COMFY SOFTWARE USE

A.1 Introduction

This is a guide to the use of the COMFY expansion to COSY Infinity 9.2. This expansion allows the user to calculate the transfer map of a beam system which includes the effects of space charge. This operates by sending a group of test particles through the system, converting their distribution into a charge density function, and then calculating the electric potential and fields for the beam. These potentials and fields are added to the map. Furthermore this expansion also includes a fixed-level fast multipole method to advance particles under circumstances that might be difficult for the map itself.

A.2 Setup and First Use

First, be sure you have installed the MSU program COSY on your system. You will need to contact them for permission from the COSY Website (http://bt.pa.msu.edu/index_cosy.htm). In order to use COMFY for the first time, simply place COSY.fox, COMFY.fox, and the COMFYFILES folder from COMFYFILES.zip in your desired folder. Use COSY to compile COSY.fox to get COSY.bin. Next, use COSY to compile COMFY.fox to get COMFY.bin. When you write your script, write INCLUDE ’COMFY’; as the first line, and you are ready to begin.

It should be noted COSY.fox used for COMFY is slightly different from the standard COSY.fox, but there are only 2 differences. First, COSY.fox includes the global variables needed for COMFY. Second, the memory values for the global MAP variables have been increased to accommodate the space charge map. These changes may be included in the standard COSY.fox easily enough.

A.3 Using COMFY

We will compare the use of a basic COSY code to one that uses COMFY and then discuss the difference. A generic COSY simulation takes the form,

```
INCLUDE ’COSY’;         {Links COSY.bin}

procedure RUN;          {Main procedure; can be any name}

OV 6 2 0;       {Sets map order 6 and other global COSY variables.}
RPE 0.1;        {Set test electron beam energy [in MeV].}

UM;     {Unmap procedure. Initialize transfer map variables.}

DL .2;  {COSY drift element [in meters]}
PM 6;   {Print transfer map to unit 6 (Screen)}

endprocedure;           {Close main procedure.}
RUN; END;       {Calls procedure and ends the program.}
```

whereas the space charge version takes the form,

```
INCLUDE 'COMFY';                {Links COMFY.bin}
procedure RUN;

VARIABLE radii 1 2;             {Variables used for COMFYINIT procedure}
VARIABLE scratch 1 2;

COV 6 2 0 8;            {Sets COSY DA order 6, COMFY space charge order 8,...}
RPE 0.1;
UM;

radii(1):=.01; radii(2):=.01; scratch(1):=.005; scratch(2):=.005;
COMFYINIT 1 30000 .005 radii scratch;       {Generates test distribution, details in Table A.1}
CDL .2;                         {COMFY drift element}
PM 6;

endprocedure;
RUN; END;
```

The first difference is the use of COV instead of OV to set the order and variables.

```
OV NO NV NP;
```

uses NO number of orders, NV number of dimensions, and NP number of parameters.

```
COV NO NV NP OM;
```

uses the same NO, NV and NP, but OM is the order that the space charge distribution will be calculated to. When using COMFY, use of COV is preferred.

The next difference is the use of the line:

```
COMFYINIT <distribution type> <particle number> <distribution radius> <aperture array> <moment order>
<current> <scratchspace>;
```

The entries and their meanings are given in Table A.1.

Table A.1: The entries used in the COMFYINIT command, as well as descriptions of what they do.

Entry	Description
<distribution type>	Determines the distribution used in the simulation, options are discussed in A.4.
<particle number>	This is the number of particles, above 1,000,000 particles there will be no increase of accuracy, above 50,000 particles the hard disk will be used causing an increase in the amount of time required. For fitting or large machines smaller numbers should be used.
<distribution radius>	This is the range within which the distribution is populated as a multiple of <scratchspace>, see A.4.
<aperture array>	This is a two dimensional array giving the size of the beam pipe for particle removal. Positive values give a rectangle, while negative values give an ellipse.
<moment order>	This is the order of moments that are calculated for the space charge model. Only relevant for the moment method.
<current>	This is the beam current in Amps, COMFY currently models an infinitely long beam, so peak current is used.
<scratchspace>	This is used for some of the distributions (see A.4).

If you are restarting a section over and over again (i.e., as part of an optimization routine) you can use the procedure:

```
COMFYREINIT <distribution type> <particle number> <distribution radius> <aperture array> <current>
<scratchspace>;
```

Finally we replace the element DL with the element CDL. CDL uses the space charge algorithms for a list of supported elements; see Section A.5. There are a number of operating modes which are discussed in Section A.6.

A.4 Distribution Options

There are multiple actions that can create the initial conditions of the beam. If you have a set of initial conditions that you wish to use, put them in a file with x,a,y,b on each line separated by a single space (COSY does not recognize tabs) then in the <distribution type> section of the COMFYINIT line put the path string of the filename. If you do this, <distribution radius> and <scratchspace> are not used.

A.4.1 Integer Distributions

The basic distributions are given in Table A.2 and how the range is determined. There is also an outlier, that is, distribution 0. In this case the <distribution radius> would take the value Rx&Ra&Ry&Rb (a vector), while scratch would be a DA vector that contains the desired distribution function. If you wish to have a zero value for one of the elements, simply zero out its radius.

Table A.2: The different possible integer distributions and their descriptions.

number	name	scratch	range
0	-	DA vector of distribution function	<distribution radius>*<scratch>
1	Uniform ellipse	Array with two entries containing the x and y semi-axes.	<distribution radius>*<scratch>
2	Gaussian	Array with two entries containing σx and σy.	6*<distribution radius>*<scratch>
3	Parabolic	Array with two entries containing the limits of x and y.	<distribution radius>*<scratch>

A.4.2 Two-Integer Distributions

If you wish to have a different distribution for the position distribution from the angle distribution, then combine the two. If we wanted to have a distribution that was uniform in xy and Gaussian in ab we would use the following line:

```
COMFYINIT 1&2 nparticles Rxy&Rab aperturearray momentorder current scratch;
```

where scratch is a four-part array as defined in Table A.2:

```
scratch(1)=xradius; scratch(2)=σa; scratch(3)=yradius; scratch(4)=σb;
```

A.4.3 Four-Integer Distribution

This allows for different distributions for all four coordinates in 2D. We would use it in this manner (allocate memory accordingly):

```
COMFYINIT 0&1&2&3 nparticles Rx&Ra&Ry&Rb aperturearray moment order current scratch;
```

```
scratch(1)=DA vector; scratch(2)=aradius; scratch(3)=σy; scratch(4)=bradius;
```

A.5 Supported Elements

In Table A.5 we see the COSY element and the COMFY element; they both take the same arguments. If you want to use an element that is not available, a stand-alone kick has been implemented with the routine,

```
UPSKICKER length h positionsin positionsout;
```

where length is the length over which you wish to kick, h is the inverse of the radius of curvature, and positionsin and positionsout are filenames for input and output when using files.

Table A.3: Table of supported COMFY elements and their COSY equivalents.

COMFY element	COSY element
CDL	DL
CMQ	MQ
CMH	MH
CMO	MO
CMD	MD
CMZ	MZ
CM5	M5
CMM	MM
CMS5	MS
CDP	DP
CDI	DI

A.6 Operating Modes

There are currently five operating modes that are supported using COMFY. The first is the default mode, where COMFY will advance the particles through the system and calculate the space charge kick with the moment method. If the procedure,

```
USEFMM nbins ncoef chargemass;
```

is called before COMFYINIT, the effects of space charge will be applied to the system using the fast multipole method, while the map will still be calculated using the moment method. For more information, see Section A.7. If the procedure,

```
COMFYOFF;
```

is called, the simulation will revert to simple COSY elements, no particles will be advanced, and space charge will not play a role in the system. This could be used if you wanted to compare the maps of a given system with space charge and without. To begin advancing particles again, call the procedure,

```
COMFYON;
```

If you would like to advance the particles without using the space charge map calculation, you can call the procedure,

```
COMFYPAUSE;
```

To start calculating the space charge map again after using COMFYPAUSE, use the procedure,

```
COMFYUNPAUSE;
```

Finally, if you are interested in only tracking the particles using the FMM you can call the procedure,

```
COMFYFMMTRACK;
```

after the COMFYINIT line. This will advance particles using the space charge determined by the FMM but will not calculate the space charge map.

A.7 FMM Use

The fast multipole method used here is a method of calculating the potential and electric fields that will work for any distribution. In order to use this method, before the COMFYINIT line you must use,

```
USEFMM nbins ncoef chargemass;
```

where nbins is the number of bins on a side, and ncoef is the number of coefficients to calculate the multipole expansion. Chargemass is a method for involving multiple species in the simulation; inputting the number 0 will use the species defined as the reference particle. If you wish to use a set of predetermined particles, first you must add a parameter in the OV procedure. COMFY automatically uses para(1) for this method. Then you must use the RPR procedure to define the particle by its magnetic rigidity. The line should look like this:

```
RPR χm*para(1) M0 Z0 ;
```

Next put the string literal for a file which contains, in space separated columns:

Charge per macroparticle [elementary charge units]	Particle energy [MeV]	Particle mass [amu]	Particle charge

As an example, if we wanted a distribution of carbon 12 at 10MeV where each macroparticle represents 1000 atoms, the line would read,

```
6000 10 12 6
```

If we wanted some carbon 14, it would read,

```
6000 10 14 6
```

It should be noted that since this method uses a Taylor expansion the difference should not be large if magnetic elements are desired.

These are read separately from the initial conditions so, if you want a random spread of particles to go along with the separately assigned particles, this can be done. Simply make sure there are enough entries for the particles you want. Please note that this significantly increases the amount of time required for each kick.

A.8 Using the smoothed FMM

The DAFMM has an option for Plummer-like smoothing of the Coulomb interactions. The parameters are 2 global variables separated for the particle-particle (COMFYSOFTD) and long range (COMFYSOFTM) interactions. The smoothing parameters equal zero by default. The smoothing parameters may be changed by using the following procedures after COMFYINIT.

PPSOFT: Changes close range smoothing parameter. Input is smoothing number.

```
PPSOFT <Value>;
```

MULTSOFT: Changes long range smoothing parameter. Input is smoothing number.

```
MULTSOFT <Value>;
```

A.9 Intrinsic COMFY Routines

In addition to the intrinsic COSY routines, a number of COMFY utilities have been included in COMFY.fox. It is recommended you familiarize yourself with them. Some general procedures you might find useful:

OUTPUTFILE: Outputs particle coordinates of the beam as a comma separated (.csv) file or space separated (.dat) file. Inputs are file name string and extension string.

```
OUTPUTFILE <File name> <extension (csv, dat)>;
```

COMFY_BPROFILE: Outputs some statistics of the beam. Gives emittance, current beam fraction, mean, and standard deviation. Unit number 0 suppresses output. Inputs are 3 UI numbers.

```
COMFY_BPROFILE <UI – Mean> <UI – Std Dev> <UI – Emittance, Current Beam Fraction>;
```

COMFY_PB: Outputs phase space plot of beam and histogram. Commented out in COMFY.fox by default. Requires global variable COMFY_PLOT in COSY.fox. Follows COSY notation for coordinates. Setting either range as 0 automatically fits range. Inputs are phase space coordinates (X=1,A=2,Y=3,B=4), X range, Y range, title string, color number, UI number.

```
COMFY_PB <Coord 1> <Coord 2> <+/- Range 1> <+/- Range 2> <Title> <Color> <Unit number>;
```

By default, the number of space charge kicks per COMFY element is 1. This may be changed by the procedure, <Element name>KN. For example, to change the number of kicks for a drift,

```
CDLKN <# of kicks>;
```

B Parallel COMFY

B.1 Introduction

This is a guide to the use of the parallelized COMFY. It is assumed the user understands how to use the parallelized version of COSY.

B.2 Setup and First Use

First, be sure you have installed the parallel version of the MSU program COSY on your system. As in the COSY manual, you can use VERSION to switch the COSY source code to MPI. The COSY documentation uses OpenMPI (http://www.open-mpi.org/) as its example. Using parallel COSY with OpenMPI is covered in the COSY documentation.

B.3 Using Parallel COMFY

At the moment, only the moment method is parallelized. The FMM will run only on the root process. The space charge maps are computed in parallel.
Currently, the root process will perform the beam initialization and distribute to the other processes. Each process needs to access the same COMFYFILES folder.

Some important global variables are

COMFYNPR	{Number of processes}
COMFYIRT	{Root process if 1, otherwise 0}
COMFYMPII	{Current process index}

The intrinsic COMFY procedures, except for COMFY_PB, should be compatible with the parallelization. Call them as you would with the serial version. Some utility procedures for the parallelization are implemented and may be used if needed.

To divide the particles among the processors using files or memory, use

```
SPLIT INPUTFILE OUTPUTFILE;
```

The input file is the base name with all the particle coordinates. The filename is INPUTFILE&'0'. The output file is the base name for the particle files for each process. The filenames will be OUTPUTFILE&'<Process index>'. These are only used for no. of particles > 50000.

To collect the particles from the processors to the root process using files or memory, use

```
JOIN INPUTFILE OUTPUTFILE;
```

This will be the reverse as SPLIT. The input file is the base name for the particle files for each process. The filenames are INPUTFILE&<Process index>. The output file is base name for the file with all particles. The filenames will be OUTPUTFILE&'0'.

There is a rebalance procedure, but it needs further testing at the moment.

To compute the split among processes, you can use

```
GET_LIMITS NUMPROC CURRPROC TOTALNUM FIRST LAST;
```

GET_LIMITS takes the number of processes, the current process index, the total number of things you wish to distribute and outputs the first and last indices for each process.
For example, if you have a vector of numbers, GET_LIMITS gives the start and end indices given the vector length as TOTALNUM, which can be used to extract the relevant part of the vector on each process.

If there are points where you wish to synchronize all processes, use

```
SYNC;
```

This will cause processes to wait until all processes have reached that point.