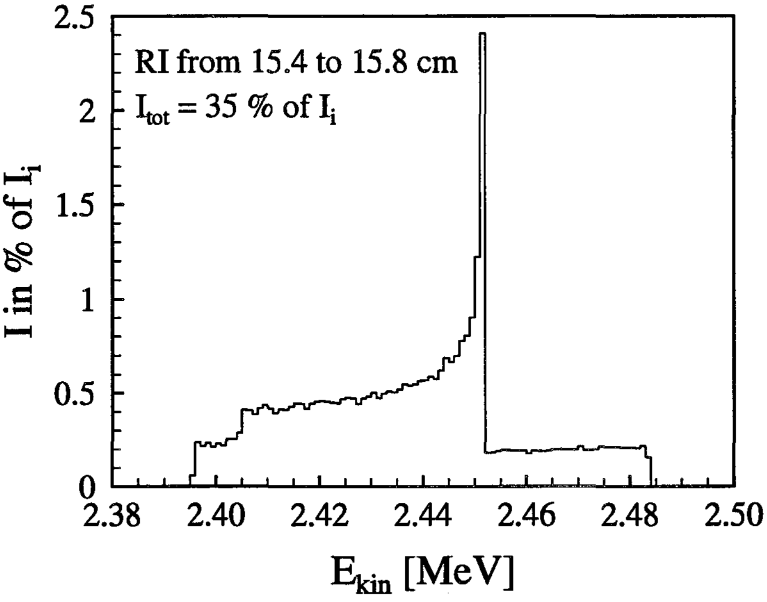
[1] Because we have the analytical expression

$$\int_0^{\pi/2} \sin^\nu \varphi d\varphi = \frac{\sqrt{\pi}}{2} \frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu}{2}+1)}$$

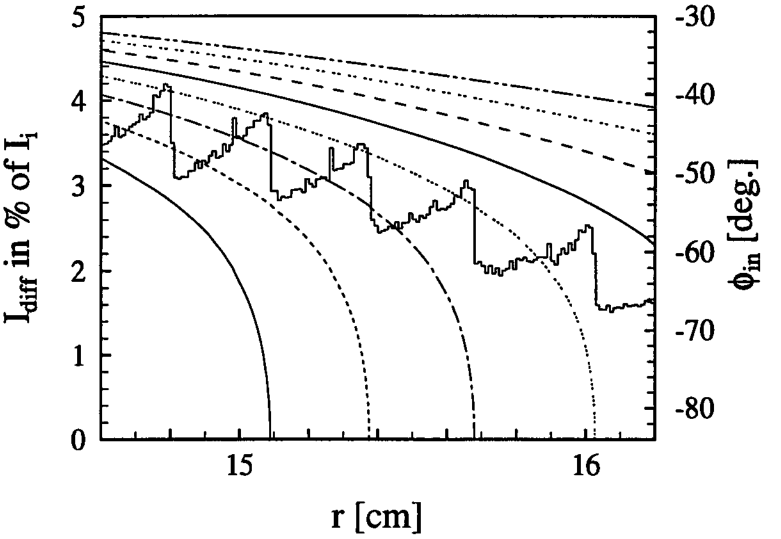
we can check that for integration with step $d\varphi = 1°$ the error is already lower than 10^{-5} %, which certainly satisfies our demand.



Figuur 4.11: The differential current calculated for $\Delta r = 0.1$ *mm with radial resolution* $\delta r = 0.01$ *mm. Here we only considered contributions up to turn 28. This means that particles with* $\varphi_{in} > -50°$ *are left out, because they need more revolutions to reach the RI.*



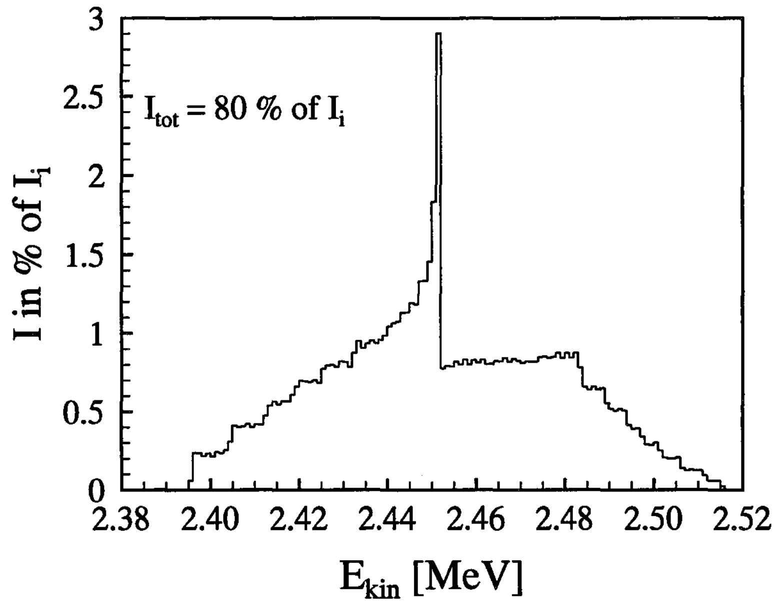
Figuur 4.12: The energy-current distribution over the radial interval RI in 4.11. Here we had $\Delta E = 1$ *keV,* $\delta E = 0.1$ *keV and* $\delta r = 0.05$ *mm. If we only take into account turns 25 to 28, we find that only 35 % of the internal current enters the extractor.*



Figuur 4.13: The calculated differential current, calculated from contributions of turns 25 to 35.

only considered turns 25 to 28, we did not consider the stationary current distribution in the RI. In figure 4.11 we can see that in this case only the particles with initial phase $-84° < \varphi_{in} < -50°$ contribute to the extracted current. In order to find contributions from particles with $\varphi_{in} > -50°$, and thus achieve a more realistic current distribution at the RI, we also should calculate higher turn numbers. Figure 4.13 depicts the calculated differential current under the same conditions as in figure 4.11, where we calculated the particle orbits over 35 revolutions. We see that now also particles with $-40° < \varphi_{in} < -50°$ contribute to the calculated differential current. Because the source current becomes higher with increasing initial phase, the differential current shows a substantial increase. The energy current-distribution in figure 4.14 shows that we find the total current that enters the extractor to be 80 % of the internal current. Because we know that eventually all particles will enter the extractor, this indicates that we should calculate the particle orbits for more than 35 revolutions, if we want to find the true differential current distribution at extraction radius. Later on we will see that this is not necessary if we want to achieve single-turn extraction, because then we already have to use an aperture to limit the initial phase interval.

Up to this point, we kept the RF frequency f_0 for which we calculated the particle orbits in the measured magnetic field fixed at $f_0 = 43.34$ MHz. From figure 2.8 this seems to be an appropriate choice to ensure isochronous acceleration. On the other hand, from figures 4.13 and 4.14 we get the impression that if we could alter the acceleration process such that the energy gain of particles from the low initial phase area increases, we could improve the turn separation in the extraction region. In the next section we shall see that this is indeed the case.

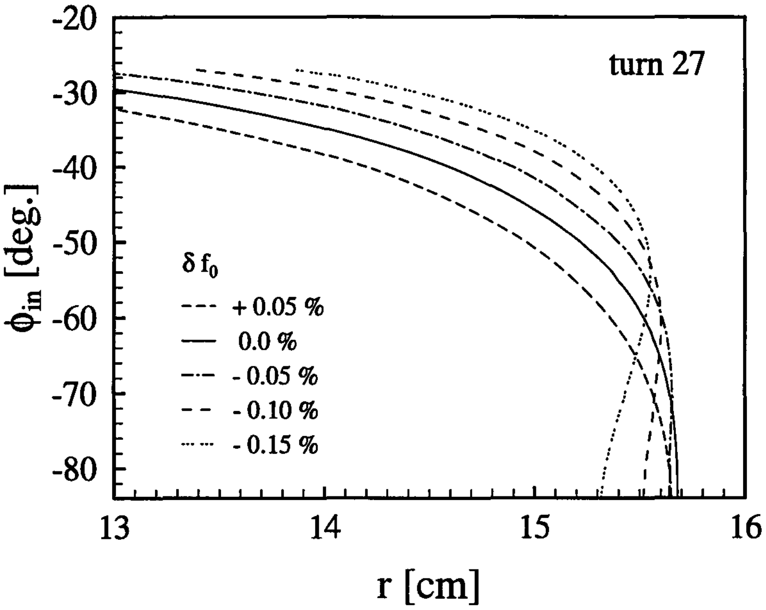


Figuur 4.14: The energy-current distribution for the conditions in figure 4.12. Here we took into account the first 35 revolutions of each particle orbit. We find that 80 % of the internal current enters the extractor entrance. The energy-spread in the RI is approximately 4 %.

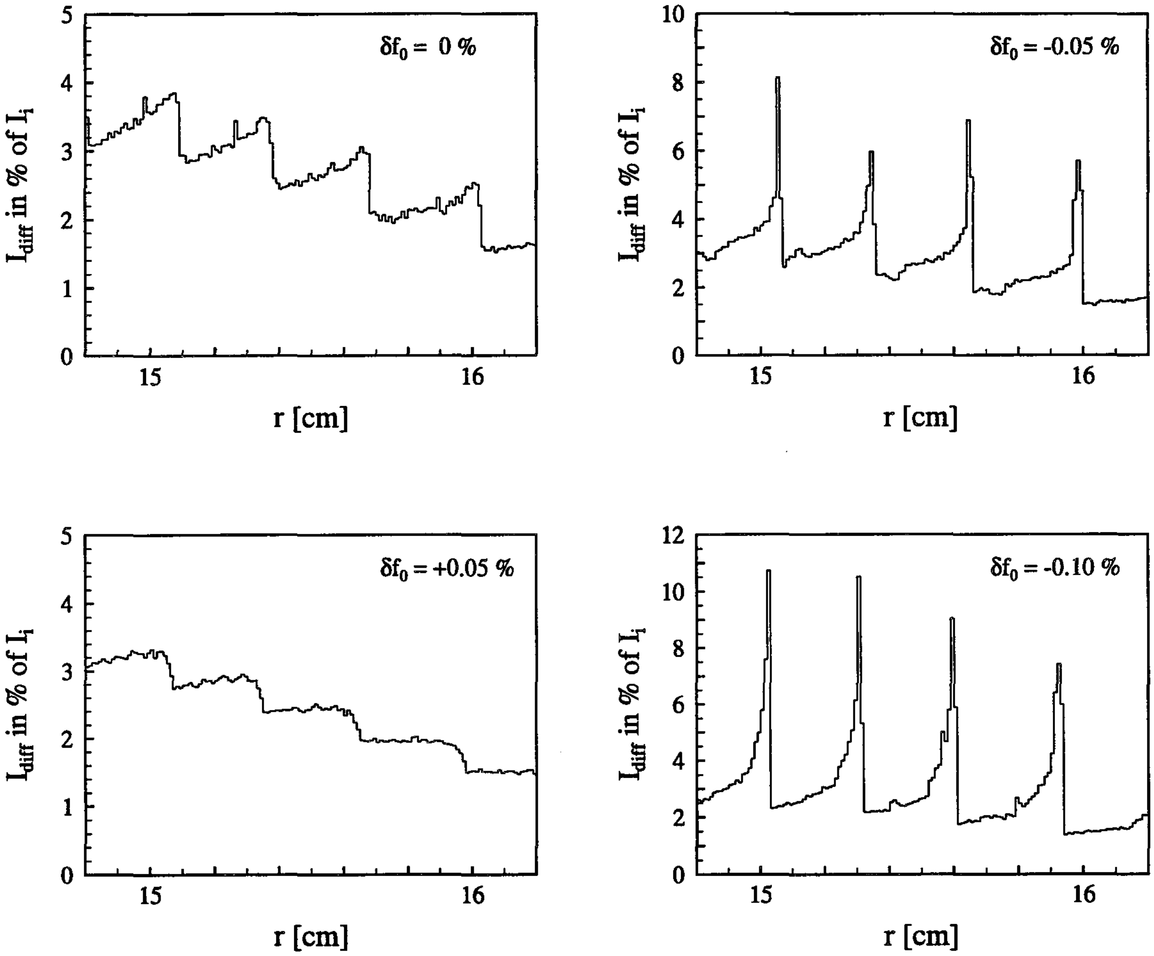
4.5 Tuning of the average magnetic field

In cyclotron operation, an essential part of the optimization procedure consists of the fine-tuning of the main magnetic field. By altering the main field we try to obtain an internal beam with the maximum achievable quality. Because the isochronous field is inversely proportional to the revolution frequency, we can easily reproduce this procedure by changing the RF frequency for which we calculate the particle orbits in the measured magnetic field.

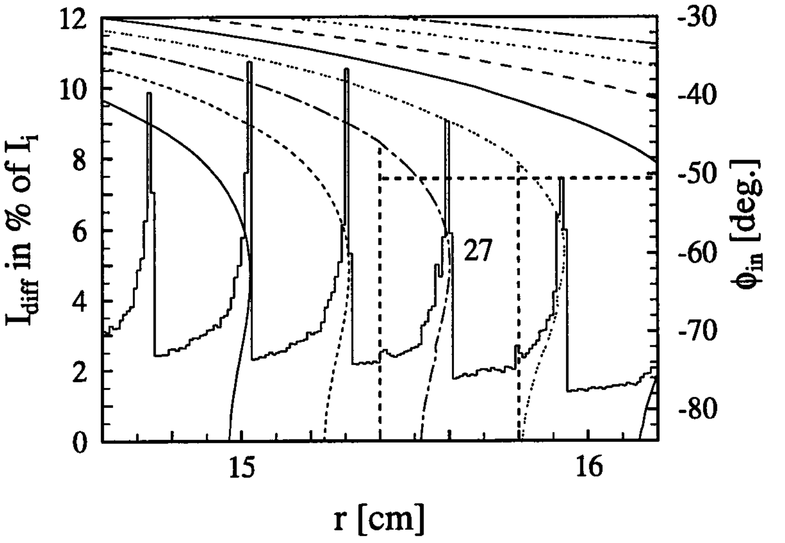
The effect of a slight variation in f_0 on the radius as a function of the initial phase at extraction radius becomes clear in figure 4.15. Here we show the radius as a function of φ_{in} for turn 27 (see figure 4.9). We varied the RF frequency f_0 with steps of 0.05 %. We see that by slightly lowering f_0, we can diminish the total radial spread of the phase area, and reduce the overlap between successive turns. We also see that the particles that reach the maximum radius originate from a lower initial phase region, and thus represent higher currents. This is confirmed by the trend we see in figure 4.16. Because the maximum in the radius also becomes more pronounced if it is shifted towards lower values of φ_{in}, there should be an optimal situation, for which the central peak in the energy-current distribution contains the largest fraction of the total internal current. We must also keep in mind that if the deviation from the isochronous field gets too large, the overall acceleration will be poor, resulting in lower beam separation at extraction radius. In figures 4.17 and 4.18 we



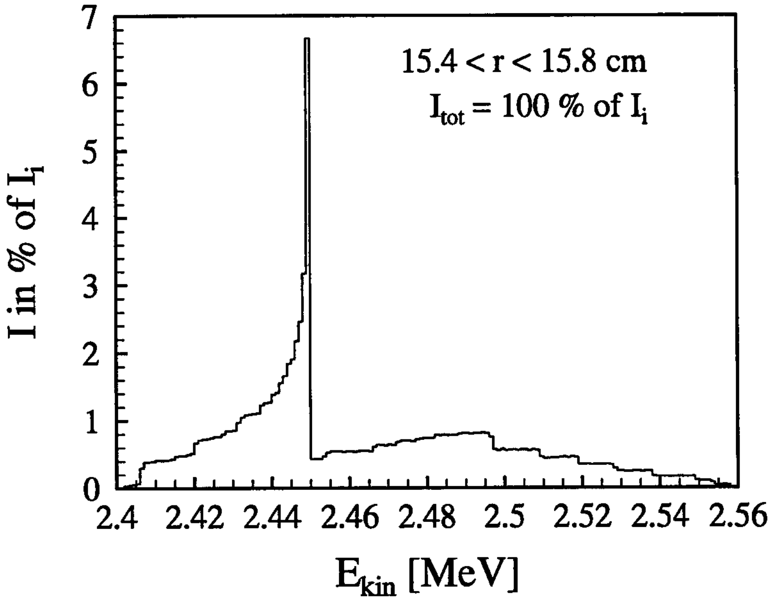
Figuur 4.15: The orbit radius for turn number 27, at the azimuth of the extractor entrance under variation of the RF frequency f_0. Lowering f_0 has the same effect as a proportional increase of the average magnetic field. We can see that a small decrease in f_0 has a remarkable effect on the differential current distribution, because the maximum in the radius shifts towards lower values of φ_{in}, which represent larger currents.



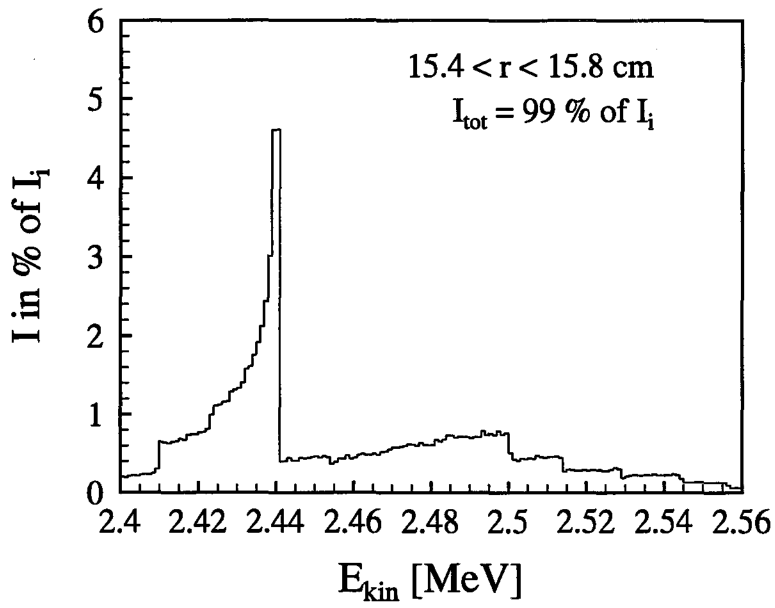
Figuur 4.16: The differential current at the azimuth of the extractor entrance under variation of the RF frequency f_0.



Figuur 4.17: The differential current distribution from figure 4.13 shows a remarkable improvement if we lower f_0 by 0.1 %. The dotted lines show the position of the extractor entrance, set for extraction of turn number 27.



Figuur 4.18: The energy-current distribution at the extractor entrance for $\delta f_0 = -0.1$ %.



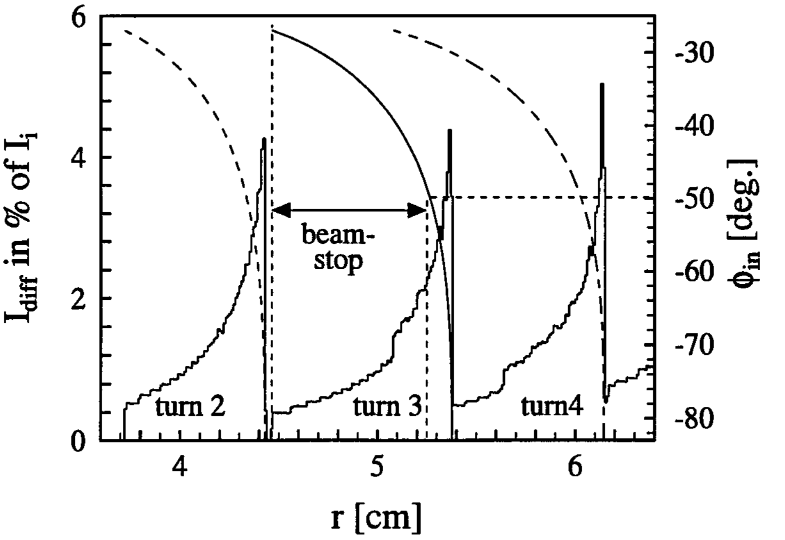
Figuur 4.19: The energy-current distribution at the extractor entrance for $\delta f_0 = -0.15$ %.

show the differential current and the energy-current distribution, calculated for the same conditions as in figures 4.13 and 4.14, with $\delta f_0 = -0.1$ %. Now the turn separation has remarkably improved, leading to a much higher and sharper bounded peak in the energy-current distribution. Because now 100 % of the internal current enters the extractor, we know that for this case the radial current distribution is stationary after 35 turns. Comparison with the energy-current distribution in figure 4.19, calculated for $\delta f_0 = -0.15$ %, shows that this situation is also quite close to an optimum: further decreasing the frequency results in a less ideal energy-current distribution.

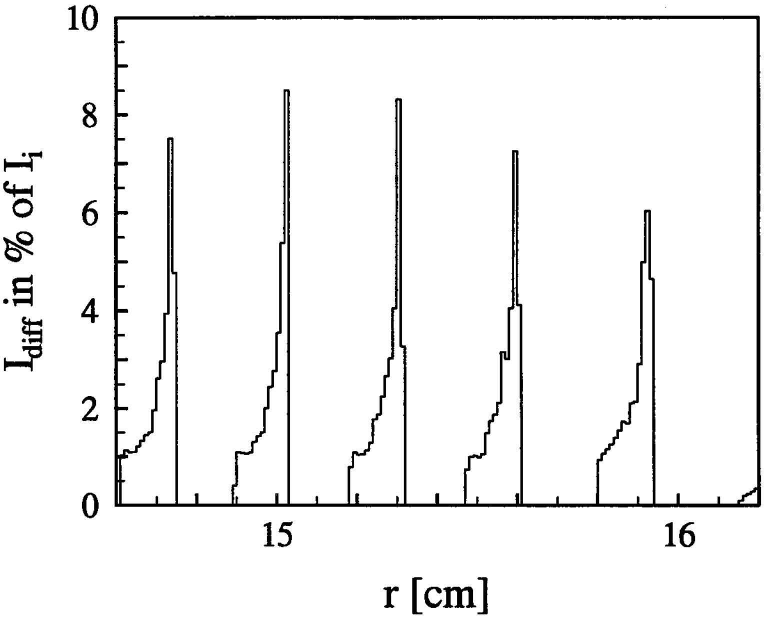
4.6 single-turn extraction

Because the width of the extractor entrance is 4 mm, and the energy discrimination of the electrostatic channel is quite low, the energy spread in the extracted beam remains quite high. From figure 4.17 it becomes clear, that if the phase width of the internal beam could be cut off at $\varphi_{in} = -50°$, we certainly have single-turn extraction, while still extracting a substantial portion of the internal beam current. Otherwise, the use of only the second harmonic acceleration system will always lead to multi-turn extraction.

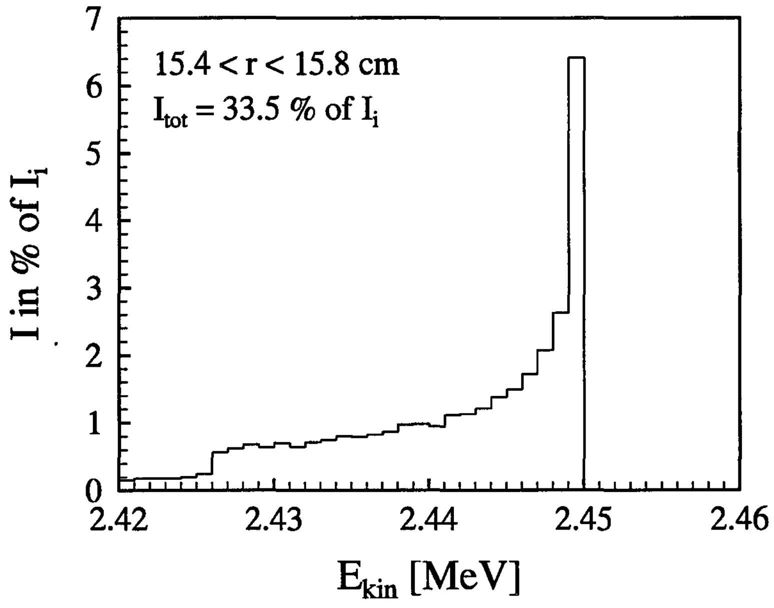
In ILEC a movable internal aperture has been mounted on one of the dummy-dee lists, which lies at an azimuth of 25°. Because the program IDIF can simultaneously evaluate the differential current at two different azimuthal positions, We can try to find an aperture position which allows single-turn extraction of turn number 27, and see how this alters the energy-current distribution of the beam at the entrance of the extractor. From the



Figuur 4.20: The differential current and the radius as a function of the initial phase for turns 2 to 4 at the azimuth of the internal aperture, for $\delta f_0 = -0.1$ %. The dotted lines indicate the beam stop we use to cut off the initial phase area above $-50°$, to achieve single-turn extraction.



Figuur 4.21: The differential current at extraction radius, for a beam stop from 4.45 to 5.25 cm. Now the turn separation is sufficient for single-turn extraction.



Figuur 4.22: The energy-current distribution at the extractor entrance, for a beam stop from 4.45 to 5.25 cm. The energy spread has been lowered to approximately 1 %.

differential current distribution at the azimuth of the aperture in figure 4.20 we can see, that if we for instance place a single beam stop in turn 3, and block the internal beam for $r = 4.45$ to 5.25 cm, we exactly cut off the initial phase area at $\varphi_{in} = -50°$. The extractor placement from figure 4.11 now leads to the energy-current distribution in figure 4.22. Although now only about 33 % of the internal current is left, the energy spread in the beam at the extractor entrance has been reduced to about 1 %. Widening the beam stop leads to further reduction of the energy spread in the extracted beam, but will also substantially decrease the extracted current, so the dimensions of the aperture to be used will depend on the required beam quality.

4.7 Involving the 6^{th} harmonic system

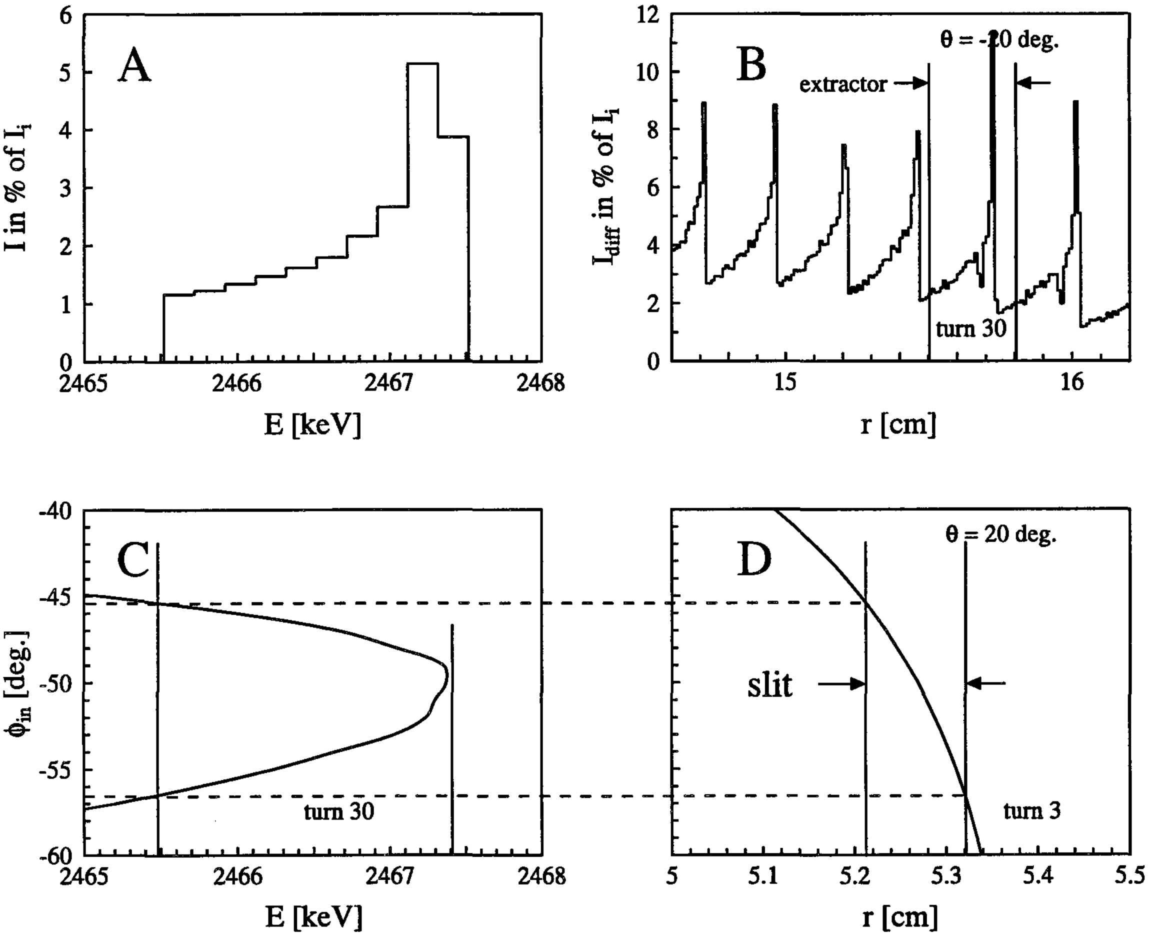
To obtain a beam with both small energy spread and high current, ILEC has been equipped with two small flattop dees, oscillating at three times the RF-frequency. The theoretical background of the flattopping-principle, as applied to ILEC, is given in [11].

The main parameters that determine the beam quality at extraction radius under 6^{th} harmonic operation are

- The main magnetic field, which in the numerical calculations can be fine-tuned by altering the frequency f_0 of the RF signal.
- The amplitude of the 6^{th} harmonic accelerating voltage, relative to the amplitude of the 2^{nd} harmonic voltage, $\hat{V}_{3/1}$.
- An additional phase shift δ_3 of the 6^{th} harmonic accelerating voltage with regard to the 2^{nd} harmonic voltage.

By varying these three parameters for a fixed dee voltage $\hat{V}_{dee,1}$ of 30 kV, we tried to maximize the extracted current, demanding an energy-spread of less than 0.1 %. In figure 4.23 [A] we see that we can extract more than 20 % of the internal beam current, and still keep the energy spread beneath 0.1 %. In this situation we had for the above parameters $\delta f_0 = -0.15$ % (with $f_0 = 43.34$ MHz), $\hat{V}_{3/1} = 0.1$, and $\delta_3 = 1.0°$. To limit the energy-spread we had to place an aperture in the third turn, restricting the initial phase area of the internal beam. On the basis of figures 4.23 [B], [C] and [D] we will discuss the positioning of the extractor and aperture with which we obtain the energy-current distribution [A].

Figure 4.23 [B] shows the calculated differential beam current at extraction radius for the azimuthal position of the extractor entrance $\theta_e = -20°$. The vertical lines indicatė an extractor entrance of 3 mm, positioned for extraction of turn number 30. Examination of the particle energy as a function of the initial phase for the 30^{th} revolution in figure 4.23 [C] indicates the desired initial phase area. In order to limit the energy-spread to about 0.1 %, we must select the initial phase region $-56° < \varphi_{in} < -46°$. Finally, in figure 4.23 [D] we show the radius as a function of the initial phase at the azimuthal position of the internal aperture $\theta_a = 20°$, for the third revolution. To select the desired phase area, we need a slit that only lets through particles with $5.22 < r < 5.32$ cm. Evaluating the internal beam for these settings with the program IDIF results in the energy-current distribution [A].



Figuur 4.23: The optimal energy-current distribution [A] for extraction of turn 30 [B], achieved with the flattop system. Figures [C] and [D] indicate how the desired initial phase region can be selected with the use of an internal aperture.

Chapter 5

The motion of the orbit centre

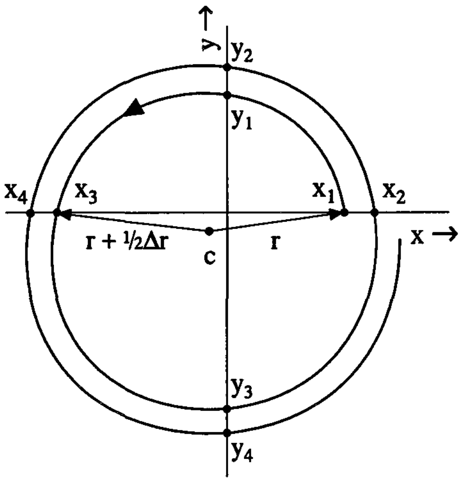
5.1 Introduction

In this chapter we discuss the results of the calculated motion of the orbit centre in ILEC under various conditions. We will show that the average centre position can be easily and accurately derived from the numerical orbit calculations. On the other hand, more insight in the separate influence of various parameters on the particle motion can be gained from the Hamilton treatment than from numerical calculations alone. Especially the movement of the orbit centre during acceleration can be attributed clearly to separate terms in the Hamilton functions derived in [18]. Therefore we will use some of the analytic expressions to describe and verify the trends we find in the numerical results. Because we will only use the Hamilton treatment to give a more qualitative interpretation of the numerical results, we shall merely state the results of the theory, and not be concerned with the exact theoretical background. We think this is justified by the good agreement between numerical and analytical calculations in [18, 11].

5.2 Calculation of the centre position

For the calculation of the position of the orbit centre, we use the method described in [18], which is illustrated in figure 5.1. From numerical calculations we can determine the radial position of a particle when it crosses the x or y axis. If we do this for two successive turns, we can immediately eliminate the turn separation Δr resulting from the energy gain, and thus find the average value $(\bar{x}_c, \bar{y}_c)$ for the centre coordinates (x_c, y_c). If we assume $\nu_r - 1$ to be small, we have for the intersections x_1 to x_4 with the x axis:

$$\begin{aligned}
x_1 &= r + x_c \\
x_2 &= r + \Delta r + x_c \\
x_3 &= r + \tfrac{1}{2}\Delta r - x_c \\
x_4 &= r + \tfrac{3}{2}\Delta r - x_c
\end{aligned}$$



Figuur 5.1: Calculation of the position of the orbit centre is based on a linear combination of the x and y coordinates at consecutive intersections of the orbit with the x and y axis.

Elimination of Δr yields

$$\bar{x}_c = \tfrac{1}{8}\left(x_1 + 3x_2 + 3x_3 + x_4\right) \tag{5.1}$$

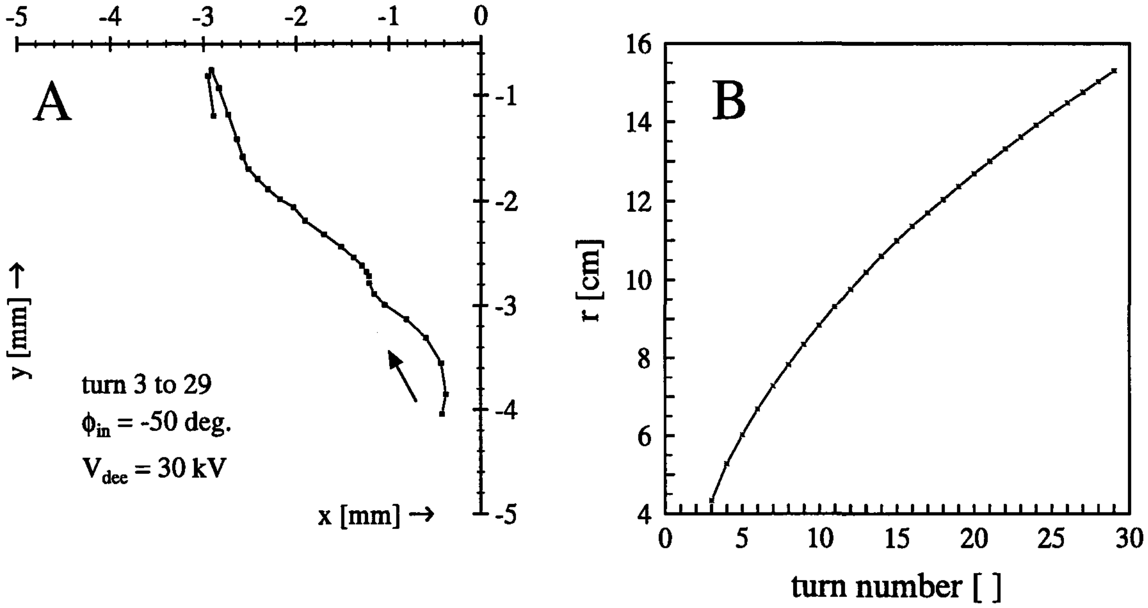
An analogous treatment of the intersections y_1 to y_4 with the y axis gives

$$\bar{y}_c = \tfrac{1}{8}\left(y_1 + 3y_2 + 3y_3 + y_4\right) \tag{5.2}$$

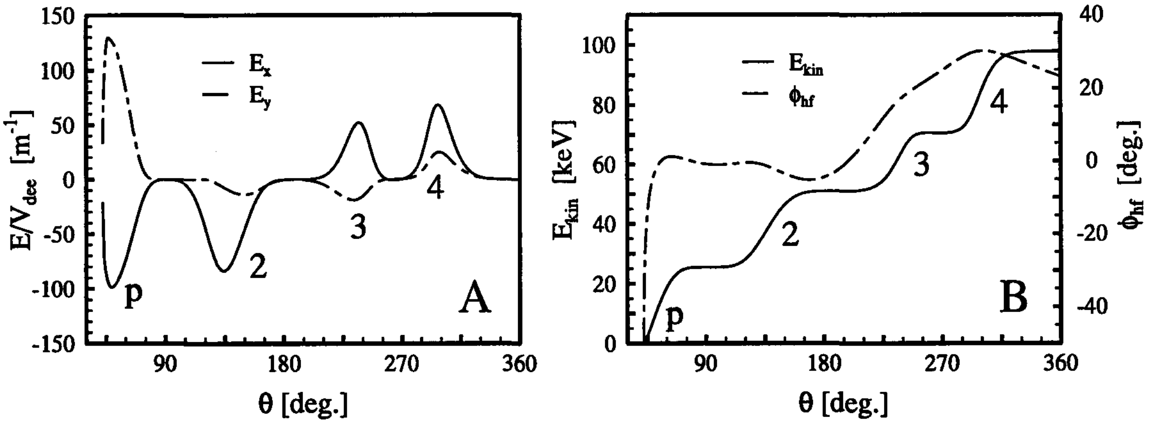
5.3 Centre motion in the symmetrical field

First we shall investigate the movement of the orbit centre for a typical particle in the ideal symmetrical magnetic field, which consists of the first four Fourier coefficients A_{4k} from figure 2.5.

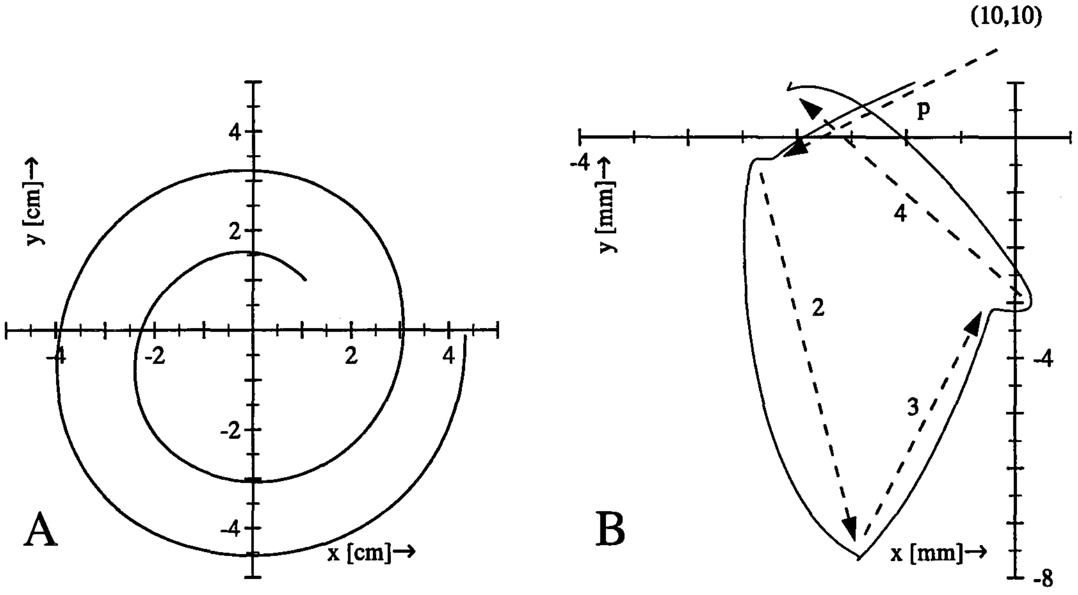
In figure 5.2 [A] we show the movement of the average centre position in the symmetrical magnetic field for a particle with initial phase $\varphi_{in} = -50°$, calculated with (5.1) and (5.2). With the aid of figure 5.2 [B], we can determine the approximate orbit radius for a certain turn. The first thing we notice, is that the initial position of the orbit centre (turn 3) does not coincide with the cyclotron centre. A more detailed study of the particle motion during the first revolution indicates that this deviation of the initial centre position originates from the first gapcrossing. From figure 5.3 it becomes clear that during gapcrossing 2 the particle experiences a much higher electrical force along the x axis than during crossings 3 and 4. The reason for this is, that at small radii the accelerating gaps become somewhat wider, and more directed along the y axis (see figure 2.1). An electrical field in the direction of the negative x axis will force the orbit centre to move in negative y-direction. Because of the large relative energy increase during gap crossing 2, the negative displacement of the



Figuur 5.2: Motion of the average centre position of the beam in the symmetrical magnetic field [A] and the orbit radius at $\theta = 0$ [B] as a function of the turn number in the symmetrical magnetic field. The arrow indicates the direction of the movement, starting at turn 3. Each marker represents the centre position for the corresponding turn.



Figuur 5.3: The x and y components of the electrical field [A] and the particle energy [B] as a function of the azimuth θ during the first revolution of the particle from figure 5.2. The dotted line in [B] indicates the high frequency phase. The particle starts at $\theta = 42.4°$, where it is first accelerated by the electrical field between source and puller (p). The following gapcrossings are numbered 2 to 4, for consistency with the numbering in fig. 2.1.



Figuur 5.4: The first two turns of the particle orbit [A] and the movement of the orbit centre during the first turn [B] for the particle from fig. 5.2. The arrows indicate the translation of the orbit centre during acceleration between source and puller (p) and in the gapcrossings (2 to 4) as in fig. 5.3.

orbit centre will not be fully compensated during crossings 2 and 3. Hence the average centre position will be at negative y values.

Because we have no modulation of the magnetic field at the radius of the first turn, we may assume the magnetic field there to be homogeneous, if we neglect the influence of the source hole. This implies that the particle will always move along a circular orbit, with the velocity vector perpendicular to the radius, which enables us to monitor the position of the orbit centre during the first revolution by the simple relations

$$\begin{aligned} x_c &= x - r(E)\cos(\theta_\nu - \tfrac{1}{2}\pi) \\ y_c &= y - r(E)\sin(\theta_\nu - \tfrac{1}{2}\pi) \end{aligned}$$

where $r(E)$ is the radius belonging to the kinetic energy E of the particle, conform equation (4.6) and θ_ν denotes the angle of the velocity vector with the positive x axis. A more detailed view of the particle motion during the first turn is given in figures 5.4 [A] and [B]. Clearly we can distinguish the shifts of the orbit centre resulting from the acceleration between source and puller and in gaps 2 to 4. It is this initial deviation of the orbit centre from the cyclotron centre that is responsible for the further motion of the orbit centre in figure 5.2.

In [18] the total motion of an accelerated particle is split into the motion of the orbit centre and a circle motion with respect to this orbit centre. To obtain the desired splitting of the motion, the original Hamiltonian $H(x, p_x, y, p_y)$ is transformed via several canonical transformations into a scaled Hamiltonian $\bar{H}(y, p_y, E, \phi)$ with

$$\begin{aligned} \bar{H} &= \frac{H}{m\omega_0^2} \\ \omega_0 &= \frac{qB_0}{m} \\ \bar{p}_i &= \frac{p_i}{qB_0} \\ \bar{t} &= \omega_0 t \end{aligned}$$

In this Hamiltonian the conjugated pair (y, p_y) represents the (slowly varying) coordinates of the orbit centre in relation to the cyclotron centre, and the pair (E, ϕ) are the action angle variables of the circle motion. The action angle variable E may be interpreted as the energy of the particle, and is related to the radius R of the circle motion by:

$$E = \frac{E_k}{m\omega^2} = \tfrac{1}{2}R^2 \tag{5.3}$$

The angle variable ϕ represents the phase of the circle motion, and is a cyclic variable.

If we neglect the influence of the circle motion on the centre motion in the Hamiltonian, by assuming that E and ϕ are constant, we find [18]:

$$H = -\tfrac{1}{2}(\nu_r - 1)\left(y^2 + P_y^2\right) - C^* y P_y + \tfrac{1}{2}D^*\left(P_y^2 \cos^2\alpha + y^2 \sin^2\alpha\right) \tag{5.4}$$

with

$$\begin{aligned} C^* &= \frac{2q\bar{V}}{\pi} h \cos h\alpha \cos \alpha sin \alpha \frac{1}{2E} \cos\left[h(\phi - \phi_0)\right] \\ D^* &= -\frac{2q\bar{V}}{\pi} h \sin h\alpha \frac{1}{2E} \sin\left[h(\phi - \phi_0)\right] \\ \bar{V} &= \frac{\hat{V}}{m\omega_0^2} \\ \sin\phi_0 &= \frac{q\bar{V}}{2E} \cos\alpha \sin\alpha \sin h\alpha \\ h &: \text{harmonic number of the accelerating voltage} \\ \alpha &: \text{halve dee angle} \\ \nu_r &: \text{radial oscillation frequency} \end{aligned}$$

In this equation we see a yP_y term, of which the coefficient C^* depends only slightly on the phase ϕ. Because for ILEC $\nu_r - 1$ is small (< 0.01), it could well be that the flow of the orbit centre in figure 5.2 is caused by an unstable motion resulting from this coupling. If we presume that the centre motion is only determined by the yP_y term, we find for the equations of motion:

$$\dot{\bar{x}}_c = -\frac{\partial H}{\partial \bar{y}_c} = C^* \bar{x}_c \tag{5.5}$$

$$\dot{\bar{y}}_c = \frac{\partial H}{\partial \bar{x}_c} = -C^* \bar{y}_c \tag{5.6}$$

where we denote the phase plane coordinates (y, P_y) by the geometrical coordinates $(\bar{y}_c, \bar{x}_c)$. According to [18] we can make a rough first-order approximation of the centre position $(\bar{x}_c, \bar{y}_c)$ after n turns by:

$$\bar{x}_c(n) \approx \bar{x}_c(n_0)\left(\frac{n}{n_0}\right)^\gamma \tag{5.7}$$

$$\bar{y}_c(n) \approx \bar{y}_c(n_0)\left(\frac{n_0}{n}\right)^\gamma \tag{5.8}$$

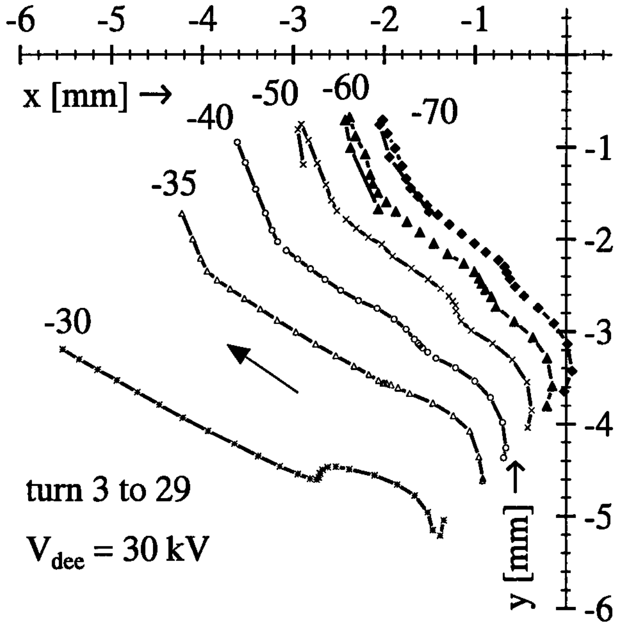
where

$$\begin{aligned} \gamma &= h\frac{\cos h\alpha}{sinh\alpha}\cos\alpha\sin\alpha \\ &= 0.64 \text{ for ILEC} \end{aligned} \tag{5.9}$$

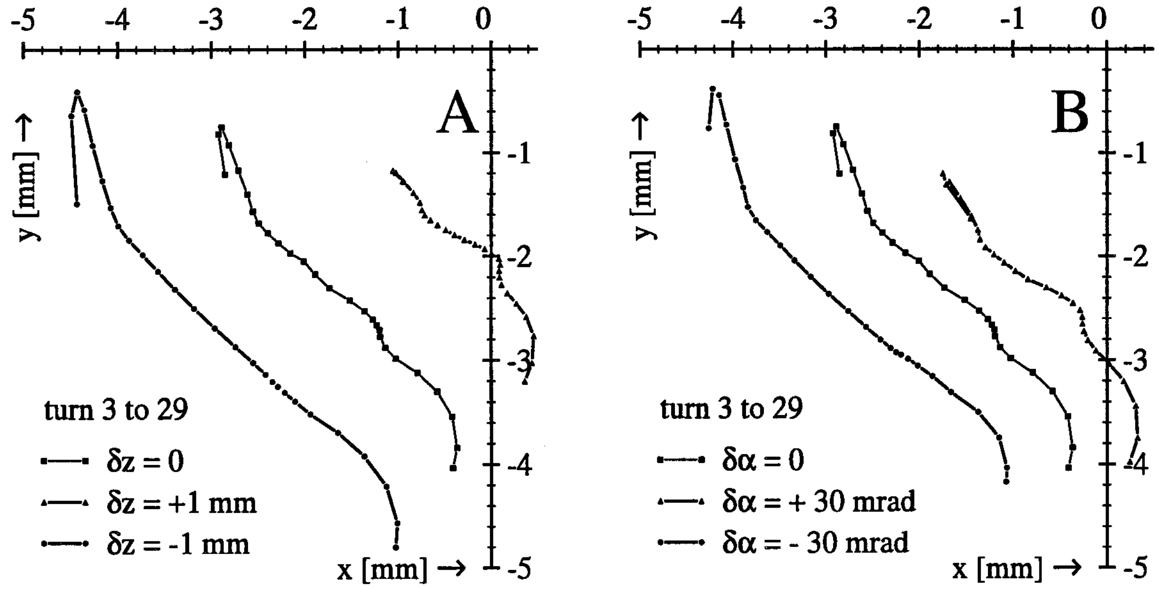
Because in these equations all phase-dependent terms were neglected, the real spread of the centre coordinates during acceleration will not depend on the initial position of the orbit centre alone. Especially at low radii, the third term in equation (5.4) can not be neglected. Therefore we might expect the centre coordinates for particles with different initial phase to diverge more than the initial spread multiplied by the deviation terms in (5.7) and (5.8), which is confirmed by the results in figure 5.5.

Another remarkable observation we can make in figure 5.5 is the irregular behaviour of the centre motion after turn number 27, for particles with initial phases of −50°, −60° and −70°. The reason for this behaviour is, that the particles enter the fringe field during their 27^{th} revolution, at the point with the greatest distance from the cyclotron centre. In the fringe field the value of $\nu_r - 1$ will decrease very rapidly, and become negative reversing the motion of the orbit centre into the observed 'backswing'.

Because we don't know the exact orientation of the puller with respect to the ion source (the angle between source and puller can be varied a few degrees), we should also consider the motion of the orbit centre under slight variations of the conditions under which the particles leave the puller. Figure 5.6 [A] shows the the centre motion for particles leaving the puller, where the radial position is varied with an amount of $\delta z = \pm 1$ mm. The effect of the alteration of the angle of the velocity vector over $\delta\alpha = \pm 30$ mrad is exposed in figure 5.6 [B].



Figuur 5.5: The movement of the average centre position of the orbit for particles with different initial phase. Because of the phase-dependency of the motion the spread in the centre coordinates becomes larger than expected from (5.7) and (5.8). We also see that extraction beyond turn 27 is not favorable, because particles with low initial phase already encounter the influence of the fringe field, resulting in poor beam quality.



Figuur 5.6: The movement of the average centre position of the orbit for particles leaving the puller at different radii and under different angles. The central particle has initial phase $\varphi_{in} = -50°$, the dee voltage was again 30 kV.

5.4 The influence of correction coils $2^{1,2}$ and $2^{A,B}$

In 2.3.2 we already stated the magnetic field perturbation caused by a first harmonic correction coil centered at (r_c, θ_c) to be:

$$B_1(r,\theta) = \hat{B}_1 e^{-\frac{1}{2}\left(\frac{r-r_c}{\sigma}\right)^2} \cos(\theta - \theta_c) \tag{5.10}$$

To be able to incorporate the influence of the correction coils $2^{1,2}$ and $2^{A,B}$ in the orbit calculations, we have to know the parameters $\hat{B}_1$ and σ. Because we have no data from measurements, we have to estimate these values.

For the characteristic coil width σ we take

$$\sigma = \tfrac{1}{2}\sqrt{(\Delta r)^2 + \left(\tfrac{g}{2}\right)^2}$$

with Δr the average radial with of the coil, and g the width of the magnet gap at the position of the coil. If we demand the total magnetic induction to equal the magnetic induction of an ideal coil with NI ampère turns we find for the amplitude $\hat{B}$ of the magnetic induction

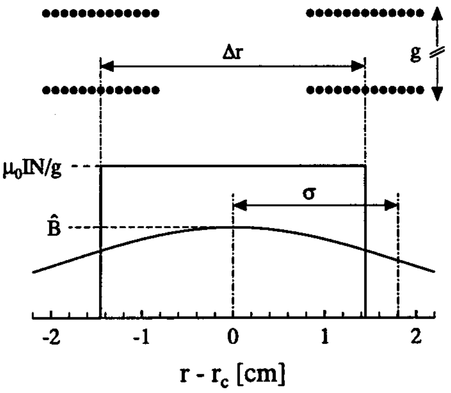
$$\int_{-\infty}^{\infty} \hat{B} e^{-\frac{1}{2}\left(\frac{r}{\sigma}\right)^2} dr = \frac{\mu_0 IN}{g}\Delta r \Rightarrow \hat{B} = \frac{1}{\sqrt{2\pi}}\frac{\mu_0 IN}{g}\frac{\Delta r}{\sigma} \tag{5.11}$$

Integration over the azimuthal width $\Delta\theta$ of the coil gives for the first harmonic component

$$\begin{aligned}\hat{B}_1 &= \frac{1}{\pi}\int\limits_{-\frac{1}{2}\Delta\theta}^{\frac{1}{2}\Delta\theta}\hat{B}\cos\theta d\theta \\ &\approx \hat{B}\frac{\Delta\theta}{\pi} \text{ for } \Delta\theta \text{ small}\end{aligned}$$

In figure 5.7 we tabulated the estimated values for $\hat{B}_1$ and σ, and the magnitude of the parameters we used in the approximation. The right-hand figure gives a graphical representation of $\hat{B}$ and σ in relation with the coil dimensions for the coil pair $2^{1,2}$. Note that for the magnetic field of one pair of radially opposite coils, with their currents in opposite direction, we have to double the value of $\hat{B}_1$ in expression (5.10).

	coils $2^{1,2}$	coils $2^{A,B}$
r_c	12 cm	14 cm
θ_c	0°	45°
Δr	2.80 cm	2.80 cm
$\Delta\theta$	40°	24°
g	50 mm	35 mm
$I \times N$	20×26 A	20×26 A
σ	1.88 cm	1.65 cm
$2\hat{B}_1$	$3.5 \cdot 10^{-3}$ T	$3.4 \cdot 10^{-3}$ T

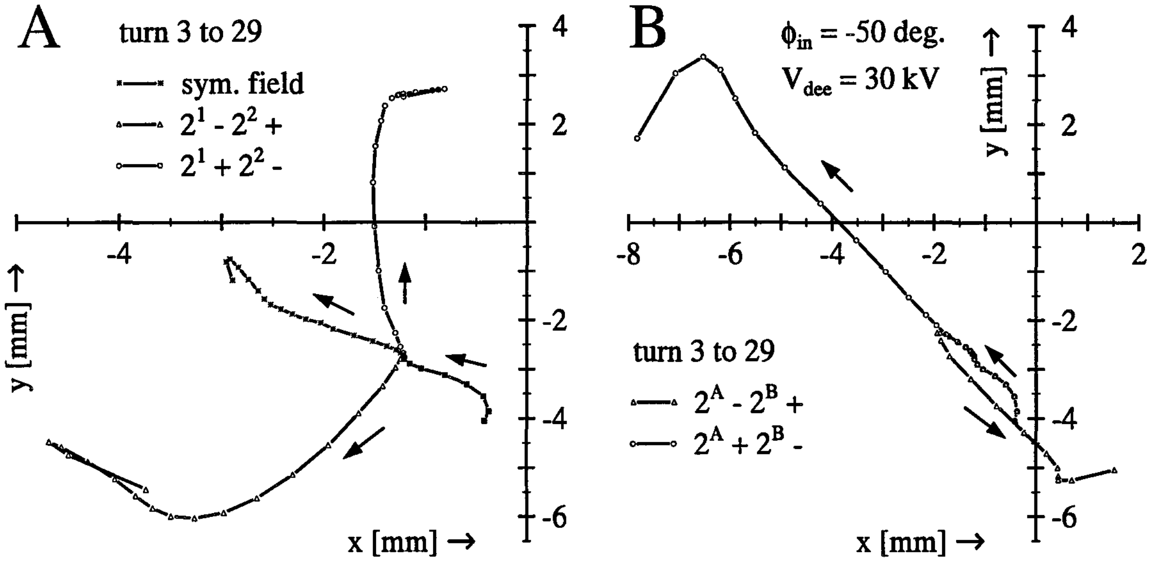


Figuur 5.7: The parameters for the correction coil pairs $2^{1,2}$ and $2^{A,B}$, used in the numerical calculations.

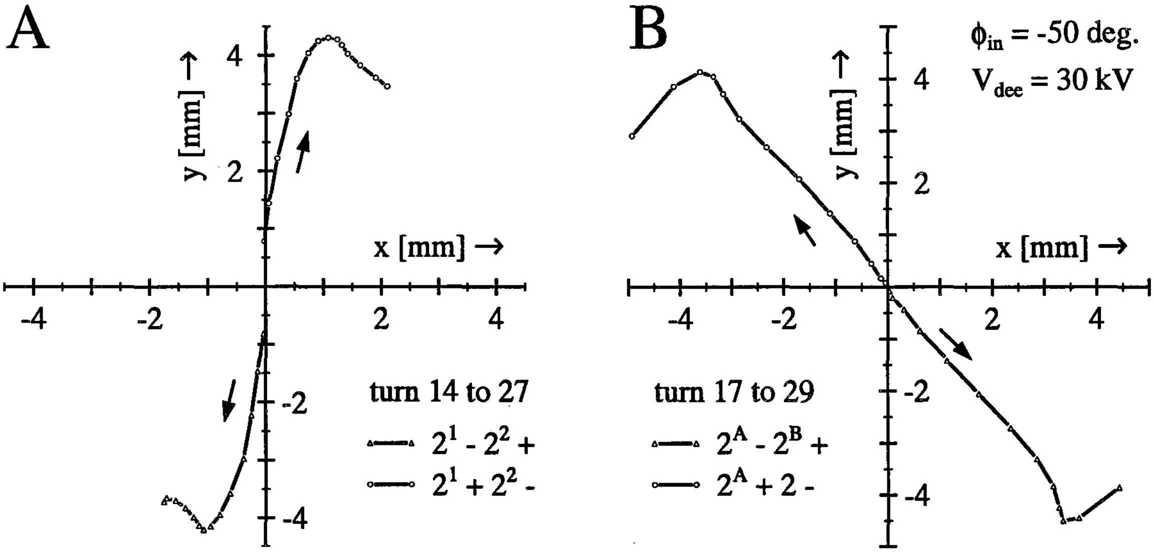
The influence of the 1^{st} harmonic coils on the orbit centre motion is shown in figures 5.8 and 5.9. The relative displacement of the orbit centre in figure 5.9 is constructed by subtracting the centre motion in the unperturbated symmetrical field from the perturbated motion. These results can be compared with the results of the measurements from [12], depicted in figure 5.10. If we take into account the possible error in the measurements, we see that the agreement is quite good.

By applying the perturbation Hamiltonian for a first harmonic component, derived in [6], we can write for the relative motion of the orbit centre:

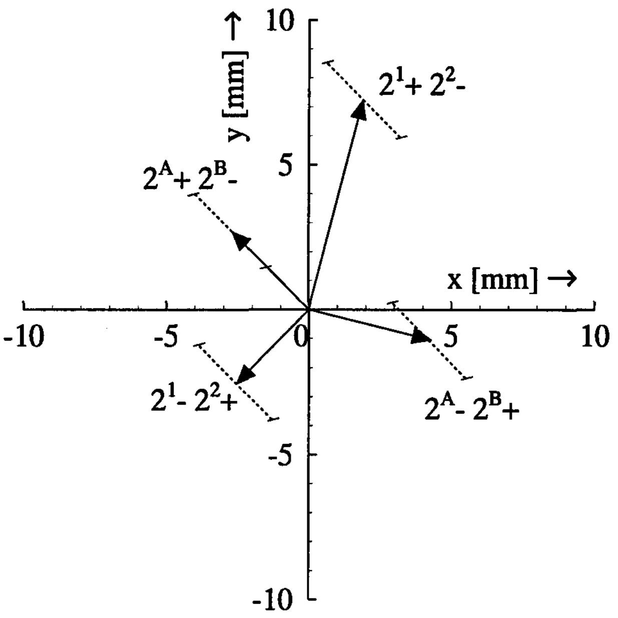
$$\begin{aligned}H_1 &= -\frac{r\tilde{B}_1(r)}{2}(x_c\cos\theta_c + y_c\sin\theta_c) \Rightarrow \\ \dot{x_c} &= \frac{r\tilde{B}_1(r)}{2}\sin\theta_c \\ \dot{y_c} &= -\frac{r\tilde{B}_1(r)}{2}\cos\theta_c\end{aligned}$$



Figuur 5.8: Motion of the orbit centre under influence of the coil pair $2^{1,2}$ [A] and $2^{A,B}$ [B] (absolute centre coordinates). The arrows indicate the direction of the displacement. Over the first ten turns the three curves coincide, because the correction coils have no influence on the centre movement st lower radii.



Figuur 5.9: Calculated relative motion of the orbit centre under influence of the coil pair $2^{1,2}$ [A] and $2^{A,B}$ [B].



Figuur 5.10: The measured relative motion of the orbit centre under influence of the coil pairs $2^{1,2}$ and $2^{A,B}$.

with $\tilde{B}_1$ the relative amplitude of the first harmonic. These equations of motion show that the orbit centre will move in a direction perpendicular to the azimuth θ_c of the first harmonic. Because for ILEC $\nu_r - 1$ is quite small, this representation of the relative centre movement is appropriate when the centre displacement does not become too large. Integration per turn gives for the displacement of the orbit centre over one turn:

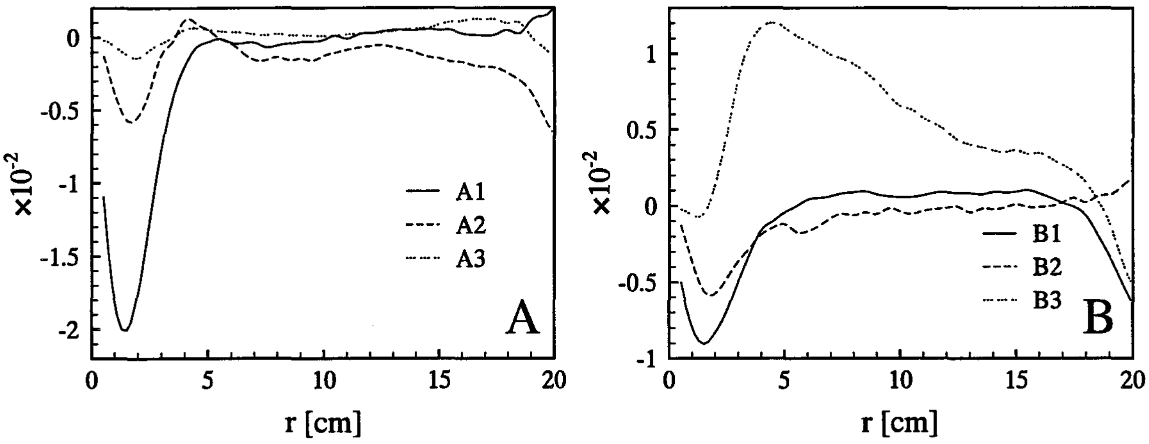
$$\begin{aligned}\Delta x_c(n) &= \pi r(n)\tilde{B}_1(r(n))\sin\theta_c \\ \Delta y_c(n) &= -\pi r(n)\tilde{B}_1(r(n))\cos\theta_c\end{aligned}$$

which enables us to approximate the center position after n successive turns by:

$$x_c(n) = x_c(n_0) + \pi n\bar{r}\bar{B}_1\sin\theta_c \tag{5.12}$$

$$y_c(n) = y_c(n_0) - \pi n\bar{r}\bar{B}_1\cos\theta_c \tag{5.13}$$

where $\bar{r}$ and $\bar{B}_1$ denote average values over turns n_0 to n. For the estimation of the centre displacement due to the coil pair $2^{1,2}$ from turn 14 to 20, we can see in figure 5.2 that $\bar{r} \approx 117.5$ mm. With $\bar{B}_1 \approx 1.5 \cdot 10^{-3}$ T, we find $\Delta y_c \approx 3.3$ mm. The numerically calculated centre movement in figure 5.9 [A] also shows that the shift over turns 14 to 20 is about 3 mm.



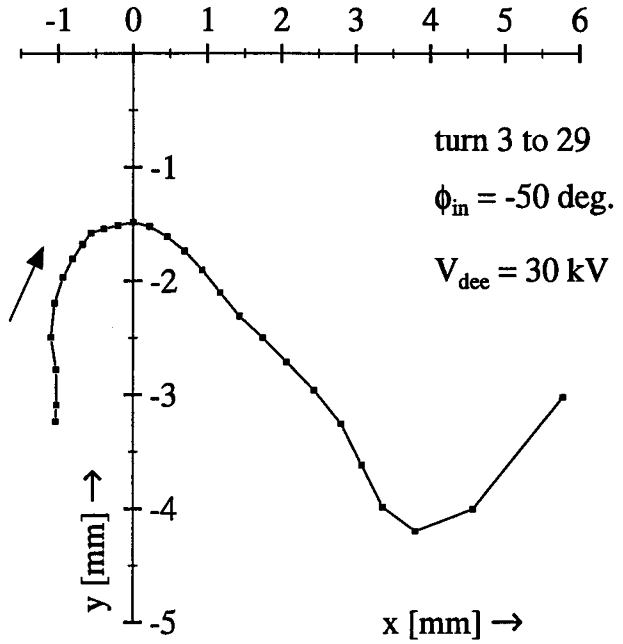
Figuur 5.11: The Fourier cosine [A] and sine [B] coefficients of the first three harmonics, derived from the magnetic field measurements. The first harmonic components clearly show the influence of the source hole on the centre field.

5.5 Lower harmonic perturbations

To achieve good agreement between the calculated and measured centre motion in the ILEC magnetic field, we should also consider the influence of perturbations from the symmetrical field. Figure 5.11 show the relative Fourier cosine and sine coefficients of the first three harmonics, as calculated from field measurements. Because the influence of the 2^{nd} and 3^{rd} harmonics is quite small, we shall only consider the the first harmonics A_1 and B_1 on the centre motion. figure 5.12 depicts the centre motion for a particle with $\varphi_{in} = -50°$, where we added the first harmonics to the symmetrical field. We see that the main influence on the centre motion comes from the B_1 component, which causes a relative first harmonic of approximately 10^{-3} in the radial interval $5 < r < 17$ cm at $\theta = 90°$. If we compare the calculated centre motion with the results of the measurements in [12], we see that the measured centre drift is of the same order of magnitude, but nearly in the opposite direction, which implies that the measured magnetic field does not agree with the reality.

The reason why the measured magnetic field differs from the real situation is not clear. Regarding the consistency of the data, it appears to be unlikely that a measuring error of this magnitude has been committed. The only alternative is that there has been an alteration in the magnetic field, which occurred after the period during which field measurements have been performed. Some years ago, the minicyclotron has been moved to it's present location. A possible explanation for the change in the magnetic field could be a small deformation of the main magnet or the pole-faces, resulting from this operation.

By repositioning the small correction slabs with which the pole faces are provided, we were able to create a permanent first harmonic, which counteracts the unwanted first harmonic. The resulting orbit centre motion is very much the same as in figure 5.12. The orbit centre drifts towards the extractor entrance, thus increasing the turn separation at



Figuur 5.12: The motion of the orbit centre due to the measured first harmonic perturbation, added to the symmetrical field (absolute centre coordinates).

the position of the extractor. A more extensive treatment of the repositioning of the orbit centre is given in [12].

Chapter 6

Conclusions and recommendations

The new programs for calculating the properties of the internal ILEC beam turned out to be a great improvement on both precision and speed. We now have the possibility to simulate the influence of various parameters on the behaviour of the beam almost interactively, and to evaluate the energy-current distribution of the extracted beam. The results of the calculations show good agreement with measurements, and allow us to dimension and position an internal aperture, in order to produce a beam of the desired quality.

The results of these calculations indicate that ILEC should be able to produce a fairly intense proton beam of high quality. With only second harmonic acceleration, single-turn extraction is feasible, resulting in extraction of approximately 100 % of the internal beam current, with an energy spread of 1 %. By applying the flattop system, this energy spread can be reduced to less than 0.1 %, while still extracting more than 20 % of the initial source-current.

For the future, it is desirable that also the emittance of the external beam can be calculated. Therefore the existing programs have to be modified. Because the numerical models for the extractor, magnetic channel and fringe-field are already incorporated in the present code, the extension should ask for a relatively small effort.

Bibliography

[1] van den Akker, J.
Graduation report, Eindhoven Univ. of Technology, 1986

[2] Bennet, J.R.J.,
First International Conference on Ion Sources, Saclay, june 1969

[3] Collatz, L.,
The numerical treatment of differential equations,
third edition, Springer-Verlag, 1960

[4] Dorn, William S., McCracken, Daniel D.,
Numerical methods with FORTRAN case studies,
John Wiley & Sons, Inc., 1972

[5] Forsythe, George E., Malcolm, Michel A., Moler, Cleve B.,
Computer methods for mathematical computations, Prentice Hall, 1977

[6] Hagedoorn, H.L. and Verster, N.F.,
Nucl.Instr. Meth **18, 19**, (1962) 201

[7] Hazewindus, N., van Nieuwland, J.M., Faber, J. and Leistra, L.,
Nucl.Instr. Meth **118**, (1974) 125

[8] Henrici, Peter,
Discrete variable methods in ordinary differentialequations,
John Wiley & Sons, Inc., 1968

[9] Jansen, Ben,
Report of practical work at the EUT, 1985, vdf/nk 85-39

[10] Kahaner, David,
Numerical methods and software, Prentice Hall, 1989

[11] Kleeven, W.J.G.M.,
Thesis, Eindhoven Univ. of Technology, 1988

[12] Keuzenkamp, Maarten,
Graduation report, Eindhoven Univ. of Technology, november 1994

[13] Kreyszig, Erwin,
Advanced engineering mathematics, sixth edition, John Wiley & Sons, Inc., 1988

[14] van Nieuwland, J.M.,
Thesis, Eindhoven Univ. of Technology, 1972

[15] Press, William H.,
Numerical recipes, Cambridge University Press, 1988

[16] de Regt, R.J.L.J.,
Graduation report, Eindhoven Univ. of Technology, 1986, vdf/nk 85-22

[17] Schlangen, Jos,
Report of practical work at the EUT, 1984, vdf/nk 84-03

[18] Schulte, W.M.,
Thesis, Eindhoven Univ. of Technology, 1978

[19] Shampine, L.F. and Watts,
ACM Trans. Math. Software **2**, (june 1976) 172

[20] Shampine, L.F., Watts, H.A., and Davenport, S.M.,
Solving non-stiff ordinary differential equations–the state of the art,
SIAM Rev. **18** (1976)

[21] Steeman, P.A.M.,
Graduation report, Eindhoven Univ. of Technology, 1985, vdf/nk 85-22

[22] Spangenberg, Karl,
Vacuum Tubes, McGraw-Hill Book Company, 1947

[23] Theeuwen, S.J.C.H.,
Report of practical work at the EUT, 1992, vdf/nk 92-19

Appendix A

Program listings

```
      PROGRAM IDIF10
C     ================================================================
C     LAST UPDATED : 21-08-94, PETER OP DE BEEK
C     ================================================================
C     UPDATE-HISTORY
C     ================================================================
C
C     ================================================================
      IMPLICIT NONE
      REAL*8  rad,rdat(3,50),edat(3,50),ddat(3,50),
     +        dr,de,fas(3),fas_in(3),fas_in2(3),diaf(4),
     +        di_r(2000),di_e(2000),res_f,res_r,res_e,
     +        rslp_mx,eslp_mx,dslp_mx,roi_mx,roi_mn,mn_e,mx_e,av_e,
     +        df,df_mx,ib_tot,i_roi,f_mn,f_mx,rmx,rmx_in


      INTEGER fnum,funit,f_in,par(6),par_in(6),i,
     +        ndf,i_mx,i_mn

      LOGICAL eof,diaf_ok
      CHARACTER*1  answ
      CHARACTER*40 files(5)
      CHARACTER*80 comm
      CHARACTER*4 name_out
      CHARACTER*7 name_re,name_d
C     ================================================================

C     INTEGER FNUM CONNECTS A FILENAME FROM ARRAY FILES TO A FIXED
C     UNIT-SPECIFIER ACCORDING TO:
C             FILENAME = FILES(FNUM)
C             UNIT     = FUNIT = FNUM + 10
      DATA files / 'in_r.rfn',
     +             'in_e.efn',
     +             'in_d.rfn',
     +             'out.idr',
     +             'out.ide'/
C     ================================================================
C  1. IN_R.RFN     INPUT FILE, R AS FUNCTION OF FIE_START
C  2. IN_E.EFN     INPUT FILE, E AS FUNCTION OF FIE_START
C  3. IN_D.RFN     INPUT FILE, R AS FUNCTION OF FIE_START
C                  AT AZIMUTH DIAFRAGMA
C  4. OUT.IDR      OUTPUT FILE, RADIAL CURRENT DISTRIBUTION
C  5. OUT.IDE      OUTPUT FILE, ENERGY-CURRENT DISTRIBUTION IN ROI
C     ================================================================
      rad = DATAN(1.0D0)/45.0D0
C     MINIMAL RESOLUTION RES_F IN INITIAL PHASE TO GUARANTEE A
C     HIGH ENOUGH ACCURACY IN INTEGRATION OF SOURCE-CURRENT.
C     A STEP OF 0.1 DEG. WILL BE SUFFICIENT, BECAUSE I(F) IS A
C     VERY WELL-BEHAVED FUNCTION.
C     NUMERICAL INTEGRATION WITH STEP = 0.1 DEG. FOR 0 < F < 90 DEG.
C     GIVES A DEVIATION LESS THAN 1.0D-5 % FROM THE ANALYTICAL VALUE:
      res_f = 1.0D-1
      res_f = res_f*rad
C     INITIAL VALUES FOR SOME VARIABLES AND ARRAYS:
      ib_tot = 0.0D0
      i_mx = 0
      i_mn = 2000
      DO 5 i = 1,3
        fas(i) = 0.0D0
        fas_in(i) = 0.0D0
        fas_in2(i) = 0.0D0
5     CONTINUE
      diaf_ok = .FALSE.
      eof = .FALSE.
C     DETERMINE NAMES OF IN- AND OUTPUT FILES:
      WRITE(6,*) 'GIVE NAME INPUTFILES FOR R EN E  (7 char)'
      READ(5,'(A7)') name_re
```

```
      WRITE(6,*) 'GIVE NAME OUTPUT FILE (4 char)'
      READ(5,'(A4)') name_out
C     ADAPTING FILE-NAMES IN ARRAY FILES TO GIVEN NAMES:
      files(1) = 'user2:[tnndrr.ilec.output]'//name_re//'.rfn'
      files(2) = 'user2:[tnndrr.ilec.output]'//name_re//'.efn'
      files(4) = 'user2:[tnndrr.ilec.idif]'//name_out//'ir.asc'
      files(5) = 'user2:[tnndrr.ilec.idif]'//name_out//'ie.asc'
C     OPEN INPUT FILE FOR R AS FUNCTION OF INITIAL PHASE:
      fnum = 1
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='old')
      CALL get_par(fnum,files,comm,par,rmx)
      WRITE(6,*) 'PHASE REGION OF INPUT FILE ',files(1),' :'
      WRITE(6,'(1X,A25,I3)') 'LOWEST PHASE (DEG.)    : ',par(4)
      WRITE(6,'(1X,A25,I3)') 'HIGHEST PHASE (DEG.)   : ',par(5)
      WRITE(6,'(1X,A25,I3)') 'PHASE INTERVAL (DEG.) : ',par(6)
      WRITE(6,*) 'SELECT PHASE REGION'
10    WRITE(6,*) 'GIVE LOWEST PHASE VALUE (DEG.) '
      READ(5,*) f_in
      IF (f_in.LT.par(4)) GO TO 10
      f_mn = f_in*rad
20    WRITE(6,*) 'GIVE HIGHEST PHASE VALUE (DEG.) '
      READ(5,*) f_in
      IF (f_in.GT.par(5)) GO TO 20
      f_mx = f_in*rad
C     IF A DIAPHRAGM IS PRESENT, ALSO R AS A FUNCTION OF THE INITIAL
C     PHASE AT THE AZIMUTH OF THIS DIAPHRAGM MUST BE READ:
      WRITE(6,*) 'PLACE DIAPHRAGM (Y/N) ?'
      READ(5,'(A1)') answ
      IF ((answ.EQ.'Y').OR.(answ.EQ.'y')) THEN
        diaf_ok = .TRUE.
        WRITE(6,*) 'GIVE NAME INPUT FILE R AT AZIMUTH DIAPR. (7 char)'
        READ(5,'(A7)') name_d
        files(3) = 'user2:[tnndrr.ilec.output]'//name_d//'.rfn'
C       OPEN INPUT FILE FOR R AS FUNCTION OF INITIAL PHASE
C       AT AZIMUTH DIAPHRAGM:
        fnum = 3
        funit = fnum + 10
        OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='old')
        CALL get_par(fnum,files,comm,par_in,rmx_in)
C       CHECK CONSISTENCY INPUT-FILES:
        DO 25 i = 1,6
          IF (par_in(i).NE.par(i)) GO TO 1600
25      CONTINUE
        IF (rmx_in.GT.rmx) GO TO 1600
C       READ DIMENSIONS OF THE DIAPHRAGM:
        WRITE(6,*)'THE DIAPHRAGM CONSISTS OF TWO BEAM STOPS,',
     +            'DEFINING THE SLIT'
        WRITE(6,*)'LOWEST RADIUS FIRST STOP (cm)'
        READ(5,*) diaf(1)
        WRITE(6,*)'HIGHEST RADIUS FIRST STOP (cm)'
        READ(5,*) diaf(2)
        WRITE(6,*)'LOWEST RADIUS SECOND STOP (cm)'
        READ(5,*) diaf(3)
        WRITE(6,*)'HIGHEST RADIUS SECOND STOP (cm)'
        READ(5,*) diaf(4)
        DO 30 i = 1,4
          diaf(i) = diaf(i)*1.0D-2
30      CONTINUE
      END IF
      WRITE(6,*) 'ENERGY-CURRENT DISTRIBUTION IN ROI (R) OR',
     +' DIFFERENTIAL CURRENT DISTRIBUTION (D) ? '
      READ(5,'(A1)') answ
      IF ((answ.EQ.'R').OR.(answ.EQ.'r')) THEN
C       OPEN INPUT FILE FOR E AS FUNCTION OF INITIAL PHASE:
        fnum = 2
        funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='old')
      CALL get_par(fnum,files,comm,par_in,rmx_in)
C     CHECK CONSISTENCY INPUT-FILES:
      DO 35 i = 1,6
        IF (par_in(i).NE.par(i)) GO TO 1600
35    CONTINUE
      IF (rmx_in.GT.rmx) GO TO 1600
C     OPEN OUTPUT FILE FOR IDE:
      fnum = 5
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='UNKNOWN')
      WRITE(6,*) 'GIVE RADIAL RESOLUTION (mm)'
      READ(5,*) res_r
      res_r = res_r*1.0D-3
C     FIRST DETERMINE MAX AND MIN ENERGY IN ROI:
C     READ POSITION ROI:
      WRITE(6,*) 'LOWEST RADIUS ROI (cm)'
      READ(5,*) roi_mn
      roi_mn = roi_mn*1.0D-2
      WRITE(6,*) 'HIGHEST RADIUS ROI (cm)'
      READ(5,*) roi_mx
      roi_mx = roi_mx*1.0D-2
C     INITIAL VALUES MIN_E, MAX_E:
      mn_e = 3.0D3
      mx_e = 0.0D0
C     CALL GET_DAT FIRST TIME TO FILL COLUMN 2 IN ARRAYS YDAT
C     WITH Y-VALUES:
      fnum = 1
      CALL get_dat(fnum,par,fas,f_mn,f_mx,rdat,eof)
      IF (eof) GOTO 1500
      fnum = 2
      CALL get_dat(fnum,par,fas_in,f_mn,f_mx,edat,eof)
      IF (eof) GOTO 1500
      IF (diaf_ok) THEN
        fnum = 3
        CALL get_dat(fnum,par,fas_in2,f_mn,f_mx,ddat,eof)
        IF (eof) GOTO 1500
      END IF
100   fnum = 1
      CALL get_dat(fnum,par,fas,f_mn,f_mx,rdat,eof)
      IF (eof) GOTO 200
      fnum = 2
      CALL get_dat(fnum,par,fas_in,f_mn,f_mx,edat,eof)
      IF (eof) GOTO 200
C     CHECK CONSISTENCY INPUT-FILES:
      DO 110 i = 1,2
        IF (fas_in(i).NE.fas(i)) GO TO 1600
110   CONTINUE
      IF (diaf_ok) THEN
        fnum = 3
        CALL get_dat(fnum,par,fas_in2,f_mn,f_mx,ddat,eof)
        IF (eof) GOTO 200
C       CHECK CONSISTENCY INPUT-FILES:
        DO 120 i = 1,2
          IF (fas_in2(i).NE.fas(i)) GO TO 1600
120     CONTINUE
      END IF
C     FILL COLUMN 3 OF YDAT WITH SLOPE:
      CALL slope(par,fas,rdat,rslp_mx)
      CALL slope(par,fas,edat,eslp_mx)
      IF (diaf_ok) THEN
        CALL slope(par,fas,ddat,dslp_mx)
C       DETERMINE DF_MAX, MAX STEP IN RADIUS < RES_R:
        df_mx = DMIN1(res_r/rslp_mx,res_r/dslp_mx)
      ELSE
        df_mx = res_r/rslp_mx
      END IF
```

```
C     CHOOSE DF SUCH THAT DF <= DF_MAX AND THE INTERVAL STP_F
C     CONTAINS INTEGER NUMBER OF DF:
      ndf = INT(fas(3)/df_mx) + 1
      df = fas(3)/ndf
      WRITE(6,1700) fas(1)/rad,fas(2)/rad,df/rad
C     ADAPT MAX_E AND MIN_E FOR CURRENT PHASE AREA:
      CALL e_range(par,df,ndf,rdat,edat,ddat,diaf,diaf_ok,
     +             mx_e,mn_e,roi_mx,roi_mn)
      GO TO 100
C     THE ENERGY-RANGE OF THE ROI IS DETERMINED
C     FILL ARRAY DI_E ACCORDING TO THE DESIRED RESOLUTION IN THE
C     ENERGY-CURRENT DISTRIBUTION
C     THE COMPLETE PHASE REGION WILL BE EVALUATED AGAIN WITH THE
C     NECESSARY RESOLUTION IN DF
C     CALCULATE THE AVERAGE OF MAX_E AND MIN_E:
200   av_e = (mx_e + mn_e)/2.
      WRITE(6,*) 'THE ENERGY-RANGE IN THE ROI GOES FROM'
      WRITE(6,'(1X,F8.2,A4,F8.2,A4)') mn_e,' TO ',mx_e,' keV'
      WRITE(6,*) 'THE AVERAGE ENERGY IN THE ROI IS'
      WRITE(6,'(1X,F8.2,A4)') av_e,' keV'
      WRITE(6,*)'GIVE WINDOW SIZE  DE (keV)'
      READ(5,*) de
C     CHECK IF THE ARRAY DI_E CAN CONTIAN THE WHOLE ENERGY-RANGE
C     WITH WINDOW SIZE DE:
      IF ((mx_e - mn_e)/de.GT.1900) THEN
        WRITE(6,*)'ARRAY DI_E IS TOO SMALL FOR THIS WINDOW'
        GO TO 200
      END IF
      WRITE(6,*)'GIVE ENERGY RESOLUTION (keV)'
      READ(5,*) res_e
C     RESET INPUT FILES:
      fnum = 1
      CALL get_par(fnum,files,comm,par_in,rmx_in)
      fnum = 2
      CALL get_par(fnum,files,comm,par_in,rmx_in)
      IF (diaf_ok) THEN
        fnum = 3
        CALL get_par(fnum,files,comm,par_in,rmx_in)
      END IF
C     RESET ARRAY DI_E:
      DO 210 i = 1,2000
         di_e(i) = 0.0D0
210    CONTINUE
C     RESET ARRAYS FAS, FAS_IN, FAS_IN1:
      DO 220 i = 1,3
         fas(i) = 0.0D0
         fas_in(i) = 0.0D0
         fas_in2(i) = 0.0D0
220   CONTINUE
      eof = .FALSE.
C     CALL GET_DAT FIRST TIME TO FILL COLUMN 2 IN ARRAYS YDAT
C     WITH Y-VALUES:
      fnum = 1
      CALL get_dat(fnum,par,fas,f_mn,f_mx,rdat,eof)
      IF (eof) GOTO 1500
      fnum = 2
      CALL get_dat(fnum,par,fas_in,f_mn,f_mx,edat,eof)
      IF (eof) GOTO 1500
      IF (diaf_ok) THEN
        fnum = 3
        CALL get_dat(fnum,par,fas_in,f_mn,f_mx,ddat,eof)
        IF (eof) GOTO 1500
      END IF
300   fnum = 1
      CALL get_dat(fnum,par,fas,f_mn,f_mx,rdat,eof)
      IF (eof) GOTO 400
      fnum = 2
```

```
        CALL get_dat(fnum,par,fas_in,f_mn,f_mx,edat,eof)
        IF (eof) GOTO 400
        IF (diaf_ok) THEN
          fnum = 3
          CALL get_dat(fnum,par,fas_in,f_mn,f_mx,ddat,eof)
          IF (eof) GOTO 400
        END IF
C       FILL COLUMN 3 OF YDAT WITH SLOPE:
        CALL slope(par,fas,rdat,rslp_mx)
        CALL slope(par,fas,edat,eslp_mx)
        IF (diaf_ok) THEN
           CALL slope(par,fas,ddat,dslp_mx)
C          DETERMINE DF_MAX, MAX STEP IN
C          RADIUS <= RES_R, MAX STEP IN ENERGY <= RES_E
C          AND MAX INTEGRATIONSTEP <= RES_F:
           df_mx = DMIN1(res_r/rslp_mx,res_r/dslp_mx,
     +                   res_e/eslp_mx,res_f)
        ELSE
           df_mx = DMIN1(res_r/rslp_mx,res_e/eslp_mx,res_f)
        END IF
C       CHOOSE DF SUCH THAT DF <= DF_MAX AND THE INTERVAL STP_F
C       CONTAINS INTEGER NUMBER OF DF:
        ndf = INT(fas(3)/df_mx) + 1
        df = fas(3)/ndf
        WRITE(6,1700) fas(1)/rad,fas(2)/rad,df/rad
        CALL evalde(par,fas,df,ndf,de,av_e,i_mx,i_mn,
     +      roi_mx,roi_mn,rdat,edat,ddat,diaf,diaf_ok,di_e,ib_tot)
        GO TO 300
C       THE COMPLETE PHASE-AREA HAS BEEN EVALUATED:
400     fnum = 1
        funit = fnum + 10
        CLOSE(funit)
        fnum = 2
        funit = fnum + 10
        CLOSE(funit)
        IF (diaf_ok) THEN
          fnum = 3
          funit = fnum + 10
          CLOSE(funit)
        END IF
C       CALCULATE RELATIVE CURRENT IN EACH ELEMENT OF DI_E
C       CALCULATE TOTAL CURRENT IN ROI:
        i_roi = 0.0D0
        DO 410 i = i_mn,i_mx
          i_roi = i_roi + di_e(i)
          di_e(i) = (di_e(i)/ib_tot)*100.0D0
410     CONTINUE
C       WRITE DI_E TO FILE:
        fnum = 5
        funit = fnum + 10
        WRITE(funit,*) 'E(keV)  I/I\L0'
        DO 420 i = i_mn - 1 ,i_mx + 1
          WRITE(funit,3000)  (i-1000.5)*de + av_e,di_e(i)
420     CONTINUE
        CLOSE(funit)
        WRITE(6,*)'THE TOTAL CURRENT IN THE ROI IS:'
        WRITE(6,'(1X,F6.2,A24)') 100.*i_roi/ib_tot,
     +     '% OF THE INTERNAL CURRENT'
      ELSE
C     ===============================================================
C     ONLY DIFFERENTIAL CURRENT-DISTRIBUTION:
C     ===============================================================
C       OPEN OUTPUT FILE FOR IDR:
        fnum = 4
        funit = fnum + 10
        OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='UNKNOWN')
        WRITE(6,*) 'GIVE WINDOW SIZE DR (mm)'
```

```
        READ(5,*) dr
        dr = dr*1.0D-3
        WRITE(6,*) 'GIVE RADIAL RESOLUTION (mm)'
        READ(5,*) res_r
        res_r = res_r*1.0D-3
C       RESET ARRAY DI_R:
        DO 430 i = 1,2000
           di_r(i) = 0.0D0
430      CONTINUE
C       CALL GET_DAT FIRST TIME TO FILL COLUMN 2 IN ARRAYS YDAT
C       WITH Y-VALUES:
        fnum = 1
        CALL get_dat(fnum,par,fas,f_mn,f_mx,rdat,eof)
        IF (eof) GOTO 1500
        IF (diaf_ok) THEN
          fnum = 3
          CALL get_dat(fnum,par,fas_in,f_mn,f_mx,ddat,eof)
          IF (eof) GOTO 1500
        END IF
500     fnum = 1
        CALL get_dat(fnum,par,fas,f_mn,f_mx,rdat,eof)
        IF (eof) GOTO 600
        IF (diaf_ok) THEN
          fnum = 3
          CALL get_dat(fnum,par,fas_in,f_mn,f_mx,ddat,eof)
          IF (eof) GOTO 600
C       CHECK CONSISTENCY INPUT FILES:
          DO 510 i = 1,2
           IF (fas_in(i).NE.fas(i)) GO TO 1600
510     CONTINUE
        END IF
C       FILL COLUMN 3 OF YDAT WITH SLOPE:
        CALL slope(par,fas,rdat,rslp_mx)
        IF (diaf_ok) THEN
          CALL slope(par,fas,ddat,dslp_mx)
C       DETERMINE DF_MAX, MAX STEP IN
C       RADIUS <= RES_R AND MAX INTEGRATIONSTEP <= RES_F:
          df_mx = DMIN1(res_r/rslp_mx,res_r/dslp_mx,res_f)
        ELSE
          df_mx = DMIN1(res_r/rslp_mx,res_f)
        END IF
C       CHOOSE DF SUCH THAT DF <= DF_MAX AND THE INTERVAL STP_F
C       CONTAINS INTEGER NUMBER OF DF:
        ndf = INT(fas(3)/df_mx) + 1
        df = fas(3)/ndf
        WRITE(6,1700) fas(1)/rad,fas(2)/rad,df/rad
C       ADD CURRENT DI FOR EACH PHASE STEP DF TO CORRESPONDING ARRAY-
C       ELEMENT OF DI_R:
        CALL evaldr(par,fas,df,ndf,dr,i_mx,i_mn,
     +             rdat,ddat,diaf,diaf_ok,di_r,ib_tot)
        GOTO 500
C       THE COMPLETE PHASE REGION HAS BEEN EVALUATED:
600     fnum = 1
        funit = fnum + 10
        CLOSE(funit)
        IF (diaf_ok) THEN
          fnum = 3
          funit = fnum + 10
          CLOSE(funit)
        END IF
C       CALCULATE RELATIVE CURRENT IN EACH ELEMENT OF DI_R:
        DO 610 i = i_mn,i_mx
          di_r(i) = (di_r(i)/ib_tot)*100.0D0
610     CONTINUE
C       WRITE DI_R TO FILE:
        fnum = 4
        funit = fnum + 10
        WRITE(funit,*) 'r(cm)  I/I\L0'
        DO 620 i = i_mn,i_mx
          WRITE(funit,2000)  (i-1)*dr*100.,di_r(i)
620     CONTINUE
        CLOSE(funit)
      END IF
C     =================================================================
C     LABELS AND FORMATS
C     =================================================================
      STOP'PROGRAM EXECUTION READY'
1000  WRITE(6,*)'ERROR OPENING FILE ' ,files(fnum)
      STOP
1500  WRITE(6,*)'FILE ',files(fnum),' DOES NOT CONTAIN VALID DATA'
      STOP
1600  WRITE(6,*)'FORMAT ',files(fnum),' NOT CONSISTENT WITH ',files(1)
      STOP
1700  FORMAT(1X,'EVALUATING PHASE AREA FROM ',F5.1,' TO ',F5.1,
     +        ' STEP = ',E8.2,' (DEG.)')
2000  FORMAT(1X,F7.3,1X,E11.4)
3000  FORMAT(1X,E13.6,1X,E11.4)
      END
C     =================================================================
C     END MAIN PROGRAM
C     =================================================================
C     SUBROUTINES
C     =================================================================
C     =================================================================
      SUBROUTINE get_par(fnum,files,comm,par,rmx)
C     =================================================================
      REAL*8  rmx
      INTEGER fnum,funit,par(6)
      CHARACTER*40 files(5)
      CHARACTER*80 comm
C     REWIND INPUT FILE:
      funit = fnum + 10
      REWIND(funit)
C     READ COMMENT LINE:
C      READ(funit,'(A80)') comm
C     READ FORMAT INPUT FILE
C     - NUMBER OF TURNS PER INITIAL PHASE VALUE
C     - NUMBER OF R-VALUES PER LINE
      READ(funit,*,END=1500) par(1),par(2),par(3)
      IF (par(1).NE.1) STOP'N_MIN .NE.1 KAN NOG NIET!'
C     READ PHASE REGION AND FASE-STEP:
      READ(funit,*,END=1500) par(4),par(5),par(6)
C     READ MAX RADIUS
      READ(funit,*,END=1500) rmx
      RETURN
1500  WRITE(6,*)'FILE ',files(fnum),' DOES NOT CONTAIN VALID DATA'
      STOP
      END
C     =================================================================
      SUBROUTINE get_dat(fnum,par,fas,f_mn,f_mx,curdat,eof)
C     =================================================================
      IMPLICIT NONE
      REAL*8   rad,curdat(3,50),fas(3),f_in,f_mn,f_mx
      INTEGER  fnum,funit,par(6),n_st,n_en,n
      LOGICAL  eof
      rad = DATAN(1.0D0)/45.0D0
      funit = fnum + 10
C     IF Y-VALUES FOR NEXT F AVAILABLE,
C     READ NEW VALUE FOR F_E, AND THE CORRESPONDING Y_VALUES.
C     AT THE END OF FUNIT, SET EOF = .TRUE.:
      READ (funit,*,END=100) f_in
      f_in = f_in*rad
C     IF F_IN ABOVE SELECTED PHASE REGION, SET EOF = .TRUE.:
      IF (f_in.GT.f_mx) GO TO 100
C     IF F_IN BELOW SELECTED PHASE REGION, KEEP READING
C     DATA UNTIL F_IN >= F_MN
C     ALS F_IN ONDER GESELECTEERDE FASEGEBIED LIGT, DAN
C     BLIJF DATA INPUFILE LEZEN TOTDAT F_IN >= F_MN
      DO WHILE (f_in.LT.f_mn)
        n_st = 1
        n_en = n_st + par(3) - 1
        DO WHILE (n_en.LE.par(2))
          READ (funit,*,END=200) (curdat(1,n),n = n_st,n_en)
          n_st = n_st + par(3)
          n_en = n_en + par(3)
        END DO
        READ (funit,*,END=100) f_in
        f_in = f_in*rad
      END DO
C     F_IN IS NOW >= F_MN
C     COLUMN 1 OF CURDAT IS NOW FILLED WITH TRASH,
C     COLUMN 2 IS STILL FILLED WITH 0 (1ST CALL) OR WITH Y-VALUES
C     COPY COLUMN 2 TO COLUMN 1:
      DO 10 n = 1,par(2)
10    curdat(1,n) = curdat (2,n)
      fas(1) = fas(2)
C     RESET COLUMN 2:
      DO 20 n = 1,par(2)
20    curdat(2,n) = 0.0D0
C     FILL COLUMN 2 WITH NEW Y-VALUES:
      n_st = 1
      n_en = n_st + par(3) - 1
      DO WHILE (n_en.LE.par(2))
        READ (funit,*,END=200) (curdat(2,n),n = n_st,n_en)
        n_st = n_st + par(3)
        n_en = n_en + par(3)
      END DO
      fas(2) = f_in
      fas(3) = fas(2) - fas(1)
      RETURN
C     TWO POSSIBILITIES:
C     1. END OF INPUT FILE HAS BEEN REACHED
C     2. END OF SELECTED PHASE REGION HAS BEEN REACHED
100   eof = .TRUE.
      RETURN
200   STOP'END INPUTFILE HAS BEEN ENCOUNTERED WHILE READING Y-VALUES'
      END
C     =================================================================
      SUBROUTINE slope(par,fas,curdat,slp_mx)
C     =================================================================
      IMPLICIT NONE
C     SLOPE FILLS COLUMN 3 OF CURDAT WITH THE SLOPE DY/DF FOR
C     ALL TURNS FROM 1 TO PAR(2): DY/DF_N = Y_N(F_E)-Y_N(F_B)/(F_E-F_B)
      REAL*8  rad,curdat(3,50),fas(3),slp_mx
      INTEGER n,par(6)
      rad = DATAN(1.0D0)/45.0D0
C     RESET COLUMN 3 OF CURDAT:
      DO 10 n= 1,par(2)
        curdat(3,n) = 0.0D0
10    CONTINUE
C     INITIAL VALUE OF SLP_MAX:
      slp_mx = 0.0D0
      DO 20 n = 1,par(2)
C     IF 1 OR BOTH Y_VALUES ARE <= 0, THE CURRENT TURN
C     DOES NOT CONTRIBUTE TO THIS PHASE INTERVAL AND THE CORRESPONDING
C     SLOPE REMAINS 0:
      IF (curdat(1,n).LE.0.0D0) GO TO 20
      IF (curdat(2,n).LE.0.0D0) GO TO 20
C     CALCULATE SLOPE:
      curdat(3,n) = (curdat(2,n)-curdat(1,n))/fas(3)
      slp_mx = DMAX1(DABS(curdat(3,n)),slp_mx)
```

```
20    CONTINUE
C     IF SLP_MAX < 1.0D-3 THEN SET SLP_MAX = 1.0D-3 BECAUSE VALUE ZERO
C     WILL CAUSE PROBLEMS IN THE CALCULATION OF DF:
      slp_mx = DMAX1(1.0D-3,slp_mx)
      RETURN
      END
C     =================================================================
      SUBROUTINE evaldr(par,fas,df,ndf,dr,i_mx,i_mn,
     +                  rdat,ddat,diaf,diaf_ok,di_r,ib_tot)
C     =================================================================
      IMPLICIT NONE
      REAL*8 fas(3),df,dr,rdat(3,50),ddat(3,50),di_r(2000),
     +       diaf(4),f_m,r_m,rdiaf_m,relf_m,ib_f,ib_tot
      INTEGER n,n_p,par(6),ndf,i_f,i_r,p_i_r(50),i_mx,i_mn

      LOGICAL diaf_ok
      DO 10 i_f = 1,ndf
C       RESET ARRAY P_I_R:
        DO 15 n_p = 1,par(2)
          p_i_r(n_p) = 0
15      CONTINUE
C       CENTRE INTERVAL DF WITH RESPECT TO FAS(1):
        relf_m = (i_f-0.5D0)*df
C       CENTRE INTERVAL DF (ABSOLUTE):
        f_m = fas(1) + relf_m
C       KEEP TRACK OF TOTAL INTERNAL CURRENT:
        ib_tot = ib_tot + ib_f(f_m)*df
        DO 20 n = 1,par(2)
c        DO 20 n = 25,28

          IF ((rdat(1,n).LE.0.0D0).OR.(rdat(2,n).LE.0.0D0))
     +    GO TO 20
C         IF A DIAPHRAGM IS PRESENT, CHECK WETHER IT INTERCEPTS
C         INTERNAL CURRENT. IF SO, THEN NO MORE CONTRIBUTIONS
C         FROM THIS AND FOLLOWING TURNS TO THE INTERNAL CURRENT:
          IF (diaf_ok) THEN
            IF ((ddat(1,n).GT.0.0D0).AND.(ddat(2,n).GT.0.0D0)) THEN
C             RDIAF_M IS CENTRE OF INTERVAL DR(DF) AT AZIMUTH
C             DIAPHRAGM FOR TURN N:
              rdiaf_m = ddat(1,n) + ddat(3,n)*relf_m
         IF ((rdiaf_m.GE.diaf(1)).AND.(rdiaf_m.LE.diaf(2))) GO TO 10
         IF ((rdiaf_m.GE.diaf(3)).AND.(rdiaf_m.LE.diaf(4))) GO TO 10
            END IF
          ENDIF
C         CENTRE OF INTERVAL DR(DF) AT TURN N:
          r_m = rdat(1,n) + rdat(3,n)*relf_m
C         CALCULATE ARRAY-INDEX I_R OF DI_R FOR R_M:
          i_r = INT(r_m/dr) + 1
          i_mx = MAX(i_r,i_mx)
          i_mn = MIN(i_r,i_mn)
C         COMPARE I_R(N) WITH THE VALUES P_I_R(1,N-1)
C         OF THE PRECEEDING TURNS.
C         IF THE PHASE-INTERVAL DF(IF) HAS ALREADY CONTRIBUTED TO THE
C         CURRENT IN INTERVAL I_R, THE CURRENT FRACTION IB_F(DF)
C         MUST NOT BE ADDED AGAIN, TO AVOID DOUBLE-COUNTING:
          DO 30 n_p = 1,n - 1
            IF (i_r.EQ.p_i_r(n_p)) THEN
              GO TO 20
            END IF
30        CONTINUE
          di_r(i_r) = di_r(i_r) + ib_f(f_m)*df
          p_i_r(n) = i_r
20      CONTINUE
10    CONTINUE
      RETURN
      END
C     =================================================================
```

```
      SUBROUTINE e_range(par,df,ndf,rdat,edat,ddat,
     +                   diaf,diaf_ok,mx_e,mn_e,roi_mx,roi_mn)
C     =================================================================
      IMPLICIT NONE
      REAL*8 df,rdat(3,50),edat(3,50),ddat(3,50),diaf(4),
     +       r_m,rdiaf_m,e_m,relf_m,mx_e,mn_e,roi_mx,roi_mn
      INTEGER n,par(6),ndf,i_f
      LOGICAL diaf_ok
      DO 10 i_f = 1,ndf
C       CENTRE OF INTERVAL DF WITH RESPECT TO FAS(1):
        relf_m = (i_f-0.5D0)*df
        DO 20 n = 1,par(2)
c         DO 20 n = 25,28
          IF ((rdat(1,n).LE.0.0D0).OR.(rdat(2,n).LE.0.0D0))
     +    GO TO 20
C         IF A DIAPHRAGM IS PRESENT, CHECK WETHER IT INTERCEPTS
C         INTERNAL CURRENT. IF SO, THEN NO MORE CONTRIBUTIONS
C         FROM THIS AND FOLLOWING TURNS TO THE INTERNAL CURRENT:
          IF (diaf_ok) THEN
            IF ((ddat(1,n).GT.0.0D0).AND.(ddat(2,n).GT.0.0D0)) THEN
C             RDIAF_M IS CENTRE OF INTERVAL DR(DF) AT AZIMUTH
C             DIAPHRAGM FOR TURN N:
              rdiaf_m = ddat(1,n) + ddat(3,n)*relf_m
          IF ((rdiaf_m.GE.diaf(1)).AND.(rdiaf_m.LE.diaf(2))) GO TO 10
          IF ((rdiaf_m.GE.diaf(3)).AND.(rdiaf_m.LE.diaf(4))) GO TO 10
            END IF
          ENDIF
C         CENTRE OF INTERVAL DR(DF) AT TURN N:
          r_m = rdat(1,n) + rdat(3,n)*relf_m
C         IF ROI_MN <= R_M <= ROI_MX THEN N IS THE LOWEST TURN
C         FOR WHICH PHASE INTERVAL DF(I_F) CONTRIBUTES TO THE CURRENT
C         IN THE ROI:
          IF ((r_m.GE.roi_mn).AND.(r_m.LE.roi_mx)) THEN
C           CALCULATE ENERGY CORRESPONDING TO INITIAL PHASE F_M:
            e_m = edat(1,n) + edat(3,n)*relf_m
            mx_e = DMAX1(e_m,mx_e)
            mn_e = DMIN1(e_m,mn_e)
            GO TO 10
          END IF
20      CONTINUE
10    CONTINUE
      RETURN
      END
C     =================================================================
      SUBROUTINE evalde(par,fas,df,ndf,de,av_e,i_mx,i_mn,
     +      roi_mx,roi_mn,rdat,edat,ddat,diaf,diaf_ok,di_e,ib_tot)
C     =================================================================
      IMPLICIT NONE
      REAL*8 fas(3),df,de,av_e,rdat(3,50),edat(3,50),ddat(3,50),
     +       diaf(4),di_e(2000),
     +       f_m,r_m,rdiaf_m,e_m,relf_m,ib_f,roi_mx,roi_mn,ib_tot
      INTEGER n,par(6),ndf,i_f,i_e,i_mx,i_mn
      LOGICAL diaf_ok
      DO 10 i_f = 1,ndf
C       CENTRE OF INTERVAL DF WITH RESPECT TO FAS(1):
        relf_m = (i_f-0.5D0)*df
C       CENTRE INTERVAL DF (ABSOLUTE):
        f_m = fas(1) + relf_m
C       KEEP TRACK OF TOTAL INTERNAL CURRENT:
        ib_tot = ib_tot + ib_f(f_m)*df
        DO 20 n = 1,par(2)
c        DO 20 n = 25,28
          IF ((rdat(1,n).LE.0.0D0).OR.(rdat(2,n).LE.0.0D0))
     +    GO TO 20
C         IF A DIAPHRAGM IS PRESENT, CHECK WETHER IT INTERCEPTS
C         INTERNAL CURRENT. IF SO, THEN NO MORE CONTRIBUTIONS
C         FROM THIS AND FOLLOWING TURNS TO THE INTERNAL CURRENT:
```

```
          IF (diaf_ok) THEN
            IF ((ddat(1,n).GT.0.0D0).AND.(ddat(2,n).GT.0.0D0)) THEN
C             RDIAF_M IS CENTRE OF INTERVAL DR(DF) AT AZIMUTH
C             DIAPHRAGM FOR TURN N:
              rdiaf_m = ddat(1,n) + ddat(3,n)*relf_m
          IF ((rdiaf_m.GE.diaf(1)).AND.(rdiaf_m.LE.diaf(2))) GO TO 10
          IF ((rdiaf_m.GE.diaf(3)).AND.(rdiaf_m.LE.diaf(4))) GO TO 10
            END IF
          ENDIF
C         CENTRE OF INTERVAL DR(DF) AT TURN N:
          r_m = rdat(1,n) + rdat(3,n)*relf_m
C         IF ROI_MN <= R_M <= ROI_MX THEN N IS THE LOWEST TURN
C         FOR WHICH PHASE INTERVAL DF(I_F) CONTRIBUTES TO THE CURRENT
C         IN THE ROI:
          IF ((r_m.GT.roi_mn).AND.(r_m.LT.roi_mx)) THEN
C           CALCULATE ENERGY CORRESPONDING TO INITIAL PHASE F_M:
            e_m = edat(1,n) + edat(3,n)*relf_m
C           CALCULATE ARRAY-INDEX I_E FOR E_M
C           THE INDEX IS CHOSEN SUCH THAT I_E = 1000 IS THE ENERGY
C           INTERVAL AV_E - 0.5*DE < E_M < AV_E + 0.5*DE:
            i_e = NINT((e_m-av_e)/de) + 1000
            i_mx = MAX(i_e,i_mx)
            i_mn = MIN(i_e,i_mn)
C           N IS THE LOWEST TURN FOR WHICH PHASE F_M CONTRIBUTES TO THE
C           CURRENT IN THE ROI, SO ADD IB_F(F_M) TO I_E:
            di_e(i_e) = di_e(i_e) + ib_f(f_m)*df
            GO TO 10
          END IF
20      CONTINUE
10    CONTINUE
      RETURN
      END
C     =================================================================
      REAL*8 FUNCTION ib_f(f)
C     =================================================================
C     CALCULATION OF THE SOURCE CURRENT AS A FUNCTION OF INITIAL PHASE
C     IB_F(F) =K*(VD(F)**3/2)
C     K = CONSTANT, VD(F) = DEE-VOLTAGE
C     F DEPENDS ON THE INITIAL AZIMUTH TH_0 OF THE PARTICLE
C     ACCORDING TO  F = H*(W0*T-TH_0). H = HARMONIC NUMBER = 2
C     FOR PARTICLES LEAVING THE CENTRE OF THE SOURCE-HOLE
C     TH_0 = 42.4 DEG. WHICH IS 0.7400 RADIANS
      REAL*8 th_0,h,f,vd
      h = 2.0D0
      th_0 = 7.400D-1
      vd = DSIN(f+h*th_0)
      IF (vd.GT.0.0D0) THEN
        ib_f = DSQRT(vd**3)
      ELSE
        ib_f = 0.0D0
      END IF
      RETURN
      END
C     =================================================================
       PROGRAM cen10
C     ================================================================
C      CALCULATION OF THE ORBIT-CENTRE MOVEMENT AND ORBIT-PARAMETERS
C      AT GIVEN AZIMUTHAL INTERVAL
C     ================================================================
C     ================================================================
C      LAST UPDATED:          08-06-94
C      BY:                    PETER OP DE BEEK
C     ================================================================
C      UPDATE-HISTORY
C     ================================================================
C     ================================================================
C      SUBROUTINES:
C     ................................................................
C      INCEN10                     VMSCPU
C      ORCEN10                     HF_FASE
C      TUCEN10                     GERK
C      INCHN10      CHN10          FEHL
C      INGAP10      GAP10          E02BAF (EXTERN, NAG-ROUTINE)
C      INRLX10      RLX10          E02BBF (EXTERN, NAG-ROUTINE)
C      INMAG10      MAG10
c
C      EQOMF
C      EFLD10       MFLD10
C      TIME (EXTERN, INTRINSIC)    DATE (EXTERN, INTRINSIC)
C     ================================================================
C      FUNCTIONS:
C     ................................................................
C      DISLINE
c
C     ================================================================
C      MAIN PROGRAM:
C     ................................................................
      CALL incen10
      CALL orcen10

      STOP 'PROGRAMMA KLAAR'

      END

C     ================================================================

      SUBROUTINE incen10
C     ................................................................
C     INCEN10 INITIALISATION OF THE ORBIT CALCULATION
C     ................................................................

      IMPLICIT NONE
      INTEGER   fnum,funit,
     +          h,z,i,
     +          ngh,hh(10),imax,jmax,
     +          n_max,hfpst,lharm,hharm
      REAL*8    c,el,pi,rad,
     +          f0,d_f0,b_iso,w0,cnstnt,vdee,v6v2,fi1,fi3,delta,
     +          mrust,intrv,
     +          xst,yst,alfast,e0,d1xst,d1yst,rr1,rr3,
     +          q,v0,abserr,relerr,r_max,d_dee,
     +          b22_m,r22_0,w22_h,b2b_m,r2b_0,w2b_h,
     +          ri(10),ra(10),t(10),yc(10),
     +          hx,hy,hz,emap(1:201,1:281,3),
     +          xmin1,xmax1,ymin1,ymax1

      LOGICAL   kan,twochan,cor22,cor2b
      CHARACTER*8  systime
      CHARACTER*9  sysdate
      CHARACTER*80 comm
      CHARACTER*7  file_out
      CHARACTER*40 namemag,namekan,nameef
      CHARACTER*40 files(8)
      COMMON/natconst/c,el,pi,rad
      COMMON/integr/  intrv,abserr,relerr,r_max,n_max

      COMMON/cycl/    f0,b_iso,w0,cnstnt,vdee,v6v2,
     +                fi1,fi3,delta,h
      COMMON/part/ mrust,z
      COMMON/baan/    xst,yst,alfast,e0,d1xst,d1yst,rr1,rr3,hfpst
      COMMON/answ/    kan,twochan,cor22,cor2b
      COMMON/harms/lharm,hharm
      COMMON/relax/xmin1,xmax1,ymin1,ymax1,hx,
     +                hy,hz,emap,imax,jmax
      COMMON/deltadee/d_dee
      COMMON/firstharm/b22_m,r22_0,w22_h,b2b_m,r2b_0,w2b_h
      COMMON/iofiles/ files
C     ................................................................
C     DETERMINATION OF FILE-NAMES
C     ................................................................
C     FNUM CONNECTS FILENAME TO UNIT-SPECIFIER ACCORDING TO
C             FILENAAM = FILES(FNUM)
C             unit = FUNIT = FNUM + 10
      DATA files / '[-.input]cen10.dat',
     +             'user2:[tnndrr.ilec]m1sft.dat',
     +             'user2:[tnndrr.ilec]mchan.dat',
     +             'user2:[tnndrr.ilec]erelx2.dat',
     +             'user2:[tnndrr.ilec]exyz1.dat',
     +             '[-.output]cen10.doc',
     +             '[-.output]cen10.nxy',
     +             '[-.output]cen10.cen'/
C     NAMES CEN10.NXY EN CEN10.DOC ARE ADJUSTED, ACCORDING TO NAME
C     READ FROM CEN10.DAT
C     ................................................................

C 1.   INTORXX.DAT  INPUTFILE, PARAM. PROGRAM           (INIT1)
C 2.   MAGN1S.DAT   DATA MAGNETFIELD                    (MAGNET0)
C 3.   MCHAN.DAT    GEOMETRY MAGNETIC CHANNEL           (INCHN10)
C 4.   ERELX2.DAT   DATA ELEKTR. FIELD                   (INRLX10)
C 5.   EXYZ1.DAT    GAP GEOMETRY                       (INGAP10)
C 6.   XXXXXXX.DOC  UITPUTFILE, DOCUMENTATION WITH OUTPUT FILES
C 7.   XXXXXXX.NXY  UITPUTFILE, CONTAINS NUMERICAL OUTPUT
C                   ORBIT COORDINATES
C 8.   XXXXXXX.CEN  UITPUTFILE, CONTAINS NUMERICAL OUTPUT
C                   ORBIT CENTRE COORDINATES
C     ================================================================

C     ................................................................
C     NATCONST
C     ................................................................
      c = 2.99792458D8
      el = 1.60217733D-19
      pi = 4.*DATAN(1.0D0)
      rad = pi/180.
C     ================================================================

C     ................................................................
C     DETERMINE SYSTEM TIME AND DATE
C     ................................................................
      CALL time (systime)
      CALL date (sysdate)
C     ================================================================

C     ................................................................
C     READ PROGRAM PARAMETERS
C     ................................................................
```

```
      fnum = 1
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='old')
      REWIND(funit)
      READ(funit,'(A80)') comm
C                 : NAME OUTPUTFILE (7 CHAR., NO BLANCS)
      READ(funit,'(A7)') file_out
C                 : OUTPUT INTERVAL IN W0*T
      READ(funit,*) intrv
C                 : HIGHEST TURN IN .XYR-FILE  (<= 40)
      READ(funit,*) n_max
C                 : MAXIMUM RADIUS
      READ(funit,*) r_max
C                 : INITIAL PHASE
      READ(funit,*) hfpst
C                 : ABSOLUTE ERROR
      READ(funit,*) abserr
C                 : RELATIEVE ERROR
      READ(funit,*) relerr
C                 : DEVIATION FROM F_0 [PROMILLE]
      READ(funit,*) d_f0
C                 : INITIAL ENERGY (KEV)
      READ(funit,*) e0
C                 : X-COORD. STARTING POINT
      READ(funit,*) xst
C                 : Y-COORD. STARTING POINT
      READ(funit,*) yst
C                 : INITIAL ANGLE WITH POS. X AXIS
      READ(funit,*) alfast
C                 : RELAXPARAMETERS
      READ(funit,*) xmin1
      READ(funit,*) xmax1
      READ(funit,*) ymin1
      READ(funit,*) ymax1
      READ(funit,*) imax
      READ(funit,*) jmax
C                 : B-FIELD, FOURIER HARMONICS
      READ(funit,*) hharm
C                 : HARM.  1*SYM...HHARM*SYM IDEAL SYMMETRICAL FIELD
      READ(funit,*) lharm
C                 : LOWER HARM. 1...LHARM ADD TO IDEAL FIELD
      READ(funit,*) kan
C                 : MAGNETIC CHANNEL (FALSE/TRUE)
      READ(funit,*) twochan
C                 : DUMMY-CHANNEL (FALSE/TRUE)
C     1E-HARM CORRECTION COILS 22 AND 2B
C     ================================
      READ(funit,*) cor22
C                 : COIL 22 ON (FALSE/TRUE)
      READ(funit,*) b22_m
C                 : AMPLITUDE GAUSS-PROFILE COIL 22 (T)
      READ(funit,*) r22_0
C                 : RADIUS B(R) = B_MAX COIL 22  (M)
      READ(funit,*) w22_h
C                 : HALF-VALUE WIDT (M)
      READ(funit,*) cor2b
C                 : COIL 2B ON  (FALSE/TRUE)
      READ(funit,*) b2b_m
C                 : AMPLITUDE GAUSS-PROFILE COIL 2B (T)
      READ(funit,*) r2b_0
C                 : RADIUS B(R) = B_MAX COIL 2B (M)
      READ(funit,*) w2b_h
C                 : HALF-VALUE WIDTH (M)

C     CYCLOTRON-PARAMETERS:
C     ====================
```

```
C                 :HARMONIC NUMBER
      READ(funit,*) h
C                 :FIRST HARMONIC RF-FREQUENCY (MHZ)
      READ(funit,*) f0
C                 : DEE-VOLTAGE (KV)
      READ(funit,*) vdee
C                 : REL. DEVIATION 2-E HARM. DEE ALONG POS.
C                 : Y-AXIS FROM DEE-VOLTAGE (%)
      READ(funit,*) d_dee
C                 : PROPORTION 6/2-HARM. DEE-VOLTAGE
      READ(funit,*) v6v2
C                 : ADDITIONAL PHASE-SHIFT FLATTOP VOLTAGE
      READ(funit,*) delta
C                 :PARTICLE CHARGE IN EL
      READ(funit,*) z
C                 : RESTMASS (KG)
      READ(funit,*) mrust

      CLOSE(funit)

C     ................................................................
C     READ MAGNETIC FIELD
C     ................................................................
      fnum = 2
      CALL inmag10(fnum,namemag,cor22,cor2b)
C     ................................................................
C     ................................................................
C     READ DATA MAGNETIC CHANNEL
C     ................................................................
      fnum = 3
      IF (kan) CALL inchn10(fnum,namekan)

C     ................................................................
C     FILENAMES OUTPUT-FILES
C     ................................................................
      files(6) =  '[-.output]'//file_out//'.doc'
      files(7) =  '[-.output]'//file_out//'.nxy'
      files(8) =  '[-.output]'//file_out//'.cen'

C     ................................................................
C     ................................................................



      vdee = vdee*1.E3

      delta = delta*rad
      intrv = intrv*rad
      alfast = alfast*rad


      q = z*el
      f0 = f0*(1 + d_f0/1000.0D0)*1.0D6
C     D_F0 = AFIJKING VAN F0 UIT INPUT-FILE IN PROMILLES
      d_dee = d_dee/100.0D0

      w0 = f0*2.*pi
      b_iso = mrust*w0/q


      cnstnt = (mrust*w0*w0)/(2.*el)
      v0 = DSQRT(1.e3*e0/cnstnt)
      d1xst = v0*DCOS(alfast)
      d1yst = v0*DSIN(alfast)
      rr1 = 2.*mrust*vdee/(q*b_iso*b_iso)
      rr3 = v6v2*rr1

C     ...............................................................
```

```
C      WRITE.DOC FILE
C     ...............................................................
      fnum = 6
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='unknown')
      WRITE (funit,2300) 'UITVOER BIJ PROGRAMMA CEN10.FOR'
      WRITE (funit,2400) 'NAAM UITVOERFILE:',file_out
      WRITE (funit,2300) '================================'
      WRITE(funit,*)     'DATUM: ',sysdate
      WRITE(funit,*)     'TIJD : ',systime
      WRITE (funit,2300) '================================'
      WRITE (funit,2300) 'INTEGRATIE-PARAMETERS'
      WRITE (funit,2300) '====================='

      WRITE (funit,2300) 'Fehlberg routine, variabele stap'
      WRITE (funit,2100) 'absolute fout                  ',abserr
      WRITE (funit,2100) 'relatieve fout                 ',relerr
      WRITE (funit,2200) 'uitvoer-interval [deg]         ',intrv/rad
      WRITE (funit,2300) 'BAAN-PARAMETERS'
      WRITE (funit,2300) '==============='
      WRITE (funit,2000) 'hoogste omloop                 ',n_max
      WRITE (funit,2200) 'maximale straal                ',r_max
      WRITE (funit,2300) 'CYCLOTRON-PARAMETERS'
      WRITE (funit,2300) '===================='
      WRITE (funit,2000) 'harmonisch_getal               ',h
      WRITE (funit,2100) 'isochroon B-veld [T]           ',b_iso
      WRITE (funit,2200) 'omloop freq. [Mhz]             ',f0/1.E6
      WRITE (funit,2200) 'afw van ingelezen f0 [prom]    ',d_f0
      WRITE (funit,2200) 'dee spanning [kV]              ',vdee/1.E3
      WRITE (funit,2200) 'afw. dee spanning y-dee (%)    ',d_dee*100.
      WRITE (funit,2200) 'verhouding 6/2 deesp.          ',v6v2
      WRITE (funit,2200) 'fasev. 6-2 [deg]               ',delta
      WRITE (funit,2300) 'PARAMETERS DEELTJE'
      WRITE (funit,2300) '=================='
      WRITE (funit,2200) 'start energie [keV]            ',e0
      WRITE (funit,2100) 'x-coord. startp. [m]           ',xst
      WRITE (funit,2100) 'y-coord. startp. [m]           ',yst
      WRITE (funit,2200) 'starthoek pos. x-as [deg]      ',alfast/rad
      WRITE (funit,2000) 'startfase [deg]                ',hfpst

      WRITE (funit,2300) 'MAFNEETVELD-SETTINGS'
      WRITE (funit,2300) '===================='
      WRITE (funit,*) 'harmonischen *sym t/m         ',hharm
      WRITE (funit,*) 'lagere harmonischen t/m        ',lharm
      WRITE (funit,*) 'magn. kanaal                   ',kan
      WRITE (funit,*) 'dummy kanaal                   ',twochan
      WRITE (funit,*) '1e-harm spoel 22               ',cor22
      WRITE (funit,2100) 'amplitude gauss-profiel (T)    ',b22_m
      WRITE (funit,2100) 'straal B(r) = B_max  (m)       ',r22_0
      WRITE (funit,2100) 'halfwaarde-breedte (m)         ',w22_h
      WRITE (funit,*) '1e-harm spoel 2b               ',cor2b
      WRITE (funit,2100) 'amplitude gauss-profiel (T)    ',b2b_m
      WRITE (funit,2100) 'straal B(r) = B_max  (m)       ',r2b_0
      WRITE (funit,2100) 'halfwaarde-breedte (m)         ',w2b_h
      CLOSE(funit)


C     ===============================================================
C     ...............................................................
C     READ DATA ELECTRIC FIELD
C     ...............................................................
      fnum = 5
      CALL ingap10(fnum,nameef,ngh,ri,ra,t,yc,hh,fi1,fi3)

      fnum = 4
      CALL inrlx10(fnum,emap,hx,hy,hz,imax,jmax)
```

```
      RETURN
C     ...............................................................
C     FORMATS
C     ...............................................................

2000  FORMAT(1X,A30,I3)
2100  FORMAT(1X,A30,D16.6)
2200  FORMAT(1X,A30,F8.3)
2300  FORMAT(1X,A40)
2400  FORMAT(1X,A30,A7)




C     ...............................................................
C     ERRORLABELS
C     ...............................................................

1000   WRITE(6,*) 'fout tijdens openen file ' ,files(fnum)
       STOP 'fout tijdens uitvoer i/o-statement!'
       END

C     ===============================================================
      SUBROUTINE orcen10
C     ...............................................................

      IMPLICIT NONE
      INTEGER    fnum,funit,n_max

      REAL       t_cpu
      REAL*8     intrv,abserr,relerr,r_max
      CHARACTER*40 files(8)


      COMMON/integr/  intrv,abserr,relerr,r_max,n_max
      COMMON/iofiles/ files
      CALL vmscpu(t_cpu)
C     OPEN .XYR-FILE:
      fnum = 7
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='unknown')
      WRITE(funit,*)'n x y th hfp th_v e_k'
      fnum = 8
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='unknown')
      WRITE(funit,*)'n x_c y_c '
      CALL tucen10

      CALL vmscpu(t_cpu)
      fnum = 7
      funit = fnum + 10
      CLOSE(funit)
      fnum = 8
      funit = fnum + 10
      CLOSE(funit)

C     ...............................................................
C     ERRORLABELS
C     ...............................................................
      RETURN
1000  WRITE(6,*) 'fout tijdens openen file ' ,files(fnum)
      STOP 'fout tijdens uitvoer i/o-statement!'
      END
```

```
C     ================================================================


      SUBROUTINE tucen10
C     ................................................................

      IMPLICIT NONE
      INTEGER   fnum,funit,hfpst,
     +          h,oml_t,z,n_max,naz_u,
     +          dim,i,n,iflag,
     +          nfe,kop,init,jflag,kflag

      REAL*8    c,el,pi,pi_2,rad,
     +          f0,b_iso,w0,cnstnt,vdee,v6v2,fi1,fi3,delta,
     +          mrust,intrv,r_max,az_u(4),x_c,y_c,
     +          xst,yst,alfast,e0,d1xst,d1yst,rr1,rr3,
     +          x(4),r,th,th_v,hfp,t,t2,d_th,err_az,r_i_n(4,40),
     +          relerr,abserr,gerror(4),e_kin,err_r,err_e,
     +          yp(4),f1(4),f2(4),f3(4),f4(4),f5(4),
     +          yg(4),ygp(4),h_s,savre,savae


      LOGICAL   kan,twochan,cor22,cor2b,
     +          az_neg,th_neg(4)

      CHARACTER*40 files(8)

      EXTERNAL eqomf

      COMMON/natconst/c,el,pi,rad
      COMMON/integr/  intrv,abserr,relerr,r_max,n_max
      COMMON/cycl/    f0,b_iso,w0,cnstnt,vdee,v6v2,
     +                fi1,fi3,delta,h

      COMMON/part/ mrust,z
      COMMON/baan/    xst,yst,alfast,e0,d1xst,d1yst,rr1,rr3,hfpst
      COMMON/answ/    kan,twochan,cor22,cor2b
      COMMON/iofiles/ files
      COMMON/gerksave/yp,h_s,f1,f2,f3,f4,f5,yg,ygp,savre,savae,
     +                nfe,kop,init,jflag,kflag


C     ................................................................
C                                STARTCONDITIES
C     ................................................................

C     ................................................................
C     ................................................................
C     CHECK N_MAX <= 40 W.R.T DIMENSION R_I_N:
      IF (n_max.GT.40) STOP'DIMENSIE ARRAY R_I_N AANPASSEN!'

      pi_2 = 2.0D0*pi

      naz_u = 4
      az_u(1) = -0.5D0*pi
      az_u(2) = 0.0D0
      az_u(3) = 0.5D0*pi
      az_u(4) = -1.0D0*pi
C     ERROR ERR_AZ IN DETERMINATION VAN AZ_U(I) MAY BE  2 DEG.:
      err_az = 2.0D0*rad
      dim = 4
      oml_t = 1
```

```
C     INITIAL VALUES X,Y,V_X,V_Y:
      x(1) = xst
      x(3) = yst
      x(2) = d1xst
      x(4) = d1yst

      az_neg = .FALSE.
      DO 50 i = 1,naz_u
        th_neg(i) = .FALSE.
50    CONTINUE

C     INITIAL RADIUS
      r=DSQRT(x(1)**2+x(3)**2)


      th=DATAN2(x(3),x(1))
      IF (th.LT.0.D0) az_neg = .TRUE.

      DO 55 i = 1,naz_u-1

        IF (th.LT.az_u(i)) th_neg(i) = .TRUE.

55    CONTINUE
C     SPECIAL CASE I = 4:

      IF (th.GT.az_u(3)) th_neg(4) = .TRUE.


      hfp = hfpst*rad
C     INITIAL VALUE HIGH-FREQUENCY PHASE
      t = hfp/h+th
      th_v = DATAN2(x(4),x(2))


      e_kin = mrust*w0*w0*(x(2)**2+x(4)**2)/(2.0D3*el)

C     ................................................................
      fnum = 7
      funit = fnum + 10
      WRITE(funit,1000)oml_t,x(1)*100.0,x(3)*100.0,
     + th/rad,hfp/rad,th_v/rad,e_kin

      WRITE(6,*)'OMLOOP',oml_t

      iflag = 1
      DO WHILE (oml_t.LE.n_max)
         t2 = t + intrv

100      CALL gerk(eqomf,dim,x,t,t2,relerr,abserr,iflag,gerror)
         IF (iflag.NE.2) THEN
                 IF (iflag.EQ.6) THEN
                   iflag = 2
                   WRITE(6,*)'INTERVAL TE KLEIN '
                   GOTO 100
                                ELSE
                   WRITE(6,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   WRITE(funit,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   STOP
                 END IF
         ENDIF

         t = DMOD(t,2.0D0*pi)

         th = DATAN2(x(3),x(1))

         r=DSQRT(x(1)**2+x(3)**2)
         IF (r.GT.r_max) GO TO 400


         IF (th.LT.0.D0)  az_neg = .TRUE.
         IF ((th.GE.0.D0).AND.(az_neg)) THEN
              oml_t = oml_t +1
              az_neg = .FALSE.
              WRITE(6,*)'OMLOOP',oml_t
         ENDIF

         CALL hf_fase(t,th,hfp,h,pi,pi_2)
         e_kin = mrust*w0*w0*(x(2)**2+x(4)**2)/(2.0D3*el)
         th_v = DATAN2(x(4),x(2))

         WRITE(funit,1000)oml_t,x(1)*100.0,x(3)*100.0,
     +    th/rad,hfp/rad,th_v/rad,e_kin



         DO 200 i = 1,naz_u-1
         IF (th.LT.az_u(i)) th_neg(i) = .TRUE.
         IF ((th.GT.az_u(i)).AND.(th_neg(i))) THEN
           th_neg(i) = .FALSE.
           d_th = th-az_u(i)
           DO WHILE (DABS(d_th).GE.err_az)
             t2 = t - d_th
300          CALL gerk(eqomf,dim,x,t,t2,relerr,abserr,iflag,gerror)
             IF (iflag.NE.2) THEN
                 IF (iflag.EQ.6) THEN
                   iflag = 2
                   GOTO 300
                                 ELSE
                   WRITE(6,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   WRITE(funit,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   STOP
                 END IF
             ENDIF
             th = DATAN2(x(3),x(1))
             d_th = th-az_u(i)

           END DO
           r=DSQRT(x(1)**2+x(3)**2)
C           ERR_R = DSQRT((X(1)*GERROR(1))**2+
C     +                   (X(3)*GERROR(3))**2)/R

C           E_KIN = MRUST*W0*W0*(X(2)**2+X(4)**2)/(2.0D3*EL)
C           ERR_E = MRUST*W0*W0*DSQRT((X(2)*GERROR(2))**2+
C     +                   (X(4)*GERROR(4))**2)/(1.0D3*EL)
C           CALL HF_FASE(T,TH,HFP,H,PI,PI_2)
C           WRITE(FUNIT,1100)HFPST,OML_T,I,TH/RAD,R,ERR_R,
C      +                      HFP/RAD,E_KIN,ERR_E
            r_i_n(i,oml_t) = r
            WRITE(6,3000) r*100.,i,th/rad



         END IF
200      CONTINUE

         i = 4
         IF (th.GT.az_u(3)) th_neg(i) = .TRUE.
         IF ((th.GT.az_u(i)).AND.(th.LT.az_u(2)).AND.(th_neg(i))) THEN
           th_neg(i) = .FALSE.
           d_th = th-az_u(i)

           DO WHILE (DABS(d_th).GE.err_az)
             t2 = t - d_th
350          CALL gerk(eqomf,dim,x,t,t2,relerr,abserr,iflag,gerror)
             IF (iflag.NE.2) THEN
                 IF (iflag.EQ.6) THEN
                   iflag = 2
                   GOTO 350
                                 ELSE
                   WRITE(6,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   WRITE(funit,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   STOP
                 END IF
             ENDIF
             th = DATAN2(x(3),x(1))
             IF (th.GT.0.0D0) THEN
                d_th = th + az_u(i)
             ELSE
                d_th = th - az_u(i)
             END IF

           END DO
           r=DSQRT(x(1)**2+x(3)**2)
C           ERR_R = DSQRT((X(1)*GERROR(1))**2+
C     +                   (X(3)*GERROR(3))**2)/R

C           E_KIN = MRUST*W0*W0*(X(2)**2+X(4)**2)/(2.0D3*EL)
C           ERR_E = MRUST*W0*W0*DSQRT((X(2)*GERROR(2))**2+
C     +                   (X(4)*GERROR(4))**2)/(1.0D3*EL)
C           CALL HF_FASE(T,TH,HFP,H,PI,PI_2)
C           WRITE(FUNIT,1100)HFPST,OML_T,I,TH/RAD,R,ERR_R,
C      +                      HFP/RAD,E_KIN,ERR_E
            r_i_n(i,oml_t) = r
            WRITE(6,3000) r*100.,i,th/rad


         END IF

      END DO

400   IF (oml_t.LE.n_max) THEN
         WRITE(6,*)'MAXIMALE STRAAL R_MAX OVERSCHREDEN'
      ELSE
         WRITE(6,*)'MAXIMALE OMLOOP N_MAX OVERSCHREDEN'
      END IF
C     CALCULATE X- AND Y-COORD. ORBIT CENTRE, AND WRITE TO FILE:
      fnum = 8
      funit = fnum + 10
      DO 550 n = 3,oml_t-2
         x_c = (r_i_n(2,n)+3.0D0*r_i_n(2,n+1)
     +          -3.0D0*r_i_n(4,n)-r_i_n(4,n+1))/8.0D0
         y_c = (r_i_n(3,n)+3.0D0*r_i_n(3,n+1)
     +          -3.0D0*r_i_n(1,n)-r_i_n(1,n+1))/8.0D0
         WRITE(funit,2000)n,x_c*100.,y_c*100.,r_i_n(2,n)*100.
550   CONTINUE

      RETURN
1000  FORMAT(1X,I4,2E14.5,3F8.2,E14.5)
2000  FORMAT(1X,I4,3E13.4)
3000  FORMAT(1X,'r (cm) ',F8.2,'   i = ',I3,'   th = ',F8.2)
      END


C     ===============================================================
      INCLUDE 'user2:[tnndrr.ilec.source]txlib10.for'
C     ===============================================================
C
C     ===============================================================
C     EXAMPLE OF THE INPUT FILE CEN10.DAT
C   .................................................................
```

```
C    PARAMETERS BAANBEREKENINGS-PROGRAMMA
C    dri3004              : naam outputfile (7 char. lang, geen spaties)
C    10.                  : integratie interval
C    30                   : hoogste omloop (<= 40)
C    0.19                 : maximale straal (cm)
C    13.78                : startfase
C    1.0E-16              : absolute fout
C    1.0E-7               : relatieve fout
C    -1.0D0               : afwijking omlfreq. van ingelezen f0 [promilles]
C    0.54369E3            : start-energie (keV)
C    7.1894E-2            : x-coord. startpunt
C    1.1297E-2            : y-coord. startpunt
C    98.72                : starthoek met pos. x_as
C    -4.E-2               : xmin1
C    4.E-2                : xmax1
C    -6.E-2               : ymin1
C    6.E-2                : ymax1
C    201                  : imax
C    281                  : jmax
C    4                    : aantal ideale harmon *sym
c    0                    : aantal niet-symmetrische harmonischen
C    TRUE                 : magn. kanaal
C    TRUE                 : dummy-kanaal
C    FALSE                : spoel 22 aan/uit
C    -3.5E-3              : amplitude gauss-profiel (T)
C    0.12                 : straal waarbij B(r) = B_max  (m)
C    .0188                : halfwaarde-breedte (m)
C    FALSE                : spoel 2b aan/uit
c    3.4E-3               : amplitude gauss-profiel (T)
C    0.14                 : straal waarbij B(r) = B_max  (m)
C    0.0165               : halfwaarde-breedte (m)
C    2                    : harmonish getal
C    21.67E0              : omloopfrqentie (MHz)
C    30.0                 : deespanning (kV)
C    0.0                  : verschilfactor 2-e h. dee langs pos. y-as (%)
C    0.0                  : verhouding 6/2-harm. deespanning
C    0.0                  : faseverschil flattop-dee
C    1                    : lading deeltje in veelvoud van el
C    1.6726E-27           : rustmassa (kg)
C    ...................................................................
```

```
       PROGRAM rad10
C     ================================================================
C      BEREKENING VAN DE BAANPARAMETERS
C      VAN DE INTERNE BUNDEL OP EEN BEPAALDE WAARDE
C      VAN HET UITVOER-AZIMUTH. DEZE HOEK WORDT GEDEFINIEERD IN DE
C      INPUT-FILE. DE UITVOER-FILES ZIJN BESTEMD ALS INVOER
C      VOOR ANDERE PROGRAMMA'S.
C     ================================================================
C      HET PROGRAMMA MAAKT EEN SCHATTING VAN DE GLOBALE FOUT IN DE
C      BEREKENDE PARAMETERS, DOOR DE INTEGRATIESTAP TE HALVEREN.
C      DE LOKALE FOUT WORDT BEREKEND DOOR VARIATIE VAN ORDE.
C      HET PROGRAMMA MAAKT GEBRUIK VAN DE INTEGRATIEROUTINE GERK,
C      ONTWIKKELD DOOR SHAMPINE & WATTS.
C      ALLE BEREKENINGEN WORDEN IN DOUBLE-PRECISION  UITGEVOERD.
C     ================================================================
C      DATUM LAATSTE UPDATE: 08-06-94
C      DOOR:                 PETER OP DE BEEK
C     ================================================================
C      UPDATE-HISTORY
C     ================================================================
C      SUBROUTINES:
C     ................................................................
C      INRAD10                    VMSCPU
C      ORRAD10                    HF_FASE
C      TURAD10                    GERK
C      INCHN10       CHN10        FEHL
C      INGAP10       GAP10        E02BAF (EXTERN, NAG-ROUTINE)
C      INRLX10       RLX10        E02BBF (EXTERN, NAG-ROUTINE)
C      INMAG10       MAG10
c
C      EQOMF
C      EFLD10        MFLD10
C      TIME (EXTERN, INTRINSIC)   DATE (EXTERN, INTRINSIC)
C     ================================================================
C      FUNCTIONS:
C     ................................................................
C      DISLINE
c
C     ================================================================
C     HOOFDPROGRAMMA:
C     ................................................................
      CALL inrad10
      CALL orrad10

      STOP 'PROGRAMMA KLAAR'

      END

C     ================================================================

      SUBROUTINE inrad10
C     ................................................................
C     INRAD10 INITIALISEERD HET BAANBEREKENINGSPROGRAMMA
C     ................................................................

      IMPLICIT NONE
      INTEGER    fnum,funit,
     +           h,hfpst_min,hfpst_max,stap_hfpst,z,i,
     +           ngh,hh(10),imax,jmax,
     +           n_min,n_max,n_reg,lharm,hharm
      REAL*8     c,el,pi,rad,
     +           f0,d_f0,b_iso,w0,cnstnt,vdee,v6v2,fi1,fi3,delta,
     +           mrust,intrv,
     +           xst,yst,alfast,e0,d1xst,d1yst,rr1,rr3,
     +           q,v0,abserr,relerr,az_u,err_th,r_max,d_dee,
     +           b22_m,r22_0,w22_h,b2b_m,r2b_0,w2b_h,
     +           ri(10),ra(10),t(10),yc(10),
```

```
     +           hx,hy,hz,emap(1:201,1:281,3),
     +           xmin1,xmax1,ymin1,ymax1

      LOGICAL   kan,twochan,cor22,cor2b
      CHARACTER*8  systime
      CHARACTER*9  sysdate
      CHARACTER*80 comm
      CHARACTER*7  file_out
      CHARACTER*40 namemag,namekan,nameef
      CHARACTER*40 files(8)
      COMMON/natconst/c,el,pi,rad
      COMMON/integr/  intrv,az_u,err_th,abserr,relerr

      COMMON/cycl/    f0,b_iso,w0,cnstnt,vdee,v6v2,
     +                fi1,fi3,delta,h
      COMMON/part/ mrust,z
      COMMON/baan/    xst,yst,alfast,e0,d1xst,d1yst,rr1,rr3
      COMMON/answ/    kan,twochan,cor22,cor2b
      COMMON/harms/lharm,hharm
      COMMON/relax/xmin1,xmax1,ymin1,ymax1,hx,
     +                 hy,hz,emap,imax,jmax
      COMMON/rfn/n_min,n_max,n_reg,hfpst_min,hfpst_max,stap_hfpst,r_max


      COMMON/deltadee/d_dee
      COMMON/firstharm/b22_m,r22_0,w22_h,b2b_m,r2b_0,w2b_h
      COMMON/iofiles/ files
C     ...............................................................
C     VASTLEGGEN VAN DE NAMEN VAN DE INPUT- EN OUTPUT FILES
C     ...............................................................

C     DE INTEGER FNUM VERBIND TIJDENS EEN I/O-OPERATIE STEEDS
C     EEN FILENAAM MET EEN UNIT-SPECIFIER VOLGENS:
C             FILENAAM = FILES(FNUM)
C             UNIT = FUNIT = FNUM + 10
      DATA files / '[-.input]rad10.dat',
     +             'user2:[tnndrr.ilec]m1sft.swp',
     +             'user2:[tnndrr.ilec]mchan.dat',
     +             'user2:[tnndrr.ilec]erelx2.dat',
     +             'user2:[tnndrr.ilec]exyz1.dat',
     +             '[-.output]rad10.doc',
     +             '[-.output]rad10.rfn',
     +             '[-.output]rad10.efn'/
C     DE NAMEN RAD10.RFN EN RAD10.DOC WORDEN, AFHANKELIJK VAN
C     DE INGELEZEN NAAM UIT RAD10.DAT, VERDEROP AANGEPAST
C     ...............................................................

C 1.   INTORXX.DAT   INVOERFILE, BEVAT PARAM. PROGRAMMA  (INIT1)
C 2.   MAGN1S.DAT   DATA MAGNEETVELD                    (MAGNET0)
C 3.   MCHAN.DAT    GEOMETRIE MAGN. KANAAL              (INCHN10)
C 4.   ERELX2.DAT   DATA ELEKTR. VELD                   (INRLX10)
C 5.   EXYZ1.DAT    GAP GEOMETRIE                       (INGAP10)
C 6.   XXXXXXX.DOC  UITVOERFILE, BEVAT DE DOCUMENTATIE BIJ
C                   DE UITVOER-FILES
C                   DE NAAM 'XXXXXXX' VAN DE UITVOERFILES WORDT
C                   GEDEFINIEERD IN DE FILE INTERXX.DAT
C 7.   XXXXXXX.NXY  UITVOERFILE, BEVAT DE NUMERIEKE UITVOER
C                   VAN DE BAANCOORDINATEN
C 8.   XXXXXXX.CEN  UITVOERFILE, BEVAT DE NUMERIEKE UITVOER
C                   VAN DE COORD. V/H BAANMIDDELPUNT
C     ===============================================================

C     ...............................................................
C     NATCONST
C     ...............................................................
      c = 2.99792458D8
      el = 1.60217733D-19
      pi = 4.*DATAN(1.0D0)
      rad = pi/180.
C     C:      LICHTSNELHEID IN VACUUM [M/S]
C     EL:     ELEMENTAIRE LADINGSEENHEID [C]
C     PI:     GETAL PI
C     RAD:    AANTAL RADIALEN PER GRAAD
C     ================================================================

C     ................................................................
C     BEPALEN SYSTEEMTIJD EN SYSTEEMDATUM
C     ................................................................
      CALL time (systime)
      CALL date (sysdate)
C     ================================================================

C     ................................................................
C     INLEZEN PROGRAMMA-PARAMETERS VAN INPUT-FILE
C     ................................................................


      fnum = 1
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='old')
      REWIND(funit)
      READ(funit,'(A80)') comm
C                   : NAAM OUTPUTFILE (7 CHAR. LANG, GEEN SPATIES)
      READ(funit,'(A7)') file_out
C                   : UITVOER INTERVAL IN W0*T
      READ(funit,*) intrv
C                   : AZIMUTH WAAROP UITVOER WEGGESCHREVEN MOET WORDEN
      READ(funit,*) az_u
C                   : TOLERANTIE IN UITVOER-AZIMUTH
      READ(funit,*) err_th
C                   : LAAGSTE OMLOOP
      READ(funit,*) n_min
C                   : HOOGSTE OMLOOP  (<= 40)
      READ(funit,*) n_max
C                   : AANT R-WAARDEN PER REGEL IN .RFN-FILE
      READ(funit,*) n_reg
C                   : MAXIMALE STRAAL VOOR UITVOER
      READ(funit,*) r_max
C                   : LAAGSTE STARTFASE
      READ(funit,*) hfpst_min
C                   : HOOGSTE STARTFASE
      READ(funit,*) hfpst_max
C                   : INTERVAL TUSSEN OPEENVOLGENDE STARTFASEN
      READ(funit,*) stap_hfpst
C                   : ABSOLUTE FOUT
      READ(funit,*) abserr
C                   : RELATIEVE FOUT
      READ(funit,*) relerr
C                   : AFWIJKING WERKELIJKE OMLFREQ. VAN
C                   INGELEZEN WAARDE F0 [PROMILLE]
      READ(funit,*) d_f0
C                   : START-ENERGIE (KEV)
      READ(funit,*) e0
C                   : X-COORD. STARTPUNT
      READ(funit,*) xst
C                   : Y-COORD. STARTPUNT
      READ(funit,*) yst
C                   : STARTHOEK MET POS. X_AS
      READ(funit,*) alfast
C                   : RELAXPARAMETERS
      READ(funit,*) xmin1
      READ(funit,*) xmax1
      READ(funit,*) ymin1
      READ(funit,*) ymax1
```

```
      READ(funit,*) imax
      READ(funit,*) jmax
C                 : ELEMENTEN B-VELD
      READ(funit,*) hharm
C                 : HARMON. VAN 1*SYM...HHARM*SYM VOOR OPBOUW IDEALE VELD
      READ(funit,*) lharm
C                 : LAGERE HARMON. VAN 1...LHARM TOEVOEGEN AAN IDEALE VELD
      READ(funit,*) kan
C                 : MAGNETISCH KANAAL (FALSE/TRUE)
      READ(funit,*) twochan
C                 : DUMMY-KANAAL (FALSE/TRUE)
C     1E-HARM CORRECTIE SPOELEN 22 EN 2B
C     ==================================
      READ(funit,*) cor22
C                 : SPOEL 22 AAN (FALSE/TRUE)
      READ(funit,*) b22_m
C                 : AMPLITUDE GAUSS-PROFIEL SPOEL 22 (T)
      READ(funit,*) r22_0
C                 : STRAAL WAARBIJ B(R) = B_MAX SPOEL 22  (M)
      READ(funit,*) w22_h
C                 : HALFWAARDE-BREEDTE (M)
      READ(funit,*) cor2b
C                 : SPOEL 2B AAN  (FALSE/TRUE)
      READ(funit,*) b2b_m
C                 : AMPLITUDE GAUSS-PROFIEL SPOEL 2B (T)
      READ(funit,*) r2b_0
C                 : STRAAL WAARBIJ B(R) = B_MAX SPOEL 2B (M)
      READ(funit,*) w2b_h
C                 : HALFWAARDE-BREEDTE (M)

C     CYCLOTRON-PARAMETERS:
C     =====================

C                 :HARMONISH GETAL
      READ(funit,*) h
C                 :OMLOOPFRQENTIE (MHZ)
      READ(funit,*) f0
C                 : DEESPANNING (KV)
      READ(funit,*) vdee
C                 : REL. AFWIJKING 2-E HARM. DEE LANGS POS.
C                 : Y-AS VAN INGESTELDE DEE-SPANNING (%)
      READ(funit,*) d_dee
C                 : VERHOUDING 6/2-HARM. DEESPANNING
      READ(funit,*) v6v2
C                 : FASEVERSCHIL FLATTOP-DEE
      READ(funit,*) delta
C                 :LADING DEELTJE IN VEELVOUD VAN EL
      READ(funit,*) z
C                 : RUSTMASSA (KG)
      READ(funit,*) mrust

      CLOSE(funit)

C     ...............................................................
C     INLEZEN MAGNEETVELD
C     ...............................................................
      fnum = 2
      CALL inmag10(fnum,namemag,cor22,cor2b)
C     ...............................................................
C     ...............................................................
C     INLEZEN DATA MAGNETISCH KANAAL
C     ...............................................................
      fnum = 3
      IF (kan) CALL inchn10(fnum,namekan)

C     ...............................................................
C     AANPASSEN FILENAMEN VOOR OUTPUT-FILES
```

```
C     ............................................................
      files(6) =  '[-.output]'//file_out//'.doc'
      files(7) =  '[-.output]'//file_out//'.rfn'
      files(8) =  '[-.output]'//file_out//'.efn'

C     ............................................................
C     OMREKENEN VAN PARAMETERS
C     ............................................................


      vdee = vdee*1.E3

      delta = delta*rad
      intrv = intrv*rad
      alfast = alfast*rad
      az_u = az_u*rad
      err_th = err_th*rad

      q = z*el
C     OMLOOPFREQUENTIE DEELTJES
      f0 = f0*(1 + d_f0/1000.0D0)*1.0D6
C     D_F0 = AFIJKING VAN F0 UIT INPUT-FILE IN PROMILLES
C     ASYMMETRIE DEE-SPANNING:
      d_dee = d_dee/100.0D0

      w0 = f0*2.*pi
      b_iso = mrust*w0/q


      cnstnt = (mrust*w0*w0)/(2.*el)
      v0 = DSQRT(1.e3*e0/cnstnt)
      d1xst = v0*DCOS(alfast)
      d1yst = v0*DSIN(alfast)
      rr1 = 2.*mrust*vdee/(q*b_iso*b_iso)
      rr3 = v6v2*rr1

C     ...........................................................
C     WEGSCHRIJVEN VAN DE PARAMETERS IN HET .DOC FILE
C     ...........................................................
      fnum = 6
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='unknown')
      WRITE (funit,2300) 'UITVOER BIJ PROGRAMMA RAD10.FOR'
      WRITE (funit,2400) 'NAAM UITVOERFILE:',file_out
      WRITE (funit,2300) '================================'
      WRITE(funit,*)      'DATUM: ',sysdate
      WRITE(funit,*)      'TIJD : ',systime
      WRITE (funit,2300) '================================'
      WRITE (funit,2300) 'INTEGRATIE-PARAMETERS'
      WRITE (funit,2300) '====================='

      WRITE (funit,2300) 'Fehlberg routine, variabele stap'
      WRITE (funit,2100) 'absolute fout                 ',abserr
      WRITE (funit,2100) 'relatieve fout                ',relerr
      WRITE (funit,2100) 'uitvoer-azimuth               ',az_u/rad
      WRITE (funit,2100) 'tolerantie in uitvoer-azimuth',err_th/rad

      WRITE (funit,2200) 'integratie-interval [deg]     ',intrv/rad
      WRITE (funit,2300) 'BAAN-PARAMETERS'
      WRITE (funit,2300) '==============='
      WRITE (funit,2000) 'startw hoogfreq.-fase [deg]   ',hfpst_min
      WRITE (funit,2000) 'eindw  hoogfreq.-fase [deg]   ',hfpst_max
      WRITE (funit,2000) 'stap in hoogfreq.-fase [deg] ',stap_hfpst
      WRITE (funit,2200) 'maximale straal uitvoer       ',r_max
      WRITE (funit,2300) 'CYCLOTRON-PARAMETERS'
      WRITE (funit,2300) '===================='
      WRITE (funit,2000) 'harmonisch_getal              ',h
```

```
      WRITE (funit,2100) 'isochroon B-veld [T]          ',b_iso
      WRITE (funit,2200) 'omloop freq. [Mhz]            ',f0/1.E6
      WRITE (funit,2200) 'afw van ingelezen f0 [prom]   ',d_f0
      WRITE (funit,2200) 'dee spanning [kV]             ',vdee/1.E3
      WRITE (funit,2200) 'afw. dee spanning y-dee (%)   ',d_dee*100.
      WRITE (funit,2200) 'verhouding 6/2 deesp.         ',v6v2
      WRITE (funit,2200) 'fasev. 6-2 [deg]              ',delta
      WRITE (funit,2300) 'PARAMETERS DEELTJE'
      WRITE (funit,2300) '=================='
      WRITE (funit,2200) 'start energie [keV]           ',e0
      WRITE (funit,2100) 'x-coord. startp. [m]          ',xst
      WRITE (funit,2100) 'y-coord. startp. [m]          ',yst
      WRITE (funit,2200) 'starthoek pos. x-as [deg]     ',alfast/rad

      WRITE (funit,2300) 'MAFNEETVELD-SETTINGS'
      WRITE (funit,2300) '===================='
      WRITE (funit,*) 'harmonischen *sym t/m       ',hharm
      WRITE (funit,*) 'lagere harmonischen t/m      ',lharm
      WRITE (funit,*) 'magn. kanaal                 ',kan
      WRITE (funit,*) 'dummy kanaal                 ',twochan
      WRITE (funit,*) '1e-harm spoel 22             ',cor22
      WRITE (funit,2100) 'amplitude gauss-profiel (T)  ',b22_m
      WRITE (funit,2100) 'straal B(r) = B_max  (m)     ',r22_0
      WRITE (funit,2100) 'halfwaarde-breedte (m)       ',w22_h
      WRITE (funit,*) '1e-harm spoel 2b             ',cor2b
      WRITE (funit,2100) 'amplitude gauss-profiel (T)  ',b2b_m
      WRITE (funit,2100) 'straal B(r) = B_max  (m)     ',r2b_0
      WRITE (funit,2100) 'halfwaarde-breedte (m)       ',w2b_h
      CLOSE(funit)


C     =============================================================
C     .............................................................
C     INLEZEN DATA ELEKTRISCH VELD
C     .............................................................
      fnum = 5
      CALL ingap10(fnum,nameef,ngh,ri,ra,t,yc,hh,fi1,fi3)

      fnum = 4
      CALL inrlx10(fnum,emap,hx,hy,hz,imax,jmax)

      RETURN
C     .............................................................
C     FORMATS
C     .............................................................

2000  FORMAT(1X,A30,I3)
2100  FORMAT(1X,A30,D16.6)
2200  FORMAT(1X,A30,F8.3)
2300  FORMAT(1X,A40)
2400  FORMAT(1X,A30,A7)




C     .............................................................
C     ERRORLABELS
C     .............................................................

1000   WRITE(6,*) 'fout tijdens openen file ' ,files(fnum)
       STOP 'fout tijdens uitvoer i/o-statement!'
       END

C     =============================================================
      SUBROUTINE orrad10
C     .............................................................
```

```
      IMPLICIT NONE
      INTEGER   fnum,funit,hfpst,hfpst_min,hfpst_max,stap_hfpst,
     +          n_max,n_min,n_reg

      REAL      t_cpu
      REAL*8    intrv,az_u,err_th,abserr,relerr,r_max

      CHARACTER*40 files(8)


      COMMON/integr/  intrv,az_u,err_th,abserr,relerr

      COMMON/iofiles/ files
      COMMON/rfn/n_min,n_max,n_reg,hfpst_min,hfpst_max,stap_hfpst,r_max


      CALL vmscpu(t_cpu)
C     OPENEN VAN .RFN-FILE:
      fnum = 7
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='unknown')
      WRITE(funit,*) n_min,n_max,n_reg
      WRITE(funit,*) hfpst_min,hfpst_max,stap_hfpst
      WRITE(funit,*) r_max
C     OPENEN VAN .EFN-FILE:
      fnum = 8
      funit = fnum + 10
      OPEN(UNIT=funit,FILE=files(fnum),ERR=1000,STATUS='unknown')
      WRITE(funit,*) n_min,n_max,n_reg
      WRITE(funit,*) hfpst_min,hfpst_max,stap_hfpst
      WRITE(funit,*) r_max


      DO 100 hfpst = hfpst_min,hfpst_max,stap_hfpst
         CALL turad10(hfpst)
100   CONTINUE


      CALL vmscpu(t_cpu)
      fnum = 7
      funit = fnum + 10
      CLOSE(funit)
      fnum = 8
      funit = fnum + 10
      CLOSE(funit)

C     ..............................................................
C     ERRORLABELS
C     ..............................................................
      RETURN
1000  WRITE(6,*) 'fout tijdens openen file ' ,files(fnum)
      STOP 'fout tijdens uitvoer i/o-statement!'
      END






C     ==============================================================


      SUBROUTINE turad10(hfpst)
C     ..............................................................
```

```
      IMPLICIT NONE
      INTEGER   fnum,funit,hfpst,hfpst_min,hfpst_max,stap_hfpst,
     +          h,oml_t,z,n_max,n_min,n_reg,n_st,n_en,
     +          dim,i,n,iflag,
     +          nfe,kop,init,jflag,kflag

      REAL*8    c,el,pi,pi_2,rad,
     +          f0,b_iso,w0,cnstnt,vdee,v6v2,fi1,fi3,delta,
     +          mrust,intrv,az_u,r_max,
     +          xst,yst,alfast,e0,d1xst,d1yst,rr1,rr3,
     +          x(4),r,th,hfp,t,t2,d_th,err_th,th_u,rf_n(40),ef_n(40),
     +          relerr,abserr,gerror(4),e_kin,err_r,err_e,
     +          yp(4),f1(4),f2(4),f3(4),f4(4),f5(4),
     +          yg(4),ygp(4),h_s,savre,savae


      LOGICAL   kan,twochan,cor22,cor2b,
     +          az_neg,th_neg

      CHARACTER*40 files(8)

      EXTERNAL eqomf

      COMMON/natconst/c,el,pi,rad
      COMMON/integr/  intrv,az_u,err_th,abserr,relerr
      COMMON/rfn/n_min,n_max,n_reg,hfpst_min,hfpst_max,stap_hfpst,r_max


      COMMON/cycl/    f0,b_iso,w0,cnstnt,vdee,v6v2,
     +                fi1,fi3,delta,h

      COMMON/part/ mrust,z
      COMMON/baan/    xst,yst,alfast,e0,d1xst,d1yst,rr1,rr3
      COMMON/answ/    kan,twochan,cor22,cor2b
      COMMON/iofiles/ files
      COMMON/gerksave/yp,h_s,f1,f2,f3,f4,f5,yg,ygp,savre,savae,
     +                nfe,kop,init,jflag,kflag


C     ..............................................................
C                                 STARTCONDITIES
C     ..............................................................

C     ..............................................................
C     VASTLEGGEN VAN DE BEGINWAARDEN
C     ..............................................................

      pi_2 = 2.0D0*pi

      dim = 4
      oml_t = 1
      DO 10 i=1,40
       rf_n(i) = 0.0D0
10    CONTINUE
      DO 15 i=1,40
       ef_n(i) = 0.0D0
15    CONTINUE



C     STARTWAARDEN X,Y,V_X,V_Y
      x(1) = xst
      x(3) = yst
      x(2) = d1xst
      x(4) = d1yst

      az_neg = .FALSE.
      th_neg = .FALSE.


      r=DSQRT(x(1)**2+x(3)**2)
C     STARTWAARDE BAANSTRAAL

      th=DATAN2(x(3),x(1))
C     STARTWAARDE AZIMUTH
      IF (th.LT.0.D0) az_neg = .TRUE.

      IF (th.LT.az_u) th_neg = .TRUE.



      hfp = hfpst*rad
C     STARTWAARDE HOOGFREQUENTFASE
      t = hfp/h+th
C     ZIE THEORIE


C     ...............................................................


      WRITE(6,2000)hfpst,oml_t

      fnum = 7
      funit = fnum + 10
      WRITE(funit,*) hfpst

      fnum = 8
      funit = fnum + 10
      WRITE(funit,*) hfpst

      iflag = 1

50       t2 = t + intrv

100      CALL gerk(eqomf,dim,x,t,t2,relerr,abserr,iflag,gerror)
         IF (iflag.NE.2) THEN
                 IF (iflag.EQ.6) THEN
                   iflag = 2
                   WRITE(6,*)'INTERVAL TE KLEIN '
                   GOTO 100
                                 ELSE
                   WRITE(6,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   WRITE(funit,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   STOP
                 END IF
             ENDIF

         t = DMOD(t,2.0D0*pi)

         th = DATAN2(x(3),x(1))
         r=DSQRT(x(1)**2+x(3)**2)
         IF (r.GT.r_max) GO TO 400

         IF (th.LT.0.D0)  az_neg = .TRUE.
         IF ((th.GE.0.D0).AND.(az_neg)) THEN
              oml_t = oml_t +1
              az_neg = .FALSE.
              WRITE(6,2000)hfpst,oml_t



         ENDIF
         IF (th.LT.az_u) th_neg = .TRUE.
         IF ((th.GT.az_u).AND.(th_neg)) THEN
```

```
           th_neg = .FALSE.
           d_th = th-az_u
           DO WHILE (DABS(d_th).GE.err_th)
             t2 = t - d_th
300          CALL gerk(eqomf,dim,x,t,t2,relerr,abserr,iflag,gerror)
             IF (iflag.NE.2) THEN
                 IF (iflag.EQ.6) THEN
                   iflag = 2
                   GOTO 300
                                 ELSE
                   WRITE(6,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   WRITE(funit,*)'INTEGRATIEFOUT. IFLAG = ',iflag
                   STOP
                 END IF
             ENDIF
             th = DATAN2(x(3),x(1))
             d_th = th-az_u
           END DO
C           WRITE(6,3000) R*100.,I,TH/RAD

           r=DSQRT(x(1)**2+x(3)**2)
           rf_n(oml_t) = r

C           ERR_R = DSQRT((X(1)*GERROR(1))**2+
C     +                   (X(3)*GERROR(3))**2)/R
           e_kin = mrust*w0*w0*(x(2)**2+x(4)**2)/(2.0D3*el)
           ef_n(oml_t) = e_kin
           IF (oml_t.LE.n_max) GO TO 50
         END IF
         IF (oml_t.LE.n_max) GO TO 50

400   fnum = 7
      funit = fnum + 10
      n_st = n_min
      n_en = n_st + n_reg - 1
      DO WHILE (n_en.LE.n_max)
        WRITE(funit,'(8E13.6)') (rf_n(i),i = n_st,n_en)
        n_st = n_st + n_reg
        n_en = n_en + n_reg
      END DO
      fnum = 8
      funit = fnum + 10
      n_st = n_min
      n_en = n_st + n_reg - 1
      DO WHILE (n_en.LE.n_max)
        WRITE(funit,'(8E13.6)') (ef_n(i),i = n_st,n_en)
        n_st = n_st + n_reg
        n_en = n_en + n_reg
      END DO


      RETURN
2000  FORMAT(1X,'STARTFASE ',I4,' OMLOOP ',I4)
3000  FORMAT(1X,'r (cm) ',F8.2,'   i = ',I3,'   th = ',F8.2)


      END
C     ===============================================================
      INCLUDE 'user2:[tnndrr.ilec.source]txlib10.for'

C     ===============================================================
C
C     ===============================================================
C     EXAMPLE OF THE INPUT FILE RAD10.DAT
C   .................................................................
C   PARAMETERS BAANBEREKENINGS-PROGRAMMA
C   radia02           : naam outputfile (7 char. lang, geen spaties)
```

```
C   10                 : integratie interval
C   20.0               : azimuth uitvoer
C   0.5                : tolerantie azimuth-iteratie
C   1                  : laagste omloop
C   35                 : hoogste omloop  (<= 40)
C   5                  : aant r-waarden per regel in .rfn-file
C   0.19               : maximale straal voor uitvoer (cm)
C   -85                : laagste startfase
C   -27                : hoogste startfase
C   1                  : interval tussen opeenvolgende startfasen
C   1.0E-16            : absolute fout
C   1.0E-6             : relatieve fout
C   -1.5D0             : afwijking omlfreq. van ingelezen f0 [promilles]
C   1.E-3              : start-energie (keV)
C   1.070E-2           : x-coord. startpunt
C   0.977E-2           : y-coord. startpunt
C   128.               : starthoek met pos. x_as
C   -4.E-2             : xmin1
C   4.E-2              : xmax1
C   -6.E-2             : ymin1
C   6.E-2              : ymax1
C   201                : imax
C   281                : jmax
C   4                  : aantal ideale harmon *sym
C   0                  : aantal niet-symmetrische harmonischen
C   TRUE               : magn. kanaal
c   TRUE               : dummy-kanaal
C   FALSE              : spoel 22 aan/uit
C   0.0E-4             : amplitude gauss-profiel (T)
C   0.19               : straal waarbij B(r) = B_max  (m)
C   .10                : halfwaarde-breedte (m)
C   FALSE              : spoel 2b aan/uit
C   2.5E-3             : amplitude gauss-profiel (T)
C   0.1375             : straal waarbij B(r) = B_max  (m)
c   .04                : halfwaarde-breedte (m)
C   2                  : harmonish getal
C   21.67E0            : omloopfrqentie (MHz)
C   30.0               : deespanning (kV)
0.0                    : verschilfactor 2-e h. dee langs pos. y-as (%)
0.10                   : verhouding 6/2-harm. deespanning
1.00                   : faseverschil flattop-dee
1                      : lading deeltje in veelvoud van el
1.6726E-27             : rustmassa (kg)
C   ..................................................................
```

```
      PROGRAM sor10
C     ==================================================================
C      SOR10 WRITES SELECTED TURN NUMBERS FROM THE .RFN AND .EFN FILES
C      FROM THE PROGRAM RAD10 TO AN ASCII-FILE
C     ==================================================================

C     LAST UPDATED:    04-08-94
C     BY:              PETER OP DE BEEK
C     ==================================================================
C     UPDATE-HISTORY
C     ==================================================================
c
C     ==================================================================
      IMPLICIT NONE
      REAL*8  y_n(50),f,rmx
      INTEGER fnum,un_in,un_out,n_low,n_hig,n,step_n,par(6)
      LOGICAL eof
      CHARACTER*1 answ
      CHARACTER*40 files(3)
      CHARACTER*7 name_re
      CHARACTER*4 name_out
      CHARACTER*80 comm
C     ==================================================================
C     INTEGER FNUM CONNECTS A FILENAME FROM ARRAY FILES TO A FIXED
C     UNIT-SPECIFIER ACCORDING TO:
C             FILENAME = FILES(FNUM)
C             UNIT     = FUNIT = FNUM + 10
      DATA files / 'in_r.rfn',
     +             'in_e.efn',
     +             'out.xxx'/
C     ==================================================================
C 1. IN_R.RFN    INPUT FILE, R AS FUNCTION OF FIE_START
C 2. IN_E.EFN    INPUT FILE, E AS FUNCTION OF FIE_START
C 6. OUT.xxx     OUTPUT FILE
C     ==================================================================
      eof = .FALSE.

C     DETERMINE NAMES OF IN- AND OUTPUT FILES:
      WRITE(6,*) 'GIVE NAME INPUTFILES FOR R EN E  (7 char)'
      READ(5,'(A7)') name_re
      WRITE(6,*) 'GIVE NAME OUTPUT FILE (4 char)'
      READ(5,'(A4)') name_out
      files(1) = 'user2:[tnndrr.ilec.output]'//name_re//'.rfn'
      files(2) = 'user2:[tnndrr.ilec.output]'//name_re//'.efn'
5     WRITE(6,*) 'CHOOSE OPTION:'
      WRITE(6,*) 'R AS A FUNCTION OF FIE_INITIAL....(1)'
      WRITE(6,*) 'E AS A FUNCTION OF FIE_INITIAL....(2)'
      READ(5,'(A1)') answ
      IF ((answ.NE.'1').AND.(answ.NE.'2')) GO TO 5
      IF (answ.EQ.'1') THEN
        fnum = 1
        files(3) = 'user2:[tnndrr.ilec.idif]'//name_out//'RF.asc'
      END IF
      IF (answ.EQ.'2') THEN
        fnum = 2
        files(3) = 'user2:[tnndrr.ilec.idif]'//name_out//'EF.asc'
      END IF

C     OPEN INPUT FILE FOR R OR E AS FUNCTION OF INITIAL PHASE:
      un_in = fnum + 10
      OPEN(UNIT=un_in,FILE=files(fnum),ERR=1000,STATUS='old')
      CALL get_par(fnum,files,comm,par,rmx)
      WRITE(6,*) 'INPUT FILE ',files(fnum),' :'
      WRITE(6,'(1X,A25,I3)') 'LOWEST TURN  : ',par(1)
      WRITE(6,'(1X,A25,I3)') 'HIGHEST TURN : ',par(2)
10    WRITE(6,*)'GIVE LOWEST TURN IN OUTPUT FILE:'
      READ(5,*) n_low
      IF (n_low.LT.par(1)) GO TO 10
20    WRITE(6,*)'GIVE HIGHEST TURN IN OUTPUT FILE:'
      READ(5,*) n_hig
      IF (n_hig.GT.par(2)) GO TO 20
      WRITE(6,*)'GIVE STEP BETWEEN TURNS:'
      READ(5,*) step_n
      IF ((n_hig-n_low)/step_n.GE.8.) GO TO 10
C     OPENING OUTPUT FILE:
      fnum = 3
      un_out = fnum + 10
      OPEN(UNIT=un_out,FILE = files(fnum),ERR=1000,STATUS = 'UNKNOWN')
      WRITE (un_out,'(1X,A1,8I6)') 'f',(n,n = n_low,n_hig,step_n)
C     CALL GET_DAT TO FILL ARRAY Y_N WITH Y VALUES:
100   CALL get_dat(un_in,par,f,y_n,eof)
      IF (eof) GOTO 200
C     wegschrijven geselekteerde omlopen in un_out:
      CALL put_dat(un_in,un_out,n_low,n_hig,step_n,f,y_n)
      GOTO 100

200   CLOSE(un_in)
      CLOSE(un_out)
      STOP'PROGRAM EXECUTION READY'
C     ==================================================================
C     LABELS AND FORMATS
C     ==================================================================
1000  WRITE(6,*)'ERROR OPENING FILE ' ,files(fnum)
      STOP
      END
C     ==================================================================
C     END MAIN PROGRAM
C     ==================================================================
C     SUBROUTINES
C     ==================================================================
C     ==================================================================
      SUBROUTINE get_par(fnum,files,comm,par,rmx)
C     ==================================================================
      REAL*8  rmx
      INTEGER fnum,funit,par(6)
      CHARACTER*40 files(3)
      CHARACTER*80 comm
C     REWIND INPUT FILE:
      funit = fnum + 10
      REWIND(funit)
C     READ COMMENT LINE:
C      READ(funit,'(A80)') comm
C     READ FORMAT INPUT FILE
C     - NUMBER OF TURNS PER INITIAL PHASE VALUE
C     - NUMBER OF R-VALUES PER LINE
      READ(funit,*,END=1500) par(1),par(2),par(3)
      IF (par(1).NE.1) STOP'N_MIN .NE.1 KAN NOG NIET!'
C     READ PHASE REGION AND FASE-STEP:
      READ(funit,*,END=1500) par(4),par(5),par(6)
C     READ MAX RADIUS
      READ(funit,*,END=1500) rmx
      RETURN
1500  WRITE(6,*)'FILE ',files(fnum),' DOES NOT CONTAIN VALID DATA'
      STOP
      END
C     ==================================================================
C     ==================================================================
      SUBROUTINE get_dat(un_in,par,f,y_n,eof)
C     ==================================================================
      IMPLICIT NONE
      REAL*8   f,y_n(50)
      INTEGER  n,un_in,par(6),n_st,n_en
      LOGICAL  eof
C     IF Y-VALUES FOR NEXT F AVAILABLE,
C     READ NEW VALUE FOR F_E, AND THE CORRESPONDING Y_VALUES.
C     AT THE END OF FUNIT, SET EOF = .TRUE.:
      READ (un_in,*,END=100) f
C     RESET ARRAY Y_N:
      DO 20 n = 1,par(2)
       y_n(n) = 0.0D0
20    CONTINUE
C     FILL ARRAY Y_N WITH NEW Y-VALUES:
      n_st = 1
      n_en = n_st + par(3) - 1
      DO WHILE (n_en.LE.par(2))
        READ (un_in,*,END=200) (y_n(n),n = n_st,n_en)
        n_st = n_st + par(3)
        n_en = n_en + par(3)
      END DO
      RETURN
100   eof = .TRUE.
      RETURN
200   STOP'END INPUTFILE HAS BEEN ENCOUNTERED WHILE READING Y-VALUES'
      END
C     ================================================================
      SUBROUTINE put_dat(un_in,un_out,n_low,n_hig,step_n,f,y_n)
C     ================================================================
      IMPLICIT NONE
      REAL*8   f,y_n(50)
      INTEGER  un_in,un_out,n_low,n_hig,step_n,n
      IF (un_in.eq.11) THEN
        WRITE (un_out,'(1X,F6.2,8E13.6)')
     +           f,(y_n(n)*100.,n = n_low,n_hig,step_n)
      ELSE
        WRITE (un_out,'(1X,F6.2,8E13.6)')
     +           f,(y_n(n),n = n_low,n_hig,step_n)
      END IF
      RETURN
      END
C     ================================================================
```