

Introduction to COSY INFINITY

1 COSY INFINITY

- is the first arbitrary order beam physics code. It is a DA (Differential Algebra) based, new generation code for the study and design of beam physics systems: Accelerators, spectrometers, beamlines, electron microscopes, and glass optical systems.
- has an object oriented language environment for use of DA (and also graphics). It is a recursive language similar to PASCAL, and has a simple yet powerful syntax.
- runs on many platforms and supports many graphics drivers.

Tips: The information on installation can be available from the web page <http://cosy.pa.msu.edu> and from the manual.

2 Basic Structure of a COSY Program

2.1 Program Segments

A complete COSY program consists of a tree-structured arrangement of nested program segments. There are three types of program segments.

MAIN Program There has to be one main program in a complete COSY program. The main program begins in the beginning and ends at the end of the whole program.

Main Program
BEGIN ;
...
END ;

Procedure Program and Function Program A COSY program can contain many procedures and functions which can be called by the main program and the other procedures or functions. A procedure program and a function program must contain at least one executable statement.

Procedure Program
PROCEDURE name { name1 ... } ;
...
ENDPROCEDURE ;

Function Program
FUNCTION name name1 { ... } ;
...
ENDFUNCTION ;

Note: { } indicates an optional expression.

name: The name of the procedure or the function.

name1 ...: The local name(s) of variable(s) that are passed into the procedure or into the function. These variables are not declared in the procedure or in the function.

A call to a procedure program
name { name1 ... } ;

A call to a function program
name (name1 {, ... })

name: The name of the procedure or the function.

name1 ...: The argument(s) that are passed into the procedure or into the function.

- The number of arguments in the procedure program or in the function program has to agree with the number of arguments in its calling statements.
- A call to a function program can be made in an arithmetic expression.

Example

1. A call to the procedure DL.

DL .1 ;

This is a drift of length .1 m.

2. A call to the function ME.

ME(1,2)

This is the (x, a) element of the map.

2.2 Three Sections inside each Program Segment

Inside each program segment, there are three sections.

1. Declaration of Local Variables The types of variables are free at the declaration time. There is no distinction among integer, real and double precision numbers. All locally declared variables are visible inside the program segment.

Variable Declaration
VARIABLE name exp { exp1 ... } ;

name: The name of the variable to be declared.

exp: The amount of memory to be allocated to the variable.

exp1 ...: In case of an array with indices, it specifies the different dimension.

Example

1. A real number variable X.

```
VARIABLE X 1 ;
```

2. A 5×7 array Y of memory length 100 per array element.

```
VARIABLE Y 100 5 7 ;
```

2. Local Procedures and Functions Any local procedures and local functions are coded inside the program segment. Any local program is visible in the segment, as long as a call statement to it is made below the local program.

3. Executable Code Executable statements are assignment statements, call statements to procedures, flow control statements, input/output statements.

Assignment Statement
variable := expression ;

variable : The name of a variable or an array element.

expression : A combination of variables and array elements visible in the segment, combined with operands and grouped by parentheses.

Example

1. An assignment of .5 to a variable Q1.

```
Q1 := .5 ;
```

2. An assignment of the summation of the absolute values of (x, a) and (y, b) element of the map to a variable OBJ.

```
OBJ := ABS(ME(1,2))+ABS(ME(3,4)) ;
```

3 Input/Output

The basic Input/Output statements are as follows.

READ statement
READ unit name ;

unit: The device unit number. 5 denotes the keyboard.

name: The name of the variable to be input.

WRITE statement
WRITE unit name ;

unit: The device unit number. 6 denotes the display.

name: The name of the variable or the string to be output.

A PM statement prints the map.

map printing statement
PM unit ;

unit: The device unit number.

4 How to use COSY INFINITY in Beam Physics Studies

There is a COSY code `cosy.fox`, which contains many procedures and functions to study beam physics. It forms a portion of a complete COSY program. To access those procedures and functions, a user has to include `cosy.fox` into the user's own COSY code. Since `cosy.fox` starts with the "BEGIN ;" statement, the user code has to have the executable code for the main program and the "END ;" statement to complete the whole COSY program.

To include COSY into the user's code, an include statement has to be made in the beginning.

Include Statement
INCLUDE 'name' ;

name: The name of a previously compiled code to be included.

Example

1. Include compiled version of cosy.fox.

```
INCLUDE 'COSY' ;
```

2. A user's COSY code may look as follows.

<pre>INCLUDE 'COSY' ; PROCEDURE RUN ; ... ENDPROCEDURE ; RUN ; END ;</pre>
--

Tips: Refer to the manual about the accessible procedures and functions in cosy.fox.

5 Example: Sequence of Elements

```
INCLUDE 'COSY' ;
PROCEDURE RUN ;
  OV 5 2 0 ; {order 5, phase space dim 2, # of parameters 0}
  RP 10 4 2 ; {kinetic energy 10MeV, mass 4 amu, charge 2}
  UM ; {sets map to unity}
  DL .1 ; {drift of length .1 m}
  MQ .2 .1 .05 ; {focusing quad; length .2 m, field .1 T, aperture .05 m}
  DL .1 ;
  MQ .2 -.1 .05 ; {defocusing}
  DL .1 ;
  PM 6 ; {prints map to display}
ENDPROCEDURE ;
RUN ; END ;
```

The first few lines of the resulting transfer map look like this:

```
.7084      -.1798      .0000E+0  .0000E+0  .0000E+0  10000
.6952      1.234      .0000E+0  .0000E+0  .0000E+0  01000
.0000E+0  .0000E+0  1.234     -.1798     .0000E+0  00100
.0000E+0  .0000E+0  .6952     .7084     .0000E+0  00010
-.7552E-1 -.5173E-1  .0000E+0  .0000E+0  .0000E+0  30000
.2751      .1728      .0000E+0  .0000E+0  .0000E+0  21000
-.4105     -.2057     .0000E+0  .0000E+0  .0000E+0  12000
.3541      .8137E-1  .0000E+0  .0000E+0  .0000E+0  03000
.0000E+0  .0000E+0  .5676E-1 -.5150E-1  .0000E+0  20100
```

The different columns correspond to the final coordinates x , a , y , b and t . The lines contain the various expansion coefficients, which are identified by the exponents of the initial condition. For example, the last entry in the third column is the expansion coefficient (y, xxy) .

6 Tips

- A comment in the COSY language can be written inside a pair of curly brackets.

Example

```
{This is a comment in COSY.}
```

- Any user executable code for a beam physics calculation should start with “OV”, then “RP” (or “RPP” or “RPE”), then “UM”. A definition of elements like “DL”, “DI”, “MQ” ... follows afterward.
- demo.fox includes many example calculations with COSY INFINITY. It is a good starting point to refer to demo.fox to find some COSY example programs.

The following are typical tips for a COSY beginner.

- An input file name has to have the extension .fox .
- Each line in an input file has to be within 80 columns.
- Don't forget to use the delimiter “;” at the end of each statement.

7 Flow Control

Like other computer languages, the COSY language has branching and looping statements. “FIT - ENDFIT structure” is a unique and unusual feature not found in other languages. It enables nonlinear optimization as a part of the syntax of the language.

IF - (ELSEIF) - ENDIF structure
IF logical-expression ;
...
{ ELSEIF logical-expression ;
... }
ENDIF ;

Example

1. If the value of X is not zero, assign the multiplicative inverse of X to Y.

```
IF X#0 ; Y:= 1/X ; ENDIF ;
```

WHILE - ENDWHILE structure
WHILE logical-expression ;
...
ENDWHILE ;

Example

1. While the value of N is positive, add N to a variable SUM.

```
SUM := 0 ; READ 5 N ;
WHILE N>0 ; SUM := SUM+N ; READ 5 N ; ENDWHILE ;
```

LOOP - ENDLOOP structure
LOOP name start end {step} ;
...
ENDLOOP ;

name: The name of the loop counter.

start: The starting value of the counter.

end: The ending value of the counter.

step: The step size of the counter.

Example

1. Compute 10! and store the result in a variable N.

```
N := 1 ;
LOOP I 1 10 ; N := N*I ; ENDLOOP ;
```

FIT - ENDFIT structure
FIT name1 { ... } ;
...
ENDFIT eps max algo {o1 ... } ;

name1 ...: The variables to be fit.

eps: The tolerance.

max: The maximum number of iterations.

algo: The number of optimizing algorithm to be used.

1: The Simplex algorithm.

4: The LMDIF optimizer. Several objective quantities can be specified.

o1 ...: The name(s) of objective quantity (quantities) to be minimized.

Example

Refer to the following section.

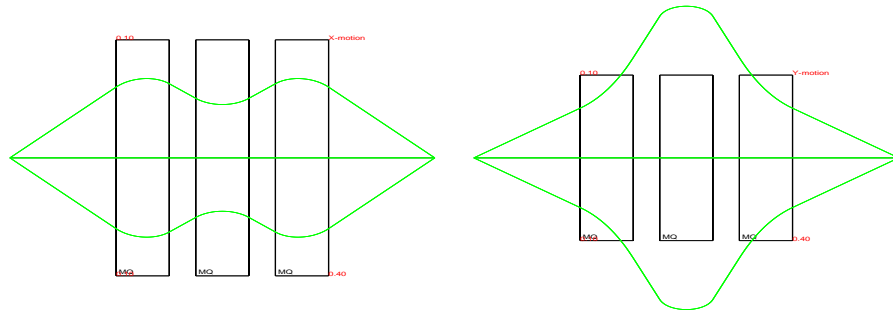
8 Example: Fitting a System

```

INCLUDE 'COSY' ;
PROCEDURE RUN ;
  VARIABLE Q1 1 ; VARIABLE Q2 1 ; VARIABLE OBJ 1 ;
  PROCEDURE TRIPLET A B ;
    MQ .1 A .05 ; DL .05 ; MQ .1 -B .05 ; DL .05 ; MQ .1 A .05 ;
    ENDPROCEDURE ;
  OV 1 2 0 ; RP 1 1 1 ;
  SB .15 .15 0 .15 .15 0 0 0 0 0 0 ;
    {sets half widths of beam .15 m in x, y and .15 rad in a, b}
  Q1 := .5 ; Q2 := .5 ; {start values of Q1, Q2}
  FIT Q1 Q2 ;
    UM ; CR ; {clears the rays}
    ER 1 3 1 3 1 1 1 1 ; {ensemble of rays, 3 in a, b}
    BP ; {begins a picture}
    DL .2 ; TRIPLET Q1 Q2 ; DL .2 ;
    EP ; {ends the picture}
    PG -1 -2 ; {outputs the x,y pictures to default windows}
    OBJ := ABS(ME(1,2))+ABS(ME(3,4)) ;
      {defines the objective OBJ.
      ME(1,2): map element (x,a), ME(3,4): map element (y,b)}
    WRITE 6 'Q1, Q2: ' Q1 Q2 'OBJECTIVE: ' OBJ ;
    ENDFIT 1E-5 1000 1 OBJ ;
      {fits OBJ by Simplex algorithm. This is point-to-point for both x, y}
  PG -10 -10 ;
    {output final pictures to PostScript files pic001.ps and pic002.ps}
  ENDPROCEDURE ;
RUN ; END ;

```

The following final x, y pictures are created in PostScript files pic001.ps and pic002.ps.



Tips: The information on the device unit numbers for various graphics drivers can be found in the manual.