CERN-PS 87-6 AA-LEA
C2

EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

BA/JB/afm

CERN PS/87-6 (AA/LEA)

**APPLICATION OF SYMBOLIC COMPUTATION TO THE SEARCH OF
COMPLICATED PRIMITIVES: THE EXAMPLE OF THE "BETATRON" INTEGRALS**

B. Autin and J. Bengtsson
CERN, Geneva, Switzerland

ABSTRACT

Symbolic computation can be a very valuable tool to calculate primitives of complicated functions in a closed form. It is applied here to the "betatron" integrals which appear in the perturbation theory of a particle motion inside an accelerator. The symbolic programme and the automatic FORTRAN coding of the generated functions are described. The verification of the formulae is essential and is performed by comparison with a conventional numerical integration. It turns out that the results are obtained with the full computer accuracy in a much shorter time by using analytical formulae than by resorting to a numerical integration.

CM-P00059132

(Paper to be published in the Journal of Symbolic Computation)

Geneva, Switzerland
February 1987

INTRODUCTION

When a function is known to have a primitive which can be expressed in a closed but complicated form, a dilemma is to be resolved: either to calculate the primitive analytically and then evaluate it numerically, or to perform a numerical integration directly. From the standpoint of an efficient use of the computer, the first alternative is likely to be preferable since it may request a shorter computing time. However, the intellectual work which is involved not only in finding the primitive but also in verifying the result and in coding the formula may be excessive. The symbolic methods provide a very valuable tool in assisting human brain for such a task as the example of the "betatron" integrals will prove it.

These integrals have their origin in the theory of the transverse motion of a particle in an accelerator (Courant and Snyder, 1958). The particle oscillation is described by a system of two second-order differential equations which can be solved by successive approximations. In the reference state, the two equations are linear and uncoupled. They have the form of a pseudo-harmonic oscillator equation in which the *restoring* force is time dependent. The solution can be expressed in cartesian or polar coordinates; the relations between the two types of coordinates are fundamental in the *algorithm* used to find the closed form of the "betatron" integrals. In the general case, the two differential equations are coupled by non-linear terms whose terminology is borrowed from the electric or magnetic field theory. A perturbation is of "dipole" type if it is independent of space coordinates (x, y) , of quadrupole, sextupole or octupole type, if it is represented by a linear, quadratic or cubic polynomial in (x, y) , respectively. The force is a function which varies

piece-wise with constant coefficients alternating between positive, zero and negative values. In each segment where the coefficients are constant, the effect of the perturbation is given by the "betatron" integrals (B. Autin, 1985) whose analytical expression is obtained by symbolic computation. As expected, the results are exceedingly complicated but their correctness and their rapid calculation can be checked by comparison with a numerical integration code.

The first part of the paper is a condensed version of reference 3 and gives the theoretical background to the "betatron" integrals. The second part is dedicated to the computational techniques.

1. BETATRON INTEGRALS

Let us consider the system of two second-order differential equations

$$\begin{aligned}\frac{d^2x}{ds^2} + K_x(s)x &= f_x(x,y) \\ \frac{d^2y}{ds^2} + K_y(s)y &= f_y(x,y) .\end{aligned}\tag{1}$$

The solution to (1) results from successive approximations applied to the reference state defined by the autonomous system:

$$\begin{aligned}\frac{d^2x}{ds^2} + K_x(s)x &= 0 \\ \frac{d^2y}{ds^2} + K_y(s)y &= 0\end{aligned}\tag{2}$$

whose solution will be expressed in terms of cartesian and polar coordinates.

1.1 Autonomous system

The two equations of (2) are decoupled and the form of the solution is discussed on the single equation

$$u'' + K(s)u = 0 \quad (3)$$

where the notation ' means a derivation with respect to s.

1.1.1 Cartesian Coordinates

We call cartesian coordinates the function $u(s)$ and its derivative $u'(s)$. The equation (3) being linear, the solution vector (u, u') is a linear function of the initial vector (u_0, u'_0) given at the origin $s = 0$.

$$\begin{pmatrix} u \\ u' \end{pmatrix} = \begin{pmatrix} a_{11}(s) & a_{12}(s) \\ a'_{11}(s) & a'_{12}(s) \end{pmatrix} \begin{pmatrix} u_0 \\ u'_0 \end{pmatrix}. \quad (4)$$

Explicit forms for the coefficients a_{11} and a_{12} will be given in section 2.1 for special cases.

1.1.2 Polar coordinates

The similarity of the equation (3) with that of an harmonic oscillator suggests a solution of the form:

$$u(s) = a(s) \cos \mu(s) . \quad (5)$$

By substituting (5) into (3), it turns out that the amplitude $a(s)$ and the phase $\mu(s)$ must satisfy the set of differential equations

$$\begin{aligned} a'' - a\mu'^2 + Ka &= 0 \\ 2a'\mu' + a\mu'' &= 0 . \end{aligned} \quad (6)$$

The second equation of (6) can be integrated once:

$$\mu' = 2J/a^2 . \quad (7)$$

The integration constant has been chosen to put in evidence the action

variable J which plays a classical role in mechanics. Still, to conform to the custom, we put

$$\beta = a^2/2J \quad (8)$$

$$\alpha = -\frac{1}{2} \beta' \quad (9)$$

$$\gamma = (1 + \alpha^2)/\beta \quad (10)$$

and the system (6) takes the simpler form

$$\alpha' = K\beta - \gamma \quad (11)$$

$$\mu' = 1/\beta$$

The relations between cartesian and polar coordinates can be obtained by expanding u as a function of u_0 and u_0' in (5). At the origin of the motion, the phase is μ_0 and the amplitude is $\sqrt{2J\beta_0}$:

$$u_0 = \sqrt{2J\beta_0} \cos\mu_0$$

Using the relations (7), (8), (9), the derivative $u'(s)$ is given by:

$$u' = -\sqrt{2J/\beta} (\sin\mu + \alpha \cos\mu) \quad (12)$$

it takes the value

$$u_0' = -\sqrt{2J/\beta_0} (\sin\mu_0 + \alpha_0 \cos\mu_0)$$

at the origin and the new form of $u(s)$ is:

$$u(s) = \sqrt{\beta/\beta_0} (\cos\mu + \alpha_0 \sin\mu)u_0 + \sqrt{\beta\beta_0} \sin\mu u_0' \quad (13)$$

By identifying the first equation of (4) with (13), one gets the formulae which will be used in the algorithm studied in section II:

$$\begin{aligned} \sqrt{\beta} \sin\mu &= a_{12}/\sqrt{\beta_0} \\ \sqrt{\beta} \cos\mu &= \sqrt{\beta_0} a_{11} - (\alpha_0/\sqrt{\beta_0})a_{12} \\ \beta &= \beta_0 a_{11}^2 - 2\alpha_0 a_{11}a_{12} + \gamma_0 a_{12}^2 \end{aligned} \quad (14)$$

1.2 Floquet Transformation

The first step in the perturbative treatment of non-autonomous system (1) consists of choosing the new set of variables

$$\begin{aligned} q &= u/\sqrt{\beta} = \sqrt{2J} \cos \mu \\ p &= -dq/d\mu = \sqrt{2J} \sin \mu \end{aligned} \quad (15)$$

whose image lies on a circle of radius $\sqrt{2J}$. The action and phase are then given by

$$\begin{aligned} J &= \frac{1}{2}(p^2 + q^2) \\ \mu &= \tan^{-1}(p/q) . \end{aligned} \quad (16)$$

The system (15) can also be regarded as the definition of the so-called Floquet's transformation (Floquet, 1883) which converts the old (u, u') coordinates into the new (q, p) coordinates or vice-versa:

$$\begin{pmatrix} u \\ u' \end{pmatrix} = \begin{pmatrix} \sqrt{\beta} & 0 \\ -\alpha/\sqrt{\beta} & -1/\sqrt{\beta} \end{pmatrix} \begin{pmatrix} q \\ p \end{pmatrix} . \quad (17)$$

The equation

$$u'' + K(s)u = f(u) \quad (18)$$

can thus be written as a system of two first-order equations:

$$\begin{aligned} \dot{q} &= -p \\ \dot{p} &= q - \beta^{3/2} f(q) . \end{aligned} \quad (19)$$

The left-hand side coefficients are constant and the notation $\dot{}$ means a derivation with respect to μ .

1.3 First-Order Perturbation of Amplitude and Phase Variables

The presence of a term at the right-hand side of the differential equation alters the function p by a quantity

$$dp = -\beta^{3/2} f(q) d\mu \quad (20)$$

which in turn produces a variation of action

$$dJ = p dp \quad (21)$$

and phase

$$d\mu = \frac{1}{2J} q dp \quad (22)$$

as it can be deduced from the relations (16). When the perturbation f is disposed periodically, the unperturbed phase increases by $2\pi Q$ in each period and one is interested in the mean change of Q over a great number of periods

$$dQ = \frac{1}{2\pi N} \sum_{n=1}^N d\mu \quad , \quad (23)$$

dQ is the variation of the number of oscillations per period, also called the tune-shift.

1.3.1 Dipole

The simplest perturbation that can be envisaged is a constant or *dipole* term.

$$f(u) = -\delta \quad . \quad (24)$$

By substitution of the expressions (15) and (21), one finds

$$dJ = \sqrt{2J} \delta \beta^{3/2} \sin\mu d\mu$$

or using the old variable s ,

$$dJ = \sqrt{2J} \delta\sqrt{\beta} \sin\mu \, ds \quad .$$

A first betatron integral can thus be introduced:

$$B_1 = \int \sqrt{\beta} \sin\mu \, ds \quad . \quad (25)$$

It is easy to show that dQ is zero in the dipole case.

1.3.2 Quadrupole

The second type of perturbation is of the same nature as the reference *quadrupole* term:

$$f(u) = -ku \quad , \quad u = \sqrt{\beta} \, q \quad . \quad (26)$$

The same algebra as in the dipole case provides the variations of action and tune

$$dJ = 2J \, k \, \beta^2 \sin\mu \cos\mu \, d\mu$$

$$dQ = k/4\pi \, \beta^2 \, d\mu \quad .$$

In the derivation of the expression of dQ , it has been assumed that the sum

$$\sum_{n=1}^N \cos 2(\mu + 2\pi nQ)$$

remain bounded, which is true if Q is different from a half integer.

Using the old variables s , we define the betatron integrals:

$$B_2 = \int \beta \sin 2\mu \, ds \quad (27)$$

$$Q_2 = \int \beta \, ds \quad (28)$$

1.3.3 Sextupole

The *sextupole* perturbation is defined by

$$f_x = -\frac{1}{2} k' (x^2 - y^2) \quad (29)$$

$$f_y = k' xy$$

and presents a new conceptual difficulty since the two equations of the system (1) become coupled. However, the same process of transformations used in the previous cases can be repeated and provides the expressions of the action variations:

$$\Delta J_x = \frac{\sqrt{2J_x}}{4} k' (J_x B_{31} - 2J_y B_{32} + J_x B_{33} - J_y B_{34} - J_y B_{35})$$

$$\Delta J_y = \frac{\sqrt{2J_x}}{2} k' J_y' (B_{34} - B_{35})$$

where five new betatron integrals have been introduced:

$$\begin{aligned} B_{31} &= \int \beta_x^{3/2} \sin \mu_x ds \\ B_{32} &= \int \sqrt{\beta_x} \beta_y \sin \mu_x ds \\ B_{33} &= \int \beta_x^{3/2} \sin 3\mu_x ds \\ B_{34} &= \int \sqrt{\beta_x} \beta_y \sin(\mu_x - 2\mu_y) ds \\ B_{35} &= \int \sqrt{\beta_x} \beta_y \sin(\mu_x + 2\mu_y) ds \end{aligned} \quad (30)$$

As for the dipole case, the tune-shift is zero. This absence of variation of the tune is in fact a general property of all the "odd" perturbations.

1.3.4 Octupole

The last perturbation which is studied here is the "octupole" which is defined by

$$\begin{aligned} f_x &= -\frac{k''}{6} (x^3 - 3xy^2) \\ f_y &= \frac{k''}{6} (3x^2y - y^3) \end{aligned} \quad (31)$$

The method developped for the sextupole can still be applied and the final results are only reported:

$$\begin{aligned}
 \Delta J_x &= \frac{J_x}{12} k'' (2J_x B_{41} - 6J_y B_{42} + J_x B_{43} - 3J_y B_{44} - 3J_y B_{45}) \\
 \Delta J_y &= \frac{J_y}{12} k'' (-6J_x B_{46} + 2J_y B_{47} + J_y B_{48} + 3J_x B_{44} - 3J_x B_{45}) \\
 \Delta Q_x &= \frac{k''}{16\pi} (J_x Q_{41} - 2J_y Q_{42}) \\
 \Delta Q_y &= -\frac{k''}{16\pi} (2J_x Q_{42} - J_y Q_{43})
 \end{aligned} \tag{32}$$

with the new betatron integrals:

$$\begin{aligned}
 B_{41} &= \int \beta_x^2 \sin 2\mu_x \, ds \\
 B_{42} &= \int \beta_x \beta_y \sin 2\mu_x \, ds \\
 B_{43} &= \int \beta_x^2 \sin 4\mu_x \, ds \\
 B_{44} &= \int \beta_x \beta_y \sin 2(\mu_x - \mu_y) \, ds \\
 B_{45} &= \int \beta_x \beta_y \sin 2(\mu_x + \mu_y) \, ds \\
 B_{46} &= \int \beta_x \beta_y \sin 2\mu_y \, ds \\
 B_{47} &= \int \beta_y^2 \sin 2\mu_y \, ds \\
 B_{48} &= \int \beta_y^2 \sin 4\mu_y \, ds \\
 Q_{41} &= \int \beta_x^2 \, ds \\
 Q_{42} &= \int \beta_x \beta_y \, ds \\
 Q_{43} &= \int \beta_y^2 \, ds
 \end{aligned} \tag{33}$$

2. SYMBOLIC COMPUTATION

2.1 Analytical Expression of the Betatron Integrals

The integrals (25), (27), (28), (30) and (33) have a closed form when $K(s)$ is a piece-wise varying function:

In each segment where K is constant, the coefficients a_{11} and a_{12} are given in the following table.

K	a_{11}	a_{12}
<0	$\cosh(\sqrt{ K } s)$	$\frac{1}{\sqrt{ K }} \sinh(\sqrt{ K } s)$
0	1	1
>0	$\cos(\sqrt{K} s)$	$\frac{1}{\sqrt{K}} \sin(\sqrt{K} s)$

In addition we impose:

$$K = K_x = -K_y \quad (34)$$

and discuss the three different cases where K may be positive, zero or negative. Last, the betatron integrals were established for a solution to the differential system defined by initial conditions; when, however, the boundary conditions are periodic (Hill's equation), the betatron integrals have the same form³ but contain cosine instead of sine functions.

2.1.1 The Integrals

Generally the following types of integrals have to be solved:

$$I_s = \int_0^L \beta_x^k(s) \beta_y^l \cdot \sin[m(\mu_x(s) + \mu_{x_1}) + n(\mu_y(s) + \mu_{y_1})] ds \quad (35)$$

$$I_C = \int_0^L \beta_x^k(s) \beta_y^l \cdot \cos[m(\mu_x(s) + \mu_{x_1}) + n(\mu_y(s) + \mu_{y_1})] ds \quad (36)$$

where k and l are integers or half-integers, m and n are integers, μ_{x_1} and μ_{y_1} are constant.

The s -dependence for μ and β depends on the value of K . There are three different cases for equations (14).

2.1.2 $K_x = -K_y = 0$

$$\begin{aligned} \sqrt{\beta(s)} \cos \mu(s) &= \sqrt{\beta_1} - (\alpha_1/\sqrt{\beta_1})s \\ \sqrt{\beta(s)} \sin \mu(s) &= s/\sqrt{\beta_1} \\ \beta(s) &= \beta_1 - 2\alpha_1 s + \gamma_1 s^2 \end{aligned} \quad (37)$$

where α_1 , β_1 and γ_1 are constants.

This is valid in both the horizontal (x) and vertical (y) planes.

2.1.3 $K_x > 0, K_y = -K_x$

$$\begin{aligned} \sqrt{\beta_x(s)} \cos \mu_x(s) &= \sqrt{\beta_{x_1}} \cos(\sqrt{K_x} s) - \frac{\alpha_{x_1}}{\sqrt{K_x} \beta_{x_1}} \sin(\sqrt{K_x} s) \\ \sqrt{\beta_x(s)} \sin \mu_x(s) &= \frac{1}{\sqrt{K_x} \beta_{x_1}} \sin(\sqrt{K_x} s) \\ \beta_x(s) &= \frac{1}{2} \left(\beta_{x_1} + \frac{\gamma_{x_1}}{K_x} \right) + \frac{1}{2} \left(\beta_{x_1} - \frac{\gamma_{x_1}}{2} \right) \cos(2\sqrt{K_x} s) - \frac{\alpha_{x_1}}{\sqrt{K_x}} \sin(2\sqrt{K_x} s) \\ \sqrt{\beta_y(s)} \cos \mu_y(s) &= \sqrt{\beta_{y_1}} \cosh(\sqrt{|K_y|} s) - \frac{\alpha_{y_1}}{\sqrt{|K_y|} \beta_{y_1}} \sinh(\sqrt{|K_y|} s) \\ \sqrt{\beta_y(s)} \sin \mu_y(s) &= \frac{\sinh(\sqrt{|K_y|} s)}{\sqrt{|K_y|} \beta_{y_1}} \\ \beta_y(s) &= \frac{1}{2} \left(\beta_{y_1} - \frac{\gamma_{y_1}}{|K_y|} \right) + \frac{1}{2} \left(\beta_{y_1} - \frac{\gamma_{y_1}}{|K_y|} \right) \cosh(2\sqrt{|K_y|} s) - \frac{\alpha_{y_1}}{\sqrt{|K_y|}} \sinh(2\sqrt{|K_y|} s) \end{aligned} \quad (38)$$

where α_{x1} , β_{x1} , γ_{x1} , α_{y1} , β_{y1} , γ_{y1} , and K are constants.

2.1.4 $K_x < 0$, $K_y = -K_x$

$$\sqrt{\beta_x(s)} \cos \mu_x(s) = \sqrt{\beta_{x1}} \cosh(\sqrt{|K_x|} s) - \frac{\alpha_{x1}}{\sqrt{|K_x| \beta_{x1}}} \sinh(\sqrt{|K_x|} s)$$

$$\sqrt{\beta_x(s)} \sin \mu_x(s) = \frac{1}{\sqrt{|K_x| \beta_{x1}}} \sinh(\sqrt{|K_x|} s)$$

$$\begin{aligned} \beta_x(s) &= \frac{1}{2} \left(\beta_{x1} - \frac{\gamma_{x1}}{|K_x|} \right) + \frac{1}{2} \left(\beta_{x1} - \frac{\gamma_{x1}}{|K_x|} \right) \cosh(2\sqrt{|K_x|} s) \\ &\quad - \frac{\alpha_{x1}}{\sqrt{|K_x|}} \sinh(2\sqrt{|K_x|} s) \end{aligned}$$

(39)

$$\sqrt{\beta_y(s)} \cos \mu_y(s) = \sqrt{\beta_{y1}} \cos(\sqrt{K_y} s) - \frac{\alpha_{y1}}{\sqrt{K_y \beta_{y1}}} \sin(\sqrt{K_y} s)$$

$$\sqrt{\beta_y(s)} \sin \mu_y(s) = \frac{1}{\sqrt{K_y \beta_{y1}}} \sin(\sqrt{K_y} s)$$

$$\beta_y(s) = \frac{1}{2} \left(\beta_{y1} + \frac{\gamma_{y1}}{K_y} \right) + \frac{1}{2} \left(\beta_{y1} - \frac{\gamma_{y1}}{K_y} \right) \cos(2\sqrt{K_y} s) - \frac{\alpha_{y1}}{\sqrt{K_y}} \sin(2\sqrt{K_y} s)$$

where α_{x1} , β_{x1} , γ_{x1} , α_{y1} , β_{y1} , γ_{y1} , K_x , and K_y are constants.

2.1.5 How to Solve the Integrals

The integrals may be solved analytically by the following procedure.

The first step is to develop the sine or cosine factor in the integrand into a polynomial in $\sin \mu_x(s)$, $\cos \mu_x(s)$, $\sin \mu_y(s)$ and $\cos \mu_y(s)$ by the following relations:

$$\sin(a_1 + a_2) = \sin a_1 \cos a_2 + \cos a_1 \sin a_2$$

$$\cos(a_1 + a_2) = \cos a_1 \cos a_2 - \sin a_1 \sin a_2$$

(40)

and relations like

$$\begin{aligned}\sin(2\alpha) &= 2 \sin\alpha \cos\alpha \\ \cos(2\alpha) &= \cos^2\alpha - \sin^2\alpha \\ \sin(3\alpha) &= \sin\alpha \cos(2\alpha) + \cos\alpha \sin(2\alpha) \\ \cos(3\alpha) &= \cos\alpha \cos(2\alpha) - \sin\alpha \sin(2\alpha) \\ \sin(4\alpha) &= \sin\alpha \cos(3\alpha) + \cos\alpha \sin(3\alpha) \\ \cos(4\alpha) &= \cos\alpha \cos(3\alpha) - \sin\alpha \sin(3\alpha)\end{aligned}\tag{41}$$

and so on, depending on the values of m and n in the integrand. One may now use relations (14.a) and 14.b) in each case to replace $\sin\mu_X(s)$, $\cos\mu_X(s)$, $\sin\mu_Y(s)$ and $\cos\mu_Y(s)$ by explicit functions of s .

If $k = m/2$ and $\ell = n/2$, this substitution is sufficient because it eliminates the original $\beta_X(s)$ and $\beta_Y(s)$ factors. In the other cases the relation (14.c) has to be used. At this stage the integrand has been transformed into a polynomial in s if $K = 0$, or into a polynomial in trigonometric and hyperbolic functions for non-zero K values. It is then further simplified by using the following relations:

$$\begin{aligned}\sin^2\alpha &= \frac{1}{2}(1 - \cos 2\alpha) \\ \cos^2\alpha &= \frac{1}{2}(1 + \cos 2\alpha) \\ \sin\alpha \cos\alpha &= \frac{1}{2} \sin 2\alpha \\ \sinh^2\alpha &= \frac{1}{2}(\cosh 2\alpha - 1) \\ \cosh^2\alpha &= \frac{1}{2}(\cosh 2\alpha + 1) \\ \sinh\alpha \cosh\alpha &= \frac{1}{2} \sinh 2\alpha\end{aligned}\tag{42}$$

The integrand thus contains bilinear terms in trigonometric and hyperbolic functions whose integration is straightforward.

The real difficulty lies in the algebra involved in the transformation of the integrand. Fortunately, there is no need to do this by hand.

2.2 A Symbolic Program for the Integration

Our purpose is to elaborate a program which performs the integration symbolically and generates the FORTRAN code for the calculated integral.

The integrals which contain the cosine and sine terms are treated separately. The FORTRAN code is a function which gives the value of the integral for given parameters which are defined in a COMMON block. The symbolic programs are divided into procedures which are called for different values of k , l , m , n in equations (35) and (36). Due to the large number of integrals a global function is defined with two parameters for each case:

<u>Section</u>	<u>FORTRAN Function</u>
3.1	DIP(NB, TYPE)
3.2	QUD(NB, TYPE)
3.3	SXT(NB, TYPE)
3.4	OCT(NB, TYPE)

NB is the number of the integral (1, 2, 3, ...), the lower numbers for the sine case and the upper for the cosine case.

TYPE 1, 2 or 3 depending on which of the equations (37), (38) and (39) is to be used.

2.3 Implementation of the Symbolic Program

The symbolic programs have been implemented in REDUCE. The program presented in Appendix A is related to the integrals which contain a cosine term in the integrand. The discussion would be exactly the same for cosine type integrands.

The program is composed of an integration procedure (INTS) followed by a FORTRAN coding of the integral (FCODE).

2.3.1 Description of the Symbolic Program

The procedure for the symbolic integration (INTS) is shown in lines 20-142 which resume the logical steps defined in section 1.5.

Line

- 24-27 β_x, β_y, μ_x and μ_y are defined as functions of s .
- 34-41 Equations (40) and (41) are defined.
- 42 The above rules are applied.
- 43-55 Equations (37) are defined and applied if TYPE = 1.
- 56-74 Equations (38) are defined and applied if TYPE = 2.
- 75-93 Equations (39) are defined and applied if TYPE = 3.
- 99-105 Substitution of $\varphi = \sqrt{K}$ s if TYPE = 2, 3.
- 106-117 Equations (42) are defined and applied.
- 121-133 Integration if TYPE = 2, 3. Observe that line 126 gives the
 primitive function and that the limits has to be substi-
 tuted, which is done in line 132.
- 134-139 Integration if TYPE = 1.

The procedure for FORTRAN coding is found at lines 145-192. It mainly contains WRITE-statements for the definition of the function and of the COMMON-block containing the parameters and for the declaration of the variables. The symbolic expression is written in FORTRAN code at line 186. REDUCE automatically divides the expression into multiple statements if the number of lines for one statement exceeds the number specified at line 4.

Finally, the main program is found at lines 195-237. It consists of a series of calls to the two procedures preceded by definitions of the constants in each case (here A, B, C, D have been used instead of k, l, m, n).

When the program is executed the result is stored in two files containing the intermediate results during the integration on one hand, and the proper functions in FORTRAN on the other hand. Some typical generated functions are presented in Appendix B.

During the development of the program we got a small problem. For some reason REDUCE did not apply the LET rules for the development of the sine and cosine terms in all cases. This was solved by changing the argument in the sine and cosine terms from $\sqrt{K} s$ (SQRT(K)*S) to φ (FI).

2.3.2 Comparison with Numerical Integration

To check the symbolic calculations we have compared the results deduced from the symbolic calculation with a numerical integration using Romberg's method (G. Dahlqvist and Å. Björk, 1974), i.e. trapezoidal rule followed by Richardson extrapolation.

We show below the CPU time in ms needed to calculate the integral using a symbolically generated function with optimized code (opt.code), non-optimized code (nonopt.code) and a numerical integration with an accuracy of twelve decimals (num.int.). Observe that the accuracy in CPU time is around 1 ms. The calculations have been done on a μ VAX using double precision.

Sextupole Case (Chapter 3.3)

Integral	K > 0 in horizontal plane K < 0 in vertical plane			K = 0		
	Opt. code	Nonopt. code	Num. int.	Opt. code	Nonopt. code	Num. int.
B ₃₁	1	3	6	1	2	19
B ₃₂	2	3	94	2	1	15
B ₃₃	1	3	214	1	2	18
B ₃₄	4	9	190	1	3	14
B ₃₅	4	9	198	1	3	15

CPU time in ms.

Octupole Case (Chapter 3.4)

Integral	K > 0 in horizontal plane K < 0 in vertical plane			K = 0		
	Opt. code	Nonopt. code	Num. int.	Opt. code	Nonopt. code	Num. int.
B ₄₁	2	4	181	2	1	30
B ₄₂	4	6	180	1	2	28
B ₄₃	2	5	191	1	1	28
B ₄₄	6	18	178	1	4	24
B ₄₅	6	18	191	2	5	27
B ₄₆	4	6	184	1	0	28
B ₄₇	2	4	181	1	2	28
B ₄₈	3	6	191	1	2	28

CPU time in ms.

4. CONCLUSION

The origin of the betatron integrals has been established using a first-order perturbative treatment of a system of two second-order differential equations which represent the motion of two forced pseudo-harmonic oscillators. The integrals have been calculated analytically and their closed expression was coded in FORTRAN with the REDUCE symbolic programs. The results are rather bulky since they may request up to two pages of dense print-out. The verification of the formulae correctness is therefore of primary importance and it was done, successfully, by comparison with a numerical integration procedure. In the course of the comparison, the initial speculation about the superiority of the symbolic integration was fully confirmed. Furthermore, the effort in investing into symbolic computation is of paramount reward when the problem is repetitive; in our case, once the methodology has been fixed for one integral, it was just a matter of routine to produce more than hundred integrals of similar nature.

REFERENCES

- E.D. Courant and H.S. Snyder, Theory of Alternating Gradient Synchrotron, Ann. Phys. 3, 1-48, (1958).
- G. Floquet, Sur les équations différentielles linéaires à coefficients périodiques, Ann. Aci. Ecole Normale Supérieure 12, 47-82 (IV 6), (1883).
- B. Autin, Non-Linear Betatron Oscillations, in Proc. Summer School on High-Energy Particle Accelerators, SLAC, Stanford, 1985, (to be published).

APPENDIX A

PROGRAM FOR A SYMBOLIC INTEGRATION

```

1. SYMBOLIC BPSLEFT()$
2. OFF ECHO$
3. ON ALLFAC$
4. CARDNO!*:=20$
5.
6. COMMENT INTEGRATION OF
7.
8. BETAX^A*BETAY^B*SIN(C*(MX+MYX1))+D*(MVV+MVV1))
9.
10. IN THE CASES (TYPE)
11.
12. 1: PURE SEXTUPOLE
13. 2: A SEXTUPOLE SUPERIMPOSED TO A F-QUADRUPOLE
14. 3: A SEXTUPOLE SUPERIMPOSED TO A D-QUADRUPOLE
15.
16. ;
17. OPERATOR INTEGRAND, INTEGRAL, INTO;
18.
19.
20. PROCEDURE INTS(TYPE);
21.
22. BEGIN
23.   ON EXP;
24.   DEPEND BX,S;
25.   DEPEND BY,S;
26.   DEPEND MX,S;
27.   DEPEND MY,S;
28.   OUT DATA;
29.   IF C=0 AND D=0 THEN
30.     WRITE INTEGRAND():=BX**A*BY**B
31.   ELSE
32.     WRITE INTEGRAND():=BX**A*BY**B*SIN(C*(MX+MX1))+D*(MV+MV1));
33.   OUT T;
34.   FOR ALL X,Y LET SIN(X+Y)=SIN(X)*COS(Y)+COS(X)*SIN(Y);
35.   FOR ALL X,Y LET COS(X+Y)=COS(X)*COS(Y)-SIN(X)*SIN(Y);
36.   FOR ALL X LET SIN(2*X)=2*SIN(X)*COS(X);
37.   FOR ALL X LET COS(2*X)=COS(X)**2-SIN(X)**2;
38.   FOR ALL X LET SIN(3*X)=SIN(X)*COS(2*X)+SIN(2*X)*COS(X);
39.   FOR ALL X LET COS(3*X)=COS(X)*COS(2*X)-SIN(2*X)*SIN(X);
40.   FOR ALL X LET SIN(4*X)=SIN(X)*COS(3*X)+SIN(3*X)*COS(X);
41.   FOR ALL X LET COS(4*X)=COS(X)*COS(3*X)-SIN(3*X)*SIN(X);
42.   INTEGRAND():=INTEGRAND();
43.   IF TYPE=1 THEN
44.     BEGIN
45.       LET COS(MX)=(SQRT(BX1)-AX1/SQRT(BX1)*S)/SQRT(BX);
46.       LET COS(MY)=(SQRT(BY1)-AY1/SQRT(BY1)*S)/SQRT(BY);
47.       LET SIN(MX)=S/SQRT(BX1)/SQRT(BX);
48.       LET SIN(MY)=S/SQRT(BY1)/SQRT(BY);
49.       INTEGRAND():=INTEGRAND();
50.       CLEAR COS(MX), SIN(MX), COS(MY), SIN(MY);
51.       LET BX=BX1-2*AX1*S+GX1*S**2;
52.       LET BY=BY1-2*AY1*S+GY1*S**2;
53.       INTEGRAND():=INTEGRAND();
54.       CLEAR BX,BY;
55.     END
56.   ELSE IF TYPE=2 THEN
57.     BEGIN
58.       LET COS(MX)=(SQRT(BX1)*COS(SQRT(K)*S)-
59.         AX1/SQRT(K*BX1)*SIN(SQRT(K)*S))/SQRT(BX);
60.       LET COS(MY)=(SQRT(BY1)*COSH(SQRT(K)*S)-
61.         AY1/SQRT(K*BY1)*SINH(SQRT(K)*S))/SQRT(BY);

```

```

62. LET SIN(MX)=SIN(SQRT(K)*S)/SQRT(K*BX1)/SQRT(BX);
63. LET SIN(MV)=SINH(SQRT(K)*S)/SQRT(K*BY1)/SQRT(BY);
64. INTEGRAND():=INTEGRAND();
65. CLEAR COS(MX), SIN(MX), COS(MV), SIN(MV);
66. LET BX=1/2*(BX1+GX1/K)+
67. 1/2*(BX1-GX1/K)*COS(2*SQRT(K)*S)-
68. AX1/SQRT(K)*SIN(2*SQRT(K)*S);
69. LET BY=1/2*(BY1-GY1/K)+
70. 1/2*(BY1-GY1/K)*COSH(2*SQRT(K)*S)-
71. AV1/SQRT(K)*SINH(2*SQRT(K)*S);
72. INTEGRAND():=INTEGRAND();
73. CLEAR BX,BY;
74. END
75. ELSE
76. BEGIN
77. LET COS(MX)=(SQRT(BX1)*COSH(SQRT(K)*S)-
78. AX1/SQRT(K*BX1)*SINH(SQRT(K)*S))/SQRT(BX);
79. LET COS(MV)=(SQRT(BY1)*COS(SQRT(K)*S)-
80. AV1/SQRT(K*BY1)*SIN(SQRT(K)*S))/SQRT(BY);
81. LET SIN(MX)=SINH(SQRT(K)*S)/SQRT(K*BX1)/SQRT(BX);
82. LET SIN(MV)=SIN(SQRT(K)*S)/SQRT(K*BY1)/SQRT(BY);
83. INTEGRAND():=INTEGRAND();
84. CLEAR COS(MX), SIN(MX), COS(MV), SIN(MV);
85. LET BX=1/2*(BX1-GX1/K)+
86. 1/2*(BX1+GX1/K)*COSH(2*SQRT(K)*S)-
87. AX1/SQRT(K)*SINH(2*SQRT(K)*S);
88. LET BY=1/2*(BY1-GY1/K)+
89. 1/2*(BY1-GY1/K)*COS(2*SQRT(K)*S)-
90. AV1/SQRT(K)*SINH(2*SQRT(K)*S);
91. INTEGRAND():=INTEGRAND();
92. CLEAR BX,BY;
93. END;
94. FOR ALL X,Y CLEAR SIN(X+Y),COS(X+Y);
95. FOR ALL X CLEAR SIN(2*X),COS(2*X);
96. FOR ALL X CLEAR SIN(3*X),COS(3*X);
97. FOR ALL X CLEAR SIN(4*X),COS(4*X);
98.
99. IF TYPE NEQ 1 THEN
100. BEGIN
101. COMMENT IF NOT PURE SEXTUPOLE SUBSTITUTE FI;
102. LET S=FI/SQRT(K);
103. INTEGRAND():=INTEGRAND();
104. CLEAR S;
105. END;
106. FOR ALL X LET SIN(X)**2=1/2*(1-COS(2*X));
107. FOR ALL X LET COS(X)**2=1/2*(1+COS(2*X));
108. FOR ALL X LET SIN(X)*COS(X)=1/2*SIN(2*X);
109. FOR ALL X LET SINH(X)**2=1/2*(COSH(2*X)-1);
110. FOR ALL X LET COSH(X)**2=1/2*(1+COSH(2*X));
111. FOR ALL X LET SINH(X)*COSH(X)=1/2*SINH(2*X);
112. INTEGRAND():=INTEGRAND();
113. FOR ALL X CLEAR SIN(X)**2,COS(X)**2,SIN(X)*COS(X);
114. FOR ALL X CLEAR SINH(X)**2,COSH(X)**2,SINH(X)*COSH(X);
115. FOR ALL X LET COS(2*X)*COS(X)=1/2*(COS(3*X)+COS(X));
116. FOR ALL X LET COS(2*X)*SIN(X)=1/2*(SIN(3*X)-SIN(X));
117. INTEGRAND():=INTEGRAND();
118. FOR ALL X CLEAR COS(2*X)*COS(X);
119. FOR ALL X CLEAR COS(2*X)*SIN(X);
120. OUT DATA;
121. IF TYPE NEQ 1 THEN
122. BEGIN
123. COMMENT IF NOT PURE SEXTUPOLE INTEGRATE WITH RESPECT TO FI.

```

```

124. DS=DFI/SQRT(K);
125. INTEGRAND():=INTEGRAND()/SQRT(K);
126. INTEGRAL():=INT(INTEGRAND(),FI);
127. COMMENT SUBTRACT INITIAL VALUE OF INTEGRAL;
128. INTO():=INTEGRAL();
129. LET FI=0;
130. INTO():=INTO();
131. CLEAR FI;
132. WRITE INTEGRAL():=INTEGRAL()-INTO();
133. END
134. ELSE
135. BEGIN
136. COMMENT INTEGRATE WITH RESPECT TO S;
137. WRITE INTEGRAL():=INT(INTEGRAND(),S);
138. COMMENT INITIAL VALUE OF INTEGRAL IS 0 IN THIS CASE;
139. END;
140. OUT T;
141. OFF EXP;
142. END;
143.
144.
145.
146.
147.
148. BEGIN
149. ON FORT;
150. OUT FORTAN;
151. OFF PERIOD;
152. WRITE " ";
153. ON PERIOD;
154. WRITE "C";
155. WRITE "C";
156. IF C=0 AND D=0 THEN
157. ELSE
158. INTEGRAND():=BETAX**A*BETAY**B*SIN(C*(MYX+MYX1)+D*(MYV+MYV1));
159. WRITE "C";
160. WRITE "C";
161. IF TYPE=1 THEN
162. WRITE "C";
163. IN THE CASE OF PURE SEXTUPOLE"
164. ELSE IF TYPE=2 THEN
165. BEGIN
166. WRITE "C";
167. WRITE "C";
168. IN THE CASE OF SEXTUPOLE SUPERIMPOSED TO AN";
169. FOCUSING QUADRUPOLE";
170. END
171. ELSE
172. BEGIN
173. WRITE "C";
174. WRITE "C";
175. IN THE CASE OF SEXTUPOLE SUPERIMPOSED TO AN";
176. DEFOCUSING QUADRUPOLE";
177. END;
178. WRITE " ";
179. IMPLICIT DOUBLE PRECISION (A-Z)";
180. IF TYPE NEQ 1 THEN
181. WRITE " ";
182. DOUBLE PRECISION FI,VAL"
183. ELSE
184. WRITE " ";
185. DOUBLE PRECISION VAL";
186. WRITE " ";
187. DOUBLE PRECISION BX1,BY1,MX1,MV1,AX1,AV1,GX1,GY1,K,S";
188. COMMON/PRM/BX1,BY1,MX1,MV1,AX1,AV1,GX1,GY1,K,S";
189. WRITE " ";
190. IF TYPE NEQ 1 THEN
191. WRITE " ";
192. FI=SQRT(K)*S";

```

```

186. WRITE VAL:=INTEGRAL();
187. WRITE " ",NAME,"=VAL";
188. WRITE " "
189. WRITE " ";
190. WRITE " ";
191. OFF FORT;
192. END;
193.
194.
195. LET A=3/2$ LET B=0$ LET C=1$ LET D=0$
196. INTS(2)$
197. FCODE(SSFQ1,2)$
198.
199. LET A=1/2$ LET B=1$ LET C=1$ LET D=0$
200. INTS(2)$
201. FCODE(SSFQ2,2)$
202.
203. LET A=3/2$ LET B=0$ LET C=3$ LET D=0$
204. INTS(2)$
205. FCODE(SSFQ3,2)$
206.
207. LET A=1/2$ LET B=1$ LET C=1$ LET D=-2$
208. INTS(2)$
209. FCODE(SSFQ4,2)$
210.
211. LET A=1/2$ LET B=1$ LET C=1$ LET D=2$
212. INTS(2)$
213. FCODE(SSFQ5,2)$
214.
215. LET A=3/2$ LET B=0$ LET C=1$ LET D=0$
216. INTS(3)$
217. FCODE(SSDQ1,3)$
218.
219. LET A=1/2$ LET B=1$ LET C=1$ LET D=0$
220. INTS(3)$
221. FCODE(SSDQ2,3)$
222.
223. LET A=3/2$ LET B=0$ LET C=3$ LET D=0$
224. INTS(3)$
225. FCODE(SSDQ3,3)$
226.
227. LET A=1/2$ LET B=1$ LET C=1$ LET D=-2$
228. INTS(3)$
229. FCODE(SSDQ4,3)$
230.
231. LET A=1/2$ LET B=1$ LET C=1$ LET D=2$
232. INTS(3)$
233. FCODE(SSDQ5,3)$
234.
235. SHUT DATA;
236. SHUT FORTRAN;
237. END;

```


APPENDIX B

EXAMPLES OF GENERATED FORTRAN FONCTIONS

1. B_{31} for a pure sextupole (1.3.3 and 2.1.1)
2. B_{44} for an octupole superimposed to a focusing quadrupole
(1.3.4 and 2.1.3)

REAL FUNCTION SPS1()

C INTEGRATION OF

C Sqrt(BETAX)*SIN(MYX1+MYX)*BETAX

C IN THE CASE OF AN PURE SEXTUPOLE

C REAL VAL

REAL BX1,BY1,MX1,MV1,AX1,AV1,GX1,GY1,K,S
COMMON/PRM/BX1,BY1,MX1,MV1,AX1,AV1,GX1,GY1,K,S

VAL=(SQRT(BX1))*((4.*(GX1*BX1+2.*AX1**2)*S**2-3.*S**3*GX1*
. AX1-18.*S*AX1*BX1+12.*BX1**2)*SIN(MX1)+(3.*S**2*GX1-8.*S*
. AX1+6.*BX1)*COS(MX1)*S)/(12.*BX1)
SPS1=VAL
END

A END JOB 8686 GHPZ115 *LIST OFFLINE ROOM XC 5.34.51 PM 19 FEB 87 R51.PR1 SYS 7890 JOB 8686 CP= 1 END A*

```

DOUBLE PRECISION FUNCTION SOFQ4()
  INTEGRATION OF
  -SIN(2*(MVY1+MVY-MYX1-MYX)) * BETAY * BETAX
  IN THE CASE OF OCTUPOLE SUPERIMPOSED TO AN
  FOCUSING QUADRUPOLE
  IMPLICIT DOUBLE PRECISION (A-Z)
  DOUBLE PRECISION FI, VAL
  DOUBLE PRECISION BX1, BV1, MX1, MY1, AX1, AV1, GX1, GV1, K, S
  COMMON/PRM/BX1, BV1, MX1, MY1, AX1, AV1, GX1, GV1, K, S
  FI=SQRT(K)*S
  ANS7=((((AX1**2-1.)*BV1**2-AV1**2*BX1**2-4.*AV1*BV1*AX1*
  . BX1-4.*BV1*BX1*BX1**2)*K+(AX1**2-1.)*AV1**2-K**2*BV1**2*
  . BX1**2+4.*AV1*AX1-AX1**2+1.)*COSH(2*FI))+2.*((AX1**2-1.)*
  . BV1**2+AV1**2*BX1**2-BX1**2)*K-2.*(AX1**2-1.)*AV1**2-2.*K
  . **2*BV1**2*BX1**2-8.*AV1*AX1+2.*AX1**2-2.)*SIN(2*FI)*SIN(
  . 2*MX1)
  ANS6=2.*(((2.*BV1-BX1)*AV1*BX1-BV1**2*AX1-2.*BV1*AX1*BX1
  . ) *K+(AX1**2-1.)*AV1-AV1**2*AX1+AX1)*COSH(2*FI)+2.*(AV1*
  . BX1**2-BV1**2*AX1)*K-2.*(AX1**2-1.)*AV1+2.*AV1**2*AX1-2.*
  . AX1)*SIN(2*FI)-2.*((AV1*BX1**2-BV1**2*AX1)*K+(AX1**2-1.)*
  . AV1-AV1**2*AX1+AX1)*SINH(2*FI)+4.*((AX1**2-1.)*AV1-AV1**2
  . *AX1+AX1)*FI+4.*(AV1*BX1**2+BV1**2*AX1)*K*FI)*COS(2*MX1)-
  . 2.*(((AX1**2-1.)*BV1**2+AV1**2*BX1**2-BX1**2)*K+(AX1**2-
  . 1.)*AV1**2+K**2*BV1**2*BX1**2+4.*AV1*AX1-AX1**2+1.)*SINH(2
  . *FI)-2.*((AX1**2-1.)*AV1**2+4.*AV1*AX1-AX1**2+1.)*FI+2.*(
  . (AX1**2-1.)*BV1**2-AV1**2*BX1**2+BX1**2)*K*FI+2.*K**2*FI*
  . BV1**2*BX1**2)*SIN(2*MX1)+ANS7
  ANS8=COS(2*MX1)
  ANS5=ANS6*ANS8
  ANS11=-4.*(((AV1*BX1**2-BV1**2*AX1)*K+(AX1**2-1.)*AV1-AV1
  . **2*AX1+AX1)*SINH(2*FI)-2.*((AX1**2-1.)*AV1-AV1**2*AX1+
  . AX1)*FI-2.*(AV1*BX1**2+BV1**2*AX1)*K*FI)*SIN(2*MX1)*SIN(2
  . *MX1)
  ANS10=(2.*(((AX1**2-1.)*BV1**2+AV1**2*BX1**2-BX1**2)*K+(
  . AX1**2-1.)*AV1**2+K**2*BV1**2*BX1**2+4.*AV1*AX1-AX1**2+1.
  . ) *SINH(2*FI)-2.*((AX1**2-1.)*AV1**2+4.*AV1*AX1-AX1**2+1.)*
  . *FI+2.*((AX1**2-1.)*BV1**2-AV1**2*BX1**2+BX1**2)*K*FI+2.*
  . K**2*FI*BV1**2*BX1**2)-(((AX1**2-1.)*BV1**2-AV1**2*BX1**
  . 2-4.*AV1*BV1*AX1*BX1-4.*BV1*BX1+BX1**2)*K+(AX1**2-1.)*AV1
  . **2-K**2*BV1**2*BX1**2+4.*AV1*AX1-AX1**2+1.)*COSH(2*FI)+
  . 2.*((AX1**2-1.)*BV1**2+AV1**2*BX1**2-BX1**2)*K-2.*(AX1**2-
  . 1.)*AV1**2-2.*K**2*BV1**2*BX1**2-8.*AV1*AX1+2.*AX1**2-2.)*
  . SIN(2*FI))*COS(2*MX1)*SIN(2*MX1)+2.*(((2.*BV1-BX1)*AV1*
  . BX1-BV1**2*AX1-2.*BV1*AX1*BX1)*K+(AX1**2-1.)*AV1-AV1**2*
  . AX1+AX1)*COSH(2*FI)+2.*(AV1*BX1**2-BV1**2*AX1)*K-2.*(AX1
  . **2-1.)*AV1+2.*AV1**2*AX1-2.*AX1)*SIN(2*FI)*SIN(2*MX1)*
  . SIN(2*MX1)+ANS11
  ANS9=-((2.*((2.*BV1+BX1)*AV1*BX1+BV1**2*AX1-2.*BV1*AX1*
  . BX1)*K-(AX1**2-1.)*AV1+AV1**2*AX1-AX1)*COS(2*MX1))-((AX1
  . **2-1.)*BV1**2-AV1**2*BX1**2+4.*AV1*BV1*AX1*BX1+4.*BV1*
  . BX1+BX1**2)*K+(AX1**2-1.)*AV1**2-K**2*BV1**2*BX1**2+4.*
  . AV1*AX1-AX1**2+1.)*SIN(2*MX1))*COS(2*MX1)+2.*((2.*BV1+
  . BX1)*AV1*BX1+BV1**2*AX1-2.*BV1*AX1*BX1)*K-(AX1**2-1.)*AV1
  . +AV1**2*AX1-AX1)*SIN(2*MX1)*SIN(2*MX1)+((AX1**2-1.)*BV1
  . **2-AV1**2*BX1**2+4.*AV1*BV1*AX1*BX1+4.*BV1*BX1+BX1**2)*K

```

```

. +(AX1**2-1.)*AV1**2-K**2*BY1**2*BX1**2+4.*AV1*AX1-AX1**2+
. 1.)*COS(2*MX1)*SIN(2*MY1))*COS(2*FI)*SINH(2*FI)+ANS10
ANS4=ANS5+ANS9
ANS3=SQRT(K)*ANS4
ANS14=-2.*(2.)*((AX1**2-1.)*AV1+2.*AX1)+K*AV1*BX1**2)*
. COSH(2*FI)*BY1-(((AX1**2-1.)*BY1+2.*BX1)*AV1+AV1**2*AX1*
. BX1+2.*BY1*AX1-AX1*BX1)-3.*(AV1*BX1-BY1*AX1)*K*BY1*BX1)-((
. ((AX1**2-1.)*BY1-2.*BX1)*AV1-AV1**2*AX1*BX1+2.*BY1*AX1+
. AX1*BX1)-(AV1*BX1+BY1*AX1)*K*BY1*BX1)*SIN(2*FI)*SINH(2*FI
. ))*COS(2*MX1)*SIN(2*MY1)*K+2.*(2.)*((AX1**2-1.)*-2.*AV1*
. AX1)+K*BX1**2)*COSH(2*FI)*BY1+(2.)*(BY1-BX1)*AV1*AX1-(AX1
. **2-1.)*BY1+AV1**2*BX1-BX1)-3.*(BY1+BX1)*K*BY1*BX1)*SIN(2
. *MY1)*SIN(2*MX1)*K+2.*(2.)*(BY1+BX1)*AV1*AX1-(AX1**2-1.)*
. BY1-AV1**2*BX1+BX1)-(BY1-BX1)*K*BY1*BX1)*SIN(2*FI)*SIN(2*
. MY1)*SIN(2*MX1)*SINH(2*FI)*K
. ANS13=2.*(2.)*(BY1+BX1)*AV1*AX1-(AX1**2-1.)*BY1-AV1**2*
. BX1+BX1)-(BY1-BX1)*K*BY1*BX1)*SIN(2*FI)*SINH(2*FI)+2.*(2.)*
. AX1**2-1.)*-2.*AV1*AX1)+K*BX1**2)*COSH(2*FI)*BY1+(2.)*(BY1-
. BX1)*AV1*AX1-(AX1**2-1.)*BY1+AV1**2*BX1-BX1)-3.*(BY1+BX1)
. *K*BY1*BX1)*COS(2*MX1)+2.*(2.)*(AX1**2-1.)*AV1+2.*AX1)+K*
. AV1*BX1**2)*COSH(2*FI)*BY1-(((AX1**2-1.)*BY1+2.*BX1)*AV1+
. AV1**2*AX1*BX1+2.*BY1*AX1-AX1*BX1)-3.*(AV1*BX1-BY1*AX1)*K
. *BY1*BX1)*SIN(2*MX1)-(((AX1**2-1.)*BY1-2.*BX1)*AV1-AV1**
. 2*AX1*BX1+2.*BY1*AX1+AX1*BX1)-(AV1*BX1+BY1*AX1)*K*BY1*BX1
. )*SIN(2*FI)*SIN(2*MX1)*SINH(2*FI))*COS(2*MY1)*K+ANS14
. ANS12=-2.*(2.)*((AX1**2-1.)*BY1+2.*BX1)*AV1+AV1**2*AX1*BX1
. +2.*BY1*AX1-AX1*BX1)-(AV1*BX1-BY1*AX1)*K*BY1*BX1)*COSH(2*
. FI)-2.*(AV1**2*AX1+2.*AV1-AX1)*BX1+2.*K*BY1**2*AX1*BX1)*
. SIN(2*MX1)-(((2.)*(BY1-BX1)*AV1*AX1-(AX1**2-1.)*BY1+AV1**2
. *BX1-BX1)+(BY1+BX1)*K*BY1*BX1)*COSH(2*FI)-2.*(AV1**2-2.*
. AV1*AX1-1.)*BX1+2.*K*BY1**2*AX1*BX1)*COS(2*MX1)-((
. ((AX1**2-1.)*BY1+2.*BX1)*AV1+AV1**2*AX1*BX1+2.*BY1*AX1-
. AX1*BX1)-(AV1*BX1-BY1*AX1)*K*BY1*BX1)*COSH(2*FI)-2.*(AV1
. **2*AX1+2.*AV1-AX1)*BX1+2.*K*BY1**2*AX1*BX1)*COS(2*MX1)*
. SIN(2*MY1)-(((2.)*(BY1-BX1)*AV1*AX1-(AX1**2-1.)*BY1+AV1**2
. *BX1-BX1)+(BY1+BX1)*K*BY1*BX1)*COSH(2*FI)-2.*(AV1**2-2.*
. AV1*AX1-1.)*BX1+2.*K*BY1**2*AX1*BX1)*SIN(2*MY1)*SIN(2*MX1))*
. COS(2*FI)*K+ANS13
. ANS2=ANS3+ANS12
. ANS1=-ANS2
. VAL=ANS1/(16.*K**3*BY1*BX1)
. SOFQ4=VAL
. END

```