1 of 3

SYNCH

RECEIVED
MAR 23 1994
OSTI

USER'S GUIDE 1993

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

SYNCH

**A PROGRAM FOR DESIGN AND ANALYSIS
OF SYNCHROTRONS AND BEAMLINES**

USER'S GUIDE

A. A. Garren and A. S. Kenney

E. D. Courant

A. D. Russell

M. J. Syphers

CONTENTS

1.0 INTRODUCTION	1
1.1 Applications	1
1.2 SYNCH Language	1
1.3 History	2
1.4 Documentation	2
2.0 COMPUTER ENVIRONMENT	3
2.1 Program Organization	3
2.2 Files	3
3.0 CALCULATIONS PERFORMED	5
3.1 Matrix Definition and Operations	5
3.2 Betatron Function Calculations	7
3.3 Particle Beam Calculations	7
3.4 Closed Orbit Calculations	7
3.5 Particle Tracking	7
3.6 Non-linear Transformation Calculations	8
3.7 Element Misalignment Calculations	8
3.8 Orbit Correction Calculations	9
3.9 Fitting Routines	9
4.0 SYNCH STATEMENTS	11
4.1 Statements	11
4.2 Conventions	11
4.3 Statement Diagram Conventions	11
4.4 General Statement Format	12
5.0 SYNCH COMMANDS	13
ACT - Activate Statement(s)	13
BEP - Plot Lattice Functions	15
BEST - Periodic Beta Functions, Stored for Plotting	17
SYNCH USER'S GUIDE 1993	iii

SHF7 – Define a 7×7 Shift Matrix	153
SIZE – Define Size of Matrices	155
SMIN – Access MINUIT from SYNCH	157
SOL – Solenoid	159
SOLV – General Fitting Routine	161
STOP – End of Job	167
SUB – Define a SYNCH Subroutine	169
SUM – Scalar Summation	171
SXTP – Sextupole Definition	173
TRK – Track Particles Through Beamline	175
TRKB – Track Betatron Functions	177
TRKE – Track Beam Envelopes	179
TRKM – Track Through Non-linear Elements	181
UPDAT – Create New Input File	183
VAR – Define Variable by Current Value in Command	185
VEC – Vector Definition	187
VPAR – Loop Through SYNCH Subroutine on a Diagonal	189
WBE – Write Betatron Functions of Matrices	191
WFL – Print Out Internal Storage	193
WMA – Write Matrices	195
= – Equate to Floating Point Number	197
** – Raise a Matrix to a Power	199
Period (.) – Comment	201

6.0 MISCELLANEOUS FEATURES 203

6.1 The Negative of a Symbolic Floating Point Number	203
6.2 Symbolic Entry for Inverse of a Matrix	203
6.3 Internally Defined Matrices	203
6.4 Predefined Constants	204
6.5 Initially Deactivated Statements	204

7.0 NON-LINEAR TRANSFORMATIONS 205

7.1 MAP-Point Transformations	205
7.2 DEQ-Differential Equation Transformations	206
8.0 MATHEMATICAL FORMULATION	209
8.1 Transfer Matrices and Beamlines	209
8.2 Linear Elements	210
8.3 Multipole Magnets	215
8.4 Betatron Functions	216
8.5 Particle Beam Envelopes	220
8.6 Closed Orbit Calculations	221
8.7 Particle Tracking	222
8.8 Magnet Misalignment Calculations	223
8.9 Parameterization of Transport and Period Matrices with X-Y Coupling	224
REFERENCES	229
APPENDIX A. FILES	A-1
APPENDIX B. SAMPLE RUNS	B-1
B.1 Standard Tevatron Cell	B.1-1
B.2 Fitting of Tunes in a Phase Trombone	B.2-1
B.3 Misalignment Calculation	B.3-1

BETA – Betatron Function of a Matrix Representing a Period	19
BMIS – Begin Misalignment Mode	21
BML – Beam Line Definition	23
BVAL – Particle Beam Definition	27
C – Comment	29
CALC – Calculator Simulator	31
CALL – Invoke a SYNCH Subroutine	35
CYA – Beta Functions and Matrices of Cyclic Permutations of Elements	37
CYAE – Beam Envelope Calculation	39
CYC – Periodic Beta Functions Through a Lattice Period	41
CYEM – Beam Emittance Calculation	43
DEACT – Deactivate Statement(s)	45
DEQ – Transformations Defined by Differential Equation Integration	47
DRF – Drift Space Definition	49
ECHO – Print Out Input Statements	53
EMIS – End Misalignment Mode	55
END – Mark End of SYNCH Subroutine	57
EQU – Equate Matrices	59
FITB – Fit Betatron Functions	61
FITQ – Fit Betatron Tunes	63
FITR – Fit matrix elements	65
FITV – Vary Two Parameters to Fit Values of Beam Coordinates	67
FXPT – Closed Orbit Calculation	69
IBET – Enter Initial Values of Betatron Functions	73
INCR – Increment an Input Parameter	75
INV – Invert a Matrix	77
INV2 – Rotate a Beamline 180° and Reflect It	79
IOUT – Write a compressed lattice file	81
KEEP – Save selected files	83
KICK – Dipole Kicker Magnet or Field Error Definition	85
LIST – Define List of Elements	87

MAG – Magnet Definition	89
MAGS – Magnet definition with errors	93
MAGV – Define a Vertically Bending Magnet	95
MAP – Non-linear Transformation Definition	97
MAT – General Matrix Definition	99
MAT3 – Define Transfer Matrix	101
MESH – Loop Through SYNCH Subroutine Variables	103
MMM – Matrix Multiplication	105
MOVE – Perform an Element Misalignment	107
MXV – Matrix-Vector Multiplication	109
NECHO – Suppress Printing of Input Statements	111
NPOL – Define <i>N</i> -pole Magnet	113
OPEN – Open special output files	115
ORBC – Calculate Closed Orbit Corrections	117
PAGE – Skip to Start of Next Page	119
PARA – Define a Parameter Value	121
PBML – Print Beamline	123
PCYC – Select Restricted CYC Print List	125
PRNT – Print Element Parameter	127
PRTV – Print a List of Vectors	129
PRV7 – Print a List of Particle Vectors	131
PVEC – Define Particle State Vector	133
RAND – Generate Random Number	135
REF – Matrix Reflection	137
REM – Insert Remark in Output Listing	139
REPL – Replace One Parameter With Another	141
ROT – Rotate a Beamline Element	143
ROTZ – Define Rotation Matrix	145
RUN – Start of Run	147
SELCT – Insert Delimiter Marker	149
SHF – Define a 3×3 Shift Matrix	151

1.0 INTRODUCTION

SYNCH is a computer program for use in the design and analysis of synchrotrons, storage rings, and beamlines. It has a large repertoire of commands that can be accessed in a flexible way. The input statements and the results of the calculations they invoke are saved in an internal database so that this information may be shared by other statements. **SYNCH** is the first accelerator program to organize its input in the form of a language. The statements, which resemble sentences, provide a natural way of describing lattices and invoking relevant calculations. The organization of the program is modular, so that it has been possible to expand its capabilities progressively.

1.1 Applications

The lattice is described by statements defining the beamlines and the elements—drifts, dipoles, quadrupoles, sextupoles, other non-linear elements, and other beamlines—of which they are composed. Beamline statements may refer to the superperiods, cells, transport lines, or substructures of these. Each element-defining statement results in the calculation of the corresponding linear transfer matrix. These matrices can be manipulated in various ways; for example, they can be multiplied together to construct the matrices corresponding to a particular beamline.

Having generated the lattice of a complete ring, superperiod, cell, or transport line, one may obtain linear properties, especially the betatron functions, of the corresponding beamline; closed orbits and linear properties corresponding to momentum deviations and/or magnet misalignments can be determined; and particles with arbitrary initial conditions can be tracked repetitively around the machine. One may also calculate emittances, damping times, and other properties of electron rings.

To design machines, the program adjusts the lengths or strengths of certain elements in order to obtain specified orbit properties. Typical examples are:

Gradients of a cell's quadrupoles may be adjusted to obtain specified cell phase advances.

Lengths and/or gradients of quadrupoles and drifts in a long straight section insertion may be varied to obtain a match of the betatron functions to those of the adjacent cells and to produce a low-beta waist at its interaction point.

1.2 SYNCH Language

The input file for a **SYNCH** run consists of a user determined sequence of statements, each containing a label or name, a command keyword specifying an operation or definition, and data. The data may be numerical or the names of previously defined statements. Most statements either define parts of a machine, invoke calculations, or both.

In order to scan systems over ranges of parameters, a set of statements can be placed together in a subroutine-like block that may be called sequentially with different parameter values. These blocks are also used by the fitting commands.

1.3 History

The first versions of SYNCH were programmed for IBM 704, 7044, and 709 computers by J. Eusebio and A. Garren in 1964-5, primarily for use in the 200 BeV Design Study. These did not contain the general fitting programs, but instead partly analytic routines were included for designing long straight sections of the type used in the Fermilab Main Ring (and later in the Tevatron). In 1966-7 the program was rewritten for the CDC 6600 by Garren and A. Kenney and later adapted to run on the CDC 7600.

With the aid of B. Miller of IBM, the CDC 7600 version was converted to run on IBM 360 computers in 1968. New facilities were subsequently added to the CDC version by Garren and Kenney, such as a general fitting routine that uses the CERN program MINUIT, and calculations of electron beam properties. Other capabilities were implemented by Courant, such as polarization effects and thin-lens multipole calculations. The IBM version was also updated from time to time, with the help of B. Wu, A. King and others at SLAC. In 1979 the current IBM version was implemented to run on a Fujitsu computer by K. Chiba at the INS laboratory of the University of Tokyo. The IBM version was compiled on a VAX 11-780 by W. Trecziak of the University of Wisconsin, which was subsequently updated by Kenney to correspond to the latest CDC version, and then used as the basis for another VAX version conforming to the ANSI FORTRAN 77 standard.

In the period 1985-92 a new, machine-independent version of SYNCH was developed using the Code Management System (CMS), which permits one to maintain functional identity between versions of the code running on different platforms.

1.4 Documentation

The first versions of the program were documented in a Berkeley Internal Report by Eusebio, Garren, and Kenney.^[1] From time to time this document was informally updated.

The first edition of the present document, written primarily by M. Syphers,^[2] was based on the rather fragmentary documentation then existing together with comments included in the FORTRAN listing. This second edition refers to the current standard FORTRAN 77 version of the program. It documents features in the program not discussed in the previous edition of the manual as well as features added to the program subsequently.

2.0 COMPUTER ENVIRONMENT

2.1 Program Organization

SYNCH has been organized to be used in a very flexible manner. The user builds the synchrotron structure by a series of input statements that describe the accelerator. These statements comprise a special-purpose language.

Each statement contains three components—a name, or label; a command; and a set of data. The name is an arbitrary set of characters which can be used as data in subsequent statements to refer to the element defined by the statement. The command is a particular set of characters that characterize the nature of the element described and of the calculations to be performed. The data completes the specification of the element and provides input parameters for the command. The statement, together with quantities calculated, such as matrices, are stored for use by subsequent statements.

The statements are processed sequentially as entered in the input file. Labels or names that are referred to in a statement must have been previously defined if the command invokes any calculations.

A SYNCH subroutine is a set of statements that are grouped together by use of the SUB and END commands for execution, when invoked by the CALL command, as a unit. A SYNCH subroutine must occur in the input file before any statement that invokes it.

2.2 Files

The SYNCH program is written in FORTRAN. Input and output are handled by associating files with the FORTRAN logical unit numbers used in the READ and WRITE statements. The details of establishing these associations are unique to each system (e.g., VAX, Cray, or Sun workstations). Input and output files are assigned to logical units 2 and 3 respectively. A complete description of files generated is given in Appendix A.

3.0 CALCULATIONS PERFORMED

This section describes the calculations that may be performed. Detailed descriptions of individual commands are left to Chapter 5, **SYNCH** Commands. Details concerning the mathematical methods employed by the program, such as descriptions of beamline element matrices, may be found in Chapter 8, Mathematical Formulation. A complete list of all commands organized by topic is given in Table 3.1.

3.1 Matrix Definition and Operations

SYNCH allows the definition of various standard beamline elements which act on the particle coordinates. Transfer matrices are calculated for linear elements upon execution of their defining statements. The **MAG** statement defines a magnet which could be bending, focusing, or a combination of both. The bending, if any, is in the horizontal plane. The **MAGV** statement is used to define a magnet which bends in the vertical plane. The **DRF** statement is used to define a drift region. Other elements include the sextupole magnet (**SXTP**), the n -pole thin lens (**NPOL**), and the delta-function dipole kick using the **KICK** statement. The **MAG**, **MAGV**, **DRF**, and **KICK** commands generate matrices which act on a state vector $(x, x', y, y', ds, dp/p)$ containing the position and slope in each transverse plane and the momentum error, dp/p , of the particle. The effects of the **SXTP** and **NPOL** commands on these vectors are not linear and are handled by separate subroutines in **SYNCH**. The user may also define linear or non-linear elements by use of the **MAT**, **MAT3**, **DEQ** and **MAP** statements.

Once defined, various operations may be performed upon the matrices of the linear elements. In particular, **MMM** multiplies two or more matrices and stores the result in a newly defined matrix; **INV** determines the inverse of a matrix; **MAGS** and **MOVE** misalign the element; **ROT** rotates a beam element about the beam direction; **REF** generates the matrix of the reflection of a beamline, and the ****** command allows the repeated multiplication of a matrix with itself. A particular sequence of beamline elements, which may contain linear and non-linear elements, may be created using the **BML** statement. This statement may be used recursively, i.e., a **BML** statement may contain as part of its defining sequence the names of other **BML** statements.

The units used for input are arbitrary, but must be consistent throughout. For instance, one could enter all lengths in meters, all field strengths in Tesla, Tesla/meter, and so forth. In this case, the betatron functions that are calculated will be in units of meters, etc.

Table 3.1. SYNCH Program Commands, by Topic.

Program Control	ACT	DEACT	RUN	SIZE	STOP
SYNCH Subroutines	CALL SUB	END VPAR	INCR	MESH	REPL
Mathematical Operations	CALC PARA	(SIN RAND	SQRT SUM	1/X VAR	etc.) =
Beamlines	BML	LIST			
Element Definitions	DEQ MAP	DRF NPOL	KICK SOL	MAG SXTF	MAGV
Operations on Transfer Matrices	EQU REF	INV ROT	INV2 ROTZ	MMM **	MXV
General Matrix/Vector Definitions	MAT	MAT3	VEC		
Betatron Function Calculations	BETA TRKE	CYA	CYC	IBET	TRKB
Particle Beam Calculations	BVAL	CYAE	CYEM		
Fitting Routines	FITB SOLV	FITQ	FITR	FITV	SMIN
Closed Orbit Determination and Particle Tracking	FXPT	PVEC	TRK	TRKM	
Element Misalignments	BMIS SHF7	EMIS	MAGS	MOVE	SHF
Orbit Correction	ORBC				
Plotting	BEP	BEST	TRKB		
Files	IOUT	KEEP	OPEN	UPDAT	SELCT
Output Statements	C PBML REM	ECHO PCYC WBE	NECHO PRNT WFL	P PRTV WMA	PAGE PRTV7

3.2 Betatron Function Calculations

The linear betatron functions describe the amplitude and phase of the transverse betatron oscillations throughout an accelerator and/or beamline. The formal language of betatron motion used is that of Courant and Snyder.^[4] See also Section 8.4, Betatron Functions.

The value of each betatron function corresponding to a particular matrix may be obtained by use of a **BETA** statement. All of them may be printed out by use of the **WBE** command. Given an initial set of betatron functions, which may be input directly by the user (**IBET**) or obtained from a previously defined matrix, the betatron functions may be tracked through a beamline by using the **TRKB** command. The betatron functions at each point of a periodic section of a circular accelerator are generated by the **CYC** command.

3.3 Particle Beam Calculations

The beam envelopes are calculated by **CYAE** or **TRKE** using the betatron functions together with specified values of the emittances and momentum spread of the beam specified by **BVAL**. For electron machines, one can compute natural emittances, damping, rf quantities, etc. using the **CYEM** command.

3.4 Closed Orbit Calculations

SYNCH has the ability to compute the closed orbit through a circular accelerator or beamline period for a particle of any momentum. The accelerator may consist of linear and/or non-linear elements. The user must input an initial guess for the closed orbit at some azimuth using a **PVEC** statement. Then, with the **FXPT** statement, the closed orbit at that point is found and propagated through the entire accelerator. The transfer matrices of the beamline elements are linearized about this closed orbit and the betatron functions from these new matrices are printed out along with the closed orbit.

The **FXPT** calculation uses 7×7 matrices and hence allows for the study of horizontal-vertical coupling of the betatron motion. The single-turn transfer matrix for the point in question is thus printed out as well as the eigenvalues and eigenvectors of the 4×4 submatrix representing x - y motion. The eigenvectors may be tracked through the accelerator.

3.5 Particle Tracking

A particle may be tracked repeatedly through a beamline using the **TRK** command. The initial trajectory and dp/p of the particle is input with the **PVEC** statement. The trajectory of the particle may be output at all points in the beamline each transit or at selected locations every so many transits. The beamline may consist of linear and/or non-linear elements.

3.6 Non-linear Transformation Calculations

Transformations, other than the standard linear ones, may be used in certain tracking and other operations by means of the **MAP** or **DEQ** statements. The former define point transformations, the latter those that result from integration of differential equations. Each transformation is implemented through a corresponding FORTRAN subroutine. **SYNCH** includes twenty subroutines of each type; ten reserved for built-in subroutines, and ten for user-defined ones. Some of the built-in subroutines are implemented, the rest of these and all of the user-defined ones are included in the form of dummy subroutines.

MAP-Point Transformations

SYNCH has 10 subroutines **MAP0**, ..., **MAP9** reserved for built-in transformations and 10 subroutines **MAP10**, ..., **MAP19** for user-defined transformations. Any of the dummy user subroutines may be replaced by a FORTRAN subroutine that produces the desired transformation of particle coordinates, compiled and linked together with a **SYNCH** object module, thus producing a new executable module.

To access a **MAP** subroutine, the user defines a beamline element with the corresponding **MAP** command and includes it in the appropriate beamline defined by a **BML** command. Whenever this **MAP** element is encountered while tracking, the current particle state vector and parameters specified in the **MAP** statement are passed to the respective subroutine, which is then executed.

DEQ-Differential Equation Transformations

Parallel to the **MAP** subroutines, **SYNCH** has 10 subroutines **DEQ0**, ..., **DEQ9** reserved for built-in transformations and 10 subroutines **DEQ10**, ..., **DEQ19** reserved for user-defined transformations. A **DEQ** subroutine is invoked by the differential equation routine in **SYNCH** for each integration step.

To access a **DEQ** subroutine, one defines a beamline element using a **DEQ** statement. When this element is encountered while tracking, the particle coordinates and the parameters specified in the **DEQ** statement are passed to the respective internal **DEQ** subroutine and to the differential equation routine, which performs the integration.

Use of the **MAP** and **DEQ** commands is described in Chapter 5, and examples of the internal FORTRAN subroutines are shown in Chapter 7.

3.7 Element Misalignment Calculations

The effects of magnet misalignments on the closed orbit of the accelerator may be studied. There are two possible approaches to the problem. One approach involves the **MAGS** command, which defines a magnet with transverse misalignments. The **BMIS** command (Begin MISalignment mode) must first be issued. In this mode, the new closed orbit due to the misalignments is calculated using a **CYC** statement instead of the dispersion function. The **EMIS** command will end the misalignment mode.

The second approach requires the use of the **MOVE** statement. Here, rotational misalignments about the s-axis as well as transverse misalignments may be invoked. This method uses the **FXPT** statement to find the new closed orbit and may be used to study coupling effects of the transverse motion in each plane.

For more information concerning these methods, see Section 8.8.

As aids for misalignment calculations, the **LIST** statement and the **RAND** statement were created. The **LIST** statement allows the user to generate a list of elements which can be used to substitute successive calls for a particular transfer matrix. The **RAND** statement generates a random number uniformly distributed on the interval $[-1/2, 1/2)$.

3.8 Orbit Correction Calculations

Using the **ORBC** statement, the correction of the closed orbit through an accelerator with field errors may be calculated. The name of a **FXPT** statement is used to define the initial orbit. The beamline used in the **FXPT** statement contains the name of the elements at which the displacements are measured and the name of the elements which are used for corrections. The optimized correction element strengths are calculated and printed out.

3.9 Fitting Routines

Several fitting routines are present allowing one to produce desired values of betatron functions at specified points. For instance, **FITQ** varies the parameters of specified magnets to create specific horizontal and vertical phase advances for a particular portion of a beamline. Likewise, **FITB** may be used to fit other betatron functions to desired values. The **SOLV** command is more general in that it can vary any number of parameters subject to constraints in order to generate desired values of several betatron functions at once.

The **SOLV** command uses the MINUIT^[6] minimization routines. All MINUIT commands may be accessed by using the **SMIN** command.

4.0 SYNCH STATEMENTS

4.1 Statements

SYNCH statements are implemented as 80-character records structured with specifically designated fields. Data must be appropriately right or left justified in their fields.

4.2 Conventions

A **SYNCH** input file is made up of a set of statements beginning with a **RUN** statement and ending with a **STOP** statement. The statements in between are composed of various parts: a label (or name), a command, and assorted input parameters which depend upon the nature of the statement. The purpose of Chapter 5 is to describe the statement associated with each command. The descriptions include a Statement Diagram, definitions of all the associated parameters, and a short paragraph about the command and its uses. Examples are included in many of the descriptions. Examples of runs are given in Appendix B. The commands are arranged alphabetically. For a cross-reference of commands organized by categories, see Table 3.1.

4.3 Statement Diagram Conventions

Each description of a command is preceded by a Statement Diagram. The conventions used in these diagrams to describe various types of elements, variables, and commands are as follows:

1. Input is fixed format. Data items must be aligned in specific fields. The lengths of the fields and the justification within the fields are determined by the type of data item (integer, floating point, character).
2. Items in upper case and special symbols (e.g., =) are keywords and must appear exactly as shown.
3. Items in lower case represent data to be supplied by the user.
4. Continuation records: certain commands require additional lines containing input data. Other commands, such as **BML**, may require them if there is too much data to fit on one line. These continuation lines must all be blank in columns 1-20. Data begins in column 21.

Note that **SYNCH** distinguishes between upper and lower case characters. Command names must always be entered in upper case. Case must always be used consistently for named data: **XYZ**, **xyz**, and **XyZ** are treated as three distinct names.

4.4 General Statement Format

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  cmd  mmm nnn  ---  ---  ---  ---  data  ---  ---  ---  ---
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

COLUMNS

- 1 Used for the commands or options C, P, "-", and ".".
- 2-6 **name**—*left justified*. Label of the statement. Used by other statements to refer to this one. In general, names are unrestricted; however there must be an exact correspondence between the original name definition and subsequent references to it. If **cmd** is **PARA** or **CALC**, the first character should be a letter.
- 8-12 **cmd**—*left justified, upper case*. The keyword of the particular command.
- 13-15 **mmm**—*right justified* Integer, up to 3 digits. Used to specify options.
- 17-19 **nnn**—*right justified* Integer, up to 3 digits. Used to specify options.
- 21-80 **data**—Input data for the particular command. The data may be alphanumeric, floating point, or integer depending upon the command:

alphanumeric—(CH) Names or labels, *left justified* in 5-column fields: 21-25, 26-30, 31-35, etc.

floating point—(FP) Decimal numbers, entered in 10-column fields 21-30, 31-40, etc. The decimal point must be included and the string can be placed anywhere within the field.

integer—(IN) Integer values, *right justified* in 5-column fields 21-25, 26-30, 31-35, etc.

symbolic floating point—(CH) Label of a floating point parameter previously defined by a **CALC**, **PARA**, **SUM**, **VAR**, or "=" statement. The character string must be *left justified* in the same 10-column data fields which would otherwise contain floating point numerical input.

special characters—The characters +, -, *, /, and # have special meanings when used within labels. In general, the characters +, -, * and / should not be used as the first character in a label. When used as the first character in a label the # symbol has special meaning to the **BEST**, **CY...**, and **FXPT**, commands. See the **PCYC** command for details. Also see Chapter 6 for use of other special symbols.

5.0 SYNCH COMMANDS

ACT – Activate Statement(s)

The ACT statement activates a specified list of SYNCH statements.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      ACT      m      stm1 stm2 stm3 ...  stmK
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) Number of statements to activate.
 = 0, or blank, Activate the command statements specified in the data fields. The number of statements activated is determined from the list. The list may be continued on successive lines.
 > 0, Activate **m** consecutive statements, beginning with **stm1**. Only one statement, **stm1**, should be named in the data field.

stm1, ... (CH) Name(s) of statement(s) to be activated.

ACT restores (activates) execution of a specified set of SYNCH command statements. During execution of the program, only active commands are executed.

The ACT statement acts only on previously defined inactive statements. The effect of the ACT statement is seen only if the newly activated statement is subsequently executed. This can occur only if the statement is contained in a SYNCH subroutine which is invoked after the ACT statement is processed.

SEE ALSO: Section 6.5, Initially Deactivated Statements
 DEACT

BEP – Plot Lattice Functions

The **BEP** statement creates a plot of the β (amplitude) and η (dispersion) functions through a beamline and a schematic diagram of the magnet layout.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  BEP      m      n ncy  mode ip1  ip2  bmax      etamn      etamx      iop1 iop2
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Selects plotting of β or $\sqrt{\beta}$.
 Sets number of tick marks for the left-hand y axis.
 ≥ 0 or blank, Plot β .
 < 0 , Plot $\sqrt{\beta}$.
 $= \pm 99, 0$, or blank, The program selects the number of tick marks to draw.
 $\neq \pm 99, 0$, or blank, Draw tick marks to divide the axis into $|m|$ intervals.

n (IN) Sets number of tick marks for the right-hand y -axis (dispersions, η_x and/or η_y).
 $= 0$ or blank, The program selects the number of tick marks to draw.
 $\neq 0$, Draw tick marks to divide the axis into $|n|$ intervals.

ncy (CH) Name of the **BEST** or **TRKB** command which computed and stored the beta functions.

mode (IN) Define horizontal plot layout.
 $= 0$ or blank, The x-axis markers are determined by the path-length stored along the beamline.
 $= 1$, Place the zero-point on the x-axis at the left-hand end.
 $= 2$, Place the zero-point in the middle of the x-axis.

ip1, ip2 (IN) The data are plotted from positions ip1 through ip2 in the beamline.
 $= 0$ or blank, Plot for the entire beamline.
 $\neq 0$, Plot from beamline position ip1 to position ip2. If either of ip1 or ip2 is omitted or 0, default to beginning or end, respectively, of beamline.

bmax (FP) Optional. Plot limit. Maximum β function value.

etamn (FP) Optional. Plot limit. Minimum dispersion function value.

BEP

etamx (FP) Optional. Maximum dispersion function value.

Note: If a value is entered for any one of **bmax**, **etamn** or **etamx** then values must be entered for all three.

iop1 (IN) Plot format selector.

= 0 or blank, Annotates axes with labels and values.

= 1, Axis labels are omitted but values are marked on the axes.

= 2, There is no axis annotation. Labels and values are omitted.

iop2 (IN) Select curves to plot.

= 0 or blank, Plot all four functions β_x , β_y , η_x , and η_y . The curves are distinguished by line type and, where available, by color, as follows:

β_x -Solid (red) line,

β_y -Chain dash (blue) line,

η_x -Dash (green) line,

η_y -Chain dot (green) line.

= 1, 2, 3, or 4, Plot β_x , β_y , η_x , or η_y , respectively. The single curve is plotted as a solid black line.

The **BEP** statement creates a plot of the amplitude (β) and/or dispersion (η) function(s) through a (portion of a) beamline. The data values plotted are those calculated and stored by the most recently executed **BEST** or **TRKB** command. The name of this most recently executed **BEST** or **TRKB** command must be entered in the **ncy** field of the **BEP** command.

To obtain plots, **SYNCH** must be run interactively. A dialogue is presented on the screen enabling the user to select on- or off-line output. On-line output immediately draws the plots to the screen. If off-line output is selected, a file will be created which the user can subsequently route to a hard-copy device.

The axis ranges are determined by the parameters **bmax**, **etamn**, and **etamx** if they are given, and by the data otherwise. Axis tick marks are generated internally based on the total range. The range for the horizontal axis is adjusted to maximize the length of the plot consistent with the value of **mode**.

SEE ALSO: **BEST**
TRKB

BEST - Periodic Beta Functions. Saved for Plotting

The **BEST** command is used to create lattice function data in a form to be plotted by **BEP**. The **BEST** statement is equivalent to **CYC** in input format and function. In addition, it saves data for plotting by **BEP**.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  BEST   m   n bmln ampl sx1  sx2  cx          cy          hdr
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

SEE ALSO: **BEP**
CYC
PCYC

(

BETA - Betatron Function of a Matrix Representing a Period

The **BETA** command assigns the value of one of the periodic betatron functions of a beamline to a named variable.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  BETA   m      mtrx
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name of the variable to which the value is assigned.

m (IN) Index value specifying the betatron function to be used.

m = 1, μ_x	m = 11, μ_y
= 2, β_x	= 12, β_y
= 3, α_x	= 13, α_y
= 4, γ_x	= 14, γ_y
= 5, η_x	= 15, η_y
= 6, η'_x	= 16, η'_y

mtrx (CH) The name of the matrix defined by the beamline and calculated by an MMM or equivalent (e.g., **CYC**) command.

The matrix **mtrx** has been calculated using an MMM or equivalent command for a beamline defined by a **BML** statement. The betatron functions are computed for the beamline assuming periodic boundary conditions: the function values are equal at the beginning and end of the line. The **BETA** command assigns to the variable **name** the value (at the ends of the beamline) of the **m**-th betatron function derived from the matrix **mtrx**.

BMIS – Begin Misalignment Mode

The **BMIS** command invokes a special mode of operation of **SYNCH** to calculate the orbit distortions resulting from magnet misalignments.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      BMIS
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

The **BMIS** command enables a special mode of operation of **SYNCH** which is used to calculate the orbit distortions which result from magnet misalignments. While in this special mode, magnets defined by **MAG** statements are assigned horizontal and vertical transfer matrix elements $M_{13} = M_{23} = 0$. The effects of magnet misalignments can then be simulated by appropriate translations of the particle's entrance and exit coordinates for the magnetic elements concerned.

The **BMIS** mode of operation is terminated and normal program operation restored by an **EMIS** statement.

SEE ALSO: **EMIS**
MAGS
 Section 8.8, Magnet Misalignment Calculations

BML - Beam Line Definition

The **BML** statement defines a beamline as a sequence of elements specified in the order in which they are encountered by the beam.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  BML      m      a1    a2    a3    a4    ...    ak
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name of the beamline.

m (IN) Reflection option selector.
 = 0 or blank, No effect.
 = -1, Create a reflected version of the defining beamline.

a1, ... (CH) Names of elements or other beamlines.

The **BML** command defines a beamline as a series of elements, **a1**, **a2**, ..., **ak**, entered in order as "encountered by the beam." The individual elements may be primary elements (drifts, magnets, etc.), matrices representing other beamlines (see, e.g., **MMM**), or beamlines defined by other **BML** commands. (The name of the beamline being defined may not itself appear in the defining list: recursive definitions are not supported.) **BML** statements may occur in the input file before the elements making up the beamline are themselves defined. The **BML** command merely sets up a list of elements; it performs no calculations.

Blank fields in the input record are ignored. The beamline definition may be continued on successive lines.

When one beamline is included in the definition of another, the included beamline is expanded in place. The effect is as if the elements in the included beamline had been explicitly named. For example,

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
.OBB  BML           0    B    B
.C    BML           .OBB QD .OBB QF
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

is completely equivalent to

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
.C    BML           0    B    B    QD  0    B    B    QF
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

for the beamline **.C**.

BML

Included BML's are properly expanded into their constituent elements, but an included MMM statement will be used exactly as it stands: as a single element. Thus, building on the previous example, adding

```
-----1-----2-----3-----4-----5-----6-----7-----8
OBB  MMM          .OBB
-----1-----2-----3-----4-----5-----6-----7-----8
```

and defining

```
-----1-----2-----3-----4-----5-----6-----7-----8
*C   BML          OBB QD  OBB QF
-----1-----2-----3-----4-----5-----6-----7-----8
```

forces *C to be used exactly as written: as a beamline containing just four elements including the lumped element OBB. *C is NOT expanded to be equivalent to .C.

N (a positive integer) repetitions of a set of elements may be obtained by placing "N(" anywhere in the 5-column field to the left of the set and ")" anywhere in the field to the right. The other spaces in the 5-column fields must be blank. As an example, consider the following set of nested BML's

```
-----1-----2-----3-----4-----5-----6-----7-----8
CELL BML          QD  2( B1  )  QF  2( B1  )
STSS BML          0   QDL LS  QFL 0
SPRD BML          STSS 5( CELL  )
-----1-----2-----3-----4-----5-----6-----7-----8
```

and the equivalent expanded form

```
-----1-----2-----3-----4-----5-----6-----7-----8
SPRD BML          0   QDL LS  QFL 0   QD  B1  B1  QF  B1  B1  QD
                   B1  B1  QF  B1  B1  QD  B1  B1  QF  B1  B1  QD
                   B1  B1  QF  B1  B1  QD  B1  B1  QF  B1  B1
-----1-----2-----3-----4-----5-----6-----7-----8
```

The beamline SPRD is exactly the same for both definitions; it comprises the 35 elements listed explicitly in the second example.

Another example of the use of repetition is provided by the following.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
.B5    BML          OO    4(  B    0    )
.FD    BML          QF    .B5  QD
.DF    BML          QD    .B5  QF
.C     BML          .FD  .DF
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

The beamline .C represents a FODO-type cell. The equivalent expanded definition is

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
.C     BML          QF  OO  B    0    B    0    B    0    B    0    QD
          QD  OO  B    0    B    0    B    0    B    0    QF
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

A reflected beamline is one in which the elements are traversed in the reverse order from the defining beamline. Reflection is obtained by setting $m = -1$. (Parameter m need not be entered except to define a reflected beamline.) Note, however, that **SYNCH** will not reflect any of the constituent elements. The constituent elements must all be self-reflecting—magnets with equal edge angles, drifts, or MMM's of symmetric beamlines—or the results will be incorrect.

BVAL - Particle Beam Definition

The **BVAL** command specifies values of the emittances of a particle beam. The emittances are used by subsequent **CYAE** and **TRKE** statements for beam envelope calculations.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  BVAL  m    n  pt          aw          epsx          epsy          epsl          sigl
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Define interpretation of **pt**.
 = 0 or blank, **pt** is the momentum of the beam particle (GeV/c)
 > 0, **pt** is the kinetic energy of the beam particle (GeV)

n (IN) Set longitudinal dispersion print option. The subsequent **CYAE** or **TRKE** command that refers to the **BVAL** will print either $\eta_s = \delta s / (\delta p / p)$, the longitudinal dispersion, or $\eta_t = c \cdot \delta t / (\delta p / p)$ according to the value of **n**. If **pt** = 0, this parameter is ignored.
 = 0 or omitted, Print the longitudinal dispersion η_s in the table of functions.
 > 0, Print η_t in the table of functions.

pt (FP) Momentum or kinetic energy of beam particles.
 = 0 or omitted, The emittances **epsx**, etc., are unnormalized* and no kinematic calculations are done.
 > 0, The emittances **epsx**, etc., are normalized* and the necessary kinematic calculations to convert to unnormalized values are done.

aw (FP) Atomic weight of beam particle.
 = 0 or omitted, The mass of the electron is used.
 > 0, Value, in multiples of proton mass.

epsx (FP) Horizontal emittance (mm-mrad).

epsy (FP) Vertical emittance (mm-mrad).

* The unnormalized emittance is the phase space area of the beam defined by $\epsilon = \int n(x, x') dx dx'$ and is not in general an invariant as the beam energy changes. The normalized emittance is defined by $\epsilon_N = \int n(x, p) dx dp$, where **p** is the momentum canonically conjugate to **x**. The two forms are related by $\epsilon = (1/\beta\gamma)\epsilon_N$, where β and γ are the usual relativistic factors.

BVAL

eps1 (FP) Longitudinal emittance (mm^2/oo). Since $\text{eps1} = \text{sig1} \times \text{sigp}$, one may input the rms value of dp/p directly by setting $\text{eps1} = dp/p$ and $\text{sig1} = 1$.

sig1 (FP) Bunch length (mm).

C - Comment

The **C** statement inserts a line of text in the output listing.

Format 1:

```

-----1-----2-----3-----4-----5-----6-----7-----8
C<-----          text                      ----->
-----1-----2-----3-----4-----5-----6-----7-----8

```

Format 2:

```

-----1-----2-----3-----4-----5-----6-----7-----8
.<-----          text                      ----->
-----1-----2-----3-----4-----5-----6-----7-----8

```

Format 3:

```

-----1-----2-----3-----4-----5-----6-----7-----8
Cs
-----1-----2-----3-----4-----5-----6-----7-----8

```

The comment commands, which do not conform to the **SYNCH** statement format standard, are used to insert a line of text in the output listing. Comments are indicated by a "**C**" or "." in column 1 of the record. Under formats 1 and 2, text from columns 2-80 is copied without change to the output file. The leading "**C**" or "." is omitted. Under format 3, the text field contains only a single character in column 2. This character is replicated in columns 2-132 of the line in the output file. Any printable character may be used.

When used within a **SYNCH** subroutine, the **C** command enters the text in the output listing when the subroutine is defined and each time it is executed, except when it is invoked by a fitting command or when echoing is suppressed by the **NECHO** command. When using comments to document a subroutine, it is normally preferable to place the comments outside the subroutine.

SEE ALSO: **REM**
PAGE
 .(Period)

CALC - Calculator Simulator

The **CALC** statement provides the functions of a hand calculator for use within a **SYNCH** program.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  CALC   m      c1   c2   c3   ...
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Name of quantity calculated.

m (IN) Print control switch.
 = 0 or blank, Do not print results from calculator stack.
 = 1, Print calculator stack at end of calculation.
 = 2, Print calculator stack after every step.

c1, ... (CH) Calculator key names and names of previously defined variables.

The **CALC** command makes available within a **SYNCH** program the functions typically provided by a scientific hand-calculator. The **CALC** command is implemented in Reverse Polish Notation (RPN) and patterned on the Hewlett-Packard family of calculators. The key names (calculator commands) are adapted from the HP scientific calculators. The available calculator key-names are:

+	-	*	/	1/X	SQRT	X*X	Y**X
EXP	LN	SIN	COS	TAN	ASIN	ACOS	ATAN
ABS	EEX	CHS	X-Y	X=Y	P-R	R-P	X=0
PI	RDN	RUP	ENTR	CLST			
STO	RCL	CLX	LSTX				
XLEY	XGTY.						

The calculator key-names are described in Table 4.1.

When issuing a **STO** (store to memory) or **RCL** (recall from memory) command, one must reference the name of a **CALC**, **=**, or **PARA** statement to provide the storage buffer.

More generally, the **CALC** command can reference itself; that is, it can be used recursively. If **name** appears also as one of the **c_i** fields in the parameter string, the value associated with **name** is used. The value defaults initially to 0 and is updated only when the **CALC**ulation is completed.

The calculator is implemented using 5 registers, **x**, **y**, **z**, **t**, **xl** in a stack. The **x**-register always contains the results of the latest calculation. The final result of the calculation is stored in **name**. The intermediate results are preserved in the stack and may be used by later **CALC** statements.

CALC

Examples:

In the following examples, the lengths `lb` and `ldr1` are subtracted from `lhct` and the result is stored as `ldr2`. They illustrate two completely equivalent ways of making the calculation.

Example 1:

```
-----1-----2-----3-----4-----5-----6-----7-----8
lhct  =          22.
lb    =          20.
ldr1  =          0.5
ldr2  =          0.0
      CALC      RCL lhct RCL lb  -    RCL ldr1 -    STO ldr2
-----1-----2-----3-----4-----5-----6-----7-----8
```

Example 2: The same result is obtained by replacing the last 2 lines of the example by the single line

```
-----1-----2-----3-----4-----5-----6-----7-----8
ldr2 CALC      RCL lhct RCL lb  -    RCL ldr1 -
-----1-----2-----3-----4-----5-----6-----7-----8
```

Table 4.1. Commands for Simulated RPN Calculator.
The notation (...) is used to denote the value contained in the ... register.

+	$(x1) \leftarrow (x), (x) \leftarrow (y) + (x), (y) \leftarrow (z), (z) \leftarrow (t)$
-	$(x1) \leftarrow (x), (x) \leftarrow (y) - (x), (y) \leftarrow (z), (z) \leftarrow (t)$
*	$(x1) \leftarrow (x), (x) \leftarrow (y) * (x), (y) \leftarrow (z), (z) \leftarrow (t)$
/	$(x1) \leftarrow (x), (x) \leftarrow (y) / (x), (y) \leftarrow (z), (z) \leftarrow (t)$
1/x	$(x1) \leftarrow (x), (x) \leftarrow 1 / (x)$
SQRT	$(x1) \leftarrow (x), (x) \leftarrow \sqrt{(x)}$
X*X	$(x1) \leftarrow (x), (x) \leftarrow (x)**2$
Y**X	$(x1) \leftarrow (x), (x) \leftarrow (y)**(x)$
EXP	$(x1) \leftarrow (x), (x) \leftarrow \exp((x))$
LN	$(x1) \leftarrow (x), (x) \leftarrow \ln((x))$
SIN	$(x1) \leftarrow (x), (x) \leftarrow \sin((x))$
COS	$(x1) \leftarrow (x), (x) \leftarrow \cos((x))$
TAN	$(x1) \leftarrow (x), (x) \leftarrow \tan((x))$
ASIN	$(x1) \leftarrow (x), (x) \leftarrow \sin^{-1}((x))$
ACOS	$(x1) \leftarrow (x), (x) \leftarrow \cos^{-1}((x))$
ATAN	$(x1) \leftarrow (x), (x) \leftarrow \tan^{-1}((x))$
ABS	$(x1) \leftarrow (x), (x) $
EEX	$(x) \leftarrow (y)*10**(x)$
CHS	$(x) \leftarrow -(x)$
X-Y	$(x) \leftarrow (y), (y) \leftarrow (x)$
X=Y	Execute the following command if $(x) = (y)$.
P-R	Replace the polar $(r=(x), \theta=(y))$ coordinates of a point by its Cartesian co-ordinates: $(x = r * \cos(\theta), \theta = r * \sin(\theta))$. $r=(x), \theta=(y) : (x1) \leftarrow (x), (x) \leftarrow x, (y) \leftarrow y$.
R-P	Replace the Cartesian (x,y) coordinates of a point by its polar (r,θ) coordinates. $x=(x), y=(y) : (x1) \leftarrow (x), (x) \leftarrow r, (y) \leftarrow \theta$.
X=0	Execute the following command if $(x) = 0$.
PI	$(x) \leftarrow \pi = 3.141\ 592\ 653\ 589\ 79$.
RDN	Roll down. Move the values in the stack thus: $(t) \leftarrow (x), (x) \leftarrow (y), (y) \leftarrow (z), \text{ and } (z) \leftarrow (t)$.
RUP	Roll up. Move the values in the stack thus: $(x) \leftarrow (t), (t) \leftarrow (z), (z) \leftarrow (y), \text{ and } (y) \leftarrow (x)$.
ENTR	$(x) \leftarrow \text{new_value}$.
CLST	Clear the stack.
STO	$\text{variable} \leftarrow (x)$.
RCL	$(x) \leftarrow \text{variable}$.
CLX	$(x) \leftarrow 0$.
LSTX	$(x) \leftarrow (x1)$
XLEY	Execute the following command if $(x) \leq (y)$.
XGTY	Execute the following command if $(x) > (y)$.

CALL – Invoke a **SYNCH** Subroutine

The **CALL** command is used to invoke a **SYNCH** subroutine.

```

-----1-----2-----3-----4-----5-----6-----7-----8
      CALL      m      name
-----1-----2-----3-----4-----5-----6-----7-----8

```

m (IN) Number of times to execute the subroutine. If absent, execute the subroutine once.

name (CH) Name of the **SYNCH** subroutine to be executed.

A **SYNCH** subroutine is a set of commands that are grouped together by **SUB** and **END** commands. When invoked by a **CALL** command, the commands in the subroutine are executed sequentially. A **SYNCH** subroutine may be executed repeatedly within a **SYNCH** program.

A **CALL** statement may be included in another **SYNCH** subroutine. However, a subroutine may not call itself.

By including **INCR** statements in the subroutine, the subroutine will be executed repeatedly using different values of the incremented variables.

SEE ALSO: **SUB**
END
INCR
MESH
REPL
VPAR

CYA – Beta Functions and Matrices of Cyclic Permutations of Elements

The **CYA** command computes the matrix products of the cyclic permutations of the matrices corresponding to the elements of a beamline, the associated betatron functions, and certain other properties of a synchrotron built utilizing the beamline as the superperiod. The table of betatron functions and other properties may be printed. The matrix products are available as named entities for subsequent use.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  CYA              n  bmln
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name. The names of the new matrices created by this command are derived from this name.

n (IN) Number of superperiods in the synchrotron. Optional.

bmln (CH) Name of a previously defined **BML** command.

The **CYA** command computes the k matrix products of the cyclic permutations of the matrices representing the k elements of the beamline **bmln** and the periodic betatron functions (the Twiss parameters) for the beamline. If the optional parameter **m** is specified, the beamline is used as the superperiod for a synchrotron built from **m** such superperiods and additional properties of the synchrotron are calculated.

A table of the betatron functions evaluated at the end of each element of the beamline is printed out. If requested, the other properties of the synchrotron are printed following the table of betatron functions. The horizontal and vertical chromaticities due to magnets which are explicitly included in the beamline are computed and printed.

The k matrices representing the cyclic permutations of the beamline elements are saved for future use. **CYA** uses the first letter of name and appends to it the integers $1, \dots, k$ to create names for the k cycled matrices. For example, if the first letter of name is N , then the names of the resulting matrices are $N1, N2, \dots, Nk$. Permutation 1 is defined by the elements ordered as they are encountered by the beam.

The table of betatron functions is labelled from 0 (start of beamline) to k (end of beamline). The k lines 0 to $k - 1$ correspond to the matrices numbered 1 to k and line k again corresponds to matrix 1.

The **CYA** command creates the new named matrices without warning. The user is cautioned not to define other elements whose names duplicate those created by **CYC** for the cycled matrices.

CYA

An example will illustrate the use of the **CYA** command. If the following **SYNCH** statements are used,

```
-----1-----2-----3-----4-----5-----6-----7-----8
L      BML          A   B   C   D
TST    CYA          L
-----1-----2-----3-----4-----5-----6-----7-----8
```

then four cycled matrices are

T1 = DCBA

T2 = ADCB

T3 = BADC

T4 = CBAD.

The matrix TST could also be referred to in a later **SYNCH** statement, TST being equivalent to T1.

CYAE - Beam Envelope Calculation

The CYAE command calculates the periodic betatron functions of the beamline and then, using specified emittances, calculates the beam envelope through the beamline.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  CYAE   m      bmln beam efact      epxco      epyco
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name.

m (IN) Selector for combining betatron and momentum displacements.
 = 0, Displacements are added in quadrature.
 = 1, Displacements are added algebraically.

bmln (CH) Name of previously defined beamline (BML statement).

beam (CH) Name of the BVAL or CYEM statement which supplies the values of the emittances, ϵ_x and ϵ_y , and dp/p .

efact (FP) The ratio ϵ/ϵ_0 of the emittances used by CYAE to calculate the beam envelopes to those defined by the referenced BVAL or CYEM statement.

epxco (FP) Closed orbit equivalent emittance (horizontal).

epyco (FP) Closed orbit equivalent emittance (vertical).

Given a beamline **bmln** together with beam emittances ϵ_x and ϵ_y and momentum error $\delta p/p$ specified by a previous BVAL or CYEM statement, CYAE uses the periodic betatron functions for the beamline and the specified emittances to calculate the beam envelopes through the beamline.

The parameter **efact** is the ratio of the emittance, ϵ , to be used by the CYAE command to the emittance obtained from the BVAL statement, $\epsilon_0 = \sigma^2/\beta$. This allows, for example, a 1σ emittance to be used by BVAL while the beam envelopes are based on a 6σ (95%) emittance. The displacements due to the betatron motion and to the momentum deviation are thus multiplied by $\sqrt{\text{efact}}$.

The "closed orbit equivalent emittances" are emittance-like parameters used to describe an ensemble of machines with closed orbit errors (cf., BVAL). The orbit distortions arising from placement errors vary as $\sqrt{\beta}$ and may be combined with the betatron displacements to estimate the total beam displacement from the design orbit.

SEE ALSO: BVAL
 TRKE

CYC – Periodic Beta Functions Through a Lattice Period

The **CYC** command computes the periodic betatron functions for a beamline. The beamline may be used as the superperiod in a synchrotron built from m such superperiods. A table of the betatron functions evaluated at the end of each element is printed out. Some other properties of the synchrotron are printed following the table. The horizontal and vertical chromaticities due to magnets which are explicitly included in the beamline are computed and printed.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  CYC      m    n bmln ampl sx1  sx2  cx          cy          hdr
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Output selector. Values of the betatron functions are printed at the downstream end of each element selected by the value of m .

$m < 0$, Print values only at those elements whose names begin with a special character (default is #). The **PCYC** command is used to specify an alternative special character or set of characters.

$0 \leq |m| \leq 10$, or $m = 11, 13, 15$: Print orbit functions at the downstream end of each beamline element (or those which have the names selected by negative m , see above).

$m = 12, 14, 16$: Print only a summary of the global characteristics of the orbit.

$|m| = 11, 12$: Generate an external binary file **JBIS** = 15, which contains information about parameters and orbit functions for each element, and may be useful for external programs.

$|m| = 13, 14$: Generate an external BCD file **CYBO** = 20, which contains the dispersion functions.

$|m| = 15, 16$: Generate an external binary file **FIL11** = 11, which contains element parameters and vertical orbit functions, to be used by the program **DEPOL**^[13] for calculating the strengths of depolarizing resonances. To implement this feature, one must execute the command "OPEN FIL11" before the **CYC**. (Note that **FXPT** also can write **FIL11**).

CYC

- n** (IN) Optional. Number of superperiods in the synchrotron. The beamline **bmln** is used to define the superperiod.
- < 0, The specified beamline is defined to be one half of a reflection-symmetric superperiod. Each element in the beamline must itself be reflection-symmetric. The total number of superperiods in the synchrotron is $|n|$. The betatron functions will be printed only for one unit of the beamline.
- ≥ 0 or blank, The beamline defines a superperiod. The table of functions is printed only for one superperiod. Setting $n = 0$ or blank is equivalent to $n = 1$.
- bmln** (CH) Name of **BML** statement which describes the beamline.
- ampl** (CH) If keyword **AMPL** is given, calculate the amplitude (emittance) dependence of the tunes.
- sx1, sx2** (CH) Optional. Names of previously defined sextupoles families. If used, the beamline must contain the two families of sextupoles **sx1** and **sx2**, to be adjusted to produce the desired chromaticities.
- cx,cy** (FP) Optional. Desired horizontal and vertical chromaticities.
- hdr** (CH) Optional. Name of a **REM**, **PAGE**, or **RUN** command. Forces the comment entered in these statements to be printed as a page header on each page of the table, and, if their **m**-value is 1, then the date and time are appended to the header.

The **CYC** command calculates k matrices representing the cyclic products of the elements $B_1, B_2, \dots, B_k$ which comprise the beamline; only the product $B_k * \dots * B_2 * B_1$ is saved and may be referenced by name. (Compare **CYA**.) Using the cyclic-product matrices, the betatron functions β, α and $\Sigma\psi/2\pi$ ($\Sigma\psi$ is the cumulative phase advance)* and the cumulative path length s are computed at the end of each element and printed out, followed by a list of the tunes of the entire synchrotron, the machine radius, the total bend angle of the machine, and the transition gamma. The horizontal and vertical chromaticities of the machine due to magnets which are explicitly used in the **BML** statement are calculated and printed. If none of $B_1, B_2, \dots, B_k$ are **MAG**'s, then no chromaticities will be computed.

The **CYC** command provides an option to adjust the chromaticity of the accelerator by varying the strengths of two families of sextupoles, **sx1** and **sx2**. The beamline **bmln** may contain more than two families of sextupoles, but only the two specified are adjusted to set the chromaticities.

SEE ALSO: **BEST, OPEN, PCYC, SXTTP**

* If any of the matrices $B_1, \dots, B_k$ represent elements with phase advance ψ greater than 2π , the integer part of $\Sigma\psi/2\pi$ may not be correct.

CYEM - Beam Emittance Calculation

CYEM calculates the horizontal and vertical electron integrals, rf output, and emittance factors for a storage ring.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  CYEM          n bmln ptcl energy   radius   freq     volt     kappa
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

n (IN) Number of superperiods. If n is negative, bmln is the first half of a reflected superperiod.

bmln (CH) Name of the beamline **BML** statement that defines the storage ring.

ptcl (CH) Type of particle
 = ELEC or blank, Electron.
 = PROT, Proton.

energy (FP) Particle beam energy (GeV).

radius (FP) Machine radius (meters).

freq (FP) Rf frequency (MHz).

volt (FP) Rf voltage (MV/turn).

kappa (FP) Coupling coefficient $\sqrt{\epsilon_y/\epsilon_x}$.

The physics of electron synchrotrons is discussed in Sands.^[14]

DEACT - Deactivate Statement(s)

The **DEACT** command deactivates a specified list of **SYNCH** statements.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      DEACT  m      stm1 stm2 stm3 ...  stmm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) Number of statements to deactivate.
 = 0 or blank, Deactivate the command statements specified in the data fields. The number of statements deactivated is determined from the list. The list may be continued on successive lines.
 > 0, Deactivate **m** consecutive statements beginning with **stm1**. Only the one statement, **stm1**, should be named in the data field.

stm1, ... (CH) Name(s) of statement(s) to be deactivated.

DEACT suspends execution of a specified set of **SYNCH** commands.

The principal use of the **DEACT** and **ACT** commands is to disable or enable calculation of matrices of elements included in **SYNCH** subroutines, to choose between different **TRKB** statements, etc.

SEE ALSO: Section 6.5, Initially Deactivated Statements
ACT

DEQ – Transformations Defined by Differential Equation Integration

The **DEQk** commands allow the use of built-in or user-written FORTRAN subroutines that specify particle coordinate transformations for each integration step of a differential equation. Integration over a finite interval constitutes a special beamline element.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  DEQk  m      p1      p2      p3      ...      pm
-----1-----2-----3-----4-----5-----6-----7-----8

```

- name** (CH) Reference name of the element defined by the **DEQk** command.
- k** (IN) Identifies the internal FORTRAN subroutine **DEQK** that specifies the transformation of the particle coordinate vector. Values $k = 0-9$ are reserved for built-in routines, and values $k = 10-19$ are available for user-written routines.
- m** (IN) Number of parameters defined by the **DEQk** command.
- p1, ...** (FP) Input parameter values to be passed to the internal FORTRAN subroutine **DEQK**, and the length of the interval **p1**, and the step size **p6**, to be passed to the differential equation routine.

A **DEQ** element may be invoked by the commands **TRK**, **FXPT**, and **TRKM**. The element **name** operates on the particle state vector $V = (x, x', y, y', -ds, dp/p, 1)$ by means of a differential equation integrator in **SYNCH**. This integrator invokes the subroutine **DEQK**, which calculates the increments in V .

Chapter 7 contains, as an example, the coding of the built-in FORTRAN subroutine **DEQ4**. The built-in **DEQ** routines are the following:

DEQ1: Integration through a wiggler magnet.

DEQ3: Integration of transverse and longitudinal beam envelopes.

DEQ4: Integration of transverse beam envelopes, linearized envelopes and single particles in a beam with space charge. This command can be used with **FXPT** to find envelopes in a periodic quadrupole array.

DEQ5: Integration through a sextupole magnet.

SEE ALSO: Section 3.6, Non-linear Transformation Calculations
Chapter 7, Non-linear Transformations
MAP, TRK, FXPT, TRKM

DRF – Drift Space Definition

The **DRF** command is used to define drift spaces in a lattice. The command implements three different ways to define the drift spaces.

METHOD 1 – Define a single drift length and calculate its transfer matrix.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  DRF          length
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

length (FP) Length of the drift space.

This simplest use of **DRF** defines a single drift region of length **length** identified by **name**.

METHOD 2 – Define a series of drifts and the associated transfer matrices.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  DRF      m      vector
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name

m (IN) Number of drift spaces to define.

vector (CH) Name of a vector with at least **m** components, defined by a **VEC** command. The values of the vector components are the lengths of the drift spaces.

This use of **DRF** defines a series of **m** drifts. The names of the drifts are constructed by appending to the first letter of **name** the digits 1,2,...,m.

DRF

Example: The statements

```
-----1-----2-----3-----4-----5-----6-----7-----8
LQ   VEC   5     5.       2.       10.3      15.       7.
DRFT DRF   5     LQ
-----1-----2-----3-----4-----5-----6-----7-----8
```

are equivalent to

```
-----1-----2-----3-----4-----5-----6-----7-----8
D1   DRF           5.
D2   DRF           2.
D3   DRF          10.3
D4   DRF          15.
D5   DRF           7.
-----1-----2-----3-----4-----5-----6-----7-----8
```

METHOD 3 – Define the length of a drift region so as to hold fixed the total length of a series of elements (drifts and magnets).

```
-----1-----2-----3-----4-----5-----6-----7-----8
name DRF          n ldnam  ltot    e1   e2   e3   ...   en
-----1-----2-----3-----4-----5-----6-----7-----8
```

name (CH) Reference name of drift space whose length is to be defined.

n (IN) Number of drifts and/or magnets, not including **name**.

ldnam (FP) Initial length of the drift region name whose length is to be adjusted.

ltot (FP) Total length of the elements **name**, **e1**, **e2**, ..., **en**.

ei (CH) Names of drifts and/or magnets to be included in the total length. There can be up to eight **ei** names on the command line. If $n > 8$, additional names can be continued on the next line, beginning in column 21. Embedded blank fields are not allowed.

This variant of the **DRF** command is used to adjust the length of the drift **name** so as to hold constant the total length of a set of drifts and/or magnets by varying the length of **name** to compensate for changes in the lengths of **e1** ... **en**. Typically such changes are made by a fitting routine, and **name** as well as **e1** ... **en** are included in a **SYNCH** subroutine. If **ltot** $\neq$ 0.0, **ldnam** is adjusted to make

$$\text{ldnam} + l_{e_1} + l_{e_2} + l_{e_3} + \dots + l_{e_n} = \text{ltot}$$

where l_{e_1} is the length of element **e1**. In this instance, **ldnam** may be left blank. If **ltot** = 0.0 or blank in the statement definition, **SYNCH** calculates and stores its value from the formula

$$\text{ltot} = \text{ldnam} + l_{e_1} + l_{e_2} + l_{e_3} + \dots + l_{e_n}$$

where **ldnam** is the original value in the input statement.

When **DRF** is used in this fashion, with **ltot** = 0, within a **SYNCH** subroutine, **ltot** will be determined on the first invocation of the subroutine. On subsequent invocations, **ldnam** will be adjusted to keep **ltot** constant.

ECHO - Print Out Input Statements

The **ECHO** command causes subsequent **SYNCH** input statements to be listed in the output file when they are read from the input file.

```

-----1-----2-----3-----4-----5-----6-----7-----8
      ECHO
-----1-----2-----3-----4-----5-----6-----7-----8

```

The **ECHO** command enables listing of the **SYNCH** input statements in the output file when they are read from the input file. The **NECHO** command is used to disable echoing. "**ECHO**" is the default mode.

SEE ALSO: **NECHO**

EMIS - End Misalignment Mode

The **EMIS** command switches the operation of **SYNCH** from the magnet misalignment mode to the normal mode.

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      EMIS
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

SEE ALSO: **BMIS**
MAGS
Section 8.8, Magnet Misalignment Calculations

END

END - Mark End of **SYNCH** Subroutine

The **END** command marks the end of the definition of a **SYNCH** subroutine.

```
-----1-----2-----3-----4-----5-----6-----7-----8
      END
-----1-----2-----3-----4-----5-----6-----7-----8
```

SEE ALSO: **SUB**
 CALL

EQU – Alternate Name for Beamline Element

The EQU command creates an alias for another beamline element.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  EQU                elmn
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name of the alias statement.

elmn (CH) Name of the original statement.

Subsequent references to **name** will have the same effect as references to **elmn**, with the following exception: if subsequently the input contains a new statement also called **elmn**, the alias **name** will continue to reference the original **elmn**.

FITB – Fit Betatron Functions

FITB invokes a two parameter fit to specified values of two betatron functions.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name FITB   m   n sbr  mtrx e1   e2   i1   i2  des1   des2   del
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name.

m, n (IN) Identify betatron functions to be fit.

= 1,	$\nu_x = \mu_x/2\pi$	= 11,	$\nu_y = \mu_y/2\pi$
= 2,	β_x	= 12,	β_y
= 3,	α_x	= 13,	α_y
= 5,	η_x	= 15,	η_y
= 6,	η'_x	= 16,	η'_y
= 7,	$\sqrt{\beta_x}$	= 17,	$\sqrt{\beta_y}$

sbr (CH) Name of **SYNCH** subroutine containing **mtrx**.

mtrx (CH) Name of matrix from which to extract betatron functions.

e1, e2 (CH) Names of elements containing parameters to be varied.

i1, i2 (IN) Parameters of **e1** and **e2**, respectively, which are to be varied.

des1, des2 (FP) Desired values to which to fit the betatron functions.

del (FP) (Default value: 1.0E-8) Fit tolerance. Inverse weight. The fit is successful when $|\text{val} - \text{des}| < \text{del}$.

FITB iterates through subroutine **sbr** varying the parameters **i1** and **i2** of elements **e1** and **e2** to obtain desired values **des1** and **des2** of the betatron functions specified by **m** and **n**, respectively.

FITB

Example 1: Fit beta functions at ends of a standard FODO cell.

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
SBR  SUB
QD   MAG      2.      gd      5000.
QF   MAG      2.      gf      5000.
B    MAG     28.      0.      5000.    25.    $
CELL MMM      B      QD      B      QF
      END
C
betx =          99.
bety =          30.
gd    =         -90.
gf    =          90.
      FITB  2  12 SBR  CELL gf  gd      1  1betx  bety
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

This code iterates through subroutine **SBR** varying the values of **gd** and **gf** until the values of the horizontal and vertical beta functions associated with matrix **CELL** become 99.0 and 30.0, respectively. The computed values of **gf** and **gd** are subsequently available for use elsewhere in the **SYNCH** program.

Example 2: Fit the beta functions at the end of a standard FODO cell. This case is a variation on Example 1.

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
SBR  SUB
QD   MAG      2.      -90.    5000.
QF   MAG      2.      90.     5000.
B    MAG     28.      0.      5000.    25.    $
CELL MMM      B      QD      B      QF
      END
C
betx =          99.
bety =          30.
      FITB  2  12 SBR  CELL QF  QD      2  2betx  bety
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

In this illustration, the values of parameter 2 of the magnets **QF** and **QD** are varied in exactly the same way as for Example 1. The distinction is that the values of the gradients are accessible only as parameters of the elements **QF** and **QD**.

SEE ALSO: **FITQ**

FITQ - Fit Betatron Tunes

FITQ invokes a two parameter fit to specified values of the horizontal and vertical tunes of a beamline.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  FITQ           n sbr  mtrx e1   e2   i1   i2   des1       des2       del
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

n (IN) Fit one or two variables.
 < 0, Fit single variable (**e1**, **i1**) to single desired value **des1**.
 ≥ 0 or blank, Fit both tunes to the desired values.

sbr (CH) Name of **SYNCH** subroutine containing **mtrx**.

mtrx (CH) Matrix from which to extract phase advances.

e1, **e2** (CH) Elements containing parameters to be varied.

i1, **i2** (IN) Parameters of **e1** and **e2** to be varied.

des1, **des2** (FP) Desired values of tunes (phase advances/ 2π)

del (FP) (Default value: 1.0E-6) Fit tolerance. Inverse weight. The fit is successful when $|val - des| < del$.

FITQ iterates through subroutine **sbr** varying parameters **i1** and **i2** of elements **e1** and **e2** respectively to obtain the desired values **des1** and **des2** of the horizontal and vertical tunes (= phase advances/ 2π) through the elements composing **mtrx**.

FITQ obtains the tunes from the matrix **mtrx** representing the beamline. Only the fractional part of the tune can be determined from **mtrx** and the solution that is found by **FITQ** may correspond to tunes whose integer part is not what is desired.

FITQ

Example: Vary the strengths of two quadrupoles in the cell represented by the beamline .C to obtain tune values equal to 0.4.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
.C      BML          QF  00  B    00  QD  00  B    00
00      DRF          0.5
BRHO =    ...
BZERO =    ...

B      MAG          6.2          0.          BRHO      BZERO

SBR    SUB
QD     MAG          2.          gd          5000.
QF     MAG          2.          gf          5000.

CELL   MMM          .C
      END
C
gd     =            -90.
gf     =            90.
nux    =            .4
nuy    =            .4
      FITQ          SBR  CELL gd  gf          1    1nux          nuy
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

The computed values of the quadrupole strengths **gf** and **gd** replace their initial values and are available for use in subsequent statements.

SEE ALSO: FITB

FITR – Fit matrix elements

FITR invokes a two parameter fit to specified values of two of the matrix elements of a transfer matrix.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name FITR  m   n sbr mtrx e1   e2   i1   i2   des1   des2
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name.

m (IN) = *ij*, indicating that the *ij*-th element of the 7×7 matrix is to be fitted.

n (IN) = *kl*, indicating that the *kl*-th element of the 7×7 matrix is to be fitted.

sbr (CH) SYNCH subroutine containing **mtrx**.

mtrx (CH) Name of matrix whose matrix elements are to be fitted.

e1, e2 (CH) Elements containing parameters to be varied.

i1, i2 (IN) Parameters of **e1** and **e2**, respectively, which are to be varied.

des1, des2 (FP) Desired values to which to fit the matrix elements of **mtrx**.

del (FP) (Default value: 1.0E-6) Fit tolerance. Inverse weight. The fit is successful when $|val - des| < del$.

FITR iterates through subroutine **sbr** varying parameters **i1** and **i2** of elements **e1** and **e2** respectively to obtain the desired values **des1** and **des2** of the matrix elements of **mtrx** specified by **m** and **n**.

FITV - Vary Two Parameters to Fit Values of Beam Coordinates

The **FITV** command invokes a two-parameter fit so that two of the components of a particle state vector achieve specified values at the end of the beamline.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name FITV  m   n sbr  vf   e1   e2   i1   i2   des1   des2   del
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name.

m, n (IN) Index of the components of **vf** to be fitted.
 $n \leq 0$, Fit only the single component indexed by **m**.

sbr (CH) **SYNCH** subroutine containing transfer matrix definition.

vf (CH) State vector giving the particle coordinates at the end of the beamline.

e1, e2 (CH) Elements containing parameters to be varied.

i1, i2 (IN) Parameters of **e1** and **e2** to be varied.

des1, des2 (FP) Desired values for the components of the vector **vf** being fitted.

del (FP) Fit tolerance. Default value is 1.0E-8.

The **FITV** command iteratively invokes a subroutine to track a particle through a beamline and adjusts two parameters of the elements which comprise the beamline to obtain specified values for two components of the particle's phase space vector at the end of the beamline. The initial phase space vector of the particle must be defined before **FITV** is invoked.

FITV

In the example, VI and VF are the state vectors of the particle at the beginning and end of the beamline. The FITV command iteratively invokes subroutine SBR to track VI through the beamline and varies two parameters until the desired values of the components of VF are obtained.

Example:

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
SBR  SUB
K1   KICK  1    001      0.0      brho    bk1
K2   KICK  1    002      0.0      brho    bk2
TM   MMM      .BL
VF   MXV      TM  VI
      END
.BL  BML      QF  001 B    00  QD  002 B    00
VI   PVEC      x      x'    y      y'
      FITV  3  4 SBR  VF  K1  K2      3  3val1    val2    del
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

In this example, the field strengths, bk1 and bk2, of the kicks K1 and K2 are adjusted to make the 3rd and 4th components of VF equal to the desired values des1 and des2.

FXPT - Closed Orbit Calculation

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  FXPT   m    n  psv  bmln nscl isav itr  iflg itap
              e1      e2      d1      d2      d3
-----1-----2-----3-----4-----5-----6-----7-----8

```

NOTE: The second record must be included with the **FXPT** command even if all the fields are blank.

name (CH) Reference name.

m (IN) Select output printed by **FXPT**.
 < 0, Print data at only those locations whose names begin with a special character (default is #). The **PCYC** command is used to specify an alternative special character.
 = 0, Print the matrix representing the beamline, its eigenvalues, the closed orbit and the dispersion vector at location 0.
 = 1, As for option 0, plus the eigenvectors at position 0.
 = 2, As for option 1, plus the closed orbit traced through the beamline **bmln**.
 = 3, As for option 1, plus the eigenvectors traced through the beamline **bmln**.
 = 4, Combine options 2 and 3.

n (IN) Number of superperiods in the ring.
 > 0, Number of superperiods.
 < 0, The reflection of **bmln** is added to **bmln** to form the complete superperiod. The number of superperiods is $|n|$.

psv (CH) Name of the **PVEC** statement defining the initial guess for the closed orbit coordinates.

bmln (CH) Name of the **BML** statement which defines the beamline for which the closed orbit is requested.

nscl (IN, default=1) Number of superperiods to close the orbit. May be greater than the number of superperiods in the ring to find a non-linear fixed point.

isav (IN) Save option for parameter **psv**.
 = 0 or blank, The vector specified by **psv** is not updated by **FXPT**.
 = 1, The equilibrium orbit found by **FXPT** replaces the initial value of **psv**.

FXPT

- itr** (IN) Number of iterations to find closed orbit (default=25).
- iflg** (IN) Ray trace option (see below). Specifies the number of rays to be traced through the beamline.
= 0 or blank, Trace only the closed orbit ray.
= 1, Trace the closed orbit ray plus 9 neighboring rays.
= 2, Trace the closed orbit ray, 9 neighboring rays and 4 rays for the linearized equations.
= 3, Trace the closed orbit ray, 9 neighboring rays and 4 rays for the linearized equations, first for the beam envelope and second for four rays inside the beam envelope. This feature is used for integrating beam envelopes under the influence of space charge, using **DEQ4**.
- itap** (IN) Switch to dispose data used by ORBC for orbit correction.
= 0 or blank, Do not save the data.
= 1, Save the data on file ORB1 = 24.
= 2, Same as itap = 1, but calculate new corrections to be added to previous ones.
- e1, e2** (FP) Closure tolerances for closed-orbit displacements and slopes.
- d1, d2** (FP) Displacements from reference ray of neighboring rays, for finding $M4$. ($M4$ is a linearized representation of the beamline. See discussion below.)
- d3** (FP) displacement in dp/p used to calculate the linearized 3×3 matrices.

Given a beamline **bmln** which may contain non-linear elements, the **FXPT** command calculates and prints the positions through **bmln** of the closed orbit for particles with momentum error dp/p . The beamline may be made up of any linear transformations, **SXTP**, **NPOL**, and **MOVE** elements and user defined **MAP** or **DEQ** elements. The second line of the **FXPT** statement may be left blank if no **MAP** or **DEQ** statements are encountered in the beamline.

Each non-linear element is linearized in a neighborhood of the closed orbit and the betatron functions of the resulting linear system are calculated and printed. A by-product of the closed orbit calculations is a 4×4 matrix, $M4$, representing the beamline. $M4$ acts on column vectors of the form $\{X - X_e, X' - X'_e, Y - Y_e, Y' - Y'_e\}$, where $\{X_e, X'_e, Y_e, Y'_e\}$ is the closed orbit. This matrix as well as its eigenvalues and eigenvectors are printed.

As a first guess for the closed orbit, one starts with the particle state vector **psv** (defined by a **PVEC** statement) which contains dp/p as its 6-th element. The particle is tracked through **bmln** and recalculated until after i iterations

$$\begin{aligned} |X_i - X_{i-1}| < e1, & \quad |X'_i - X'_{i-1}| < e2 \\ |Y_i - Y_{i-1}| < e1, & \quad |Y'_i - Y'_{i-1}| < e2 \end{aligned}$$

or until the maximum number of iterations, **itr**, is reached (default: **itr**=25). The ray starting from (X_i, X'_i, Y_i, Y'_i) defines the closed orbit.

FXPT assumes that horizontal-vertical coupling may be present. Coupling is automatically present whenever the orbit contains vertical displacements together with nonlinearities, or vertical or tilted deflecting magnets, tilted quadrupoles, or solenoids. Beta functions and phases for the normal modes are computed using the formalism of Edwards and Teng^[10] as well as the coupling angle, the angle between the normal-mode axes and the horizontal and vertical axes. The theory is described in Section 8.9.

FXPT also can generate the binary file **FIL11** = 11, containing element parameters and vertical orbit functions, to be used by the program **DEPOL**^[13] for calculating the strengths of depolarizing resonances. To implement this feature, one must execute the command "OPEN **FIL11**" before the **FXPT** command.

SEE ALSO: **PVEC**

IBET - Enter Initial Values of Betatron Functions

The **IBET** command provides one method to enter the initial values of the betatron functions which are needed by **TRKB**, **TRKE** or **TRKM** commands.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  IBET              nux0      betax0    alphax0   gammax0   dx0      dpx0
              nuy0      betay0    alphay0   gammay0   dy0      dyp0
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

nux0 (FP) Initial value of horizontal tune.

betax0 (FP) Initial value of horizontal β function.

alphax0 (FP) Initial value of horizontal α function.

gammax0 (FP) Initial value of horizontal γ function. Not needed if **betax0** is not blank.

dx0 (FP) Initial value of horizontal dispersion function.

dpx0 (FP) Initial value of slope of horizontal dispersion function.

nuy0 (FP) Initial value of vertical tune.

betay0 (FP) Initial value of vertical β function.

alphay0 (FP) Initial value of vertical α function.

gammay0 (FP) Initial value of vertical γ function. Not needed if **betay0** is not blank.

dy0 (FP) Initial value of vertical dispersion function.

dyp0 (FP) Initial value of slope of vertical dispersion function.

SEE ALSO: **TRKB**
TRKE
TRKM

INCR - Increment an Input Parameter

The **INCR** command causes one parameter of a specified command to be incremented by a designated amount.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  INCR   m    n stmt type step
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Name of **INCR** statement.

m (IN) Number of the parameter to be varied.

n (IN) Increment mode switch. Parameters can be incremented either additively (the default) or multiplicatively.
 = 0 or blank, The parameter is altered by adding **step** to the current value of the parameter.
 ≠ 0, The parameter is altered by multiplying the current value by **step**.

stmt (CH) Name of the **SYNCH** statement containing the parameter to be altered.

type (CH) Type of parameter to be altered.
 = F (default) Floating point
 = I Integer.
 = KA Integer, corresponding to parameter **m**.
 = KB Integer, corresponding to parameter **n**.

step (As appropriate) Value used to alter the selected parameter. The type of **step** must agree with that of the parameter being altered. The value may be positive or negative.

The **INCR** command increments or multiplies the **m**-th parameter of type **type** in **stmt** by **step**. The **INCR** command is often used by placing it within a **SYNCH** subroutine, which is subsequently invoked repetitively by a **CALL** command. Upon completion of the **CALL** command the incremented variable is restored to its original value. Note that a matrix depending on the incremented variable is modified only when the defining command is re-executed.

INCR

Example:

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
XMPL  SUB
QD    MAG      2.0      -50.0    brho    b0      $
      INCR    2      QD    F      -.05
      --
      END
      --
      CALL  10      XMPL
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

Here, the second parameter (the gradient) of the MAG statement QD has initially the value -50.0 and is incremented by -.05 on each of ten invocations of XMPL. After the INCR statement is executed, reference to parameter 2 of the MAG statement would use the properly incremented value. However, the matrix corresponding to QD is not updated until the MAG statement is executed on the next iteration.

SEE ALSO: MESH
 REPL
 VPAR

INV – Invert a Matrix

The INV command computes the inverse of a matrix.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  INV          mtrx
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name of the inverted matrix.

mtrx (CH) Name of the matrix to be inverted.

The matrix **name** is defined to be the inverse of the matrix **mtrx**.

SEE ALSO: Section 6.2, Symbolic Entry for Inverse of a Matrix

INV2 – Rotate a Beamline 180° and Reflect It

The **INV2** command produces the matrix representing a beamline rotated through 180° about the longitudinal axis and reflected.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  INV2      mtrx
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name of the rotated-reflected matrix.

mtrx (CH) Name of the matrix to be rotated and reflected.

The **INV2** command rotates **mtrx** through 180° about the longitudinal axis and mirror-reflects the result.

Example:

The **INV2** command is equivalent to a **ROT** command followed by a **REF** command. Thus, if

```

-----1-----2-----3-----4-----5-----6-----7-----8
A    MAGV      len      grad      brho      b0      e1      e2
-----1-----2-----3-----4-----5-----6-----7-----8

```

then

```

-----1-----2-----3-----4-----5-----6-----7-----8
K    INV2      A
-----1-----2-----3-----4-----5-----6-----7-----8

```

is equivalent to

```

-----1-----2-----3-----4-----5-----6-----7-----8
KK   ROT      A      180.0
K    REF      KK
-----1-----2-----3-----4-----5-----6-----7-----8

```

(

(

IOUT – Write a compressed lattice file

The **IOUT** statement writes a new **SYNCH** input file containing current definitions and values. Action commands and comments are omitted from the file.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      IOUT      m      n
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

- m** (IN) Controls statements to be written to the file.
- = 0 or blank (default), Write parameters, beam elements and beamlines. The original BML definitions are preserved.
 - = 1, Like **m** = 0, but omit **MMM**, **REF**, and **BML** statements. The resulting file contains only the parameter and element definitions.
 - = 2, Like **m** = 0 with the beam lines expanded, except that parentheses, indicating repeated structures, are retained. This can be a long file.
 - = 3, Like **m** = 2, but beamlines **BMLs** are completely expanded into a flat file. This file is often very long.
- n** (IN) Controls the definition of a reflected beam line.
- = 0 or blank, Replace a **REF** statement with a **BML** statement with **n** = -1.
 - = 1, Retain command **REF** as it was defined.

The **IOUT** command extracts from the input data stream the commands **=**, **PARA**, **CALC**, **DRF**, **MAG**, **MAGS**, **MAGV**, **MMM**, **NPOL**, **REF**, **SXTP**, and **BML** current at the time the **IOUT** command is invoked. The output is written to file **LFIL** = 70.

The **CALC** command is always translated into a **PARA** command. The **=** command is translated into a **PARA** unless its value is 0 or a Symbolic Floating Point datum.

Like commands are grouped together and alphabetized by name.

Record 1 of **LFIL** contains the text 'setops synch' in columns 1-13. This line is required as part of the input to the **SYNCH-to-MAD** translator incorporated in versions 6 and 7 of the **MAD**^[11] program. This translator does not translate irregular edge angles (fifth and sixth parameters of the **MAG** statements) if these are different from blank, 0, or \$. Subsequent lines contain the **RUN** command from the input data set and the commands listed above, ending with the **BMLs**.

The output file, **LFIL**, provides a handy listing of parameter values and beam line elements for reference or checking. It can be used, as it stands, for input to the translator. It can be used as **SYNCH** input by removing the first line, appending **SYNCH** action commands, e.g. **SOLV**, **CYC**, and a **STOP** statement.

KEEP - Save selected files

Force selected files to be saved at job completion for further use.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      KEEP          f11  f12  ...  fln
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

f1i (CH) Name(s) of the file(s) which are to be saved at completion of job.

A number of files written by **SYNCH** are normally deleted at completion of the job. Using the **KEEP** command, one can force such files to be retained for subsequent use. The files which may be saved by use of **KEEP** are: **SVOUT** (logical unit 4) and **ORB1** (logical unit 24).

SEE ALSO: Appendix A, Files
FXPT
SOLV

KICK - Dipole Kicker Magnet or Field Error Definition

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  KICK   m    n elmn                      brho      bk
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Define direction of dipole field.
 = 1, Field is in *x*-direction; deflection, vertical.
 = 2 or blank, Field is in *y*-direction; deflection, horizontal.

n (IN) Kick strength option.
 = 0 or blank, Field strength given by **bk**.
 ≥ 1 , Field strength is random within range $\pm bk$.

elmn (CH) Element name into which the kick or error is to be inserted.
 = blank, A zero-length drift is assumed and the strength is defined by
 $bk = (\text{kicker length}) * (\text{field strength})$. The effect is to provide a delta function kick
 which changes only the direction of the particle's trajectory.
 = name of DRF of length ℓ . **KICK** returns the matrix for a zero-gradient dipole
 magnet of length ℓ and field strength **bk**.
 = name of MAG of length ℓ . **KICK** adds a delta function kick of field strength **bk**
 (deflection angle $\ell * bk / b\rho$) at the center of the magnet.

brho (FP) Magnetic rigidity.

bk (FP) Strength of the kick or error field.

The **KICK** command creates the matrix representing either a dipole kick or a magnet having a dipole field error.

If **elmn** is blank or is that of a zero-length drift a matrix representing a delta function kick is created. The net effect of this matrix is to change only the direction of the particle's trajectory.

If **elmn** refers to a drift of nonzero length, **KICK** produces the matrix of a dipole magnet having the same length as **elmn**.

If **elmn** refers to a magnet, **KICK** produces the matrix of the magnet with a dipole field error.

LIST - Define List of Elements

The **LIST** statement defines a list of elements which will replace successive instances of a named element in a beamline.

```

-----1-----2-----3-----4-----5-----6-----7-----8
rpnm LIST      el1 el2 el3 ...   elk
-----1-----2-----3-----4-----5-----6-----7-----8

```

rpnm (CH) Name of the lattice element to be replaced, on successive occurrences in the beamline, by the listed elements.

eli (CH) Name of the specific elements which are to be substituted.

The **LIST** command defines a list of elements **el1**, ..., **elk** which will replace successive instances of the element **rpnm** in a beamline.

The **LIST** command is useful when studying the effects of magnet errors and misalignments on particle orbits in a beamline, including the closed orbits in a circular machine. A beamline is normally defined to consist of ideal, perfectly aligned elements. To study the orbit distortions caused by magnet imperfections and misalignments, modified elements which incorporate the errors can be defined using, for example, the **KICK**, **MAGS** and/or **MOVE** commands. Then, when named in a **LIST** command, these modified elements can replace the ideal elements which define the beamline.

LIST

Example:

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
QF      MAG      L      K      1.
QD      MAG      L      -K     1.
QF1     MOVE     QF      x1      xp1      y1      yp1
QF2     MOVE     QF      x2      xp2      y2      yp2
|       |       |       |       |       |       |
QF25    MOVE     QF      x25     xp25     y25     yp25
QD1     MOVE     QD      x26     xp26     dy26     yp26
|       |       |       |       |       |       |
QD25    MOVE     QD      x50     xp50     y50     yp50
C
.C      BML      QFL .B  QDL .B
.SP     BML      25( .C ) .SS
QFL     LIST     QF1  QF2  QF3  QF4  QF5  QF6  QF7  QF8  QF9  QF10
          QF11   ...   QF25
QDL     LIST     QD1  QD2  QD3  QD4  QD5  QD6  QD7  QD8  QD9  QD10
          QD11   ...   QD25
C
      FXPT  2   1 PV  .SP      1   0  10   0
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

In this example, the beamline .SP is made up of 25 standard cells and a long straight section. When the FXPT command is invoked, the beamline will use elements QF1,... and QD1,... from the LIST commands in place of the elements QFL and QDL when calculating the closed orbit.

2

of

3

MAG – Magnet Definition

The **MAG** statement defines the transfer matrix of a dipole, quadrupole, or combined function magnet, or of several magnets of the same type.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  MAG      m      n length   gradient brho      b0          e1          e2
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Number of magnets to define.
 = 0, 1, or blank, Define one magnet using parameters specified.
 > 1, The number of similar magnets to define using parameters specified in **VEC** statements.

n (IN) Select definition of vertical part of magnet matrix.
 = 0 or blank, Normal magnet definition.
 ≠ 0, A fictitious element for which the vertical part of the magnet matrix is replaced by the horizontal part.

length (FP) Effective length of the magnet, ℓ .

gradient (FP) Field gradient, B' .

brho (FP) Magnetic rigidity of the particle beam, $B\rho$.

b0 (FP) Reference bend field, B . For quadrupoles, set $b0 = 0$ or blank.

e1, e2 (FP) Angles in degrees between the entering and exiting central rays and the normal directions to the entrance and exit faces of the magnet. The angle is positive when the central ray is between the normal direction and the bend center of the magnet.
 = 0, or blank, The central ray enters and/or exits the magnet perpendicular to the magnet face(s).
 = \$ (left justified), If the \$ character is present, the magnet is assumed to be a parallel-faced rectangular bending magnet which bends the beam by θ . Then $e1 = e2 = \theta/2$.

The **MAG** command calculates the transfer matrix for a magnet of the specified length, gradient, magnetic rigidity **brho**, reference bend field **b0**, and entrance and exit angles **e1** and **e2**. The gradient, $B' = dB_y/dx$, is evaluated at the reference orbit.

MAG

The gradient, magnetic rigidity and reference bend field may be scaled by an arbitrary factor leading to equivalent definitions of a magnet. The choice is dictated by convenience. Three particularly convenient choices which emphasize the geometric nature of the problem are illustrated.

1. As multiples of brho:

$$\begin{aligned}\text{gradient} &= (B'/B\rho)(B\rho) = K(B\rho) \leftarrow K = \text{kk} \\ \text{brho} &= 1.0 (B\rho) \leftarrow 1.0 \\ \text{b0} &= (B/B\rho)(B\rho) = (1/\rho)(B\rho) \leftarrow (1/\rho) = 1/\text{rho}\end{aligned}$$

Then

	1	2	3	4	5	6	7	8
name	MAG	length	kk	1.0	1/rho	e1	e2	

where rho is the radius of curvature of the particle trajectory in the reference bend field.

2. As multiples of brho/length:

The focal length, f , and quadrupole strength, q , of the equivalent thin lens quadrupole may be written

$$(1/f) = q = K\ell = B'\ell/B\rho.$$

Then,

$$\begin{aligned}\text{gradient} &= (B'\ell/B\rho)(B\rho/\ell) = q(B\rho/\ell) \leftarrow q \\ \text{brho} &= \ell(B\rho/\ell) \leftarrow \ell \\ \text{b0} &= (B\ell/B\rho)(B\rho/\ell) \leftarrow (B\ell/(B\rho)) = \theta\end{aligned}$$

where θ is the dipole bend angle. Then

	1	2	3	4	5	6	7	8
name	MAG	length	q	length	theta	e1	e2	

Here, the length of the magnet is entered twice, but using the bend angle, θ , as an input parameter emphasizes the geometric nature of the problem and is often useful.

3. As multiples of b0:

```
gradient = (B'/B)B = kB ← k = k
brho = Bρ ← ρ = rho
b0 ← ω (relative horizontal curvature)
    ω = 0, for a quadrupole
    ω = 1, -1, for dipole
```

where ω , the relative horizontal curvature, is defined by $\omega = \text{curvature} \times \rho$. The sign of ω is defined so that ω is positive when the center of curvature of the bend lies in the negative x direction. Then,

```
-----1-----2-----3-----4-----5-----6-----7-----8
name MAG          length  k      rho      omega    e1      e2
-----1-----2-----3-----4-----5-----6-----7-----8
```

where omega is the value of ω .

Multiple magnets defined by one command line.

A series of lengths and/or gradients may be defined as the elements of vectors using the VEC command. The vectors can then be used to define several magnets with a single MAG statement (cf., DRF). The procedure is illustrated schematically as

```
-----1-----2-----3-----4-----5-----6-----7-----8
lvec VEC  m    l1      l2 ...  lm
gvec VEC  m    g1      g2 ...  gm
name MAG  m    lvec    gvec    br      b0
-----1-----2-----3-----4-----5-----6-----7-----8
```

The magnet names are constructed by appending to the first letter of name the digits 1, 2, ... The length of any vector used in this way must at least be equal to the number of requested magnets.

Several examples will illustrate this method.

```
-----1-----2-----3-----4-----5-----6-----7-----8
gf    =          <value>
gd    =          <value>
ql    =          <value>
ql1   =          <value>
ql2   =          <value>
gfv   VEC    2    gf      gd
glv   VEC    2    ql1     ql2
-----1-----2-----3-----4-----5-----6-----7-----8
```

MAG

(a) The following statement

```
-----1-----2-----3-----4-----5-----6-----7-----8
gmv  MAG    2    ql      gfv      brho
-----1-----2-----3-----4-----5-----6-----7-----8
```

is equivalent to

```
-----1-----2-----3-----4-----5-----6-----7-----8
g1   MAG          ql      gf      brho
g2   MAG          ql      gd      brho.
-----1-----2-----3-----4-----5-----6-----7-----8
```

(b) The following statement

```
-----1-----2-----3-----4-----5-----6-----7-----8
gmv  MAG    2    glv      gf      brho
-----1-----2-----3-----4-----5-----6-----7-----8
```

is equivalent to

```
-----1-----2-----3-----4-----5-----6-----7-----8
g1   MAG          ql1     gf      brho
g2   MAG          ql2     gf      brho.
-----1-----2-----3-----4-----5-----6-----7-----8
```

(c) The following statement

```
-----1-----2-----3-----4-----5-----6-----7-----8
gmv  MAG    2    glv      gfv      brho
-----1-----2-----3-----4-----5-----6-----7-----8
```

is equivalent to

```
-----1-----2-----3-----4-----5-----6-----7-----8
g1   MAG          ql1     gf      brho
g2   MAG          ql2     gd      brho.
-----1-----2-----3-----4-----5-----6-----7-----8
```

SEE ALSO: MAGV

MAGS – Magnet definition with errors

The MAGS command defines the transfer matrix representing a previously defined magnet with the effects of dipole field errors and/or misalignments added.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  MAGS   m    n  mg          kxin      kxout      kyin      kyout      db
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Name of the newly defined magnet which includes the effects of the errors.

m (IN) Selector to determine the interpretation of db.
 = 1 or blank, $db = dB/B$ and the MAG statement defining mg must have the scaling

```

      mg      MAG          length      k          rho          1.0

```

= 2, $db = dB$ and the MAG statement must have the scaling

```

      mg      MAG          length      gradient  brho          bo

```

n (IN) Randomizing switch.
 < 1 or blank, The errors are assigned the values kxin, kxout, kyin, kyout, and db.
 ≥ 1, The errors are assigned random values in the ranges [0, kxin), [0, kxout), [0, kyin), [0, kyout), and [0, db).

mg (CH) Previously defined magnet (MAG command).

kxin (FP) Radial displacement of magnet axis at magnet entrance.

kxout (FP) Radial displacement of magnet axis at magnet exit.

kyin (FP) Vertical displacement of magnet axis at magnet entrance.

kyout (FP) Vertical displacement of magnet axis at magnet exit.

db (FP) Field strength error. Interpretation controlled by parameter m, above.

Given the previously defined magnet mg, the transfer matrix for a new magnet, name, is calculated. Magnet name has the same magnet properties as mg, modified to account for the field error db (dB/B or dB) and misalignments.

MAGS

The **MAGS** command is used in conjunction with the **BMIS** and **EMIS** statements. The resulting closed orbit is found using a **CYC** command, in the columns containing the x , y dispersions and their slopes.

Alternate method: A more general approach, which also includes coupling effects, is to use the **MOVE** and **FXPT** commands.

SEE ALSO: **BMIS, EMIS, CYC**
MOVE, FXPT
Section 8.8, Magnet Misalignment Calculations

MAGV - Vertical Bending Magnet Definition

The MAGV command defines the transfer matrix for a vertically bending magnet.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  MAGV          length  gradient  brho      b0        e1        e2
-----1-----2-----3-----4-----5-----6-----7-----8

```

The parameter specifications for the MAGV command are identical to those for the MAG command, to which the user is referred for details.

The matrix created by the MAGV command is identical to that produced by the following sequence of commands

```

-----1-----2-----3-----4-----5-----6-----7-----8
R+   ROTZ          90.0
R-   ROTZ         -90.0
B1   MAG           length  gradient  brho      b0
B1V  MMM           R+   B1   R-
-----1-----2-----3-----4-----5-----6-----7-----8

```

SEE ALSO: MAG

MAP - Non-linear Transformation Definition

The **MAPk** commands allow use of built-in or user-written FORTRAN subroutines that specify particle coordinate transformations corresponding to special (usually nonlinear) beamline elements.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  MAPk   m      p1      p2      p3      ...      pm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name of the element defined by the **MAPk** command.

k (IN) Identifies the internal FORTRAN subroutine **MAPk** that specifies the transformation of the particle coordinate vector. Values $k = 0-9$ are reserved for built-in routines, and values $k = 10-19$ are available for user-written routines.

m (IN) Number of parameters defined by the **MAPk** command.

p1, ... (FP) The input parameter values. They are passed to the internal FORTRAN subroutine **MAPk**.

The element **name** operates on the particle state vector $V = (x, x', y, y', -ds, dp/p, 1)$ by means of the internal FORTRAN subroutine **MAPk**. It is included in a **BML** statement and invoked by a **TRK**, **FXPT**, or **TRKM** statement.

The use of the **MAPk** commands is explained further in Section 3.6. Chapter 7 contains an example with the FORTRAN subroutine **MAP0**, a quadratic mapping. **SYNCH** includes two built-in routines: **MAP0** and **MAP3**, which simulates the beam-beam force of a round beam.

SEE ALSO: Section 3.6, Non-linear Transformation Calculations
 Chapter 7, Non-linear Transformations
DEQ, **TRK**, **FXPT**, **TRKM**

MAT - General Matrix Definition

The MAT command is used to define an arbitrary matrix.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name MAT      m  n A11      A21      A31      A41      A51      A61
                   A71      ...      Am1
                   A12      A22      A32      A42      A52      A62
                   ...      Am2
                   A13      A23      A33      ...      Am3
                   .
                   .
                   .
                   A1n      A2n      A3n      A4n      A5n      A6n
                   A7n      ...      Amn
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name of the matrix.

m (IN) Row dimension of matrix.

n (IN) Column dimension of matrix.

A_{ij} (FP) The *ij*-th element of the matrix. Each column of the matrix must begin on a new line.

The MAT command is used to define an arbitrary m-row by n-column matrix. The matrix elements are read in sequentially by columns. Note that each column must start on a new line.

*

MAT3 - Define Transfer Matrix

The **MAT3** command defines a transfer matrix in terms of the 3×3 submatrices describing uncoupled motion.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  MAT3          rx11      rx12      rx13      rx21      rx22      rx23
                        ry11      ry12      ry13      ry21      ry22      ry23
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name of the matrix which is created.

rx11, ... (FP) Components of the 3×3 horizontal transfer matrix. The third row of the matrix is (0 0 1).

ry11, ... (FP) Components of the 3×3 vertical transfer matrix. The third row of the matrix is (0 0 1).

MESH - Loop Through SYNCH Subroutine Variables

The **MESH** command is used to repeatedly invoke a **SYNCH** subroutine while incrementing a specified set of defining variables.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      MESH      m      sbr
                   k1      a1      min1      max1      inc1
                   k2      a2      min2      max2      inc2
                   .
                   .
                   .
                   km      am      minm      maxm      incm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) Number of variables to be varied.

sbr (CH) Name of subroutine containing elements a_i .

k_i (IN) Position of floating point variable in defining statement of element a_i .

a_i (CH) Name of element containing the variable.

min_i (FP) Minimum value of variable k_i of element a_i .

max_i (FP) Maximum value of variable k_i of element a_i .

inc_i (FP) Increment of the variable k_i of element a_i .

The **MESH** command simulates a nested FORTRAN Do-loop. The subroutine **sbr** is executed repeatedly while the control variables k_i of elements a_i are incremented by inc_i . The first variable controls the outer loop; the m -th, the innermost loop.

MESH

Example:

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
SR      SUB
QF      MAG          2.0          0.5          1.0
QD      MAG          2.0         -0.05          1.0
.C      BML          QF  0      QD  QD  0      QF
C      CYC          .C
      END
      MESH    2      SR
                2      QF  0.5          1.0          0.1
                2      QD -1.0         -0.5          0.1
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

This example will produce a **CYC** printout of the cell **C** for all of the combinations of the gradients of the magnets **QF** and **QD** from 0.5 to 1.0 for **QF** and -1.0 to -0.5 for **QD**.

SEE ALSO: **INCR**
VPAR

MMM – Matrix Multiplication

The MMM command multiplies together a series of matrices representing magnets, drifts, other beamline elements, or whole beamlines to create a single matrix representing the sequence.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  MMM      m    n q1    q2    q3    q4    ...    qm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Control variable. Number of matrices to be multiplied. The use of this variable is optional.

n (IN) Control variable.
 = 0 or blank, Create a matrix representing the beamline by multiplying the matrices of the given elements $q_1, \dots, q_m$.
 < 0, A matrix corresponding to the reflection of the input beamline is appended to the input beamline. Thus only half of a symmetric superperiod need be specified.
 $|n| > 1$, The resulting matrix is raised to the $|n|$ th power.

q1, ... (CH) Names of previously defined elements (drifts, magnets, etc.) or beamlines.

For many purposes it is not necessary to step through a beamline element by element. Instead, a single matrix can be used to transport the particles through the beamline. The MMM command multiplies the matrices of $q_1, q_2, \dots, q_m$ taken in beamline order, and stores the result $M(\text{name}) = M(q_m) M(q_m - 1) \dots M(q_1)$ in the matrix **name**. The names $q_1, \dots, q_m$ represent the constituents (drifts, magnets, other MMMs, or beamlines) which comprise the beamline.

MOVE – Perform an Element Misalignment

The **MOVE** command defines a new element as an existing element that has been misaligned by translations along and/or rotations about one or more of the x , y , and s axes. Coupling effects are included automatically.

Format 1–

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  MOVE          n elem ray  disp
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

n (IN) Control variable to use specified or randomly generated misalignments.
 = 0 or blank, Use misalignments specified by command line parameters.
 = 1, Values of the misalignment parameters are randomly generated in the ranges $[-hx/2, hx/2]$, etc. New values are generated on subsequent invocations of the **MOVE** command.
 = 2, For the first execution of the command, random values are generated as for $n = 1$. For subsequent invocations, as in a repeatedly executed subroutine, the same (random) values are used. That is, the random number seed is preserved and used to restart the random number generator each time, thus generating the same sequence of random numbers and the same displacements.

elem (CH) Name of the element to be misaligned.

ray (CH) (Optional) Name of a particle vector defined by a **PVEC** command. The ray is mapped by the **MOVE** command to illustrate the effects of the misalignment; it is used in tracking and closed orbit calculations by, e.g., the **TRK**, **TRKB**, and **FXPT** commands.

disp (CH) Name of a vector (defined by a previous **VEC** command) which specifies the misalignment parameters. The six component vector is $(hx, dhx, hy, dhy, hs, dhs)$, where

hx = Translation distance along x -axis (meters).
 dhx = Rotation angle about y -axis (radians).
 hy = Translation distance along y -axis (meters).
 dhy = Rotation angle about x -axis (radians).
 hs = Translation distance along s -axis (meters).
 dhs = Rotation angle about s -axis (radians).

MOVE

Format 2-

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  MOVE          n elem ray          hx          dhx          hy          dhy
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

All command parameter definitions are the same as those for Format 1 except that the **disp** field must be blank and the transverse displacements are specified on the command line. Format 2 cannot be used to specify longitudinal moves.

The **MOVE** command defines a new element, **name**, which represents the element **elem** when misaligned as specified by the vector **disp**. The misalignment vector **disp** must be defined by a **VEC** command.

When using the **MOVE** statement, closed orbit calculations are to be made using the **FXPT** command. The effects of coupling are included.

Alternate method: Element misalignments can also be studied using the **BMIS-EMIS-MAGS** commands; then the closed orbit calculations are done using a **CYC** command. This method does not allow for coupling between the horizontal and vertical motion.

SEE ALSO: Section 8.8, Magnet Misalignment Calculations

BMIS
EMIS
FXPT
MAGS
TRK

MXV - Matrix-Vector Multiplication

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
vf      MXV              mtrx vi
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

vf (CH) Name of the 7-component product vector.

mtrx (CH) Name of a matrix.

vi (CH) Name of a particle vector defined by a **PVEC** or by another **MXV** command.

The **MXV** command transports the particle represented by **vi** through the beamline represented by **mtrx**, by multiplying the particle coordinate vector **vi** by **mtrx**. The particle's state vector after traversing the beamline is **vf**.

SEE ALSO: **PVEC**
TRK

88

NECHO - Suppress Printing of Input Statements

The **NECHO** command disables echoing of **SYNCH** input commands to the output file.

```
-----1-----2-----3-----4-----5-----6-----7-----8
      NECHO
-----1-----2-----3-----4-----5-----6-----7-----8
```

SYNCH normally echoes commands in the input stream to the output stream. The **NECHO** command disables echoing. Echoing may be restored using the **ECHO** command.

SEE ALSO: **ECHO**

(

56

NPOL – Define N -pole Magnet

The NPOL command defines the particle-coordinate transformation through an N -pole magnet.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name NPOL  m  k length  coeff  brho
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name of element created.

m (IN) Order of perturbation in Hamiltonian, defined as $m = N/2$. The corresponding multipole order is $n = m - 1$. Some examples are given in the table.

field	N	m	n
dipole	2	1	0
quadrupole	4	2	1
sextupole	6	3	2
octopole	8	4	3
decapole	10	5	4
...			

k (IN) Select normal or skew field.
 = 0, Normal N -pole thin lens.
 = 1, Skew N -pole thin lens.

length (FP) Effective length of element.

coeff (FP) Taylor series expansion coefficient b_n of the median-plane field. The strength of the thin-lens kick is given by

$$S = \begin{cases} \frac{\text{coeff}}{\text{brho}} & \text{length} = 0 \\ \frac{\text{length} \times \text{coeff}}{\text{brho}} & \text{length} \neq 0 \end{cases}$$

brho (FP) Magnetic rigidity.

The NPOL command is used to define the particle-coordinate transformation through an N -pole magnet. The multipole field itself is represented by thin-lens approximation. If the element has non-zero length, the transformation is obtained by concatenating a drift, the thin-lens transformation, and another drift. Each drift is half the length of the element.

NPOL

The thin-lens multipole field is expressed as the term of appropriate order in the Taylor series expansion of the (median plane) field:

$$B_y + iB_x = \begin{Bmatrix} 1 \\ i \end{Bmatrix} B_n \begin{cases} \text{normal; } k = 0 \\ \text{skew; } k = 1 \end{cases}$$

where $B_n = [b_n/n!]z^n$ and $z = x + iy$. The N -pole thin-lens magnet changes the particle momenta according to:

$$p_z \rightarrow p_z - (1 + i)(B_n/B\rho),$$

where

$$p_z = p_x + ip_y.$$

In general, NPOL is used to implement a nonlinear beamline element. For closed orbit and other calculations performed by **FXPT**, which map rays slightly displaced from the reference ray, the transformation is linearized and represented by the matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -Q & 1 & -iQ & 0 \\ 0 & 0 & 1 & 0 \\ -iQ & 0 & 0 & 1 \end{pmatrix}$$

where

$$Q = \begin{Bmatrix} 1 \\ i \end{Bmatrix} \frac{b_n}{(n-1)!} z^{n-1} \begin{cases} \text{normal; } k = 0 \\ \text{skew; } k = 1 \end{cases}$$

SEE ALSO: **SXTP**

OPEN – Open special output files

The **OPEN** command opens files intended to receive output from particular commands and to be saved at completion of the job.

```

-----1-----2-----3-----4-----5-----6-----7-----8
      OPEN      fn1 fn2 ... fni
-----1-----2-----3-----4-----5-----6-----7-----8

```

fn1, ... (CH) Names of the files to be opened.

It is sometimes convenient to write files required for input to another program. The **OPEN** command allows such files to be created only when needed.

The files controlled by **OPEN** are **FIL11** (logical unit 11) and **PLFIL** (logical unit 17).

SEE ALSO: Appendix A, Files
BEST
FXPT

ORBC – Calculate Closed Orbit Corrections

The **ORBC** command calculates optimized correction element strengths necessary to correct a given closed orbit in an accelerator with field errors.

```

-----1-----2-----3-----4-----5-----6-----7-----8
      ORBC  m  n  fxpt      mon  cor  deltax  deltam
-----1-----2-----3-----4-----5-----6-----7-----8

```

- m** (IN) Select correction in horizontal or vertical plane.
 = 0, Horizontal corrections.
 = 1, Vertical corrections.
- n** (IN) Iteration limit and print option selector. For any n , $|n|$ is the maximum allowed number of iterations.
 < 0, All intermediate results as well as the matrix relating the orbit displacements to the correction elements are printed.
 > 0, Only the results of the first and last iterations are printed. The matrix relating the orbit displacements to the correction elements is not printed.
- fxpt** (CH) Name of **FXPT** statement defining initial orbit. The **FXPT** statement used must have a 1 or 2 in column 55 to tell the program to write closed orbit information to the file **ORB1** (logical unit 24).
- mon** (CH) Name of elements in beamline where displacements are to be measured. This name will occur at many points in the beamline.
- cor** (CH) Name of correction elements defined by **KICK** statements in the beamline. There must be many correctors, all with this name, distributed along the beamline.
- deltax** (FP) Tolerable range of final residual orbit errors, in meters. The correction is successful if all the residual orbit errors are in the range $(-\text{deltax}/2, \text{deltax}/2)$.
- deltam** (FP) Assumed measurement error, in millimeters. **ORBC** calculates the corrections assuming that the uncorrected error at each monitor is $\pm \text{deltam}/2$.

NOTE: **ORBC** makes specific assumptions about the units used for **deltax** and **deltam** (m and mm, respectively). Meters must be used as the unit for all lengths in the lattice description.

ORBC

The **ORBC** command calculates optimized correction element strengths necessary to correct a given closed orbit in an accelerator with field errors. The command uses the MICADO^[9] procedure.

The orbit to be corrected is obtained using the **FXPT** command. The beamline submitted to **FXPT** must contain not more than 150 identically named elements at which the orbit is measured, and some number of identically named correction elements, defined by a **KICK** command, where the correction kicks are to be applied. The **FXPT** parameter *itap* (col. 55) must be set to 1 or 2. Before the first use (or pair of uses—see below) of **ORBC** it should be set to 1; this causes the corrections to be calculated for the original closed orbit with its errors. If only one iteration of corrections is desired, invoke **FXPT** with *itap* = 2 and then run **ORBC** again; this will cause further corrections to be superimposed on the ones calculated in the previous pass. To see the corrected orbit, invoke **FXPT** with *itap* = 0 or 2 (not 1) after the last **ORBC**.

The correction calculations are performed when **ORBC** is invoked following the **FXPT** calculation with *itap* = 1 or 2. **ORBC** only calculates corrections for one plane (*x* or *y*) in each invocation; therefore if both horizontal and vertical corrections are wanted, two separate **ORBC** commands are needed after each **FXPT**.

The maximum number of iterations is $|n|$ and must not exceed the lesser of 150 and the number of correctors. If *n* is negative, all intermediate iterations as well as the matrix relating orbit displacements to correction elements is printed out. If *n* is positive, only first and last iterations are printed and matrix print is suppressed.

Iteration is terminated when the iteration limit is reached, or when the residual errors in the corrected orbit are in the range $(-\text{deltax}/2, \text{deltax}/2)$.

The final corrected orbit is computed using another **FXPT** command, this time with *itap* = 0.

SEE ALSO: **FXPT**
KEEP
KICK

PAGE – Skip to Start of Next Page

The **PAGE** or **P** command forces output to skip to the beginning of the next page. Header text may be inserted and a date-time stamp printed.

Format 1:

Format 1 (**P** in col. 1) is used to insert a line of text at the top of the next page. The output then continues normally. The date-time stamp option is not available with this format.

```

-----1-----2-----3-----4-----5-----6-----7-----8
P<-----                                text                                ----->
-----1-----2-----3-----4-----5-----6-----7-----8

```

Format 2:

In addition to inserting a line of text at the top of the next page, this statement may be referenced by **CYC** to insert header text for its output, as well as the time/date stamp on option.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name PAGE m <-----                                text                                ----->
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name. Must be provided if the command is used to define a header for **CYC** output.

m (IN) Date-time stamp selector.
 ≠ 1, The date-time stamp is omitted.
 = 1, A date-time stamp is appended to the header.

text (CH) Page header text. The maximum string length is 79 characters for Format 1 or 60 characters for Format 2. The text field may be blank.

When **PAGE** or **P** is in a **SYNCH** subroutine, a page will be ejected each time the subroutine is invoked by a **CALL** command. If the subroutine is invoked by an iterative process, e.g. **FITQ**, **SOLV**, etc., the **PAGE** or **P** command is bypassed. It is recommended that these commands only be used outside subroutines.

SEE ALSO: **REM**
C

RUN

PARA - Define a Parameter Value

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  PARA          fpval    exp
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name.

fpval (FP) A floating point number.

exp (IN) An exponent. If **exp** = 0 or blank, the floating point value itself is stored. If **exp** ≠ 0, then the value of **fpval** times 10 to the power **exp** is stored for use by the program.

The parameter values specified by **PARA** commands may be updated in the output when a new **SYNCH** input file is created by the **UPDAT** command. Otherwise, the **PARA** command functions like the = command.

SEE ALSO: **SELCT**
UPDAT
 =

PBML - Print Beamline

The PBML command prints a sequential list of the elements that comprise the beamline, together with their position, name, cumulative and element lengths, command, and input data.

```
-----1-----2-----3-----4-----5-----6-----7-----8
      PBML      bmln
-----1-----2-----3-----4-----5-----6-----7-----8
```

bmln (CH) Name of beamline.

PCYC - Select Restricted CYC Print List

The **PCYC** command defines the initial characters of the names of elements to be printed in the **CYC** or **FXPT** output when abbreviated output is requested.

```

-----1-----2-----3-----4-----5-----6-----7-----8
      PCYC      a1  a2  a3  ...      ak  ...                        a12
-----1-----2-----3-----4-----5-----6-----7-----8

```

a1, ... (CH) Any single letter or special character.

The **PCYC** command defines a list of characters used to select elements at which output is to be produced. If, in the **CYC** or **FXPT** commands, the **m** parameter is negative, output is suppressed except at elements whose names begin with one of the characters in the list. Up to 12 characters can be specified in a **PCYC** command. Only one **PCYC** list is active at a time; the last one defined will be used.

If a **PCYC** command is placed in a subroutine, it is not active until the subroutine is invoked.

If no **PCYC** command precedes the **CYC** or **FXPT** command, the default character for selective printing, the **#** character, is used.

SEE ALSO: **CYC**
FXPT

PRNT - Print Element Parameter

The **PRNT** command causes printing of specified parameters of a list of elements.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      PRNT   m   n a1   a2   a3   ...   ak
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) Number of the parameter to be printed for each element **a1** The default is **m = 1**.

n (IN) Parameter type identifier
 = 1, Floating point input number in the statement defining **a1**
 = 2, Alphanumeric input character string in the statement defining **a1**
 = 3, Integer input number in the statement defining **a1**
 = 4, Not used.
 = 5 or blank, Internally calculated floating point number of **a1** (Default).
 = 7, Floating point number from LQ2 storage (for debugging use).
 = 8, Floating point number from LQ3 storage (for debugging use).

a1 ... (CH) Names of statements for which data is to be printed.

The **PRNT** command prints out the current value of the **m**-th parameter of type **n** of each element or variable listed, **a1**, **a2**, **a3**, ..., **ak**.

Example 1: Printing parameter values:

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
K      =                2.
K1     =                .002
K2     =                -2.      /      1000.
K3     PARA             .003
K4     PARA             -3.      -3
K5     CALC             RCL K    CHS .001 *
      PRNT             K    K1   K2   K4   K5
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

The above **PRNT** command generates the following output:

K	K1	K2	K3	K4	K5
2.0	.002	-.002	.003	-.003	-.002

PRNT

Example 2: In order to print out magnet parameters, etc., one uses the n = 1 option:

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
LQ      =                1.8
QF      MAG              2.          .0025      1.
QD      MAG              2.          -.0025      1.
QF1     MAG              LQ          K1          1.
QD1     MAG              LQ          K2          1.
QF2     MAG              LQ          K3          1.
QD2     MAG              LQ          K4          1.
        PRNT  1  1  QF  QD  QF1  QD1  QF2  QD2
        PRNT  2  1  QF  QD  QF1  QD1  QF2  QD2
        REPL  1    QF  F   2.2
        REPL  1    LQ  F   1.9
        PRNT  1  1  QF  QD  QF1  QD1  QF2  QD2
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

The above three PRNT statements generate the following output:

QF	QD	QF1	QD1	QF2	QD2
2.0	2.0	1.8	1.8	1.8	1.8
QF	QD	QF1	QD1	QF2	QD2
0.0025	-0.0025	0.002	-0.002	0.003	-0.003
QF	QD	QF1	QD1	QF2	QD2
2.2	2.0	1.9	1.9	1.9	1.9

PRTV - Print a List of Vectors

The **PRTV** command prints the values of the components of a list of vectors defined by **VEC** statements.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      PRTV          v1   v2   v3   ...   vm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

vi (CH) Names of the vectors to be printed.

The **VEC** command is used to define vectors of any length. The **PRTV** command is used to display their values.

The command pair (**VEC**, **PRTV**) is to be distinguished from the pair (**PVEC**, **PRV7**).

SEE ALSO: **VEC**
PVEC
PRV7

PRV7 - Print a List of Particle Vectors

The **PRV7** command prints the values of the components of a list of particle vectors defined by **PVEC** statements.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      PRV7          v1    v2    v3    ...    vm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

vi (CH) Names of the particle vectors to be printed.

The **PVEC** command is used to define seven-component particle vectors. The **PRV7** command is used to display the values of these vectors.

The command pair (**VEC**, **PRTV**) is to be distinguished from the pair (**PVEC**, **PRV7**).

SEE ALSO: **VEC**
PVEC
PRTV

1		2		3		4		5		6		7		8	
pv	PVEC	m	n	x1	xp1	y1	yp1	-ds1	dp/p1						
				x2	xp2	y2	yp2	-ds2	dp/p2						
				.	.	.	.	.	.						
				.	.	.	.	.	.						
				.	.	.	.	.	.						
				xm	xpm	ym	ypm	-dsm	dp/pm						

The **PVEC** is used to define a set of particle state vectors for use as initial values in tracking calculations. The state vectors are vectors in the 7-dimensional phase space.

Particle vectors are to be distinguished from more general vectors defined by the **VEC** command.

SEE ALSO: PRV7
VEC

RAND - Generate Random Number

RAND returns a pseudo-random number in a range defined by the user.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
rn      RAND   m    n x              y              z
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

rn (CH) Name of variable to be assigned the value of the random number.

m, n Repetition counters.

$m = 0, n = 0$, Run the random number generator once.

$m \neq 0, n \neq 0$, Run the random number generator $m * n$ times so as to obtain a new seed for the next use of random numbers.

x, y (FP) Scale factors to determine the range of values returned in rn. The returned value is determined by

$$rn = x + (g - .5) * y$$

where g is a random number in the range $[0, 1)$. Then the range of values spanned by rn is $[x - y/2, x + y/2)$: an interval of width y centered at x .

z (FP) Initialization value to reset the seed for random number generator. The seed is reset before any random numbers are generated. Seed values should be large odd integers, with a decimal point included.

The internal random number generator generates pseudo-random numbers in the range $[0, 1)$. The input parameters x and y are used to scale the internal value into the range required for the calculations.

REF - Matrix Reflection

REF creates the transfer matrix of the beamline that is the mirror reflection of another beamline.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
r      REF      p
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

r (CH) Name of the transfer matrix of the mirror-reflected beamline.

p (CH) Name of the transfer matrix to be reflected.

REF creates a new transfer matrix r which represents the reflection of p; i.e., if p represents the transfer matrix corresponding to the sequence of elements $A_1, A_2, \dots, A_n$, then r represents the transfer matrix corresponding to the sequence $\bar{A}_n, \bar{A}_{n-1}, \dots, \bar{A}_2, \bar{A}_1$, where $\bar{A}_i$ is the matrix corresponding to the reflection of A_i . Thus, if the horizontal portion of p is represented by

$$P_x = \begin{bmatrix} a & b & e \\ c & d & f \\ 0 & 0 & 1 \end{bmatrix},$$

then

$$R_x = \begin{bmatrix} d & b & -de + bf \\ c & a & -ce + af \\ 0 & 0 & 1 \end{bmatrix}.$$

is the matrix for the reflected beamline.

The primitive elements, magnets and drifts, which comprise a beamline are naturally self-reflecting. It is possible, however, that a beamline may include in its definition a matrix representing a portion of another beamline that is not self-reflecting. But even in this case, the r matrix will be a correct representation of the reflected beamline, thought of in terms of its primitive elements.

REM - Insert Remark in Output Listing

The **REM** command inserts a comment in the output file and/or defines headers for tables of lattice functions written by **CYC** commands. A date-time stamp in the table header is optional.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  REM      m      <-----                        text                        ----->
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name. Used by **CYC** to reference the **REM** statement.

m (IN) Date-time stamp option switch.
 ≠ 1, No date-time stamp is appended.
 = 1, A date-time stamp is appended to header on each page of the tables written by **CYC** commands.

text (CH) Comment text. Maximum of 60 characters.

The **REM** command inserts the text string in the output listing, as do the **C** and **.** commands.

SEE ALSO: **C**

.
PAGE
P
CYC

REPL - Replace One Parameter With Another

REPL substitutes a new value for a parameter of another statement.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      REPL      m      stmt type value
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) Index number of the parameter to be replaced.

stmt (CH) Reference name of the statement for which the parameter is to be replaced.

type (CH) The type of data value to be replaced. The new value replaces the **m**-th parameter of the corresponding type.

- = F, (Default) The new data value is type FLOATING POINT.
- = C, The new data value is type CHARACTER.
- = I, The new data value is type INTEGER.
- = KA, The new data value is an INTEGER value for the parameter **m**.
- = KB, The new data value is an INTEGER value for the parameter **n**.
- = SF, The new data value is a SYMBOLIC FLOATING POINT value, which replaces a floating point value (either type F or SF).

value Numeric or symbolic value as required by **type** defining the replacement value for the parameter.

REPL causes the value of the **m**-th parameter of type **type** associated with **stmt** to be replaced by **value**.

REPL changes only the input data, not the corresponding matrix. The matrix will be updated only if the command is re-executed, as in a SYNCH subroutine.

REPL

Example:

```
-----1-----2-----3-----4-----5-----6-----7-----8
QD  MAG          2.0      -50.0      brho      b0      $

:

      REPL  2      QD  F      -50.05
-----1-----2-----3-----4-----5-----6-----7-----8
```

Here, the second parameter (gradient = -50.0) of the MAG statement QD is replaced by -50.05.

The following example shows the use of a Symbolic Floating Point value to effect the same change.

```
-----1-----2-----3-----4-----5-----6-----7-----8
GRAD  =          -50.
QD  MAG          2.0      GRAD      brho      b0      $

:

GRAD2 =          -50.05

:

      REPL  2      QD  SF      GRAD2
-----1-----2-----3-----4-----5-----6-----7-----8
```

Equivalently one may write

```
-----1-----2-----3-----4-----5-----6-----7-----8
      REPL  1      GRAD      -50.05
-----1-----2-----3-----4-----5-----6-----7-----8
```

In both examples, the matrix associated with QD is not updated until the MAG command is re-executed.

SEE ALSO: INCR

ROT – Rotate a Beamline Element

The **ROT** command calculates the transfer matrix of a beamline element, rotated about the longitudinal *s*-axis.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  ROT              elmn      theta
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name of the rotated element.

elmn (CH) Reference name of the element which is to be rotated.

theta (FP) Value of the rotation angle, in degrees. In the right-handed *xys* system, a positive theta rotates the *x*-axis towards the *y*-axis.

SEE ALSO: **ROTZ**
Section 8.2, Linear Elements

ROTZ - Define Rotation Matrix

The **ROTZ** command defines a matrix representing a rotation in the transverse x - y plane about the longitudinal s axis.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  ROTZ          theta
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name of the matrix representing the rotation.

theta (FP) Angle of rotation about the s -axis, in degrees.

SEE ALSO: **ROT**
Section 8.2, Linear Elements

RUN - Start of Run

The **RUN** statement signals the start of a **SYNCH** run, and must be the first of the set of statements that comprise the input.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  RUN      m      <-----+-----+-----+-----+-----+-----+-----+-----+-----+----->
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) A name which will be printed in the output listing as a run-identifier. This name is required only if the **RUN** statement is referenced by a **CYC** command.

m (IN) Date-time stamp selector. The **RUN** statement may be referenced by **name** by a **CYC** statement, in which case the **text** is used as a header for the tables written by the **CYC** command. A date-time stamp is appended to the text string in the header.
 $\neq 1$, The date-time stamp is omitted.
 $= 1$, The date-time stamp is appended to the text string.

text (CH) Arbitrary comments (up to 60 characters in columns 21-80) which will be printed in the output listing.

The **RUN** statement must be used to signal the start of a **SYNCH** run. The **name** and any comments in the text field are printed at the beginning and end of the output listing for identification. The date and time are also printed at the start of the output listing.

SEE ALSO: **STOP**

SELECT - Delimiter Marker

The **SELECT** command is a marker to delimit a set of **PARA** commands, which may be updated by an **UPDAT** command.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  SELECT
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name to identify a group of **PARA** commands which may be updated. The group must be enclosed by a pair of **SELECT** commands having the same reference name.

The **SELECT** command is used to delimit groups of one or more **PARA** commands, the parameter values of which are to be updated by the results of the current run before being written to a file by an **UPDAT** command. The group to be updated is identified by preceding and following it by **SELECT** commands bearing the same name.

SELECT commands cannot be nested.

Each pair of **SELECT** commands must have a unique name.

SEE ALSO: **UPDAT**

SHF - Define a 3×3 Shift Matrix

The **SHF** command defines 3×3 shift matrices in the transverse dimensions.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  SHF              mx1      mx2      my1      my2
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

The **SHF** command defines shift matrices, used to describe alignment errors, in the transverse coordinates. The matrices are of the form

$$S_x = \begin{bmatrix} 1 & 0 & mx1 \\ 0 & 1 & mx2 \\ 0 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} 1 & 0 & my1 \\ 0 & 1 & my2 \\ 0 & 0 & 1 \end{bmatrix}.$$

The equivalent 7×7 matrix is

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & mx1 \\ 0 & 1 & 0 & 0 & 0 & 0 & mx2 \\ 0 & 0 & 1 & 0 & 0 & 0 & my1 \\ 0 & 0 & 0 & 1 & 0 & 0 & my2 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

SEE ALSO: **SHF7**

Section 8.8, Magnet Misalignment Calculations

SHF7 – Define a 7×7 Shift Matrix

The **SHF7** command defines a 7×7 shift matrix, used for misalignment calculations.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name SHF7          r17      r27      r37      r47      r57      r67
-----1-----2-----3-----4-----5-----6-----7-----8

```

The shift matrix defined by **SHF7** is of the form

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & r17 \\ 0 & 1 & 0 & 0 & 0 & 0 & r27 \\ 0 & 0 & 1 & 0 & 0 & 0 & r37 \\ 0 & 0 & 0 & 1 & 0 & 0 & r47 \\ 0 & 0 & 0 & 0 & 1 & 0 & r57 \\ 0 & 0 & 0 & 0 & 0 & 1 & r67 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

SEE ALSO: **SHF**

Section 8.8, Magnet Misalignment Calculations

SIZE - Define Size of Matrices

The **SIZE** command is used to override or reset the default specification for the dimensionality of the state vectors and transformation matrices used by **SYNCH** and to control the formatting of printed output.

```

-----1-----2-----3-----4-----5-----6-----7-----8
      SIZE      k
-----1-----2-----3-----4-----5-----6-----7-----8

```

k (IN) Defines the dimension of matrices and state vectors used within **SYNCH**.
 = 3, The state vectors are 3-dimensional and the matrices, 3×3 . This is the default size with most commands.
 = 7, The state vectors are 7-dimensional and the matrices, 7×7 .

The matrices and state vectors used by **SYNCH** are, for some commands, intrinsically 3-dimensional and for others, 7-dimensional. Internally, **SYNCH** elevates 3-dimensional entities to 7-dimensions and performs all calculations in a 7-dimensional space.

The **MMM** command provides an exception to this general rule. When a matrix is defined by **MMM** to be the product of a set of intrinsically 3×3 matrices (for example, drifts, dipoles and quadrupoles) the matrix will be stored as a 7×7 matrix if **SIZE** has been used to specify 7-dimensional vectors/matrices. This provides for improved efficiency of execution at the expense of larger required memory. If any of the matrices referenced by the **MMM** command are 7×7 , the resulting matrix will be 7×7 regardless of any **SIZE** command.

When **SYNCH** is instructed to print a state vector or a matrix, it will by default print the entity in its intrinsic dimension. This may be inconvenient when comparing results. In this case, using the **SIZE** command allows the user to force all the matrices and state vectors to be printed as either 3- or 7-dimensional entities. It is only with the **WMA** command that this is a concern.

SIZE

The state vectors on which **SYNCH** operates are, in 3-dimensions, defined to be,

$$\begin{bmatrix} x \\ x' \\ \delta \end{bmatrix}, \begin{bmatrix} y \\ y' \\ \delta \end{bmatrix}$$

and, in 7 dimensions,

$$\begin{bmatrix} x \\ x' \\ y \\ y' \\ h \\ \delta \\ 1 \end{bmatrix}$$

where x, y = displacements, x', y' = slopes, $\delta = dp/p$, the fractional momentum error, and $h = -ds = -(\text{azimuthal position of particle relative to position of reference particle})$.

SEE ALSO: WMA

SMIN - Access MINUIT from SYNCH

The **SMIN** command allows the user to directly submit **MINUIT**^[6] commands from within a **SYNCH** program.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      SMIN      m      stat1
                      stat2
                      .
                      .
                      .
                      statm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) The total number of **MINUIT** commands to be submitted.

stat*i* **MINUIT** command string. The data are formatted like the corresponding **MINUIT** command: a 10-character field for the command name followed by 5 floating point data, each in a field 10 characters wide, all offset to start in column 21 rather than column 1. See the **MINUIT** manual for descriptions of the available commands.

MINUIT is a standard fitting package, distributed by CERN, used by the **SYNCH** command **SOLV**. The **SMIN** command allows the user to control the use of **MINUIT** by **SOLV**. If the input does not contain an **SMIN** statement, the **MINUIT** commands

```

PRINTOUT      0.
MINIMIZE      1000.      1.0
END RETURN.

```

will be executed by **SOLV**.

Example:

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
FIT  SMIN      4      PRINTOUT  2.
                      SEEK      500.
                      SIMPLEX   1500.      0.1
                      END RETURN
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

SEE ALSO: **SOLV**

SOL - Solenoid

The SOL command defines a solenoid magnet.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name SOL      length  (blank)  brho    b0
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name of the element defined by the SOL command.

length (FP) Effective solenoid length.

brho (FP) Magnetic rigidity of the particle beam, $B\rho$.

b0 (FP) Solenoid magnetic field, B .

SOL computes a transfer matrix (which effects x - y coupling and focusing) for a solenoid.

As in the case of MAG, the parameters brho and b0 may be scaled by an arbitrary factor without changing the actual computational result; only the ratio of these two quantities is significant. Convenient choices are:

1.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name SOL      length  (blank)  brho    b0
-----1-----2-----3-----4-----5-----6-----7-----8

```

brho = magnetic rigidity

b0 = solenoid field

2.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name SOL      length  (blank)  1.0      1/rho
-----1-----2-----3-----4-----5-----6-----7-----8

```

brho = 1

1/rho = field/rigidity = $1/(\text{radius of curvature of particle in field of solenoid})$

SOL

3.

----	1	----	2	----	3	----	4	----	5	----	6	----	7	----	8
name	SOL		length		(blank)		length		theta						
----	1	----	2	----	3	----	4	----	5	----	6	----	7	----	8

$\theta = \text{length} \times \text{field} / \text{rigidity}$
= deflection angle of the particle if it were to traverse the distance
length in a transverse field equal to the solenoid's longitudinal field

SEE ALSO: **TRK**
FXPT
Section 8.2, Linear Elements

SOLV – General Fitting Routine

SOLV is used to obtain specified values of the betatron functions at various points along a beamline. The betatron function are tracked from given initial values through the beamline by **TRKB**, then **SOLV** invokes the **MINUIT** optimization program to minimize a goodness-of-fit function, *FCN*. **SOLV** may also be used to fit matrix elements or parameters defining non-linear **MAP** or **DEQ** statements.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      SOLV  m   n sbr  cmd pos1 pos2 itr  itol prn  sav  pri          lambda
              { m lines specifying the constraints }
              { n lines specifying the fitting variables }
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) Number of constraint lines.

n (IN) Number of variable lines.

sbr (CH) Name of **SYNCH** subroutine in which the fitting variables and the statement named **cmd** are located.

cmd (CH) Name of a **TRKB** statement that implements tracking of the betatron functions along the beamline or of an **MMM** statement which defines a transfer matrix that is to be fitted using the **MTRX** constraint.

pos1,pos2 (IN) Optional. Start and end positions to be used in the **TRKB** statement. If present, these values override the values given in the **TRKB** statement.

itr (IN) Maximum number of iterations in the fitting process (default = 1000).

itol (IN) Exponent of desired tolerance on *FCN*, (see 'The Fitting Procedure' below).

prn (IN) **MINUIT** control parameter. Printout level selector. The **MINUIT** summary output is written to FORTRAN logical unit 4 (see Appendix A). Setting **prn** = 0 produces the least output.

sav (IN) Controls disposition of fitted variable values.
 = 0 or blank, The fitted values of the variables replace the starting values.
 = 1, The fitted variables retain their original values upon completion of fitting.

SOLV

- pri** (IN) Control printing of betatron functions.
= 0 or blank, Print the betatron functions before and after fitting.
= 1, Print betatron functions after fitting only.
= 2, Do not print betatron functions.
- lambda** (FP) Factor used if the magnitudes of the variables are to be included in the goodness-of-fit function. See the section on the variables, below.

Specifying the Constraints

The **m** lines following the **SOLV** command line specify the fitting constraints. Each constraint line has the following format.

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
              bcon type jp1  jp2  jp3  jp4  jp5  jp6  des          sigma
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

- bcon** (CH) Name of the quantity or quantities fitted to the value **des** at location(s) **jp1**, **jp2**,
Listed below are the possible choices for **bcon** (with one exception to be explained below, under **type**).

- AX, AY** Fit alpha function: $\alpha_x = \text{des}$ or $\alpha_y = \text{des}$.
AXAY Fit $\alpha_x = \alpha_y = \text{des}$.
AXMY Fit $\alpha_x - \alpha_y = \text{des}$.
ARFL Fit $\alpha_x = -\alpha_y$ and $\beta_x = \beta_y$. If $\text{des} \neq 0$ then set $\beta_x = \beta_y = \text{des}$.
BX, BY Fit beta function: $\beta_x = \text{des}$ or $\beta_y = \text{des}$.
BXBY Fit $\beta_x/\beta_y = \text{des}$.
BXMY Fit $\beta_x - \beta_y = \text{des}$.
DX, DY Fit slope of dispersion function: $\eta'_x = \text{des}$ or $\eta'_y = \text{des}$.
GX, GY Fit gamma function: $\gamma_x = \text{des}$ or $\gamma_y = \text{des}$.
MTRX Constrain element (**jp1**, **jp2**) of matrix **cmd** on the **SOLV** command line to be equal to **des**. Note that **jp3**, ..., **jp6** should be left blank.
NUX, NUY Fit cumulative phase advance, $\nu_x = \text{des}$, or $\nu_y = \text{des}$, in units of 2π .
RND Fit $\alpha_x = \alpha_y = \eta'_x = 0$ and $\beta_x/\beta_y = \text{des}$.
RND2 Fit $\alpha_x = \alpha_y = 0$ and $\beta_x/\beta_y = \text{des}$.
S Fit cumulative path length: $S = \text{des}$.
THET Fit cumulative bending angle: $\theta = \text{des}$.

VX, VY Constrain x or y -coordinate of tracked ray to be equal to **des**.
VDX, VDY Constrain slope of x or y -coordinate of tracked ray to be equal to **des**.
WST Fit $\alpha_x = \alpha_y = \eta'_x = \text{des}$.
X, Y Fit dispersion function: $\eta_x = \text{des}$ or $\eta_y = \text{des}$.
XDX Fit $\eta_x = \eta'_x = \text{des}$.

type (CH) Modifies the effect of the constraint.
 In the case of **type** = **BFIT**, a new constraint is specified. This is the exception, noted previously, to the rule that **bcon** should have one of the values **AX**, **AY**, ..., listed above.

blank Unmodified constraint, as defined in the table above under **bcon**.
NO Ignore this constraint.

DIF Make the values of the quantity **bcon** at positions **jp2**, **jp3**, ... equal to the value at position **jp1**.

SUM Make the values of the quantity **bcon** at positions **jp2**, **jp3**, ... equal to the negative of the value at position **jp1**.

LSTN Add $(\text{bcon} - \text{des})^{\text{jp1}}$ to *FCN* at every position in the ranges **jp2**–**jp3** and **jp4**–**jp5**.

< Add $(\text{bcon} - \text{des})^4$ to *FCN* if **bcon** > **des**.

> Adds $(\text{bcon} - \text{des})^4$ to *FCN* if **bcon** < **des**.

<> Adds $(|\text{bcon}| - \text{des})^4$ to *FCN* if $|\text{bcon}| > \text{des}$.

BFIT Fit all of the betatron functions tracked to position **jp1** to be equal to those of the previously defined matrix whose name is specified by **bcon**.
 In this case, set **des** = 0.

jp*i* (IN) Position number(s) in beamline at which the constraint **bcon** is to be applied. See also constraint type **MTRX** under **bcon** and **LSTN** under **type** for other meanings of the indices **jp*i***.

Alternatively, the positions **jp*i*** can be specified by the character names of the corresponding beamline elements. If the position represents the k^{th} instance of the element, then k ($1 \leq k \leq 9$) should be placed in the 5th position of the field.

des (FP) Desired value of the quantity to be fitted.

sigma (FP) Desired tolerance on fitted value. (The inverse weight.)

SOLV

Each of the *m* constraint lines specifies a constraint to be applied at one or more beamline positions. If no positions are specified on a constraint line, those positions which have been most recently given on a preceding constraint line will be used.

If the constraint name, *bcon*, is not recognized, the constraint will be ignored. However, any positions entered on such a line will be available for use by subsequent lines.

Specifying the Variables

Each variable (element) has one or more associated parameters. For example, the parameters associated with a magnet are its length, gradient, beam rigidity, strength, and entrance- and exit-face angles. Each parameter is referred to by its sequence number in the defining command.

The next *n* lines specify the variables (parameters) that are to be varied in the fitting.

There are two formats available to specify the variables. Each Format 1 line can specify up to five variables and the number of the parameter to be fitted. (The number of the parameter is the same for each of the variables.) Each Format 2 record specifies one variable and up to five of its parameters. Each record also specifies the upper and lower bound on the fitting variables and the step size to be used.

Format 1:

```
-----1-----2-----3-----4-----5-----6-----7-----8
          vr1 vr2 vr3 vr4 vr5 ipm bdlo      bdhi      stpsz
-----1-----2-----3-----4-----5-----6-----7-----8
```

Format 2:

```
-----1-----2-----3-----4-----5-----6-----7-----8
          vr  ipm1 ipm2 ipm3 ipm4 ipm5 bdlo      bdhi      stpsz
-----1-----2-----3-----4-----5-----6-----7-----8
```

vri, *vr* (CH) Name of variable(s) to be fitted.

ipm, *ipmi* (IN) Parameter(s) of *vr*(s) to be varied in the fitting process.

bdlo (FP) Lower bound of the *ipm*-th parameter of *vr*, *vri*.

bdhi (FP) Upper bound of the *ipm*-th parameter of *vr*, *vri*.

stpsz (FP) Step size used when varying *ipm*-th parameter of *vr*, *vri*.
= blank, Default to $(bdhi - bdlo)/2$.
< 0, Variable enters into *FCN* with fit value = $(bdlo + bdhi)/2$ and with
tolerance = $(\lambda) \times (bdhi - bdlo)/2$.

The Fitting Procedure

The i pm-th parameters of the vr are varied through various minimization routines to satisfy the constraints, $bcon$. A fit has been accomplished when the value of the function

$$FCN = \Sigma \{[(value - des)/sigma]^2\} / m$$

where m , the number of fitting constraints,* is below $10^{2 \times itol}$. If the value of FCN does not reach $10^{2 \times itol}$ within itr calls to MINUIT, execution of the SOLV statement is halted.

If the parameter to be fitted, the i pmth of the variable vr , was originally defined symbolically, then it is preferable to fit the first parameter of the symbol. Thus, use

```

-----1-----2-----3-----4-----5-----6-----7-----8
L1      =          7.6
K1      =          0.25
Q1      MAG      L1      K1      1.
      SOLV      ...
      ...
      K1          1-.5      .5      .01
-----1-----2-----3-----4-----5-----6-----7-----8

```

rather than

```

-----1-----2-----3-----4-----5-----6-----7-----8
      Q1          2-.5      .5      .01
-----1-----2-----3-----4-----5-----6-----7-----8

```

since in the latter case the symbol $K1$ in the $Q1$ definition will be replaced with the numerical value obtained in the optimization.

The default MINUIT commands for the SOLV statement are

```

PRINTOUT    0.
MINIMIZE    1000.    1.0
END RETURN.

```

These defaults may be changed by the SMIN command.

SEE ALSO: SMIN

Appendix B, Sample SYNCH programs

* Some of the constraint lines, e.g. AXAY, generate more than one fitting constraint.

STOP - End of Job

The **STOP** command terminates a **SYNCH** job.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      STOP          stat
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

A **SYNCH** job consists of one run, which is initiated by a **RUN** command and terminated by a **STOP** command. The **STOP** command also marks the end of the job.

stat (CH) If the keyword **STAT** is present, a summary table of storage usage will be printed at the end of the output.

STOP closes various files while complying with any **KEEP** requests.

NOTE: In earlier versions of **SYNCH**, a **FIN** command terminated the run, allowing one to stack multiple runs in one job. This feature has been eliminated.

SEE ALSO: **RUN**

SUB - Define a SYNCH Subroutine

The **SUB** command marks the beginning of a **SYNCH** subroutine.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  SUB
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Name by which the subroutine is referenced.

A **SYNCH** subroutine is a set of commands that are included between **SUB** and **END** commands. These commands are not executed until **name** is invoked by a command such as **CALL**, **FITQ**, **MESH**, **SOLV**, **VPAR**. When the subroutine is invoked, the included commands are executed sequentially.

A **SYNCH** subroutine may be executed repeatedly within a **SYNCH** program when referenced by **CALL**, ... statements or by appearing in other commands. When the name of a subroutine is used as input to a command other than a **CALL**, the named subroutine is called during execution of that command. By including **INCR** commands and calling the subroutine repetitively, one may perform calculations over a range of variable values.

A **CALL** statement may be included in a **SYNCH** subroutine but a subroutine may not call itself. **CALL** statements may be deactivated but **SUB** and **END** statements should never be deactivated.

Any **SYNCH** subroutines except **RUN** and **STOP** may be included in a subroutine to be invoked by **CALL**, **MESH**, **VPAR**, but it is preferable to include in the **SYNCH** subroutine only commands that cause some calculation to be made. Commands that only store information, like

BML	EQU	MAT3	PAGE	PARA	REM	VEC
------------	------------	-------------	-------------	-------------	------------	------------

are best placed outside of **SYNCH** subroutines.

If the subroutine is to be invoked by any other commands, e.g. **FITQ**, **SOLV**, then only the following commands may be included:

**	=	CALC	CYC	DRF	INV	KICK
MAG	MAGV	MMM	MOVE	MXV	NPOL	REF
ROT	SHF	SHF7	SQL	SUM	TAB	TRKB
TRKE	TRKM					

SEE ALSO: **CALL**
END
INCR

SUM - Scalar Summation

The SUM command calculates the sum of a set of scalars.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name SUM      m      s1      s2      s3      ...      sm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Number of values to be summed.

si (FP) Numbers to be summed.

SEE ALSO: CALC

SXTP – Sextupole Definition

The **SXTP** command is used to define a sextupole magnet.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  SXTP      n length  d2b      brho      exac
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name.

n (IN) Defines normal or skew field.
 = 0 or blank, A normal sextupole field is used.
 $\neq 0$, A skew sextupole field is used and the matrix elements $R(2,6)$ and $R(4,6)$ are set to zero.

length (FP) The length of the magnet. The thin lens kick is applied at the mid-point of the magnet, $\text{length}/2$.

d2b (FP) The sextupole coefficient, $d2b = B''(0)$.
 The sextupole strength is defined in terms of d2b by

$$S = (\text{length})(d2b)/(\text{brho}) .$$

If the input specifies $\text{length} = 0$, $\text{length} = 1$ is used in this calculation.

brho (FP) The magnet rigidity of the beam.

exac (IN) Select thin-lens approximation or exact calculation.
 = 0 or blank, The sextupole magnet is approximated by a thin-lens located at the center of the magnet.
 $\neq 0$, Use an exact calculation using a method by B. Autin, CERN^[9] based on elliptic functions. The method is valid only if the particle trajectory lies entirely in the horizontal plane and if the length of the magnet is non-zero.

The **SXTP** command defines a sextupole magnet, usually in the thin-lens approximation. In the thin lens approximation the parameter **length** may take any non-negative value. If $\text{length} > 0$ the thin lens acts at the center of the magnet.

A more exact approximation^[9] is available when $\text{length} > 0$ and the particle trajectory is in the horizontal ($y = 0$) plane. It is invoked by setting **exac** $\neq 0$.

SEE ALSO: NPOL
 CYC

TRK – Track Particles Through Beamline

The **TRK** command tracks particles, one at a time, through a series of linear and nonlinear transformations which comprise a beamline.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  TRK      m      n namv bmln mtrx refv ntr  iof  isav imod ipr      npr
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Number of particles to track.

n (IN) Define interpretation of the state vectors. See the parameters **namv** and **refv**.
 = 0, $(x, dx, y, dy, -ds, dp/p)$ (m, rad, 1)
 = 1, $(x, dx, y, dy, -ds, dp/p)$ (mm, mrad, ‰)
 = 2, $(ex, \psi_x, ey, \psi_y, -ds, dp/p)$ (mm-mrad, deg, m, 1)

namv (CH) Name of **PVEC** statement defining initial state vectors of the **m** particles. These values are defined relative to the state vector of the reference particle (see **refv**).

bmln (CH) Name of the beamline (defined by a **BML** command) through which to track.

mtrx (CH) Optional. Name of matrix which gives beta functions at print positions. Some output is conditional on **mtrx** being defined:
 – If **mtrx** is given, the output includes the action variables e_x and e_y .
 – If both **mtrx** and **refv** are given, the output includes the action variables e_x and e_y and the phases ψ_x and ψ_y .

refv (CH) Optional. Name of **PVEC** statement specifying the initial state vector of the reference particle to be tracked through the beamline.

ntr (IN) Number of transits of **bmln** to be made by each particle.

iof (IN) Output frequency.
 = 00001, Write output at each position of the beamline each transit.
 = 0r00p, Write output at the p-th position every r-th transit.

TRK

- isav** (IN) Save option for tracked vectors.
= 0, Do not save the tracked vectors.
= 1, Save tracked vectors for subsequent use. The saved vectors overwrite the contents of **namv**. In subsequent **TRK** statements using the same **namv** one must set **n** = 0.
- imod** (IN) Tracking mode.
= 0 or 1, Track using stored matrices for the linear elements.
= 2, Track, recalculating the effect of each beam element on the ray rather than using stored matrices. Save the cumulative matrix representing linearized motion relative to the tracked ray.
= 5, Same as case 2, but do not save cumulative matrix.
- ipr** (IN) Print option.
= -1, Printing is suppressed.
= 0, Print full output listing.
= 2, Print only lines with elements tagged by symbol # or those defined by a **PCYC** command.
- npr** (IN) Print frequency. Print every **npr**-th position along beam line. The default is to print every line.

The **TRK** command tracks **m** particles through the beamline **bmln** **ntr** times. The beamline may contain non-linear elements.

WARNING: A blank line **MUST** be included after the **TRK** line of the input.

SEE ALSO: **FXPT**

TRKB – Track Betatron Functions

The **TRKB** command tracks the values of the betatron functions through a beamline. Initial values must be specified at the beginning of the beamline.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  TRKB    m    n bmln bet0 refv efmt pos1 pos2 itbl irad s0          th0
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m, n (IN) Allocate storage to save the table of betatron functions, for use by **SOLV**.
 $m = -1$, No storage is reserved; function table is not stored.
 $m = n = 0$, or blank, Save the entire function table.
 $0 < m < n$, Allocate storage to save the table for positions m through n . (See parameters $pos1$ and $pos2$.)

bmln (CH) Name of beamline, defined by a **BML** statement, through which to track betatron functions.

bet0 (CH) Source of initial values of the betatron function.
 = name of **IBET** statement. The initial values are defined by the **IBET** statement.
 = name of matrix. Name of a matrix previously defined, by, for example, an **MMM** statement, from which to extract initial values of the betatron functions.

refv (CH) Optional. Name of a **PVEC** statement which defines the initial transverse coordinates of the particle to be tracked through the beamline.

efmt (CH) Output format.
 = blank, Use the default F-format code in the output format specifications.
 = **EFORM**, Use the E-format code to allow display of values which overflow the default F-format in the standard output.

pos1 (IN) Begin tracking at position number $pos1$ of beamline **bmln**. Position 0 is the upstream end of the first element; position $n > 0$, the downstream end of the n -th element.
 = 0 or blank, Begin tracking at position 0, the start of the beamline.
 > 0 , Begin tracking at position $pos1$ in the beamline.

Note: If a value $pos1 > 0$ is given then a value $pos2 > 0$ must also be given.

TRKB

- pos2** (IN) End tracking at position number **pos2** of beamline **bmln**.
= 0 or blank, Track to the end of the beamline.
> 0, End tracking at position **pos2** in the beamline.
- itb1** (IN) Select option to get initial values from **TRKB** table.
= 0 or blank, The initial values of the betatron functions are determined by the **bet0** option.
= 1, Get the initial values from a previously calculated **TRKB** table for the beamline: On the first invocation of **TRKB** for the beamline, determine the initial values by the **bet0** option. Otherwise, take the initial values at position **pos1** from the **TRKB** table.
- To use this option, the **TRKB** statement must be included in a **SYNCH** subroutine. By changing the value of **pos1** by using the **REPL** command, one can track a beamline in successive segments.
- irad** (IN) Define units for phase advance.
= 0 or blank, Print the tune, $\nu = \psi/2\pi$.
= 1, Print value of ψ in radians.
- s0** (FP) Optional. Initial value of path length at position 0.
- th0** (FP) Optional. Initial value of horizontal angle at position 0.

TRKB propagates the initial values of the betatron functions through the specified beamline. No periodicity conditions are implied for the betatron functions. One may think of this as describing the evolution of a prescribed phase space ellipse through the beamline.

The values of the functions are printed at the beginning of the beamline and at the end of each element. They may also be saved for use by a **SOLV** command.

The table produced by **TRKB** lists the position sequence number, the element name, the accumulated path length, and, for both horizontal and vertical planes, the tune, β , α , η , and η' . In addition, if the parameter **refv** is given, the table is extended to include the coordinates of a particle vector that evolve from the initial values given by the **PVEC** statement. If there is no particle vector specified, the quantity $\mathcal{H} = \gamma_x \eta_x^2 + 2\alpha_x \eta_x \eta'_x + \beta_x \eta_x'^2$, the Courant-Snyder invariant [4] of the dispersion, is printed.

SEE ALSO: **SOLV**

TRKE - Track Beam Envelopes

The **TRKE** command calculates the beam envelopes defined by the betatron functions and the (transverse) beam emittances through a beamline.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  TRKE   m    n bmln bet0 refv eps  pos1 pos2 itbl iadd epxco      epyco
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m, n (IN) Allocate storage to save the table of betatron functions calculated, for use by **SOLV**.
m = -1, No storage is reserved; function table is not stored.
m = **n** = 0 or blank, Save the entire function table.
0 < **m** < **n**, Allocate storage to save the table for positions **m** through **n**. (See parameters **pos1** and **pos2**.)

bmln (CH) Name of beamline, defined by a **BML** statement, through which to track betatron functions.

bet0 (CH) Source of initial values of the betatron function.
= name of **IBET** statement. The initial values are defined by the **IBET** statement.
= name of matrix. Name of a matrix previously defined, by, for example, an **MMM** statement, from which to extract initial values of the betatron functions.

refv (CH) Optional. Name of a **PVEC** statement which defines the initial state vector of the particle to be tracked through the beamline.

eps (CH) Name of a preceding **BVAL** statement which defines the beam emittances.

pos1 (IN) Begin tracking at position number **pos1** of beamline **bmln**. Position 0 is the upstream end of the first element; position **n** > 0, the downstream end of the **n**-th element.
= 0 or blank, Begin tracking at position 0, the start of the beamline.
> 0, Begin tracking at position **pos1** in the beamline.

Note: If a value **pos1** > 0 is given then a value **pos2** > 0 must also be given.

TRKE

- pos2** (IN) End tracking at position number **pos2** of beamline **bmln**.
= 0 or blank, Track to the end of the beamline.
> 0, End tracking at position **pos2** in the beamline.
- itb1** (IN) Select option to get initial values from **TRKE** table.
= 0 or blank, The initial values of the betatron functions are determined by the **bet0** option.
= 1, Get the initial values from a previously calculated **TRKE** table for the beamline: On the first invocation of **TRKE** for the beamline, determine the initial values by the **bet0** option. Otherwise, take the initial values at position **pos1** from the **TRKE** table.
- To use this option, the **TRKE** statement must be included in a **SYNCH** subroutine. By changing the value of **pos1** by using the **REPL** command, one can track a beamline in successive segments.
- iadd** (IN) Defines how the contributions to the beam envelope from momentum spread, dp/p , and emittance, **epxco** and **epycp**, are combined.
= 0 or blank, Add the contributions in quadrature.
= 1, Add the contributions algebraically.
- epxco** (FP) Closed orbit equivalent horizontal emittance. (See **CYAE** command.)
- epycp** (FP) Closed orbit equivalent vertical emittance. (See **CYAE** command.)

TRKE, like **TRKB**, propagates the betatron functions through the specified beamline. The betatron functions and beam emittances are then used to calculate the beam sizes $\sigma_x = \sqrt{\beta_x \epsilon_x}$ and $\sigma_y = \sqrt{\beta_y \epsilon_y}$, which are displayed in the printed output.

The parameters **epxco** and **epycp** are equivalent emittances for an ensemble of machines which allow the contribution to the beam envelope size due to misalignments to be estimated. See the command **CYAE** for discussion.

SEE ALSO: **CYAE**
TRKB

TRKM - Track Through Non-linear Elements

The **TRKM** command tracks particles or beam envelopes through a beamline containing elements defined by **MAP** or **DEQ** statements.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  TRKM   m   n bmln bet0 map  par  pos1 pos2      irad step
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Reference name.

m, n (IN) Allocate storage to save the table of betatron functions calculated, for use by **SOLV**.
 $m = -1$, No storage is reserved; function table is not stored.
 $m = n = 0$ or blank, Save the entire function table.
 $0 < m < n$, Allocate storage to save the table for positions m through n . (See parameters **pos1** and **pos2**.)

bmln (CH) Name of beamline, defined by a **BML** statement.

bet0 (CH) Source of initial values.
 = name of **IBET** statement. The initial values are defined by the **IBET** statement.

map (CH) Name of **MAPk** or **DEQk** statement that is referenced in **bmln**.

par (CH) Name of a **VEC** statement that contains data to be passed to the internal subroutine that executes the transformation.

pos1 (IN) Begin tracking at position number **pos1** of beamline **bmln**. Position 0 is the upstream end of the first element; position $n > 0$, the downstream end of the n -th element.
 = 0 or blank, Begin tracking at position 0, the start of the beamline.
 > 0 , Begin tracking at position **pos1** in the beamline.

Note: If a value **pos1** > 0 is given then a value **pos2** > 0 must also be given.

TRKM

- pos2** (IN) End tracking at position number **pos2** of beamline **bmln**.
= 0 or blank, Track to the end of the beamline.
> 0, End tracking at position **pos2** in the beamline.
- irad** (IN) Define units for phase advance.
= 0 or blank, Print the tune, $\nu = \psi/2\pi$, where ψ is the phase advance in radians.
= 1, Print value of ψ .
- step** (FP) **DEQk** commands (e.g., the built-in example describing space charge effects) invoke a differential equation integrator. Parameter **step** is the path length step size passed to the integrator.

The **TRKM** command is used to track a particle through a beamline all of whose elements are defined by the same **MAPk** or **DEQk** subroutine. **TRKM** can be used in conjunction with **SOLV** to fit associated parameters to desired values.

One use of **TRKM** with **SOLV** is to design a quadrupole beamline to focus a beam influenced by space charge. The **DEQ4** command, which integrates the Kapchinsky-Vladimirsky envelope equations, is used. The particle coordinates in this case represent the quantities $(\sigma_x, \sigma'_x, \sigma_y, \sigma'_y)$ of the beam envelope.

SEE ALSO: Section 7.2, DEQ-Differential Equation Transformations

UPDAT - Create New Input File

The **UPDAT** command creates a new **SYNCH** input file by copying the original one and updating it with values resulting from the current run.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      UPDAT  m      lmb1 lmb2 lmb3 ... lmbm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) Select option for updating **PARA** statements.
 = 0 or blank, All **PARA** instructions are updated by replacing the initial parameter values by the latest values.
 > 0, Number of groups of **PARA** statements, delimited by named **SELCT** statements, to be updated.

lmb1, ... (CH) Names of the **SELCT** commands which delimit groups of **PARA** statements.

Parameter values specified by **PARA** commands in the input file may be updated, either selectively or in total, by the results of the current run. The new file may be used as input for a subsequent **SYNCH** run.

SEE ALSO: **SELCT**
PARA

3 of 3

VAR – Define Variable by Current Value in Another Statement

The **VAR** command assigns to a named variable the current value of a specified parameter in a particular instance of a statement.

```

-----1-----2-----3-----4-----5-----6-----7-----8
name  VAR      m      stmt
-----1-----2-----3-----4-----5-----6-----7-----8

```

name (CH) Name of the variable to which the extracted value is to be assigned.

m (IN) Parameter position within the named statement from which the value is to be extracted.

stmt (CH) Reference name of the statement from which the parameter value is to be extracted.

The **VAR** command provides another means to copy the value of a particular parameter from one statement to another.

Example 1–

```

-----1-----2-----3-----4-----5-----6-----7-----8
QF   MAG          1.5      0.06      1.
GQF  VAR      2      QF
QFH  MAG          0.75      GQF      1.
-----1-----2-----3-----4-----5-----6-----7-----8

```

In this example, the value of the gradient, parameter 2, of magnet **QF** is extracted to define the value assigned to **GQF** which is then used to define **QFH**, a half length version of **QF**.

VAR

Example 2-

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
L      DRF              2.
SR      SUB
QF      MAG              1.          0.05          1.
QD      MAG              1.          -.05           1.
C      MMM              QF   L      QD   L
      END
      FITQ              SR   C      QF   QD          2   2   .25          .25
GF      VAR      2      QF
GD      VAR      2      QD
QFH     MAG              .75          GF           1.
QDH     MAG              .75          GD           1.
      CYC              -1 QFH   L      QDH
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

The values of the gradients of QF and QD are not in fact the original values ± 0.05 indicated but rather the values that result from the FITQ. The use of the VAR statement ensures that the adjusted values are transferred to the half-length magnets QFH and QDH.

VEC - Vector Definition

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  VEC      m      n  v11      v12      v13      ...      v1m
                        v21      v22      v23      ...      v2m
                        .
                        .
                        .
                        vn1      vn2      vn3      ...      vnm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Dimension of the vectors being defined.

n (IN) Number of independent vectors being defined. Each vector must begin on a new line.
 = blank, 0, or 1, Define one **m**-component vector.
 > 1, Define **n** **m**-component vectors.

vi_j (FP) Values of the vector components.

The **VEC** command is used to define one or more vectors of dimension **m**. The vectors are arranged consecutively in storage. To refer to the *i*-th component of the *k*-th vector one must refer to the $(m * (k - 1) + i)$ -th parameter of the **VEC** command.

SEE ALSO: **PVEC**

VPAR – Loop Through SYNCH Subroutine on a Diagonal

The **VPAR** command repeatedly invokes a **SYNCH** subroutine while simultaneously incrementing several variables.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  VPAR      m      sbr
           k1      a1      min1      max1      inc1
           k2      a2      min2      max2      inc2
           .
           .
           .
           km      am      minm      maxm      incm
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

m (IN) Number of variables to be varied.

sbr (CH) Name of **SYNCH** subroutine containing elements **a1**, ..., **am**.

ki (IN) Refers to the **ki-th** floating point variable in definition of element **ai**.

ai (CH) Name of element containing the variable.

mini (FP) Minimum value of variable **ki** of element **ai**.

maxi (FP) Maximum value of variable **ki** of element **ai**.

inci (FP) Increment of the variable **ki** of element **ai**.

The **VPAR** command implements the logical structure of a single FORTRAN DO-loop. The subroutine **sbr** is executed repeatedly while the control variables **ki** of elements **ai** are incremented by **inci**. All **m** variables are incremented for each invocation of **sbr**. Compare this command to **MESH** wherein the variables are incremented one by one as in a nested DO-loop structure.

The number of steps required to satisfy the range is calculated for each variable and the largest value determines the number of steps actually calculated. If the intervals are not integer multiples of the increments some truncation of the range may occur.

VPAR

Example:

```
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
SR      SUB
QF      MAG          2.0          0.5          1.0
QD      MAG          2.0          -.05          1.0
.C      BML          QF    0      QD    QD    0      QF
C       CYC          .C
      END
      VPAR    2      SR
              2      QF          0.5          1.0          0.1
              2      QD          -1.0          -0.5          0.1
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
```

The above code will produce a **CYC** print-out of the cell **C** for the following combinations of the gradients of the magnets **QF** and **QD**: (0.5, -1.0), (0.6, -0.9), (0.7, -0.8), (0.8, -0.7), (0.9, -0.6), and (1.0, -0.5).

SEE ALSO: MESH

WBE - Write Betatron Functions of Matrices

The **WBE** command writes out the betatron functions corresponding to a list of transfer matrices.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      WBE          n A1    A2    A3    ...    Am
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

n Print option selector. The betatron functions β , α , $\mu/2\pi$, D , D' , and W (defined below) for each plane are printed.
 = 0, Use $W = \sqrt{\beta}$. If $|\cos \mu| > 1$, all the functions are set to 0.
 = 1, Use $W = \text{Tr } A/2$. The functions are not set to 0 when $|\cos \mu| > 1$.

A1, ... (CH) Names of the matrices from which the betatron functions are determined.

The **WBE** command writes out the betatron functions for the transfer matrices labelled A1, A2, A3, ..., Am. Each transfer matrix can be represented as

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} \cos \mu + \alpha \sin \mu & \beta \sin \mu & A_{13} \\ -[(1 + \alpha^2)/\beta] \sin \mu & \cos \mu - \alpha \sin \mu & A_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

in terms of the betatron functions and from this the function values can be determined. The other functions are

$$W = \begin{cases} \sqrt{\beta}, & \text{if } n = 0 \\ \text{Tr } A/2 = (A_{11} + A_{22})/2, & \text{if } n = 1 \end{cases}$$

and

$$D = \frac{1}{2}[A_{13} + (A^{-1})_{13}]/[1 - \cos \mu],$$

$$D' = \frac{1}{2}[A_{23} + (A^{-1})_{23}]/[1 - \cos \mu].$$

Here, (A^{-1}) is the inverse of A .

The betatron phase advance is determined from $\cos \mu = \text{Tr } A/2 = (A_{11} + A_{22})/2$, where

$$\mu = \begin{cases} \arccos(\text{Tr } A/2), & \text{if } |\text{Tr } A| \leq 2; \\ \text{arccosh}(\text{Tr } A/2), & \text{if } |\text{Tr } A| > 2. \end{cases}$$

WFL - Print Out Internal Storage

The **WFL** command is used to print details of the internal storage used by **SYNCH**.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
      WFL      m      n A1      A2      A3      ...      Am
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

m (IN) Select time/condition for execution.
 = 0 or blank, Execute only on error.
 = 1, Execute immediately.

n (IN) Select information to be printed.
 = 0 or blank, Print portions of arrays **CINFF** and **INFF** relating to instructions named in this command. These arrays contain the command name, the reference name, the storage locations of input integer, floating point and character data, how many of each, and the value of each datum.
 = 1, Print as for $n = 0$ plus storage allocated for the instructions. The storage contains the matrices and/or whatever calculated output has been stored for the command.
 = 2, Print as for $n = 1$ plus the contents of the working scratch storage area.

A1, ... (CH) Names of commands for which the internal storage data are required. If no names are listed, data for all instructions encountered prior to the **WFL** command are printed, including that of the internally defined matrices.

(

(

8:

WMA – Write Matrices

```

-----1-----2-----3-----4-----5-----6-----7-----8
      WMA          A1   A2   A3   ...   Am
-----1-----2-----3-----4-----5-----6-----7-----8

```

A1, ... (CH) Names of previously defined transfer matrices.

The **WMA** command writes out the previously defined pairs of transfer matrices labelled A1, A2, A3, ..., Am. If a **SIZE** command is in effect when **WMA** is invoked, the matrices will be printed accordingly. If no **SIZE** command has been issued, the matrices will be printed in 3×3 format if all the matrices in the list are represented internally as 3×3 structures. If any one of them is represented internally as a 7×7 structure, all the matrices in the list are printed in 7×7 format.

SEE ALSO: **SIZE**

= - Equate to Floating Point Number

The = command is used to define a scalar variable by equating it to a floating point numeric value, to another scalar variable, or to the result of an arithmetic operation on two such data.

Use 1:

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  =                float
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name of the variable being defined or replaced.

float (FP) A floating point number or the name of a previously defined scalar variable.

The variable **name** is given the value **float**. The variable **name** may be substituted for a floating point variable in any **SYNCH** input statement and the value **float** will be used.

Use 2: Arithmetic Operations

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  =                oprnd1  operator  oprnd2
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Reference name.

oprnd1, oprnd2 (FP) The operands. Floating point numbers, or names of previously defined scalar variables.

operator (CH) Operator identifying the operation to be performed. The allowed operators are +, -, *, /, or ** indicating the operations of addition, subtraction, multiplication, division, and exponentiation, respectively.

The = command assigns to **name** the value resulting from performing the designated operation on the operands. Operands can be either numerical values or the names of other variables.

SEE ALSO: **PARA**
CALC

**** - Raise a Matrix to a Power**

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
name  **      k      mtrx
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

name (CH) Name of the matrix which is created by the operation.

k (IN) Power to which the matrix **mtrx** is to be raised.

mtrx (CH) Name of the matrix to be raised to the **k**-th power.

SEE ALSO: MMM

Period (.) - Comment

A period (.) in column 1 of any line identifies the line as a comment.

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
.  ----  ---  text  ---  ----
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

A period (.) in column 1 of a line causes any text entered in columns 2 through 80 to be treated as a comment. See the C command for more details.

SEE ALSO: C
 REM
 P
 PAGE

6.0 MISCELLANEOUS FEATURES

6.1 The Negative of a Symbolic Floating Point Number

If a Symbolic Floating Point variable, V , has been previously defined, the negative of that variable can be used in a SYNCH statement by denoting it by $-V$.

Use of this feature requires that the variable name contain not more than 4 characters.

Example:

```

-----1-----2-----3-----4-----5-----6-----7-----8
grd  =          10.0
QF   MAG          len      grd      brho      b0      $
QD   MAG          len      -grd     brho      b0      `
-----1-----2-----3-----4-----5-----6-----7-----8

```

Here, magnet QF will have gradient 10.0 and magnet QD will have gradient -10.0 .

6.2 Symbolic Entry for Inverse of a Matrix

If a matrix has been previously defined, its inverse may be used in a computation by preceding its name with a slash (/).

Use of this feature requires that the variable name contain not more than 4 characters.

Example:

```

-----1-----2-----3-----4-----5-----6-----7-----8
A   MAG          .9      280.      13373.    0.
B   MAG          .9      -280.      13373.    0.
P   MMM          A      /B
-----1-----2-----3-----4-----5-----6-----7-----8

```

Here, the matrix multiplication will be $P = \text{inv}(B) \times A$.

6.3 Internally Defined Matrices

a) Unit matrix, (1):

$$(1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Symplectic matrix, (S):

$$(S) = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The parentheses are part of the name and must be included when referring to these matrices.

6.4 Predefined Constants

The following constants are defined in **SYNCH** and may be referenced by name in the user's data sets.

PI	= 3.141592653589793	PI02	= PI/2
PI2	= 2 × PI	RADEG	= PI/180
TPI	= 2 × PI	CFAC	= 0.299792458

6.5 Initially Deactivated Statements

A **SYNCH** statement can be initially placed in the deactivated state (see **DEACT**) by having a dash (–) in column 1 of the statement line. The statement will not be executed unless it has been activated by an **ACT** statement (see **ACT**). Execution of **SYNCH** subroutines can be bypassed until needed by placing a dash in front of the **CALL** statement.

7.0 NON-LINEAR TRANSFORMATIONS

7.1 MAP-Point Transformations

The MAP subroutines all have the form $\text{MAP}k(V, PAR)$ where:

k is an integer, $0 \leq k \leq 9$ for built-in routines, $10 \leq k \leq 19$ for users.

V is a subscripted variable of dimension 7 which contains elements of a particle state vector $(x, x', y, y', -ds, dp/p, 1)$ on which transformations will be made. At the end of the routine, the transformed values will replace the original ones in the array V .

PAR is an array containing parameters needed for the transformation. The dimension depends on the transformation. The input variable m in the MAP statement corresponds to the dimension of PAR .

As an example, consider the following non-linear transformation, which is included in the code as MAP0:

$$\begin{aligned} X &= x \\ X' &= x' - (3Ax^2 + 2Bxy + Cy^2) \\ Y &= y \\ Y' &= y' - (Bx^2 + 2Cxy + 3Dy^2) \\ -dS &= -ds \\ dP/P &= dp/p \\ 1 &= 1 \end{aligned}$$

where A , B , C , and D are four parameters associated with the transformation. The subroutine that accomplishes the above transformation follows:

```
SUBROUTINE MAPO(V,PAR)
  DIMENSION V(7),W(7),PAR(4)
C   Compute the transformed values...
  W(1) = V(1)
  W(2) = V(2) - (3.*PAR(1)*V(1)*V(1) + 2.*PAR(2)*V(1)*V(3)
1      + PAR(3)*V(3)*V(3) )
  W(3) = V(3)
  W(4) = V(4) - (PAR(2)*V(1)*V(1) + 2.*PAR(3)*V(1)*V(3)
1      + 3.*PAR(4)*V(3)*V(3) )
  W(5) = V(5)
  W(6) = V(6)
  W(7) = V(7)
```

```

C   Put transformed values in V array...
    DO 2 L=1,7
2   V(L) = W(L)
    RETURN
    END

```

Having compiled and linked this routine, one includes line(s) such as the following in the SYNCH input file:

```

XMPL  MAP0  4      A          B          C          D

```

with appropriate values for A, B, C, and D, as well as beamline statements that reference XMPL. Initial values of the V components are given in the arguments of the TRK or other command used.

7.2 DEQ-Differential Equation Transformations

The DEQ subroutines all have the form $DEQk(S, W, DW)$ where:

k is an integer, $0 < k < 10$ for built-in routines, $10 < k < 20$ for users.

S is the cumulative path length.

W is an array giving the state vectors of a set of particles.

DW is an array giving increments to the particle state vectors W in one integration step.

An example is the following differential equation routine, which is included in the code as DEQ4. It is used to integrate envelopes of beams with space charge, as well as of single particles within such beams.

```

SUBROUTINE DEQ4(S,W,DW)

  IMPLICIT DOUBLE PRECISION (A-H,O-Z)
  COMMON/CDERIV/IFLAG,MR,NR,DP,NV(11),DAT(20),UZ,STOT,
1   DAT2(7),MP2FLG,NF,HO,LOCALF,NUMALF,NPR
  EQUIVALENCE (L,DAT(1)),(GRAD,DAT(2)),(BRHO,DAT(3)),
1   (EPS,DAT(4)),(Q,DAT(5))
  DIMENSION W(4,1),DW(4,1)
  DOUBLE PRECISION L,K,KX,KY,KBX,KBY

  IF (IFLAG) 1,1,2
1  BRHO = BRHO*(1+DP)
    K = GRAD/BRHO
    EPS2 = EPS*EPS
    RETURN

```

C ENVELOPE EQUATIONS

```
2  AX=W(1,1)
   AY=W(3,1)
   AX3=AX*AX*AX
   AY3=AY*AY*AY
   A=AX+AY
   EAX3=EPS2/AX3
   EAY3=EPS2/AY3
   QA=Q/A

   DW(1,1) = W(2,1)
   DW(3,1) = W(4,1)
   DW(2,1) = -K*AX + EAX3 + QA
   DW(4,1) = K*AY + EAY3 + QA

   GO TO (7,3,3,5), IFLAG
```

C LINEARIZED ENVELOPE EQUATIONS

```
3  EAX4=3*EAX3/AX
   EAY4=3*EAY3/AY
   QA2=QA/A
   KBX=-K-EAX4-QA2
   KBY= K-EAY4-QA2

   DO 4 J=2,NR
     DW(1,J) = W(2,J)
     DW(3,J) = W(4,J)
     DW(2,J) = KBX*W(1,J) - QA2*W(3,J)
4    DW(4,J) = KBY*W(3,J) - QA2*W(1,J)

7  RETURN
```


C SINGLE PARTICLE EQUATIONS

```

5   QAX=QA/AX
    QAY=QA/AY
    KX=-K+QAX
    KY= K+QAY

    DO 6 J=2,NR
      DW(1,J) = W(2,J)
      DW(3,J) = W(4,J)
      DW(2,J) = KX*W(1,J)
6   DW(4,J) = KY*W(3,J)

    RETURN
    END

```

Note that the first 'particle' in this application is the beam envelope. To use this routine, one includes line(s) such as the following in the SYNCH input file:

```

-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
QSPCH DEQ4   6      L           G           BR           EPS           Q           DS
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8

```

where QSPCH is the name of a quadrupole or drift, L and G are its length and gradient, BR is the rigidity of the particles, EPS the beam emittance, Q a factor proportional to the beam current and DS is the integration step size. The quantities L and DS are passed to the differential equation routine, and the other parameters to DEQ4 through the array DAT.

8.0 MATHEMATICAL FORMULATION

8.1 Transfer Matrices and Beamlines

The coordinate system employed by the program SYNCH is a curvilinear system of right-handed orthogonal coordinates $x - y - s$ where s is the longitudinal position along the reference orbit, x is the horizontal (radial) displacement from the reference orbit, and y is the vertical displacement from the reference orbit. (The reference orbit itself follows an isomagnetic line through each element.) The trajectory, or state of a particle in this coordinate system is represented by the vectors $X = (x, x', dp/p)$, and $Y = (y, y', dp/p)$, where $x' = dx/ds$, $y' = dy/ds$ and dp/p is the momentum error with respect to the reference particle. Most accelerator or beamline elements can be represented by a pair of 3×3 transfer matrices R_x and R_y which act on the above particle state vectors:

$$\begin{aligned} X_{\text{out}} &= R_x X_{\text{in}} \\ Y_{\text{out}} &= R_y Y_{\text{in}} \end{aligned}$$

where the matrices are of the form

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$\det(R) = 1.$$

For some types of calculations one may wish to investigate coupling between horizontal and vertical motion. For these cases, a 7×7 matrix is used to describe each beamline element. This matrix, M , acts on the particle state vector $X = (x, x', y, y', -ds, dp/p, 1)$:

$$X_{\text{out}} = M X_{\text{in}}.$$

The matrix M for a non-rotated magnet, for example, would be of the form

$$M = \begin{bmatrix} (R_x)_{11} & (R_x)_{12} & 0 & 0 & 0 & (R_x)_{13} & 0 \\ (R_x)_{21} & (R_x)_{22} & 0 & 0 & 0 & (R_x)_{23} & 0 \\ 0 & 0 & (R_y)_{11} & (R_y)_{12} & 0 & (R_y)_{13} & 0 \\ 0 & 0 & (R_y)_{21} & (R_y)_{22} & 0 & (R_y)_{23} & 0 \\ M_{51} & M_{52} & M_{53} & M_{54} & 1 & M_{56} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

(A 7×7 matrix is used in order to facilitate magnet misalignment calculations; see Section 8.8.) Bending magnets defined by **MAG** are assumed to bend the orbit in the horizontal plane. The matrix for a vertical bending magnet is defined by the **MAGV** command should be used to in effect perform 90° rotations of coordinates before entering and upon exiting the magnet. (See below.)

For simplicity, the 3×3 matrices for the various types of elements will be introduced under the assumption that the appropriate 7×7 matrices can be derived if necessary.

Beamlines

A beamline is defined as a particular sequence of accelerator elements. These elements can include primitive ones such as drifts and magnets, compound elements represented by a single matrix, or other beamlines. Beamlines are defined in **SYNCH** by **BML** statements; compound elements, by **MMM** statements.

8.2 Linear Elements

Drift Region—length ℓ

$$R_x = \begin{bmatrix} 1 & \ell & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} 1 & \ell & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Magnets

$$B' = (dB_y/dR) \quad \text{at } R_0$$

$$B_0 = (B_y) \quad \text{at } R_0$$

$$k = B'/B_0$$

where R_0 is the reference orbit and the radius of curvature, $\rho = (B\rho)/B_0$, the magnetic rigidity divided by the magnetic field of the bending magnet. (See Figure 1.) Then

$$k_x = k + 1/\rho \quad k_y = -k$$

$$K_x = |k_x/\rho| \quad K_y = |k_y/\rho|$$

$$\Phi_x = \sqrt{K_x} \ell \quad \Phi_y = \sqrt{K_y} \ell$$

$$\theta = \ell/\rho$$

Figure 1. Geometry Through Bending Magnet.

It is useful to separate the description of motion through a magnet into matrices describing the body field of the magnet and the edge effects. The transfer matrices R_x and R_y for motion through a magnet are then given by

$$R_x = E_{2x} T_x E_{1x}$$

$$R_y = E_{2y} T_y E_{1y}$$

where the T -matrices describe the body of the magnet and the E -matrices, the edge-focusing effects. The T -matrices for various magnets and the E -matrices as calculated by SYNCH are shown below.

Horizontally (Radially) Focusing Combined-Function Magnet:

$$T_x = \begin{bmatrix} \cos \Phi_x & \ell \frac{\sin \Phi_x}{\Phi_x} & \ell \theta \frac{1 - \cos \Phi_x}{\Phi_x^2} \\ -\frac{\Phi_x \sin \Phi_x}{\ell} & \cos \Phi_x & \theta \frac{\sin \Phi_x}{\Phi_x} \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_y = \begin{bmatrix} \cosh \Phi_y & \ell \frac{\sinh \Phi_y}{\Phi_y} & 0 \\ \frac{\Phi_y \sinh \Phi_y}{\ell} & \cosh \Phi_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Vertically Focusing Combined-Function Magnet:

$$T_x = \begin{bmatrix} \cosh \Phi_x & \ell \frac{\sinh \Phi_x}{\Phi_x} & \ell \theta \frac{\cosh \Phi_x - 1}{\Phi_x^2} \\ \frac{\Phi_x \sinh \Phi_x}{\ell} & \cosh \Phi_x & \theta \frac{\sinh \Phi_x}{\Phi_x} \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_y = \begin{bmatrix} \cos \Phi_y & \ell \frac{\sin \Phi_y}{\Phi_y} & 0 \\ -\frac{\Phi_y \sin \Phi_y}{\ell} & \cos \Phi_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

two-quadrant
Horizontally Bending ~~Wedge~~ Magnet ($K = 0$):

gradient

$$T_x = \begin{bmatrix} \cos \theta & \ell \frac{\sin \theta}{\theta} & \ell \frac{1 - \cos \theta}{\theta} \\ -\frac{\theta \sin \theta}{\ell} & \cos \theta & \sin \theta \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_y = \begin{bmatrix} 1 & \ell & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

For wedge magnets the entrance and exit angles are zero so the E -matrices are the identity and $R_x = T_x$; $R_y = T_y$.

Rectangular Bending Magnets:

The transfer matrices for a rectangular bending magnet are constructed using the T -matrices for a horizontally bending wedge magnet and edge matrices corresponding to $e_1 = e_2 = \theta/2$. See Figure 1.

Horizontally (radially) Focusing Quadrupole Magnet:

$$T_x = \begin{bmatrix} \cos \Phi_x & \frac{1}{\sqrt{K_x}} \sin \Phi_x & 0 \\ -\sqrt{K_x} \sin \Phi_x & \cos \Phi_x & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_y = \begin{bmatrix} \cosh \Phi_y & \frac{1}{\sqrt{K_y}} \sinh \Phi_y & 0 \\ \sqrt{K_y} \sinh \Phi_y & \cosh \Phi_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Vertically Focusing Quadrupole Magnet:

$$T_x = \begin{bmatrix} \cosh \Phi_x & \frac{1}{\sqrt{K_x}} \sinh \Phi_x & 0 \\ \sqrt{K_x} \sinh \Phi_x & \cosh \Phi_x & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_y = \begin{bmatrix} \cos \Phi_y & \frac{1}{\sqrt{K_y}} \sin \Phi_y & 0 \\ -\sqrt{K_y} \sin \Phi_y & \cos \Phi_y & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Edge Focusing Matrices:

$$E_1 = \begin{bmatrix} 1 & 0 & 0 \\ \pm \frac{1}{\rho} \tan e_1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$E_2 = \begin{bmatrix} 1 & 0 & 0 \\ \pm \frac{1}{\rho} \tan e_2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where + is used for x , - is used for y .

Rotated Magnet:

The matrix for a rotated magnet is obtained by altering the coordinate system through the magnet via a rotational transformation. To perform this calculation, 7×7 matrices are used. In the 7×7 mode, a rotation of the coordinates of θ radians about the s -axis is represented by

$$R_\theta = \begin{bmatrix} \cos \theta & 0 & \sin \theta & 0 & 0 & 0 & 0 \\ 0 & \cos \theta & 0 & \sin \theta & 0 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 & 0 & 0 & 0 \\ 0 & -\sin \theta & 0 & \cos \theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

If R_θ represents a rotation θ and $R_{-\theta}$ represents a rotation $-\theta$ then the transfer matrix of a rotated magnet is given by $M = R_{-\theta} M_0 R_\theta$, where M_0 is the original magnet matrix. Thus, a vertically bending magnet, for example, would be represented by the matrix

$$M_v = R_{-\pi/2} M_0 R_{\pi/2}.$$

Any previously defined beamline element (including those defined by **MMM** and **REF** commands) may be rotated in this fashion by using the **ROT** command. A rotation matrix R_θ may be defined using **ROTZ**.

Kicker Magnet:

The effect of the **KICK** command is to introduce a delta-function kick in the center of a previously defined zero-length drift region, or produce a field error in a previously defined magnet. If the element was defined as a zero-length drift, then the specified slope (x' or y') will be changed by the amount θ . The corresponding matrix (shown here for vertical deflection) would be

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\theta & \theta \\ 0 & 0 & \theta & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

If the original element was a magnet, then the **KICK** statement will alter the transfer matrix to represent a magnet with a field error of dB/B . This is accomplished by changing the matrix elements $R_{17} = 0$ to $R_{17} = (\ell^2/2\rho)(dB/B)$ and $R_{27} = 0$ to $R_{27} = (\ell/\rho)(dB/B)$.

On the other hand, if the element was previously defined as a drift region of length ℓ , then the **KICK** statement turns the drift into a bending (zero-gradient dipole) magnet, neglecting any edge focusing. The transfer matrix for this new element is

$$R = \begin{bmatrix} 1 & \ell & 0 & 0 & 0 & \ell\theta/2 & -\ell\theta/2 \\ 0 & 1 & 0 & 0 & 0 & \theta & -\theta \\ 0 & 0 & 1 & \ell & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -\theta & -\ell\theta/2 & 0 & 0 & 1 & \ell\theta^2/3 & -\ell\theta^2/6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(horizontal kick)}$$

or

$$R = \begin{bmatrix} 1 & \ell & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \ell & 0 & -\ell\theta/2 & \ell\theta/2 \\ 0 & 0 & 0 & 1 & 0 & -\theta & \theta \\ 0 & 0 & \theta & \ell\theta/2 & 1 & \ell\theta^2/3 & -\ell\theta^2/6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{vertical kick}).$$

8.3 Multipole Magnets

Sextupole Magnet:

The effect of a sextupole on the reference ray V is to replace x' and y' with $x' - P$ and $y' + Q$, where

$$V = (x, x', y, y', ds, dp/p, 1)$$

and

$$P = \bar{S}(x^2 - y^2)/2$$

$$Q = \bar{S}xy$$

$$\bar{S} = S/(1 + dp/p)$$

$$S = \ell B''/B\rho.$$

The effect of the sextupole on rays near the reference ray, V , is given by the following 7×7 matrix:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\bar{S}x & 1 & \bar{S}y & 0 & 0 & P & P \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \bar{S}y & 0 & \bar{S}x & 1 & 0 & -Q & -Q \\ -P & 0 & Q & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

N-Pole Magnet:

The effect of a thin-lens multipole kick of m -th order can be accommodated by **SYNCH** using the **NPOL** statement. The user enters the order of the multipole, m , the effective length of the element, L , and its Taylor series expansion coefficient, b_n , where $m = n - 1$. The magnetic field is then given by

$$B = B_y + iB_x = \begin{pmatrix} 1 \\ i \end{pmatrix} \frac{b_n}{n!} z^n, \quad \text{for } \begin{pmatrix} \text{normal} \\ \text{skew} \end{pmatrix} \text{ fields,}$$

where $z = x + iy$. A particle being tracked through such an element is then subject to this field over the distance ℓ .

Solenoid Magnet:

A solenoid magnet has the following transfer matrix:

$$R = \begin{bmatrix} c^2 & 2\rho sc & sc & 2\rho s^2 & 0 & 0 & 0 \\ -sc/2\rho & c^2 & -s^2/2\rho & sc & 0 & 0 & 0 \\ -sc & -2\rho s^2 & c^2 & 2\rho sc & 0 & 0 & 0 \\ s^2/2\rho & -sc & -sc/2\rho & c^2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

where, with

B = solenoid field, l = its length, $B\rho$ = particle rigidity,

$$\rho = B\rho/B$$

$$\theta = l/\rho$$

$$c = \cos \theta/2$$

$$s = \sin \theta/2$$

Note that ρ is the radius of curvature in a dipole field B .

8.4 Betatron Functions

Definition

Consider a beamline representing an accelerator (or a part of one) made up of p identical periodic sections. Each section is composed of N elements, each represented by a 2×2 transfer matrix, $R(i)$. The matrix M_0 corresponding to one passage through a period (neglecting coupling) is given by

$$M_0 = R(N) R(N-1) R(N-2) \dots R(2) R(1).$$

Here, the matrices are taken to act on the vectors (x, x') or (y, y') . These matrices are submatrices of the corresponding 3×3 matrices used earlier in this section. The elements of these submatrices may be parameterized as follows:

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} \cos \mu + \alpha \sin \mu & \beta \sin \mu \\ -\gamma \sin \mu & \cos \mu - \alpha \sin \mu \end{pmatrix},$$

where α , β , $\gamma = (1 + \alpha^2)/\beta$, and μ are the betatron functions as defined by Courant and Snyder.^[4] Thus, the values of the betatron functions at point 0 of the accelerator (i.e., at the entrance to element 1, exit of element N) are given by

$$\begin{aligned}\mu &= \arccos[(a + d)/2] \\ \beta &= b/\sin \mu \\ \alpha &= (a - d)/(2 \sin \mu) .\end{aligned}$$

The parameter μ represents the phase advance of the betatron oscillation for one passage through the period and can be denoted as $\mu = 2\pi\nu/p$, where p is the number of periods in the machine and ν is the tune of the accelerator. The amplitude of the betatron oscillation is proportional to $\sqrt{\beta}$. The slope of the function β is given by $d\beta/ds = -2\alpha$. The betatron oscillations which a particle undergoes while traversing the accelerator may be expressed as

$$\begin{aligned}x(s) &= A_x \sqrt{\beta_x(s)} \cos[\psi_x(s) + \phi_x] \\ y(s) &= A_y \sqrt{\beta_y(s)} \cos[\psi_y(s) + \phi_y]\end{aligned}$$

where A_x , A_y , ϕ_x , and ϕ_y are arbitrary constants, and $\psi_x(s)$, $\psi_y(s)$ are the horizontal and vertical phase advances of the betatron oscillations from the arbitrary $s = 0$ point in the accelerator.

To calculate the values of α , β , and ψ for all points in the machine, SYNCH computes the matrices representing a single pass through a period starting at the end of each element. Once the single-pass transfer matrix, M_0 , for point 0 has been found, the values of $\beta(i)$ and $\alpha(i)$ at the end of the i -th element can be found from the matrix

$$M_i = R(i) M_{i-1} R^{-1}(i) , \quad i = 1, 2, \dots, N .$$

Here, $R(i)$ is the transfer matrix for element i . The phase advance through the i -th element is given by

$$\mu(i) = \arctan\{R_{12}(i)/[\beta(i-1)R_{11}(i) - \alpha(i-1)R_{12}(i)]\}$$

and the phase advance from point 0 to the end of the i -th element is given by

$$\psi(i) = \sum_{k=1}^i \mu(k) .$$

The tunes of the accelerator are given by $\nu_x = p\psi_x/2\pi$, $\nu_y = p\psi_y/2\pi$ where ψ_x and ψ_y are the phase advances through a full period.

The third row and column of each 3×3 transfer matrix is used to compute the closed orbit for an off-momentum particle. If M is the matrix for one complete revolution (i.e., through p periods), then the particle state vector $X_{eq} = (x, x', dp/p)$ representing a closed orbit must satisfy $X_{eq} = M X_{eq}$, from which

$$X_{eq} = \begin{pmatrix} (M_{13} + M_{13}^{-1})/(2 - M_{11} - M_{22}) \\ (M_{23} + M_{23}^{-1})/(2 - M_{11} - M_{22}) \\ 1 \end{pmatrix}.$$

The dispersion function, $\eta(s)$, is given by

$$\begin{aligned} X_{eq} &= (dp/p)\eta_x & Y_{eq} &= (dp/p)\eta_y \\ X'_{eq} &= (dp/p)\eta'_x & Y'_{eq} &= (dp/p)\eta'_y \end{aligned}.$$

Thus,

$$\begin{aligned} \eta &= (M_{13} + M_{13}^{-1})/(2 - M_{11} - M_{22}) \\ \eta' &= (M_{23} + M_{23}^{-1})/(2 - M_{11} - M_{22}). \end{aligned}$$

Hence, the dispersion function can be calculated at the end of each element in the accelerator using the 3×3 matrices corresponding to the matrices M_i described earlier.

It must be stressed that in the above treatment of betatron function calculations, coupling between horizontal and vertical motion has been completely ignored. This is true for most **SYNCH** computations of β and α functions. However, accurate values for the machine tunes and dispersion functions may be obtained by using 7×7 matrices and performing a **FXPT** calculation. This is discussed under "Closed Orbit Calculations" later in this section.

Other machine parameters calculated by **SYNCH** are the transition energy and the natural chromaticities. The transition energy is that energy of the particle beam for which the period of revolution about the machine is independent of particle momentum. This energy is given by $E = \gamma_t m_0$, where m_0 is the rest mass of the orbiting particles. The parameter γ_t , called the transition gamma, is found from the relationship

$$\gamma_t = 1/\sqrt{\alpha},$$

where α represents, for this discussion only, the momentum compaction factor

$$\alpha = (dC/C)/(dp/p), \quad C = \text{the machine's circumference}.$$

By tracking the vector $(\eta_x, \eta'_x, \eta_y, \eta'_y, 0, 1, 0)$ through a superperiod of ideal length S_0 , where η represents the momentum dispersion function, the change in path length, ΔS , for a particle with $dp/p = 1$ can be found. The resulting vector at the end of the superperiod will be $(\eta_x, \eta'_x, \eta_y, \eta'_y, -\Delta S, 1, 0)$. Hence, the value of the transition gamma will be given by

$$\gamma_t = \sqrt{S_0/\Delta S}.$$

Since it can occur that $\alpha < 0$, the complex root of γ_t^2 is taken. This root is displayed at the end of the CYC output.

Chromaticity is defined, in either the horizontal or vertical plane, as the change in the tune of the accelerator per unit change in dp/p . It can be shown^[4] that the tune change, $\Delta\nu$, due to errors, k , in the field gradient function $K = B'/B\rho$ (i.e., $K = K_0 + k$), is given by

$$\Delta\nu = \frac{1}{4\pi} \int k \beta ds.$$

If $K = K_0(1 - dp/p)$, then the chromaticity can be written as

$$\xi_Q = -\frac{1}{4\pi} \int K \beta ds \quad (\text{due to quadrupoles}).$$

Likewise, an expression for the chromaticity due to sextupoles can be obtained:

$$\xi_S = \frac{1}{4\pi} \int K' \eta \beta ds \quad (\text{due to sextupoles}).$$

In a similar fashion, chromatic effects due to the edge focusing of the bending magnets can be taken into account. **SYNCH** calculates the total machine chromaticity due to these three effects and displays the result at the end of the CYC output. The chromaticities computed are due only to magnets which are *explicitly* listed in the beamline used in the CYC statement.

Another method for studying chromatic properties is to perform an **FXPT** calculation for various values of dp/p and look at the behavior of the total machine tunes.

Tracking Betatron Functions Through a Beamline

The 2×2 matrix M representing passage through one periodic section of the accelerator may be written as

$$M = I \cos \mu + J \sin \mu,$$

where I is the identity matrix, and J is given by

$$J = \begin{pmatrix} \alpha & \beta \\ -\gamma & -\alpha \end{pmatrix}.$$

Let R represent the transfer matrix from point 0 to point 1, J_0 the above matrix with betatron functions evaluated at point 0, and J the above matrix with betatron functions evaluated at point 1. Then the two J -matrices are related by

$$J = R J_0 R^{-1}.$$

If R has the matrix elements

$$R = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

then the betatron functions at point 1 in terms of the betatron functions at point 0 are given by the relations

$$\begin{aligned}\alpha_1 &= (ad + bc)\alpha_0 - ac\beta_0 - bd\gamma_0 \\ \beta_1 &= a^2\beta_0 - 2ab\alpha_0 + b^2\gamma_0 \\ \gamma_1 &= d^2\gamma_0 - 2cd\alpha_0 + c^2\beta_0 \\ \psi_1 &= \psi_0 + \arctan\{b/[a\beta_0 - b\alpha_0]\}.\end{aligned}$$

Using these relationships, the betatron functions can be tracked through a beamline given their initial values.

Dispersion functions are tracked through a beamline using the relations

$$\begin{aligned}\eta_1 &= a\eta_0 + b\eta'_0 + e \\ \eta'_1 &= c\eta_0 + d\eta'_0 + f\end{aligned}$$

where e and f are the 1-3 and 2-3 elements of the 3×3 transfer matrices.

8.5 Particle Beam Envelopes

Particle beam envelopes are calculated by SYNCH using betatron functions computed by the program and beam emittances specified by the user. The beam emittance is defined as the area of the xx' (or yy') phase space ellipse which contains some certain fraction of the beam (95%, say). The user may enter his/her own favorite value of the beam emittance for the machine in question. If β , α and γ are the betatron functions at a particular longitudinal location in the accelerator, then the phase space ellipse representing the beam at that point will have the form shown in the figure below:

Figure 2. Phase Space Ellipse.

Here, the area is represented by $\pi\epsilon$, where ϵ is the beam emittance. The beam size, σ , as measured from the reference orbit, is thus given by

$$\sigma = \sqrt{\epsilon\beta}.$$

If the beam has an average momentum error of dp/p , then this ellipse will be translated along the x -axis by an amount $\eta(dp/p)$ and translated along the x' -axis by an amount $\eta'(dp/p)$. η and η' are the dispersion function and its slope. The closed orbit may not lie on the reference orbit if magnet misalignments and errors are taken into account, and so the beam ellipse may be displaced even further. The total beam envelope displayed by the **CYAE** command is given either by

$$\begin{aligned}\sigma &= \sqrt{\epsilon\beta}[1 + \sqrt{\epsilon_{co}/\epsilon}] + |\eta dp/p| \\ \sigma' &= \sqrt{\epsilon\gamma}[1 + \sqrt{\epsilon_{co}/\epsilon}] + |\eta' dp/p|\end{aligned}$$

or by

$$\begin{aligned}\sigma &= \sqrt{\epsilon\beta + (\eta dp/p)^2} + \sqrt{\epsilon_{co}\beta} \\ \sigma' &= \sqrt{\epsilon\gamma + (\eta' dp/p)^2} + \sqrt{\epsilon_{co}\gamma}\end{aligned}$$

depending upon the option chosen. Here, ϵ_{co} is the equivalent emittance representing the closed orbit:

$$\epsilon_{co} = (X_{co})^2/\beta.$$

The calculation is performed for both planes.

For electron machines, the **CYEM** statement may be used to calculate beam emittances, rf integrals, etc. For these calculations, **SYNCH** follows the conventions of Morton.^[5]

8.6 Closed Orbit Calculations

To calculate the closed orbit around an accelerator composed of various linear and non-linear elements, **SYNCH** begins by tracking an initial "first guess" particle state vector through one complete revolution. This initial vector will be denoted as V_0 , while the particle state vector after one revolution will be called V_1 . Next, all of the transfer matrices of the accelerator elements are linearized about this initial single-turn trajectory, generating new transfer matrices, R . For details of how the various element matrices are modified on this step, the reader is referred to [3].

One may now track a particle vector, X , in the neighborhood of V (i.e., $X = V + Z$, where $Z \ll V$) around the machine. Let Z_0 be the initial difference vector, $Z_0 = X_0 - V_0$, which is input by the user. After one revolution, the vector $Z_1 = X_1 - V_1$ will be given by $Z_1 = TZ_0$, where T is the linearized single-turn transfer matrix found by $T = R_N R_{N-1} \dots R_2 R_1$.

For the trajectory to be a closed orbit,

$$X_1 = V_1 + Z_1 = X_0 = V_0 + Z_0$$

or,

$$X_0 = V_1 + TZ_0 = V_1 + T(X_0 - V_0)$$

or,

$$X_0 = -(T + I)^{-1}(V_1 - V) = X_{eq}$$

where I is the identity matrix. Hence, the vector X_{eq} may be used as a new "best guess" V_0 for the closed orbit and the entire operation described above may be repeated n times until $|X_{eq}(n) - X_{eq}(n-1)| = Z_0(n)$ is less than some tolerance.

Once the final vector X_{eq} is found, the closed orbit throughout the machine is given by the last tracking of the particle. Also, the betatron functions about this closed orbit may be extracted from the matrices T and R_i in the manner described earlier in this section. Again, the computations of β and α functions ignore coupling between horizontal and vertical motion.

Generalized momentum dispersion functions may also be extracted from T and R_i , since $TD = D$, where $D = (x, x', y, y', 0, 1, 1)$. This computation of D gives the change in the closed orbit from X_{eq} per unit dp/p .

The program also calculates eigenvalues and eigenvectors of the 4×4 submatrix M , the single-turn matrix which operates on (x, x', y, y') . The eigenvalues and eigenvectors will contain any coupling information and hence will provide the user with accurate values for the "eigen" tunes of the machine. The eigenvectors may be tracked around the accelerator as an output option of the **FXPT** statement.

8.7 Particle Tracking

The procedure for particle tracking with the **SYNCH** program is rather straightforward. The options allowed by the **TRK** command allows one to track a particle through a beamline consisting only of linear elements, or to track a particle through a beamline consisting of linear and non-linear elements. In the first case, the single-turn transfer matrix corresponding to the location of interest is computed and used in the calculation. In the second case, the trajectory of the particle through each individual sub-beamline (previously defined **BML** or **MMM**) and non-linear element obviously must be computed upon every passage.

8.8 Magnet Misalignment Calculations

SYNCH provides two methods of generating magnet misalignments. One method employs the **MAGS** command in conjunction with the **BMIS**, **EMIS** and **SHF** commands. In this procedure, the user may specify transverse misalignments but not rotations about the s -axis. If this method is used, the orbit distortions brought about by the misalignments are shown in the **CYC** output in the columns which usually contain the momentum dispersion functions. The second method employs the **MOVE** command. This method allows for both x and y translations as well as rotational misalignments about the s -axis. In this procedure, the **FXPT** statement is used to compute the new closed orbit of the machine.

The **MAGS** procedure was developed first. The **MOVE** procedure, developed subsequently, is more commonly used today.

The MAGS Method

One way to perform magnet misalignment calculations is to think of the element matrices as acting on the particle state vectors $(x, x', 1)$ and $(y, y', 1)$. One sees from Figure 3 that the coordinates relative to the element's centerline must be transformed in the following manner when entering or exiting a misaligned element.

At entrance to the misaligned element:

$$x \rightarrow x - a; \quad x' \rightarrow x' - \theta.$$

At exit from the misaligned element:

$$x \rightarrow x + b; \quad x' \rightarrow x' + \theta.$$

Figure 3. Misaligned Magnet.

Thus, if the coordinates are transformed upon entrance to and exit from the magnet by the "shift" matrices

$$S_i = \begin{bmatrix} 1 & 0 & -a \\ 0 & 1 & -\theta \\ 0 & 0 & 1 \end{bmatrix}, \quad S_o = \begin{bmatrix} 1 & 0 & b \\ 0 & 1 & \theta \\ 0 & 0 & 1 \end{bmatrix},$$

then the transfer matrix for the misaligned magnet is given by $M = S_o M_o S_i$, where M_o is the original 3×3 matrix representing the aligned magnet with $M_o(1,3)$ and $M_o(2,3)$ set to zero.

In this manner, the same procedure used to calculate the closed orbit for an off-momentum particle in an aligned accelerator may be followed to calculate the closed orbit in the misaligned accelerator. Hence, CYC is used.

The MOVE Method

The other procedure for studying the effects of misaligned elements is to invoke the MOVE and SHF7 commands and calculate the new closed orbit using FXPT. This method requires the use of 7×7 matrices and allows the study of horizontal and vertical coupling. The seventh column of these matrices is the analog of the third column in the BMIS method. Again, shift matrices are employed as well as rotation matrices to handle rotations about the s -axis. The appropriate 7×7 shift matrices are of the form

$$S_i = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -a_x \\ 0 & 1 & 0 & 0 & 0 & 0 & -\theta_x \\ 0 & 0 & 1 & 0 & 0 & 0 & -a_y \\ 0 & 0 & 0 & 1 & 0 & 0 & -\theta_y \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad S_o = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & b_x \\ 0 & 1 & 0 & 0 & 0 & 0 & \theta_x \\ 0 & 0 & 1 & 0 & 0 & 0 & b_y \\ 0 & 0 & 0 & 1 & 0 & 0 & \theta_y \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

8.9 Parameterization of Transport and Period Matrices with X-Y Coupling

A parameterization of 4×4 matrices describing linear beam transport systems has been obtained by Edwards and Teng.^[10] Here we extend their formalism^[12] to include dispersive effects, and give prescriptions for incorporating it in the program SYNCH. A period of a beam transport system, or an element or segment of such a system (periodic or not) is characterized by a 6×6 transfer matrix, which we write in the form

$$T = \begin{pmatrix} M & n & d_1 \\ m & N & d_2 \\ e_1 & e_2 & F \end{pmatrix}. \quad (1)$$

Here we have written the 6×6 matrix T in terms of 2×2 submatrices M, m , etc. The dynamic variables are taken to be $x, x' \equiv dx/ds, y, y', -\Delta s, \Delta p/p$, in that order. We consider only transport systems without acceleration or damping; then the elements in the fifth column and sixth row of T vanish except for $T_{55} = T_{66} = 1$, and the matrix T is symplectic, which means that the inverse of T is given by

$$T^{-1} = \bar{T} \equiv \begin{pmatrix} \bar{M} & \bar{m} & \bar{e}_1 \\ \bar{n} & \bar{N} & \bar{e}_2 \\ \bar{d}_1 & \bar{d}_2 & \bar{F} \end{pmatrix} \quad (2)$$

where the "symplectic conjugate" $\bar{a}$ of any 2×2 matrix a is defined as

$$\bar{a} \equiv \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix} \quad (3)$$

and the symplectic conjugate of a 4×4 or 6×6 matrix is defined by (2).

Parameterization of 4×4 Matrices for a Complete Period

When T is a matrix describing a complete period (a circular accelerator or storage ring, or a cell of a periodic system), Edwards and Teng find a similarity transformation that transforms the $x - y$ 4×4 submatrix of T (which we also designate by T) into uncoupled form:

$$T = R U \bar{R} \quad \text{with} \quad U = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \quad (4)$$

where R has the form

$$R = \begin{pmatrix} I \cos \varphi & \bar{D} \sin \varphi \\ -D \sin \varphi & I \cos \varphi \end{pmatrix} \quad (5)$$

and A, B and D are 2×2 unimodular (symplectic) matrices, I and 0 are the 2×2 unit and null matrices, and φ is an equivalent rotation angle. The eigenvalues of T and the matrix D and the angle φ are determined as follows:^[4] Eigenvalues are $\exp(\pm i\mu_1), \exp(\pm i\mu_2)$, where μ_1 and μ_2 are the phase advances of the normal modes, with

$$\cos \mu_1 + \cos \mu_2 = \text{Tr} T / 2 = \text{Tr}(M + N) / 2. \quad (6)$$

Define

$$t = \text{Tr}(M - N) / 2; \quad \Delta = \det(m + \bar{n}); \quad (7)$$

then

$$\cos \mu_1 - \cos \mu_2 = \delta \equiv t \sqrt{1 + \Delta / t^2} \quad (8)$$

and

$$D = -\frac{m + \bar{n}}{\sqrt{\Delta}}; \quad D \sin \varphi = -\frac{m + \bar{n}}{\sqrt{2\delta(\delta + t)}}; \quad (9)$$

with

$$\cos \varphi = \sqrt{\frac{\delta + t}{2\delta}}; \quad \sin \varphi = \sqrt{\frac{\delta - t}{2\delta}}.$$

We confine ourselves to the case where the phase advances are real, i.e. the motion is stable; then $\Delta + t^2$ has to be positive so that δ is real. Note that we resolve the ambiguity of the sign of square roots by requiring δ to have the same sign as t . (If Δ is negative then $\sin \varphi$ and $\mathbf{D}$ are imaginary, but $\mathbf{D} \sin \varphi$ is still real, which is all that really matters). In [10] it is shown that the transfer matrices $\mathbf{A}$ and $\mathbf{B}$ for the uncoupled normal modes are given by

$$\mathbf{A} = \mathbf{M} + \frac{\mathbf{n}(\mathbf{m} + \bar{\mathbf{n}})}{\delta + t}; \quad \mathbf{B} = \mathbf{N} - \frac{\mathbf{m}(\mathbf{n} + \bar{\mathbf{m}})}{\delta + t}. \quad (10)$$

These are 2×2 unimodular matrices, and can be parameterized in terms of phase advances and Twiss (Courant-Snyder) parameters in the usual way:

$$\mathbf{A} = \begin{pmatrix} \cos \mu_x + \alpha_x \sin \mu_x & \beta_x \sin \mu_x \\ -\gamma_x \sin \mu_x & \cos \mu_x - \alpha_x \sin \mu_x \end{pmatrix} \quad (11)$$

$$\mathbf{B} = \begin{pmatrix} \cos \mu_y + \alpha_y \sin \mu_y & \beta_y \sin \mu_y \\ -\gamma_y \sin \mu_y & \cos \mu_y - \alpha_y \sin \mu_y \end{pmatrix}. \quad (12)$$

The parameters α, β, γ , and μ in (11) and (12) may be taken as the definitions of the generalized Twiss parameters of the matrix $\mathbf{T}$.

Parameterization of Elements of a Periodic System

The parameterization just found applies to the matrix for a complete period. It does not apply to the individual elements or components of the period, since when the beam traverses an element the α and β functions are generally different at the beginning and the end.

Consider a periodic system $\mathbf{G}$. The matrix elements of $\mathbf{G}$ are periodic in s , as are the parameters α, β, γ . At each azimuth s the parameters can be determined as detailed above. Now suppose the matrix for going from azimuth s_1 to s_2 is $\mathbf{T}$, so that

$$\mathbf{G}_2 = \mathbf{T} \mathbf{G}_1 \bar{\mathbf{T}}. \quad (13)$$

We reduce $\mathbf{G}_1$ and $\mathbf{G}_2$ to semi-diagonal form by the methods of the previous section:

$$\mathbf{G}_1 = \mathbf{R}_1 \mathbf{U} \bar{\mathbf{R}}_1; \quad \mathbf{G}_2 = \mathbf{R}_2 \mathbf{U} \bar{\mathbf{R}}_2. \quad (14)$$

Then $\mathbf{T}$ may be written as

$$\mathbf{T} = \mathbf{R}_2 \mathbf{V} \bar{\mathbf{R}}_1 \quad (15)$$

so that

$$\mathbf{V} = \bar{\mathbf{R}}_2 \mathbf{T} \mathbf{R}_1 \quad (16)$$

which, with (13) and (14), gives

$$\mathbf{U}_2 = \mathbf{V} \mathbf{U}_1 \bar{\mathbf{V}}$$

or

$$\mathbf{U}_2 \mathbf{V} = \mathbf{V} \mathbf{U}_1. \quad (17)$$

Since U_1 and U_2 are semi-diagonal, so is V , i.e. V may be regarded as the semi-diagonalization of the component matrix T in the context of T as an element of G .

To find V explicitly we write R_1 and R_2 in the form (5), and use (15) in the form $R_2 V = T R_1$ with T in the form (1):

$$A \cos \varphi_2 = M \cos \varphi_1 - n D_1 \sin \varphi_1; \quad (18a)$$

$$B \cos \varphi_2 = N \cos \varphi_1 + m \bar{D}_1 \sin \varphi_1. \quad (18b)$$

Thus the element matrix T is semi-diagonalized, with the help of the semi-diagonalization parameters of G_1 and G_2 .

For computational purposes it would be preferable if one did not first have to carry out the procedure for both the matrices G_1 and G_2 . In fact the explicit computation of $\cos \varphi_2$ can be avoided: We note that the uncoupled matrices A and B must be unimodular. Therefore we may simply compute the right-hand sides of (18), and then normalize by dividing by the square root of the determinant of the resulting matrices. Using (9) we have

$$M \cos \varphi_1 - n D_1 \sin \varphi_1 = \left[M + \frac{n(m_1 + \bar{n}_1)}{\delta + t_1} \right] \sqrt{\frac{\delta + t_1}{2\delta}} \quad (19)$$

and similarly for the second line of (18). Here the subscript 1 refers to the global matrix G_1 and its components, while M, N, m, n without subscripts are the 2×2 submatrices of the matrix T . Thus the uncoupled transfer matrices A and B for the two normal modes for the matrix T are found as follows:

Find t_1 and δ for the global matrix G_1 . From the 2×2 submatrices of G_1 and T form the matrices

$$A' = M + n(m_1 + \bar{n}_1)/(\delta + t_1) \quad (20)$$

$$B' = N - m(n_1 + \bar{m}_1)/(\delta + t_1). \quad (21)$$

Find the determinants of these matrices (they should be equal). The uncoupled matrices A and B are the unimodular 2×2 matrices

$$A = A' / \sqrt{\det(A')} \quad B = B' / \sqrt{\det(B')}. \quad (22)$$

The phase advances for going through T can be found using the parameterization

$$A = \begin{pmatrix} \sqrt{\frac{\beta_{x2}}{\beta_{x1}}} (\cos \psi_x + \alpha_{x1} \sin \psi_x) & \sqrt{\beta_{x1} \beta_{x2}} \sin \psi \\ \dots & \sqrt{\frac{\beta_{x1}}{\beta_{x2}}} (\cos \psi_x - \alpha_{x2} \sin \psi_x) \end{pmatrix} \quad (23)$$

where the (21) element is obtained by requiring A to be unimodular; the B matrix has the same form with the y parameters. The phase advances are therefore

$$\psi_x = \arctan[A_{12}/(\beta_{x1} A_{11} - \alpha_{x1} A_{12})] \quad (24)$$

$$\psi_y = \arctan[B_{12}/(\beta_{y1} B_{11} - \alpha_{y1} B_{12})] \quad (25)$$

which are again expressed in terms of the parameters of the previous global matrix G_1 and of T .

Dispersion

The fifth and sixth rows and columns of the full matrices refer to the change in path length $-\Delta s$ and to relative momentum deviation $\Delta p/p$. In the uncoupled case, the x -variables still depend on momentum; this is customarily described (as in the SYNCH program), by augmenting the 2×2 matrix with a third column, where the elements A_{13} and A_{23} describe the dependence of excursion and slope on momentum. The corresponding elements in the decoupled matrices developed here can be obtained by augmenting the matrices $\mathbf{R}$ effecting the similarity transformations with fifth and sixth rows and columns, all zero except for $R_{55} = R_{66} = 1$.

Printed Output

In the SYNCH program matrices with $x - y$ coupling are generally formulated in a 7×7 format (the seventh column describes perturbations, and need not concern us here). However, in the CYC and FXPT operations, which calculate the betatron functions at the end of each element of a lattice or transport line, all 7×7 matrices are truncated in previous versions of SYNCH to two 2×3 matrices each, with coupling information lost. We have attempted to remedy this truncation algorithm in the present version of SYNCH. In the FXPT operation in the new version, the results of the previous sections are used to generate α and β functions, dispersion functions, and phase advances pertaining to the normal oscillation modes of the coupled system (called x and y but not necessarily horizontal and vertical in space), while the closed orbit FXPT produces should still be in space coordinates. The transformations between normal modes and space coordinates (e.g. the matrix $\mathbf{D}$) are not exhibited in the SYNCH output, but the coupling angle φ is printed out at each point of the lattice, together with the other orbit functions.

REFERENCES

1. A. A. Garren, A. S. Kenney, J. W. Eusebio, SYNCH-A Computer System for Synchrotron Design and Orbit Analysis, LBL Internal Report UCID-10153, 1965.
2. A. A. Garren, A. S. Kenney, E. D. Courant, M. J. Syphers, A Users Guide to SYNCH, Fermilab Report, unpublished, June, 1985
3. A. A. Garren, SYNCH Closed Orbit and Related Calculations, PEP Technical Memo 49, 1977. A discussion of the methods used in SYNCH to calculate closed orbit information.
4. E. D. Courant, H. S. Snyder, Theory of the Alternating Gradient Synchrotron, Annals of Physics, Vol. 3, No. 1, 1958.
5. P. L. Morton, Effects of Transverse Coupling in the SLAC Storage Ring, SLAC-PUB-863, 1971 Particle Accelerator Conference, Chicago, Illinois.
6. F. James, M. Roos, MINUIT-Function Minimization and Error Analysis, CERN Computer Center Program Library, D506, 1967, revised 1983. CERN write-up referring to FORTRAN 77 and FORTRAN 4 implementation of MINUIT.
7. Proceedings from the 1981-1984 Summer Schools on High Energy Particle Accelerators. Provide many excellent articles, tutorials, problems in accelerator physics.
8. Theoretical Aspects of the Behavior of Beams in Accelerators and Storage Rings, Proceedings of the First Course of the International School of Particle Accelerators, November 1976; CERN, 1977.
9. B. Autin and Y. Marti, Closed Orbit Correction of Alternating Gradient Machines Using a Small Number of Magnets, CERN/ISR-MA/73-17, CERN, 1973.
10. D. A. Edwards, L. C. Teng, Parameterization of Linear Coupled Motion in Periodic Systems. IEEE Trans. Nucl. Sci., 20, No. 3, 85 (1973)
11. H. Grote, F. Christoph Iselin, The MAD Program User's Reference Manual, Version 8.1, CERN/SL/90-13 (AP). Some earlier versions of MAD contained a SYNCH to MAD translator.
12. E. D. Courant, Parameterization of Transport and Period Matrices with X-Y Coupling, BNL Report, to be published.
13. DEPOL is an unpublished program to compute depolarization effects written by E. D. Courant.
14. M. Sands, The Physics of Electron Storage Rings, An Introduction, SLAC-121, Nov. 1970

APPENDIX A FILES

This Appendix describes disk files used by the program during execution and some special purpose files that may contain diagnostic information or data for user-written post-processors.

Files generated by commands whose major purpose is to create them are automatically saved. Some files are made as option requests in action-type commands or by **OPEN**. Other files are deleted unless a save request is made by **KEEP**. See **OPEN** and **KEEP** for internal files names to be used as their data. Files saved will bear the names **FOROXX.DAT**, where **XX** is the unit number.

The files used by **SYNCH** are listed in Table A below.

Table A. Files Used by SYNCH.

Internal Name	Logical Unit	Description
JFILE	2	Input data file submitted by user.
OUT3	3	ASCII Output file from SYNCH program.
SVOUT	4	Summary and diagnostic file created by fitting program MINUIT , which is invoked by SOLV . It is not saved unless requested by KEEP command. If one encounters a problem with solving, sometimes valuable information may be found in this file.
INFILE	8	Corrected copy of input file 2. If file 2 is an old input file containing outdated command names, they are replaced with current names on this file. If changes have occurred, this file is saved, so that the user may rename it to the file 2 and submit it in future runs, thus avoiding editing the old one.
FIL11	11	Binary file of polarization parameters made by CYC and FXPT upon request. In both cases, it must be requested by OPEN . The CYC command must also choose the proper option in its <i>m</i> -value.
JBIS	15	Binary file of betatron functions made by CYC on option.
PLFILE	17	ASCII file containing values that are stored for plotting program. It is not used by plot program, but could be looked at by user or function as input to any other outside program. It is written only by a request in OPEN .
CYBO	20	ASCII file of the dispersion functions created by CYC on option.
ORB1	24	Binary file written by FXPT , read by ORBC . It is saved only by a request in KEEP .
OFIL	26	ASCII file consisting of updated input requested by UPDAT . This file may be used as input for a future SYNCH run.
LFIL	70	ASCII file written by IOUT .

APPENDIX B SAMPLE RUNS

This section provides example runs of **SYNCH** programs. The individual programs have been run on a DEC VAX systems. Both the input file to and the output file produced by **SYNCH** are included.

- B.1 Calculation of Periodic Lattice Functions**
- B.2 Fitting of Tunes in a Phase Trombone**
- B.3 Closed Orbit Calculations**

B.1 Calculation of Periodic Lattice Functions

This example illustrates the way in which magnetic elements and beamlines are defined and how periodic accelerator lattice functions can be computed. A standard cell of the Fermilab Tevatron is used. Also exhibited is the use of the "matrix multiply" command (MMM).

```

TEUC  RUN          Tevatron Standard Cell
C
C          This particular SYNCH run calculates the betatron
C          functions through a Tevatron standard FODO cell.
C
C
C-----
C
C          Lengths in meters, field strengths in kG, kG/m, etc.
C
C  Magnetic Rigidity at 1 TeV ...
C
C  BRHO  =          33387.702
C
C  Bend field strength ...
C
C  BY    =          44.27664
C
C  Quadrupole gradients ...
C
C  GF    =          760.32056
C  GD    =          -760.32056
C
C  Magnet lengths ...
C
C  BL    =          6.1214
C  QL    =          1.67894
C
C
C
C          Drift Definitions
C
C  O      DRF          0.2794
C  OO     DRF          2.29616
C  OOO    DRF          0.4445
C
C          Magnet Definitions
C
C  B      MAG          BL          0.0          BRHO          BY          $
C  QF     MAG          QL          GF          BRHO
C  QD     MAG          QL          GD          BRHO
C
C          Standard Cell Beamline Definition
C
C  HC      BML          OO      B      O      B      O      B      O      B      OOO
C
C  CELL    BML          HC      QF      HC      QD
C

```

```

P      Computation of Betatron Functions
C
      CVC      CELL
P
C      Define half-cell matrices in order to print out betatron functions
C      only at the ends of the quadrupoles...
C
MHCF  MMM      HC  QF
MHCD  MMM      HC  QD
C
C      Define a new "beamline" made up of the above two matrices.  Print
C      out betatron functions at the ends of the quadrupoles...
C
CL    BML      MHCF MHCD
      CVC      CL
C
C      STOP

```

SYNCH RUN TEVC
16-JAN-93 10:01:09

Tevatron Standard Cell

SYNCH VERSION

This particular SYNCH run calculates the betatron
functions through a Tevatron standard FODO cell.

Lengths in meters, field strengths in kG, kG/m, etc.

Magnetic Rigidity at 1 TeV ...

*** BRHO = // 33387.702

Bend field strength ...

*** BY = // 44.27664

Quadrupole gradients ...

*** GF = // 760.32056

*** GD = // -760.32056

Magnet lengths ...

*** BL = // 6.1214

*** QL = // 1.67894

Drift Definitions

*** O DRF // 0.2794

*** OO DRF // 2.29616

*** OOO DRF // 0.4445

Magnet Definitions

*** B MAG // BL 0.0 BRIHO BY \$

*** QF MAG // QL GF BRIHO

*** QD MAG // QL GD BRHO

Standard Cell Beamline Definition

*** HC BML // OO B O B O B O B OOO

*** CELL BML // HC QF HC QD

Computation of Betatron Functions

***	CYC		// CELL									
POS	S(M)	NUX	NUY	BETAX(M)	BETAY(M)	XEQ(M)	YEQ(M)	ZEQ (M)	ALPHAX	ALPHAY	DXEQ	DYEQ
0	0.0000	0.00000	0.00000	29.04507	97.80421	2.26773	0.00000	0.0000	-0.58154	1.87141	0.04412	0.00000
1 OO	2.2962	0.01201	0.00391	31.95858	89.45278	2.36903	0.00000	0.0000	-0.68733	1.76572	0.04412	0.00000
2 B	8.4176	0.03869	0.01627	42.09965	69.55521	2.66393	0.00000	0.0204	-0.96935	1.48467	0.05223	0.00000
3 O	8.6970	0.03974	0.01691	42.64492	68.72917	2.67852	0.00000	0.0204	-0.98222	1.47180	0.05223	0.00000
4 B	14.8184	0.05966	0.03317	56.39634	52.43252	3.02311	0.00000	0.0435	-1.26425	1.19036	0.06035	0.00000
5 O	15.0978	0.06044	0.03403	57.10640	51.77095	3.03997	0.00000	0.0435	-1.27712	1.17748	0.06035	0.00000
6 B	21.2192	0.07540	0.05575	74.46816	39.07971	3.43425	0.00000	0.0697	-1.55915	0.89571	0.06847	0.00000
7 O	21.4986	0.07600	0.05690	75.34301	38.58278	3.45338	0.00000	0.0697	-1.57202	0.88283	0.06847	0.00000
8 B	27.6200	0.08744	0.08594	96.31512	29.50044	3.89735	0.00000	0.0996	-1.85405	0.60083	0.07659	0.00000
9 OOO	28.0645	0.08817	0.08836	97.97247	28.97542	3.93140	0.00000	0.0996	-1.87453	0.58032	0.07659	0.00000
10 QF	29.7434	0.09087	0.09768	97.97247	28.97530	3.93310	0.00000	0.0996	1.87453	-0.58024	-0.07457	0.00000
11 OO	32.0396	0.09477	0.10972	89.60695	31.88317	3.76189	0.00000	0.0996	1.76874	-0.68617	-0.07457	0.00000
12 B	38.1610	0.10711	0.13646	69.67928	42.01003	3.33029	0.00000	0.1283	1.48671	-0.96811	-0.06645	0.00000
13 O	38.4404	0.10776	0.13751	68.85210	42.55461	3.31172	0.00000	0.1283	1.47384	-0.98100	-0.06645	0.00000
14 B	44.5618	0.12398	0.15747	52.53477	56.28955	2.92982	0.00000	0.1536	1.19181	-1.26268	-0.05833	0.00000
15 O	44.8412	0.12483	0.15826	51.87238	56.99873	2.91352	0.00000	0.1536	1.17894	-1.27556	-0.05833	0.00000
16 B	50.9626	0.14652	0.17325	39.16540	74.33796	2.58131	0.00000	0.1759	0.89691	-1.55690	-0.05021	0.00000
17 O	51.2420	0.14766	0.17384	38.66780	75.21155	2.56728	0.00000	0.1759	0.88404	-1.56977	-0.05021	0.00000
18 B	57.3634	0.17663	0.18531	29.57116	96.15030	2.28476	0.00000	0.1955	0.60201	-1.85069	-0.04209	0.00000
19 OOO	57.8079	0.17904	0.18604	29.04507	97.80466	2.26605	0.00000	0.1955	0.58154	-1.87115	-0.04209	0.00000
20 QD	59.4868	0.18835	0.18874	29.04507	97.80421	2.26773	0.00000	0.1955	-0.58154	1.87141	0.04412	0.00000
CIRCUMFERENCE = 59.4868 M				THETX = 0.06494248 RAD		NUX = 0.18835		DNUX/(DP/P) = -0.21240				
RADIUS = 9.4676 M				THETY = 0.00000000 RAD		NUY = 0.18874		DNUY/(DP/P) = -0.21294				
(DS/S)/(DP/P)= 0.0032870				TGAM=(17.44210,		0.00000)						
MAXIMA	---	BETX(10) =	97.97247	BETY(19) =	97.80466	XEQ(10) =	3.93310	YEQ(20) =	0.00000			
MINIMA	---	BETX(19) =	29.04507	BETY(10) =	28.97530	XEQ(19) =	2.26605	YEQ(20) =	0.00000			

Define half-cell matrices in order to print out betatron functions
only at the ends of the quadrupoles...

```
*** MHCF  MMM      // HC  QF
*** MHCD  MMM      // HC  QD
```

Define a new "beamline" made up of the above two matrices. Print
out betatron functions at the ends of the quadrupoles...

```
*** CL      BML      // MHCF MHCD
***        CYC      // CL
```

POS	S(M)	NUX	NUY	BETAX(M)	BETAY(M)	XEQ(M)	YEQ(M)	ZEQ (M)	ALPHAX	ALPHAY	DXEQ	DYEQ
0	0.0000	0.00000	0.00000	29.04507	97.80421	2.26773	0.00000	0.0000	-0.58154	1.87141	0.04412	0.00000
1 MHCF	29.7434	0.09087	0.09768	97.97247	28.97530	3.93310	0.00000	0.0996	1.87453	-0.58024	-0.07457	0.00000
2 MHCD	59.4868	0.18835	0.18874	29.04507	97.80421	2.26773	0.00000	0.1955	-0.58154	1.87141	0.04412	0.00000

```
CIRCUMFERENCE = 59.4868 M      THETX = 0.06494248 RAD      NUX = 0.18835      DNUX/(DP/P) = 0.00000
RADIUS = 9.4676 M      THETY = 0.00000000 RAD      NUY = 0.18874      DNUY/(DP/P) = 0.00000
(DS/S)/(DP/P) = 0.0032870      TGAM=( 17.44210, 0.00000)
```

```
MAXIMA --- BETX( 1) = 97.97247      BETY( 2) = 97.80421      XEQ( 1) = 3.93310      YEQ( 2) = 0.00000
MINIMA --- BETX( 2) = 29.04507      BETY( 1) = 28.97530      XEQ( 2) = 2.26773      YEQ( 2) = 0.00000
```

=====

END OF SYNCH RUN TEVC

B.2 Fitting of Tunes in a Phase Trombone

In the following example, a "trombone" is constructed: a beam line in which quadrupoles can be varied so as to adjust the tunes (phase advances) to any desired value while preserving the values of the orbit functions at the ends of the beam line—thus such a trombone can be inserted in a lattice without changing the optical properties of the rest of the lattice. This is accomplished using the SOLV instruction. The quadrupoles are initially set the same as those in the standard cell (which gives a phase advance of $5/6$ times 2π), and then adjusted by SOLV so as to vary this phase advance while maintaining the orbit function matching.

TRMB RUN

SYNCH Example to Illustrate Fitting

In this example, the phase advance through a "phase trombone" is fitted to specified values. The phase trombone consists of 5 standard-sized cells with full bending, but the quadrupoles are powered separately. The idea is to match the Courant-Snyder parameters at the beginning and end of the trombone to those of the standard cell but to make the phase advance through the trombone any value one wishes.

Phase Trombone:

Adjust quad power supplies A,B,C,D,E to make phase advances from middle of QD to middle of QD desired values.

Lengths in meters, Fields in Tesla, Tesla/meter

BRHO	=	66712.8	
BO	=	6.0	
RHO	=	11118.8	
RHI	=	1.	/ RHO

60-degree Standard Cells:

MUX	=	.16666667
MUY	=	.16666667
BR	=	1.0
KF	=	.003
KD	=	-.003
LB	=	16.6
LQ	=	2.5

(Note: LQ is the half-quadrupole length)

Bending Magnets ...

B	MAG	LB	O.	BR	RHI
---	-----	----	----	----	-----

Standard Cell Quadrupole Magnets ...

SRC	SUB			
QF	MAG	LQ	KF	BR
QD	MAG	LQ	KD	BR
CL	MM	C		
	END			

0	DRF	0.5
00	DRF	2.5
00C	DRF	5.0

Beamlines

Standard Cell

.BB	BML	B	0	B	0	B	0	B	0	B
.OBO	BML	00C	.BB	00C						
BB	MM	.BB								
OBO	MM	.OBO								
FD	BML	QF	.OBO	QD						
DF	BML	QD	.OBO	QF						
C	BML	DF	FD							

Phase Trombone

.TR	BML	QD	OBO	QFA	QFA	OBO	QDB	QDB	OBO	QFC
		QFC	OBO	QDD	QDD	OBO	QFE	QFE		
		OBO	QDD	QDD	OBO	QFC	QFC			
		OBO	QDB	QDB	OBO	QFA	QFA	OBO	QD	

Fit the Cells ...

	FITQ	SRC	CL	KF	KD	1	1MUX	MUY
CELL	CYC	C						

Fit Phase Trombone ...

Trombone Quadrupole Magnets

```
IBT  IBET          115.95190 0.      0.0086243 2.24716      0.0
          344.95041 0.      0.0028990 0.0      0.0
KAA  PARA          KF
KBB  PARA          KD
KCC  PARA          KF
KDD  PARA          KD
KEE  PARA          KF
TRMB SUB
QFA  MAG          LQ      KAA      BR
QDB  MAG          LQ      KBB      BR
QFC  MAG          LQ      KCC      BR
QDD  MAG          LQ      KDD      BR
QFE  MAG          LQ      KEE      BR
TTR  TRKB  0  30 .TR  IBT          0  30
      END
```

(NOTE: In the TRKB statement a CL could
have been appropriately placed in line 1
instead of IBT.
The program would have then retrieved the
values of the Courant-Snyder parameters
from the matrix CL.)

Specify Desired Phase Advance Through
the Trombone...

```
MUXT - .85
MUYT - .85
```

Vary trombone parameters so as to obtain the desired phase advance. Repeat with different values of the phase advance.

(The following line may optionally be inserted anywhere in an input file to assist the programmer in finding the appropriate columns for entries)

...;...1....;...2....;...3....;...4....;...5....;...6....;...7....;...

```

UAPPH SUB
  SOLU  5  1  TAMB  TTR          3000  -3          2
          AX          15          0.0          .0001
          AY          15          0.0          .0001
          DX          15          0.0          .0001
          MUX          30          MUXT          .001
          MUY          30          MUYT          .001
          KAA  KBB  KCC  KDD  KEE      1  -.004      .004      .00001
INS  CVC  -1  .TR
      INCR  1  MUXT          .05
      INCR  1  MUYT          .05
      END
      CALL  3  UAPPH
      PAGE
      CALL  TAMB
      STOP

```

SYNCH RUN TRMB
13-JAN-93 14:51:58

SYNCH VERSION

=====

SYNCH Example to Illustrate Fitting

In this example, the phase advance through a "phase trombone" is fitted to specified values. The phase trombone consists of 5 standard-sized cells with full bending, but the quadrupoles are powered separately. The idea is to match the Courant-Snyder parameters at the beginning and end of the trombone to those of the standard cell but to make the phase advance through the trombone any value one wishes.

Phase Trombone:

Adjust quad power supplies A,B,C,D,E to make phase advances from middle of QD to middle of QD desired values.

Lengths in meters, Fields in Tesla, Tesla/meter

```
*** BRHO = // 66712.8
*** B0 = // 6.0
*** RHO = // 11118.8
*** RHI = // 1. / RHO
```

60-degree Standard Cells:

```
*** MUX = // .166666667
*** MUY = // .166666667

*** BR = // 1.0
*** KP = // .003
*** KD = // -.003
*** LB = // 16.6
*** LQ = // 2.5
```

(Note: LQ is the half-quadrupole length)

Bending Magnets ...

```
*** B MAG // LB 0. BR RHI
```

Standard Cell Quadrupole Magnets ...

```
*** SRC SUB 0 0 //
```

.....

```
*** QF MAG // LQ KP BR
*** QD MAG // LQ KD BR
```

```

*** CL      MMM      // C
*** END      0  0  //

```

```

*** O      DRF      // 0.5
*** OO     DRF      // 2.5
*** OOC    DRF      // 5.0

```

Beamlines

Standard Cell

```

*** .BB  BML      // B  O  B  O  B  O  B  O  B
*** .OBO BML      // OOC .BB OOC
*** BB   MMM      // .BB
*** OBO  MMM      // .OBO
*** FD   BML      // QF  .OBO QD
*** DF   BML      // QD  .OBO QF
*** C    BML      // DF  FD

```

Phase Trombone

```

*** .TR  BML      // QD  OBO QFA QFA OBO QDB QDB OBO QFC
*      // QFC OBO QDD QDD OBO QFE QFE
*      // OBO QDD QDD OBO QFC QFC
*      // OBO QDB QDB OBO QFA QFA OBO QD

```

Fit the Cells ...

*** FITQ // SRC CL KF KD 1 1MUX MUY

PARAMETER REPLACEMENTS MADE BY FITTING

1 OF KF = 0.002034033492 1 OF KD = -0.002034156078

*** CELL CYC // C

POS	S(M)	NUX	NUY	BETAX(M)	BETAY(M)	XEQ(M)	YEQ(M)	ZEQ (M)	ALPHAX	ALPHAY	CXEQ	DYEQ
0	0.0000	0.00000	0.00000	115.95190	344.95041	2.24716	0.00000	0.0000	0.00000	0.00000	0.00000	0.00000
1 QD	2.5000	0.00342	0.00116	117.48644	340.60149	2.26146	0.00000	0.0000	-0.61641	1.73219	0.01145	0.00000
2 OOC	7.5000	0.01001	0.00356	123.94423	323.57321	2.31872	0.00000	0.0000	-0.67514	1.67346	0.01145	0.00000
3 B	24.1000	0.02946	0.01248	149.59536	271.25075	2.52121	0.00000	0.0036	-0.87010	1.47849	0.01294	0.00000
4 O	24.6000	0.02999	0.01277	150.46840	269.77519	2.52769	0.00000	0.0036	-0.87598	1.47262	0.01294	0.00000
5 B	41.2000	0.04595	0.02352	182.78703	224.12080	2.75496	0.00000	0.0076	-1.07093	1.27765	0.01444	0.00000
6 O	41.7000	0.04639	0.02388	183.86089	222.84610	2.76217	0.00000	0.0076	-1.07680	1.27177	0.01444	0.00000
7 B	58.3000	0.05945	0.03695	222.84674	183.85978	3.01422	0.00000	0.0119	-1.27174	1.07680	0.01593	0.00000
8 O	58.8000	0.05981	0.03738	224.12142	182.78592	3.02218	0.00000	0.0119	-1.27762	1.07093	0.01593	0.00000
9 B	75.4000	0.07056	0.05334	269.77413	150.46768	3.29901	0.00000	0.0166	-1.47255	0.87595	0.01742	0.00000
10 O	75.9000	0.07086	0.05387	271.24961	149.59466	3.30772	0.00000	0.0166	-1.47842	0.87008	0.01742	0.00000
11 B	92.5000	0.07978	0.07332	323.56875	123.94449	3.60932	0.00000	0.0217	-1.67334	0.67511	0.01891	0.00000
12 OOC	97.5000	0.08217	0.07992	340.59570	117.48703	3.70389	0.00000	0.0217	-1.73206	0.61638	0.01891	0.00000
13 QF	100.0000	0.08333	0.08333	344.94429	115.95257	3.72756	0.00000	0.0217	0.00000	0.00000	0.00000	0.00000
14 QF	102.5000	0.08449	0.08675	340.59570	117.48703	3.70389	0.00000	0.0217	1.73206	-0.61638	-0.01891	0.00000
15 OOC	107.5000	0.08689	0.09335	323.56875	123.94449	3.60932	0.00000	0.0217	1.67334	-0.67511	-0.01891	0.00000
16 B	124.1000	0.09581	0.11280	271.24961	149.59466	3.30772	0.00000	0.0269	1.47842	-0.87008	-0.01742	0.00000
17 O	124.6000	0.09611	0.11333	269.77413	150.46768	3.29901	0.00000	0.0269	1.47255	-0.87595	-0.01742	0.00000
18 B	141.2000	0.10686	0.12928	224.12142	182.78592	3.02218	0.00000	0.0316	1.27762	-1.07093	-0.01593	0.00000
19 O	141.7000	0.10721	0.12972	222.84674	183.85978	3.01422	0.00000	0.0316	1.27174	-1.07680	-0.01593	0.00000
20 B	158.3000	0.12028	0.14279	183.86089	222.84610	2.76217	0.00000	0.0359	1.07680	-1.27177	-0.01444	0.00000
21 O	158.8000	0.12072	0.14314	182.78703	224.12080	2.75496	0.00000	0.0359	1.07093	-1.27765	-0.01444	0.00000
22 B	175.4000	0.13667	0.15389	150.46840	269.77519	2.52769	0.00000	0.0399	0.87598	-1.47262	-0.01294	0.00000
23 O	175.9000	0.13720	0.15419	149.59536	271.25075	2.52121	0.00000	0.0399	0.87010	-1.47849	-0.01294	0.00000
24 B	192.5000	0.15665	0.16311	123.94423	323.57321	2.31872	0.00000	0.0435	0.67514	-1.67346	-0.01145	0.00000
25 OOC	197.5000	0.16325	0.16551	117.48644	340.60149	2.26146	0.00000	0.0435	0.61641	-1.73219	-0.01145	0.00000
26 QD	200.0000	0.16667	0.16667	115.95190	344.95041	2.24716	0.00000	0.0435	0.00000	0.00000	0.00000	0.00000

CIRCUMFERENCE = 200.0000 M
RADIUS = 31.8310 M
(DS/S)/(DP/P) = 0.0002174

THETX = 0.01492967 RAD
THETY = 0.00000000 RAD
TGAM = (67.82970, 0.00000)

NUX = 0.16667
NUY = 0.16667
DNUX/(DP/P) = -0.18367
DNUY/(DP/P) = -0.18370

MAXIMA --- BETX(13) = 344.94429 BETY(1) = 344.95041 XEQ(13) = 3.72756 YEQ(26) = 0.00000
MINIMA --- BETX(1) = 115.95190 BETY(13) = 115.95257 XEQ(1) = 2.24716 YEQ(26) = 0.00000

Fit Phase Trombone ...

Trombone Quadrupole Magnets

```

*** IBT IBET // 115.95190 0. 0.0086243 2.24716 0.0
* // 344.95041 0. 0.0028990 0.0 0.0
*** KAA PARA // KF
*** KBB PARA // KD
*** KCC PARA // KF
*** KDD PARA // KD
*** KEE PARA // KF

*** TRMB SUB 0 0 //
.....
*** QFA MAG // LQ KAA BR
*** QDB MAG // LQ KBB BR
*** QFC MAG // LQ KCC BR
*** QDD MAG // LQ KDD BR
*** QFE MAG // LQ KEE BR
*** TTR TRKB 0 30 // .TR IBT 0 30
*** END 0 0 //
.....

```

(NOTE: In the TRKB statement a CL could have been appropriately placed in line 1 instead of IPT. The program would have then retrieved the values of the Courant-Snyder parameters from the matrix CL.)

Specify Desired Phase Advance Through the Trombone...

```

*** MUXT = // .85
*** MUXT = // .85

```

Vary trombone parameters so as to obtain the desired phase advance. Repeat with different values of the phase advance.

(The following line may optionally be inserted anywhere in an input file to assist the programmer in finding the appropriate columns for entries)
1.....2.....3.....4.....5.....6.....7.....8

```

*** VARPH SUB 0 0 //
.....
*** SOLV 5 1 // TRMB TTR 3000 -3 2
* // AX 15 0.0 .0001
* // AY 15 0.0 .0001
* // DX 15 0.0 .0001
* // NUX 30 MUXT .001
* // NUZ 30 MUXT .001
* // KAA KBB KCC KDD KEE 1 -.004 .004 .00001
*** INS CYC -1 // .TR
*** INCR 1 // MUXT .05
*** INCR 1 // MUXT .05
*** END 0 0 //
.....

```

*** CALL 3 // VARPH

--- SOLV --- BETA-FUNCTION FITTING

INITIAL VALUES OF VARIABLES

1	KAA	1	0.20340335E-02
2	KBB	1	-0.20341561E-02
3	KCC	1	0.20340335E-02
4	KDD	1	-0.20341561E-02
5	KEE	1	0.20340335E-02

CONSTRAINT/	FUNCTION	POSITIONS OR INDICES	PRESENT VALUE	DESIRED VALUE	ERROR	TOLERANCE
1	AX	15	0.0000017	0.0000000	0.0000017	0.0001000
2	AY	15	0.0000049	0.0000000	0.0000049	0.0001000
3	DX	15	0.0000000	0.0000000	0.0000000	0.0001000
4	NUX	30	0.8333318	0.8500000	-0.0166682	0.0010000
5	NUY	30	0.8333291	0.8500000	-0.0166709	0.0010000

FCN = SUM[(ERROR/TOL.)**2]/ 5 = 0.1111502E+03

FINAL VALUES OF VARIABLES

1	KAA	1	0.20743279E-02
2	KBB	1	-0.20634133E-02
3	KCC	1	0.20558081E-02
4	KDD	1	-0.20970949E-02
5	KEE	1	0.20926992E-02

KAA	PARA	0.20743279	-2
KBE	PARA	-0.20634133	-2
KCC	PARA	0.20558081	-2
KDD	PARA	-0.20970949	-2
KEE	PARA	0.20926992	-2

CONSTRAINT/	FUNCTION	POSITIONS OR INDICES	PRESENT VALUE	DESIRED VALUE	ERROR	TOLERANCE
1	AX	15	0.0000000	0.0000000	0.0000000	0.0001000
2	AY	15	0.0000000	0.0000000	0.0000000	0.0001000
3	DX	15	0.0000002	0.0000000	0.0000002	0.0001000
4	NUX	30	0.8499987	0.8500000	-0.0000013	0.0010000
5	NUY	30	0.8500012	0.8500000	0.0000012	0.0010000

FCN = SUM[(ERROR/TOL.)**2]/ 5 = 0.1780953E-05

POS	S(M)	NUX	NUY	BETAX(M)	BETAY(M)	XEQ(M)	YEQ(M)	ZEQ(M)	ALPHAX	ALPHAY	DXEQ	DYEQ
-----	------	-----	-----	----------	----------	--------	--------	--------	--------	--------	------	------

CIRCUMFERENCE = 1000.0000 M
 RADIUS = 159.1549 M
 (DS/S)/(DP/P) = 0.0002095

THETX = 0.07464834 RAD
 THETY = 0.00000000 RAD
 TGAM=(69.09004, 0.00000)

NUX = 0.85000
 DNUX/(DP/P) = -0.94230
 NUY = 0.85001
 DNUY/(DP/P) = -0.94180

MAXIMA	---	BETX(15) =	366.40541	BETY(24) =	351.94331	XEQ(3) =	3.72725	YEQ(30) =	0.00000
MINIMA	---	BETX(24) =	109.16296	BETY(15) =	104.79072	XEQ(18) =	2.09170	YEQ(30) =	0.00000

***	INCR	1	//	MUXT	0.050000	VALUE =	0.900000
***	INCR	1	//	MUYT	0.050000	VALUE =	0.900000

SUB. VARPH, ITER. 2

--- SOLV --- BETA-FUNCTION FITTING

INITIAL VALUES OF VARIABLES

1	KAA	1	0.20743279E-02
2	KBB	1	-0.20634133E-02
3	KCC	1	0.20558081E-02

4 KDD 1 -0.20970949E-02
5 KEE 1 0.20926992E-02

CONSTRAINT/	FUNCTION	POSITIONS OR INDICES	PRESENT VALUE	DESIRED VALUE	ERROR	TOLERANCE
1	AX	15	0.0000000	0.0000000	0.0000000	0.0001000
2	AY	15	0.0000000	0.0000000	0.0000000	0.0001000
3	DX	15	0.0000002	0.0000000	0.0000002	0.0001000
4	NUX	30	0.8499987	0.9000000	-0.0500013	0.0010000
5	NUY	30	0.8500012	0.9000000	-0.0499988	0.0010000

FCN = SUM[(ERROR/TOL.)**2]/ 5 = 0.1000003E+04

FINAL VALUES OF VARIABLES

1	KAA	1	0.21714587E-02
2	KBB	1	-0.21333424E-02
3	KCC	1	0.21510513E-02
4	KDD	1	-0.23049062E-02
5	KEE	1	0.22472816E-02
KAA	PARA		0.21714587 -2
KBB	PARA		-.21333424 -2
KCC	PARA		0.21510513 -2
KDD	PARA		-.23049062 -2
KEE	PARA		0.22472816 -2

CONSTRAINT/	FUNCTION	POSITIONS OR INDICES	PRESENT VALUE	DESIRED VALUE	ERROR	TOLERANCE
1	AX	15	0.0000000	0.0000000	0.0000000	0.0001000
2	AY	15	0.0000000	0.0000000	0.0000000	0.0001000
3	DX	15	0.0000000	0.0000000	0.0000000	0.0001000
4	NUX	30	0.9000002	0.9000000	0.0000002	0.0010000
5	NUY	30	0.8999997	0.9000000	-0.0000003	0.0010000

FCN = SUM[(ERROR/TOL.)**2]/ 5 = 0.3679713E-07

POS	S(M)	NUX	NUY	BETAX(M)	BETAY(M)	XEQ(M)	YEQ(M)	ZEQ (M)	ALPHAX	ALPHAY	DXEQ	DYEQ
-----	------	-----	-----	----------	----------	--------	--------	---------	--------	--------	------	------

CIRCUMFERENCE = 1000.0000 M THETX = 0.07464834 RAD NUX = 0.90000 DNUX/(DP/P) = -1.02308
RADIUS = 159.1549 M THETY = 0.00000000 RAD NUY = 0.90000 DNUY/(DP/P) = -1.02042
(DS/S)/(DP/P) = 0.0001883 TGAM=(72.86502, 0.00000)

MAXIMA	---	BETX(15) =	414.98694	BETY(6) =	369.18844	XEQ(27) =	3.72598	YEQ(30) =	0.00000
MINIMA	---	BETX(6) =	93.93700	BETY(15) =	77.68731	XEQ(12) =	1.65997	YEQ(30) =	0.00000

***	INCR	1	//	MUXT	0.050000	VALUE =	0.950000
***	INCR	1	//	MUYT	0.050000	VALUE =	0.950000

SUB. VARPH, ITER. 3

--- SOLV --- BETA-FUNCTION FITTING

INITIAL VALUES OF VARIABLES

1	KAA	1	0.21714587E-02
2	KBB	1	-0.21333424E-02
3	KCC	1	0.21510513E-02
4	KDD	1	-0.23049062E-02
5	KEE	1	0.22472816E-02

CONSTRAINT/	FUNCTION	POSITIONS OR INDICES	PRESENT VALUE	DESIRED VALUE	ERROR	TOLERANCE
1	AX	15	0.0000000	0.0000000	0.0000000	0.0001000
2	AY	15	0.0000000	0.0000000	0.0000000	0.0001000
3	DX	15	0.0000000	0.0000000	0.0000000	0.0001000
4	NUX	30	0.9000002	0.9500000	-0.0499998	0.0010000
5	NUY	30	0.9999997	0.9500000	-0.0500003	0.0010000

FCN = SUM[(ERROR/TOL.)**2]/ 5 = 0.1000001E+04

FINAL VALUES OF VARIABLES

1	KAA	1	0.22388332E-02
2	KBB	1	-0.21539953E-02
3	KCC	1	0.22721651E-02
4	KDD	1	-0.25636852E-02
5	KEE	1	0.24011955E-02
KAA	PARA	0.22388332	-2
KBB	PARA	-0.21539953	-2
KCC	PARA	0.22721651	-2
KDD	PARA	-0.25636852	-2
KEE	PARA	0.24011955	-2

CONSTRAINT/	FUNCTION	POSITIONS OR INDICES	PRESENT VALUE	DESIRED VALUE	ERROR	TOLERANCE
1	AX	15	0.0000001	0.0000000	0.0000001	0.0001000
2	AY	15	0.0000000	0.0000000	0.0000000	0.0001000
3	DX	15	-0.0000001	0.0000000	-0.0000001	0.0001000
4	NUX	30	0.9499998	0.9500000	-0.0000002	0.0010000
5	NUY	30	0.9500000	0.9500000	0.0000000	0.0010000

FCN = SUM[(ERROR/TOL.)**2]/ 5 = 0.2246301E-06

POS	S(M)	NUX	NUY	BETAX(M)	BETAY(M)	XEQ(M)	YEQ(M)	ZEQ(M)	ALPHAX	ALPHAY	DXEQ	DYEQ
-----	------	-----	-----	----------	----------	--------	--------	--------	--------	--------	------	------

CIRCUMFERENCE =	1000.0000 M	THETX =	0.07464834 RAD	NUX =	0.95000	DNUX/(DP/P) =	-1.11066
RADIUS =	159.1549 M	THETY =	0.00000000 RAD	NUY =	0.95000	DNUY/(DP/P) =	-1.12958
(DS/S)/(DP/P) =	0.0001706	TGAM =	(76.57227, 0.00000)				

MAXIMA	---	BETX(15) =	449.99906	BETY(6) =	381.54035	XEQ(27) =	3.72501	YEQ(30) =	0.00000
MINIMA	---	BETX(6) =	84.30636	BETY(15) =	52.29574	XEQ(12) =	1.28234	YEQ(30) =	0.00000

***	INCR	1	//	MUXT	0.000000	VALUE =	0.850000
***	INCR	1	//	MUYT	0.000000	VALUE =	0.850000

CALL

// TRMB

POS	S	QX	BX	AX	X	DX	HX(CM)	QY	BY	AY	Y	DY
0	0.0000	0.00000	115.9519	0.00000	2.24716	0.00000	4.35504	0.00000	344.9504	0.00000	0.00000	0.00000
1 QD	2.5000	0.00342	117.4864	-0.61642	2.26146	0.01145	4.35504	0.00116	340.6015	1.73219	0.00000	0.00000
2 OBO	97.5000	0.08217	340.5960	-1.73206	3.70388	0.01891	4.02810	0.07992	117.4873	0.61638	0.00000	0.00000
3 QFA	100.0000	0.08333	344.5052	0.17571	3.72518	-0.00190	4.02810	0.08333	116.1020	-0.05967	0.00000	0.00000
4 QFA	102.5000	0.08450	338.8553	2.07369	3.69441	-0.02269	4.02810	0.08674	118.0896	-0.73907	0.00000	0.00000
5 OBO	197.5000	0.17842	86.0106	0.58778	1.89350	-0.01523	4.21346	0.16110	376.6833	-1.98297	0.00000	0.00000
6 QDB	200.0000	0.18310	84.3065	0.09688	1.86812	-0.00511	4.21346	0.16215	381.5425	0.04802	0.00000	0.00000
7 QDB	202.5000	0.18781	85.0330	-0.38878	1.86791	0.00494	4.21346	0.16320	376.2074	2.07643	0.00000	0.00000
8 OBO	297.5000	0.29313	281.0666	-1.67469	2.69167	0.01240	2.94911	0.24087	109.1069	0.73515	0.00000	0.00000
9 QFC	300.0000	0.29453	285.4726	-0.07937	2.70351	-0.00294	2.94911	0.24456	107.0417	0.09484	0.00000	0.00000
10 QFC	302.5000	0.29593	281.8528	1.52043	2.67701	-0.01824	2.94911	0.24827	108.1495	-0.54006	0.00000	0.00000
11 OBO	397.5000	0.39221	99.0047	0.40424	1.29885	-0.01078	2.00040	0.33379	318.5495	-1.67468	0.00000	0.00000
12 QDD	400.0000	0.39625	98.6305	-0.25377	1.28226	-0.00251	2.00040	0.33503	321.8311	0.36907	0.00000	0.00000
13 QDD	402.5000	0.40023	101.5695	-0.92813	1.28624	0.00571	2.00040	0.33627	314.8981	2.38930	0.00000	0.00000
14 OBO	497.5000	0.47411	443.2940	-2.66890	2.18278	0.01317	1.07484	0.46744	53.2049	0.36537	0.00000	0.00000
15 QFE	500.0000	0.47500	449.9999	0.00000	2.19926	0.00000	1.07484	0.47500	52.2960	0.00000	0.00000	0.00000
16 QFE	502.5000	0.47589	443.2940	2.66890	2.18278	-0.01317	1.07484	0.48256	53.2049	-0.36537	0.00000	0.00000
17 OBO	597.5000	0.54977	101.5695	0.92813	1.28623	-0.00571	2.00033	0.61373	314.8981	-2.38930	0.00000	0.00000
18 QDD	600.0000	0.55375	98.6304	0.25377	1.28224	0.00251	2.00033	0.61497	321.8311	-0.36907	0.00000	0.00000
19 QDD	602.5000	0.55779	99.0046	-0.40424	1.29883	0.01077	2.00033	0.61621	318.5495	1.67468	0.00000	0.00000
20 OBO	697.5000	0.65407	281.8528	-1.52043	2.67696	0.01824	2.94900	0.70173	108.1495	0.54006	0.00000	0.00000
21 QFC	700.0000	0.65547	285.4726	0.07937	2.70346	0.00294	2.94900	0.70544	107.0417	-0.09484	0.00000	0.00000
22 QFC	702.5000	0.65687	281.0665	1.67469	2.69162	-0.01240	2.94900	0.70913	109.1069	-0.73515	0.00000	0.00000
23 OBO	797.5000	0.76219	85.0330	0.38878	1.86788	-0.00494	4.21333	0.78680	376.2074	-2.07643	0.00000	0.00000
24 QDB	800.0000	0.76690	84.3065	-0.09688	1.86809	0.00511	4.21333	0.78785	381.5425	-0.04802	0.00000	0.00000
25 QDB	802.5000	0.77158	86.0106	-0.58778	1.89348	0.01523	4.21333	0.78890	376.6833	1.98297	0.00000	0.00000
26 OBO	897.5000	0.86550	338.8553	-2.07369	3.69437	0.02269	4.02801	0.86326	118.0896	0.73907	0.00000	0.00000
27 QFA	900.0000	0.86667	344.5052	-0.17571	3.72514	0.00190	4.02801	0.86667	116.1020	0.05967	0.00000	0.00000
28 QFA	902.5000	0.86782	340.5961	1.73206	3.70385	-0.01891	4.02801	0.87008	117.4873	-0.61638	0.00000	0.00000
29 OBO	997.5000	0.94658	117.4864	0.61642	2.26145	-0.01145	4.35502	0.94884	340.6015	-1.73219	0.00000	0.00000
30 QD	1000.0000	0.95000	115.9519	0.00000	2.24715	0.00000	4.35502	0.95000	344.9504	0.00000	0.00000	0.00000

=====

END OF SYNCH RUN TRMB

MUXT = .85
 MUYP = .85

Vary trombone parameters so as to obtain the desired phase advance. Repeat with different values of the phase advance.

(The following line may optionally be inserted anywhere in an input file to assist the programmer in finding the appropriate columns for entries)

.....1.....2.....3.....4.....5.....6.....7.....8

```
VARPH SUB
  SOI,V  5  1  TRMB TTR          3000  -3          2
          AX      15          0.0      .0001
          AY      15          0.0      .0001
          DX      15          0.0      .0001
          NUX     30          MUXT     .001
          NUY     30          MUYP     .001
          KAA KBB KCC KDD KEE  1 -.004      .004      .00001
INS  CYC  -1  .TR
     INCR  1  MUXT      .05
     INCR  1  MUYP      .05
     END
     CALL  3  VARPH
     PAGE
     CALL  TRMB
     STOP
```

B.3 Closed Orbit Calculations

Here, the closed orbit in a synchrotron due to a single misaligned quadrupole magnet is computed using two different methods. In the first part of the example, the misaligned magnet is defined using a MAGS command within the BMIS/EMIS environment and the closed orbit is computed from a CYC command. In the second part of the example, a MOVE command is used to define the misaligned element and the new closed orbit is sought using a FXPT command. Several other features are exhibited in this example, including the use of a subroutine, the use of parentheses in defining a beamline, and the printing of matrix elements.

```

MSG  RUN
C
C      Calculation of Closed Orbit for a Synchrotron with
C      a Misaligned Element.
C
C
C      The calculation will be performed using two different
C      methods -- 1) using BMIS and CYC
C      2) using MOVE and FXPT
C
C
C-----
C
C      Lengths in meters, field strengths in kG, kG/m, etc.
C
C      Magnetic Rigidity at 1 TeV ...
C
C      BRHO = 33387.702
C
C      Bend field strength ...
C
C      BY = 44.622553
C
C      Quadrupole gradients ...
C
C      GF = 760.32056
C      GD = -760.32056
C
C      Magnet lengths ...
C
C      BL = 6.1214
C      QL = 1.67894
C
C      Magnet definitions ...
C
C      Magnet Definitions
C
C      MDEFN SUB
C      B MAG BL 0.0 BRHO BY $
C      QF MAG QL GF BRHO
C      QD MAG QL GD BRHO
C      END
C
C      Drift Definitions
C
C      0 DRF 0.2794
C      00 DRF 2.29616
C      000 DRF 0.4445
C
C      Standard Cell Beamline Definition
C
C      HC BML 00 B 0 B 0 B 0 B 000
C
C      CELL BML HC QF HC QD
C
C
C      ----- METHOD 1:

```



```

C
C      Begin the Magnet Misalignment Mode...
C
C      BMIS
C
C      CALL 1      MODFN
C
C      ARC  MMM      76( CELL      )
C
C
C      Define half-cell matrices in order to print out betatron functions
C      only at the ends of the quadrupoles...
C
C      MHC  MMM      HC
C      MHCF MMM      HC  QF
C      MHCD MMM      HC  QD
C
C      Define a new "beamline" made up of the above two matrices.
C
C      BML      MHCF MHCD
C
C
C      Define the misaligned element and a corresponding
C      "misaligned" standard cell
C
C      MSQ  MAGS  2      QF      0.0      0.0      -.010      -.010      0.0
C
C      MSC  BML      MHC  MSQ  MHCD
C
C      Define a ring made up of the above standard cells with one
C      quadrupole being misaligned.
C
C      RNG  BML      10( C      ) MSC  9( C      ) ARC
C
C
C      Calculate the Closed Orbit using CYC.      (In BMIS mode, the
C      closed orbit will appear in the columns marked XEQ, YEQ. in
C      the CYC output.)      Note that in this CYC output, each line
C      will correspond to the end of a quadrupole. Also, the "tunes"
C      at the end of the CYC output will be incorrect because the
C      phase advance through the matrix ARC is greater than 2pi.
C      See Section IV -- CYC.
C
C      CYC      RNG
C
C
C      EMIS

```

```

P
C          ----- METHOD 2:
C
C
C      CALL    1      MDEFN
C
C      MHC      MMM      HC
C      MHCF      MMM      HC      QF
C      MHCD      MMM      HC      QD
C      ARC      MMM      76( CELL      )
C      MSC      BML      MHC      QFM      MHCD
C      RNG      BML      10( C      ) MSC      9( C      ) ARC
C
C      Re-define a misaligned quadrupole magnet using the MOVE command...
C
C
C      UC      UEC      6      dx      dxp      dy      dyp      ds      dphi
C      QFM      MOVE      QF      UC      -0.010      0.0      0.0      0.0
C
C      Look at the magnet matrices ...
C
C      UMA      2      QF      QFM
C
C
C      Now the closed orbit may be calculated using FXPT...
C
C      Provide initial guess of closed orbit for the FXPT routine:
C
C      PU      PUEC      ( x ,      x' ,      y ,      y' ,      ds ,      dp/p )
C      PU      PUEC      0.      0.      0.      0.      0.      0.
P
S      FXPT      2      1 PU      RNG      1      0      10      0
C
C      STOP

```

SYNCH RUN MSG
16-JAN-93 09:59:17

SYNCH VERSION

=====

Calculation of Closed Orbit for a Synchrotron with
a Misaligned Element.

The calculation will be performed using two different
methods -- 1) using BMIS and CYC
2) using MOVE and FXPT

Lengths in meters, field strengths in kG, kG/m, etc.

Magnetic Rigidity at 1 TeV ...

*** BRIIO = // 33387.702

Bend field strength ...

*** BY = // 44.622553

Quadrupole gradients ...

*** GF = // 760.32056

*** GD = // -760.32056

Magnet lengths ...

*** BL = // 6.1214

*** QL = // 1.67894

Magnet definitions ...

Magnet Definitions

*** MDFN SUB 0 0 //

=====

*** B	MAG	//	BL	0.0	BRHO	BY	\$
*** QF	MAG	//	QL	GF	BRHO		
*** QD	MAG	//	QL	GD	BRHO		
*** END			0	0	//		

=====

Drift Definitions

*** O DRF // 0.2794

*** OO DRF // 2.29616

*** OOO DRF // 0.4445

Standard Cell Beamline Definition

*** HC BML // OO B O B O B O B OOO

*** CELL BML // HC QF HC QD

----- METHOD 1:

begin the Magnet Misalignment Mode...

*** BMIS

*** CALL 1 // MDFN

*** ARC MMM // 76(CELL)

Define half-cell matrices in order to print out betatron functions
only at the ends of the quadrupoles...

*** MHC MMM // HC
*** MHCF MMM // HC QF
*** MHCD MMM // HC QD

Define a new "beamline" made up of the above two matrices.

*** C BML // MHCF MHCD

Define the misaligned element and a corresponding
"misaligned" standard cell

*** MSQ MAGS 2 // QF 0.0 0.0 -.010 -.010 0.0

*** MSC BML // MHC MSQ MHCD

Define a ring made up of the above standard cells with one
quadrupole being misaligned.

*** RNG BML // 10(C) MSC 9(C) ARC

Calculate the Closed Orbit using CYC. (In BMIS mode, the
closed orbit will appear in the columns marked XEQ, YEQ. in
the CYC output.) Note that in this CYC output, each line
will correspond to the end of a quadrupole. Also, the "tunes"
at the end of the CYC output will be incorrect because the
phase advance through the matrix ARC is greater than 2pi.
See Section IV -- CYC.

*** CYC // RNG

POS	S(M)	NUX	NUY	BETAX(M)	BETAY(M)	XEQ(M)	YEQ(M)	ZEQ (M)	ALPHAX	ALPHAY	DXEQ	DYEQ
0	0.0000	0.00000	0.00000	29.04507	97.80157	0.00000	0.02419	0.0000	-0.58154	1.87136	0.00000	-0.00060
1 MHCF	29.7434	0.09087	0.09769	97.97246	28.97421	0.00000	0.00663	0.0003	1.87453	-0.58022	0.00000	-0.00033
2 MHCD	59.4868	0.18835	0.18875	29.04507	97.80157	0.00000	-0.00313	0.0005	-0.58154	1.87136	0.00000	-0.00022
3 MHCF	89.2302	0.27922	0.28643	97.97246	28.97421	0.00000	-0.00997	0.0008	1.87453	-0.58022	0.00000	-0.00059
4 MHCD	118.9736	0.37669	0.37749	29.04507	97.80157	0.00000	-0.02654	0.0011	-0.58154	1.87136	0.00000	0.00043
5 MHCF	148.7170	0.46756	0.47518	97.97246	28.97421	0.00000	-0.01412	0.0014	1.87453	-0.58022	0.00000	-0.00011
6 MHCD	178.4604	0.56504	0.56624	29.04507	97.80157	0.00000	-0.01680	0.0016	-0.58154	1.87136	0.00000	0.00054
7 MHCF	208.2038	0.65591	0.66393	97.97246	28.97421	0.00000	-0.00063	0.0019	1.87453	-0.58022	0.00000	0.00050
8 MHCD	237.9472	0.75339	0.75499	29.04507	97.80157	0.00000	0.01393	0.0022	-0.58154	1.87136	0.00000	-0.00002
9 MHCF	267.6906	0.84426	0.85267	97.97246	28.97421	0.00000	0.01365	0.0025	1.87453	-0.58022	0.00000	0.00049
10 MHCD	297.4340	0.94173	0.94374	29.04507	97.80157	0.00000	0.02726	0.0027	-0.58154	1.87136	0.00000	-0.00056
11 MHCF	327.1774	1.03260	1.04142	97.97246	28.97421	0.00000	0.01088	0.0030	1.87453	-0.58022	0.00000	-0.00014
12 MHCD	356.9208	1.13008	1.13248	29.04507	97.80157	0.00000	0.00654	0.0033	-0.58154	1.87136	0.00000	-0.00040
13 MHCF	386.6642	1.22095	1.23017	97.97246	28.97421	0.00000	-0.00548	0.0036	1.87453	-0.58022	0.00000	-0.00059
14 MHCD	416.4076	1.31843	1.32123	29.04507	97.80157	0.00000	-0.02235	0.0038	-0.58154	1.87136	0.00000	0.00026

15	MHCF	446.1510	1.40930	1.41891	97.97246	28.97421	0.00000	-0.01499	0.0041	1.87453	-0.58022	0.00000	-0.00031
16	MHCD	475.8944	1.50677	1.50998	29.04507	97.80157	0.00000	-0.02332	0.0044	-0.58154	1.87136	0.00000	0.00060
17	MHCF	505.6378	1.59764	1.60766	97.97246	28.97421	0.00000	-0.00578	0.0046	1.87453	-0.58022	0.00000	0.00036
18	MHCD	535.3812	1.69512	1.69872	29.04507	97.80157	0.00000	0.00484	0.0049	-0.58154	1.87136	0.00000	0.00018
19	MHCF	565.1246	1.78599	1.79641	97.97246	28.97421	0.00000	0.01065	0.0052	1.87453	-0.58022	0.00000	0.00058
20	MHCD	594.8680	1.88347	1.88747	29.04507	97.80157	0.00000	0.02695	0.0055	-0.58154	1.87136	0.00000	-0.00046
21	MHC	622.9325	1.97164	1.97583	97.97246	28.97434	0.00000	0.01410	0.0057	-1.87453	0.58030	0.00000	-0.00046
22	MSQ	624.6114	1.97434	1.98516	97.97246	28.97421	0.00000	0.01410	0.0057	1.87453	-0.58022	0.00000	0.00046
23	MHCD	654.3548	2.07181	2.07622	29.04507	97.80157	0.00000	0.02685	0.0060	-0.58154	1.87136	0.00000	-0.00058
24	MHCF	684.0982	2.16269	2.17390	97.97246	28.97421	0.00000	0.01002	0.0063	1.87453	-0.58022	0.00000	-0.00018
25	MHCD	713.8416	2.26016	2.26496	29.04507	97.80157	0.00000	0.00438	0.0066	-0.58154	1.87136	0.00000	-0.00036
26	MHCF	743.5850	2.35103	2.36265	97.97246	28.97421	0.00000	-0.00658	0.0068	1.87453	-0.58022	0.00000	-0.00060
27	MHCD	773.3284	2.44851	2.45371	29.04507	97.80157	0.00000	-0.02357	0.0071	-0.58154	1.87136	0.00000	0.00031
28	MHCF	803.0718	2.53938	2.55140	97.97246	28.97421	0.00000	-0.01496	0.0074	1.87453	-0.58022	0.00000	-0.00026
29	MHCD	832.8152	2.63685	2.64246	29.04507	97.80157	0.00000	-0.02207	0.0077	-0.58154	1.87136	0.00000	0.00059
30	MHCF	862.5586	2.72773	2.74014	97.97246	28.97421	0.00000	-0.00465	0.0079	1.87453	-0.58022	0.00000	0.00040
31	MHCD	892.3020	2.82520	2.83121	29.04507	97.80157	0.00000	0.00699	0.0082	-0.58154	1.87136	0.00000	0.00014
32	MHCF	922.0454	2.91607	2.92889	97.97246	28.97421	0.00000	0.01146	0.0085	1.87453	-0.58022	0.00000	0.00056
33	MHCD	951.7888	3.01355	3.01995	29.04507	97.80157	0.00000	0.02732	0.0087	-0.58154	1.87136	0.00000	-0.00049
34	MHCF	981.5322	3.10442	3.11764	97.97246	28.97421	0.00000	0.01326	0.0090	1.87453	-0.58022	0.00000	0.00002
35	MHCD	1011.2756	3.20190	3.20870	29.04507	97.80157	0.00000	0.01352	0.0093	-0.58154	1.87136	0.00000	-0.00050
36	MHCF	1041.0190	3.29277	3.30638	97.97246	28.97421	0.00000	-0.00151	0.0096	1.87453	-0.58022	0.00000	-0.00055
37	MHCD	1070.7624	3.39024	3.39745	29.04507	97.80157	0.00000	-0.01717	0.0098	-0.58154	1.87136	0.00000	0.00011
38	MHCF	1100.5058	3.48111	3.49513	97.97246	28.97421	0.00000	-0.01439	0.0101	1.87453	-0.58022	0.00000	-0.00043
39	MHCD	1130.2492	3.57859	3.58619	29.04507	97.80157	0.00000	-0.02641	0.0104	-0.58154	1.87136	0.00000	0.00059
40	MHCF	1159.9926	3.66946	3.68388	97.97246	28.97421	0.00000	-0.00930	0.0107	1.87453	-0.58022	0.00000	0.00022
41	MHCD	1189.7360	3.76694	3.77494	29.04507	97.80157	0.00000	-0.00266	0.0109	-0.58154	1.87136	0.00000	0.00033
42	ARC	5710.7328	4.08129	4.11972	29.04507	97.80157	0.00000	0.02419	0.0525	-0.58154	1.87136	0.00000	-0.00060

CIRCUMFERENCE = 5710.7328 M
RADIUS = 908.8914 M
(DS/S)/(DP/P) = 0.0000092

THETX = 6.28318525 RAD
THETY = 0.00000000 RAD

NUX = 4.08129
NUY = 4.11972

DNUX/(DP/P) = 0.00000
DNUY/(DP/P) = 0.00000

MAXIMA	---	BETX(38) =	97.97246	BETY(42) =	97.80157	XEQ(42) =	0.00000	YEQ(33) =	0.02732
MINIMA	---	BETX(8) =	29.04507	BETY(15) =	28.97421	XEQ(42) =	0.00000	YEQ(4) =	-0.02654

*** EMIS

----- METHOD 2:

*** CALL 1 // MDFN

*** MHC MMM // HC
 *** MHCF MMM // HC QF
 *** MHCD MMM // HC QD
 *** ARC MMM // 76(CELL)
 *** MSC BML // MIIC QFM MHCD
 *** RNG BML // 10(C) MSC 9(C) ARC

Re-define a misaligned quadrupole magnet using the MOVE command...

*** VC VEC 6 dx dxp dy dyp ds dphi 0.0
 *** QFM MOVE // QF VC

Look at the magnet matrices ...

*** WMA 2 // QF QFM

TRANSFER MATRICES

				R(I,J)			
ELEMENT	X	DX/DS	Y	DY/DS	-DS	DP/P	1
QF	0.96807535	1.66103516	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
	-0.03782588	0.96807535	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
	0.00000000	0.00000000	1.03226803	1.69696015	0.00000000	0.00000000	0.00000000
	0.00000000	0.00000000	0.03864398	1.03226803	0.00000000	0.00000000	0.00000000
	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000	0.00000000
	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000
	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000
LENGTH =		1.67894000	THETA =		0.00000000		
QFM	0.96807535	1.66103516	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
	-0.03782588	0.96807535	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
	0.00000000	0.00000000	1.03226803	1.69696015	0.00000000	0.00000000	0.00032268
	0.00000000	0.00000000	0.03864398	1.03226803	0.00000000	0.00000000	0.00038644
	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000	0.00000000
	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000
	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000
LENGTH =		1.67894000	THETA =		0.00000000		

Now the closed orbit may be calculated using FXPT...

Provide initial guess of closed orbit for the FXPT routine:

*** PV PVEC (x , // 0. x' , 0. y , 0. y' , 0. ds , 0. dp/p) 0.

*** S FXPT 2 1 // PV RNG 1 0 10 0

CALCULATION OF THE EQUILIBRIUM ORBIT AND BETATRON FUNCTIONS OF S

INITIAL REFERENCE RAY DEFINED BY PV

X = 0.00000000 DX = 0.00000000 Y = 0.00000000 DY = 0.00000000 DS = 0.00000000 DP/P = 0.00000000 1.00000000
 ITERATION = 1 XO= 0.00000000 DXO= 0.00000000 YO= 0.02419259 DYO= -0.00059762
 ITERATION = 2 XO= 0.00000000 DXO= 0.00000000 YO= 0.02419259 DYO= -0.00059762
 CLOSED ORBIT FOUND AFTER 1 ITERATIONS.

7X7 MATRIX FOR S

0.58805296	14.19878280	0.00000000	0.00000000	0.00000000	0.31012944	0.00000000
-0.02252462	1.15666084	0.00000000	0.00000000	0.00000000	0.04450951	0.00000000
0.00000000	0.00000000	2.00878983	66.82241613	0.00000000	-0.03722769	0.01552896
0.00000000	0.00000000	-0.03145116	-0.54841106	0.00000000	0.00039429	-0.00016447
-0.03315950	-0.27326629	0.00037881	-0.00593122	1.00000000	-18.97781377	0.00000926
0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000	0.00000000
0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000	1.00000000

EIGENVALUES OF THE 4X4 SUBMATRIX

X... LMD1 = (0.87235690 0.48886955), C(1) = 1.00000000, MU(1) = 0.51079342 RAD, Q(1) = 0.08129530
 1/LMD1 = (0.87235690 -0.48886955), C(2) = 1.00000000, MU(2) = -0.51079342 RAD, Q(2) = 0.91870470
 Y... LMD3 = (0.73018939 0.68324480), C(3) = 1.00000000, MU(3) = 0.75219723 RAD, Q(3) = 0.11971591
 1/LMD3 = (0.73018939 -0.68324480), C(4) = 1.00000000, MU(4) = -0.75219723 RAD, Q(4) = 0.88028409

EIGENVALUE = (0.87235690, 0.48886955), EIGENVECTOR = (5.38925913, 0.00000000)
 (0.10790979, 0.18555426)
 (0.00000000, 0.00000000)
 (0.00000000, 0.00000000)

EIGENVALUE = (0.87235690, -0.48886955), EIGENVECTOR = (5.38925913, 0.00000000)
 (0.10790979, -0.18555426)
 (0.00000000, 0.00000000)
 (0.00000000, 0.00000000)

EIGENVALUE = (0.73018939, 0.68324480), EIGENVECTOR = (0.00000000, 0.00000000)
 (0.00000000, 0.00000000)
 (9.88946778, 0.00000000)
 (-0.18922809, 0.10111768)

EIGENVALUE = (0.73018939, -0.68324480), EIGENVECTOR = (0.00000000, 0.00000000)
 (0.00000000, 0.00000000)
 (9.88946778, 0.00000000)
 (-0.18922809, -0.10111768)

	X	DX	Y	DY	DS	DP/P
EQ ORBIT	0.00000000	0.00000000	0.02419259	-0.00059762	0.00000000	0.00000000 1.00000000
ET ORBIT	2.28526160	0.04445996	-0.05799709	0.00143267	0.00000000	1.00000000 0.00000000

EIGENVECTORS 1 AND 3 IN POLAR COORDINATES

POS	X1 X3		DX1 DX3		Y1 Y3		DY1 DY3	
0	5.389259	0.000000	0.214651	1.044051	0.000000	0.000000	0.000000	0.000000
	0.000000	0.000000	0.000000	0.000000	9.889468	0.000000	0.214551	2.650829

BETATRON FUNCTIONS THRU RNC

POS	S (M)	QX	QY	BX (M)	BY (M)	AX	AY	Cpl ang (deg)	EX (M)	EXP	EY (M)	EYP	XCO (mm)	DXCO (mr)	YCO (mm)	DYCO (mr)
0	0.00	0.00	0.00	29.0441	97.8016	-0.58	1.87	0.0000	2.2853	0.0445	-0.058	0.001	0.000	0.000	24.193	-0.598
1MHCF	29.74	0.09	0.10	97.9745	28.9742	1.87	-0.58	0.0000	3.9636	-0.0751	-0.016	0.001	0.000	0.000	6.634	-0.331
2MHCD	59.49	0.19	0.19	29.0465	97.8016	-0.58	1.87	0.0000	2.2855	0.0445	0.007	0.001	0.000	0.000	-3.128	-0.220
3MHCF	89.23	0.28	0.29	97.9742	28.9742	1.87	-0.58	0.0000	3.9641	-0.0751	0.024	0.001	0.000	0.000	-9.972	-0.586
4MHCD	118.97	0.38	0.38	29.0440	97.8016	-0.58	1.87	0.0000	2.2856	0.0445	0.064	-0.001	0.000	0.000	-26.541	0.432
5MHCF	148.72	0.47	0.48	97.9680	28.9742	1.87	-0.58	0.0000	3.9642	-0.0752	0.034	0.000	0.000	0.000	-14.122	-0.109
6MHCD	178.46	0.57	0.57	29.0452	97.8016	-0.58	1.87	0.0000	2.2856	0.0445	0.040	-0.001	0.000	0.000	-16.801	0.545
7MHCF	208.20	0.66	0.66	97.9771	28.9742	1.87	-0.58	0.0000	3.9639	-0.0752	0.002	-0.001	0.000	0.000	-0.632	0.504
8MHCD	237.95	0.75	0.75	29.0460	97.8016	-0.58	1.87	0.0000	2.2854	0.0445	-0.033	0.000	0.000	0.000	13.926	-0.023
9MHCF	267.69	0.84	0.85	97.9703	28.9742	1.87	-0.58	0.0000	3.9635	-0.0751	-0.033	-0.001	0.000	0.000	13.647	0.488
10MHCD	297.43	0.94	0.94	29.0436	97.8016	-0.58	1.87	0.0000	2.2852	0.0445	-0.065	0.001	0.000	0.000	27.258	-0.562
11MHCF	327.18	1.03	1.04	97.9710	28.9742	1.87	-0.58	0.0000	3.9635	-0.0751	-0.026	0.000	0.000	0.000	10.879	-0.138
12MHCD	356.92	1.13	1.13	29.0462	97.8016	-0.58	1.87	0.0000	2.2854	0.0445	-0.016	0.001	0.000	0.000	6.540	-0.399
13MHCF	386.66	1.22	1.23	97.9768	28.9742	1.87	-0.58	0.0000	3.9639	-0.0751	0.013	0.001	0.000	0.000	-5.479	-0.591
14MHCD	416.41	1.32	1.32	29.0449	97.8016	-0.58	1.87	0.0000	2.2856	0.0445	0.054	-0.001	0.000	0.000	-22.347	0.263
15MHCF	446.15	1.41	1.42	97.9678	28.9742	1.87	-0.58	0.0000	3.9642	-0.0752	0.036	0.001	0.000	0.000	-14.993	-0.306
16MHCD	475.89	1.51	1.51	29.0442	97.8016	-0.58	1.87	0.0000	2.2856	0.0445	0.056	-0.001	0.000	0.000	-23.320	0.596
17MHCF	505.64	1.60	1.61	97.9748	28.9742	1.87	-0.58	0.0000	3.9640	-0.0752	0.014	-0.001	0.000	0.000	-5.779	0.362
18MHCD	535.38	1.70	1.70	29.0465	97.8016	-0.58	1.87	0.0000	2.2854	0.0445	-0.012	0.000	0.000	0.000	4.837	0.185
19MHCF	565.12	1.79	1.80	97.9738	28.9742	1.87	-0.58	0.0000	3.9636	-0.0751	-0.026	-0.001	0.000	0.000	10.654	0.577
20MHCD	594.87	1.88	1.89	29.0439	97.8016	-0.58	1.87	0.0000	2.2853	0.0445	-0.065	0.001	0.000	0.000	26.951	-0.457
21MHCF	622.93	1.97	1.98	97.9681	28.9743	-1.87	0.58	0.0000	3.9617	0.0772	-0.034	0.001	0.000	0.000	14.101	-0.458
22QFM	624.61	1.97	1.99	97.9681	28.9742	1.87	-0.58	0.0000	3.9635	-0.0751	-0.034	-0.001	0.000	0.000	14.101	0.458
23MHCD	654.35	2.07	2.08	29.0440	97.8016	-0.58	1.87	0.0000	2.2853	0.0445	-0.064	0.001	0.000	0.000	26.851	-0.577
24MHCF	684.10	2.16	2.17	97.9739	28.9742	1.87	-0.58	0.0000	3.9636	-0.0751	-0.024	0.000	0.000	0.000	10.017	-0.184
25MHCD	713.84	2.26	2.26	29.0465	97.8016	-0.58	1.87	0.0000	2.2854	0.0445	-0.010	0.001	0.000	0.000	4.375	-0.362
26MHCF	743.58	2.35	2.36	97.9747	28.9742	1.87	-0.58	0.0000	3.9640	-0.0751	0.016	0.001	0.000	0.000	-6.579	-0.597
27MHCD	773.33	2.45	2.45	29.0441	97.8016	-0.58	1.87	0.0000	2.2856	0.0445	0.056	-0.001	0.000	0.000	-23.565	0.305
28MHCF	803.07	2.54	2.55	97.9678	28.9742	1.87	-0.58	0.0000	3.9642	-0.0752	0.036	0.001	0.000	0.000	-14.958	-0.264
29MHCD	832.82	2.64	2.64	29.0450	97.8016	-0.58	1.87	0.0000	2.2856	0.0445	0.053	-0.001	0.000	0.000	-22.070	0.591
30MHCF	862.56	2.73	2.74	97.9769	28.9742	1.87	-0.58	0.0000	3.9639	-0.0752	0.011	-0.001	0.000	0.000	-4.652	0.399
31MHCD	892.30	2.83	2.83	29.0461	97.8016	-0.58	1.87	0.0000	2.2854	0.0445	-0.017	0.000	0.000	0.000	6.994	0.139
32MHCF	922.05	2.92	2.93	97.9708	28.9742	1.87	-0.58	0.0000	3.9635	-0.0751	-0.027	-0.001	0.000	0.000	11.465	0.563
33MHCD	951.79	3.01	3.02	29.0436	97.8016	-0.58	1.87	0.0000	2.2852	0.0445	-0.065	0.001	0.000	0.000	27.321	-0.487
34MHCF	981.53	3.10	3.12	97.9704	28.9742	1.87	-0.58	0.0000	3.9635	-0.0751	-0.032	0.000	0.000	0.000	13.260	0.024
35MHCD	1011.28	3.20	3.21	29.0461	97.8016	-0.58	1.87	0.0000	2.2854	0.0445	-0.032	0.001	0.000	0.000	13.521	-0.504
36MHCF	1041.02	3.29	3.31	97.9770	28.9742	1.87	-0.58	0.0000	3.9639	-0.0751	0.004	0.001	0.000	0.000	-1.508	-0.545
37MHCD	1070.76	3.39	3.40	29.0450	97.8016	-0.58	1.87	0.0000	2.2856	0.0445	0.041	0.000	0.000	0.000	-17.169	0.108
38MHCF	1100.51	3.48	3.50	97.9680	28.9742	1.87	-0.58	0.0000	3.9642	-0.0752	0.035	0.001	0.000	0.000	-14.393	-0.433
39MHCD	1130.25	3.58	3.59	29.0441	97.8016	-0.58	1.87	0.0000	2.2856	0.0445	0.063	-0.001	0.000	0.000	-26.412	0.585
40MHCF	1159.99	3.67	3.68	97.9743	28.9742	1.87	-0.58	0.0000	3.9640	-0.0752	0.022	-0.001	0.000	0.000	-9.299	0.220
41MHCD	1189.74	3.77	3.77	29.0465	97.8016	-0.58	1.87	0.0000	2.2854	0.0445	0.006	-0.001	0.000	0.000	-2.663	0.331
42ARC	5710.73	4.08	4.12	29.0441	97.8016	-0.58	1.87	0.0000	2.2853	0.0445	-0.058	0.001	0.000	0.000	24.193	-0.598

CIRCUMFERI = 5710.7328 M

THETX = 6.28318525 RAD

= 4.08130

DNUX/(DP/P) = -0.30128

RADIUS = 908.8914 M
(DS/S)/(DP/P) = 0.0033386

THETY = 0.00000000 RAD
TGAM=(17.30688, 0.00000)

NUY = 4.11972

DNUY/(DP/P) = 0.08717

MAXIMA --- BETX(7) = 97.97710 BETY(33) = 97.80158 XEQ(15) = 3.96421 YEQ(4) = 0.06363
MINIMA --- BETX(33) = 29.04362 BETY(7) = 28.97420 XEQ(33) = 2.28524 YEQ(33) = -0.06550
MAXIMA XCO(42) = 0.00000 YCO(33) = 27.32137
MINIMA XCO(42) = 0.00000 YCO(4) = -26.54149

.....

=====

END OF SYNCH RUN MSMG

DATE

FILMED

4 / 20 / 94

END