Breaking up few macro-particles into a many micro-particle Poissonian system

Program Specification for MASP

Andrew Colin

Version 3 : August 16, 2015

Note : In this specification, full stops are omitted in places where they might be mistaken for parts of file names. Thus:

.....the file name should have the extension **.csv** *An example is .....*

1.1 Introduction

Many interesting systems are built of two components:

- A **pulse generator** which produces a pulse of particles with known statistical properties, and
- A **device** which receives the particles and extracts some useful property.

These devices are generally expensive to build and operate. A simulator is a vital tool in achieving the optimum design and selecting the best working conditions for any particular application.

The design of a simulator for such a system must solve a key practical problem. Current limitations on computers mandate that simulated pulse generators are (as yet) unable to produce streams of individual particles in sufficient numbers, but generate such streams as*macroparticles'*, each of which may include some 10^4 particles with identical properties. Likewise, the simulated device does not necessarily need individual particles as inputs, but will only produce accurate results if supplied with small groups of particles, called *microparticles* - perhaps with an average of 25 particles each. Therefore a simulated pulse of massive macroparticles with given statistical properties needs to be split into a much larger number of microparticles which obey the same statistics. This is the job of the Macroparticle splitting program **"MASP".**

The immediate application for the program is as a component of a simulator for a Free Electron Laser. We use electrons in a beam as an example throughout this speciification. **MASP** could be useful in other areas; for example, a pulse of enantiomorphic molecules may be sent to an analyser that determines its chirality [1].

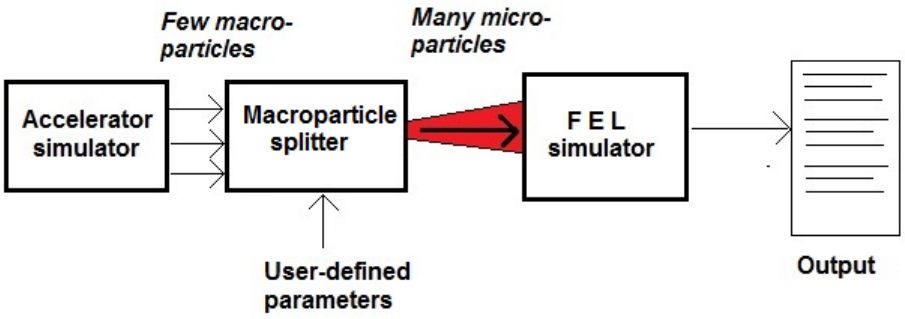

Figure 1.1: Simulator configuration (for Free Electron Laser)

This program, known as **"MASP"** (for **MA**croparticle **S**plitter **P**rogram), helps to solve the problem by reading in the properties of macroparticles, and dividing each macroparticle into much smaller microparticles before passing them on for analysis. Figre 1.1 shows how it is used.

1.2 Nomenclature

MASP is designed for use over a range of scientific areas that need similar or identical calculations. The program description deliberately uses generic terms which avoid links with specific disciplines.

- The basic entity is called a *particle*. This could be an atom, a molecule, an electron or a proton. The state of a particle can be given as its position, in X, Y and Z coordinates, and its momentum, in Px, Py and Pz coordinates - that is, six dimensions in all. The units of these quantities must be consistent with one another but are otherwise undefined, as the output of **MASP** will be in the same units as the input.
- Each particle has an invariant property. In the case of an electron or proton this is its charge; for a particle not moving at relativistic speed, this is its mass. We refer to this property as the *weight* of the particle, and assign it the arbitrary value of 1. The *size* of a macroparticle or a microparticle is the number of particles it contains.
- To be specific, a *macroparticle* is a heavy-weight group of particles produced by the particle generator simulator; a *microparticle* is a light-weight group of particles generated by the macroparticle splitter and fed to the property extraction module (such as the FEL simulator of figure 1.1) with the correct noise properties of the equivalent particle distribution..

1.3 Overview of the Particle Conditioner

A useful two-dimensional analogy for this section is the behaviour of a cartridge fired from a shotgun. Initially all the pellets move towards the target, but as they approach they spread out, in an assumed gaussian pattern, in both direction and speed.

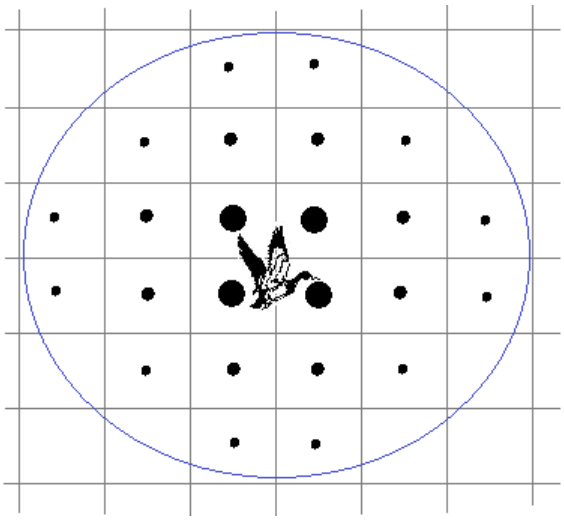

Figure 1.2: The Shotgun effect

Figure 1.2 shows a duck flying in front of a rectangular grid. When the hunter takes a well-aimed shot at the duck, the majority of the pellets hit cells close to the target (some of them may even hit the duck), but a substantial fraction hit cells some distance away. The black blobs in figure 1.2 shows roughly how many pellets hit each cell.

The splitting process in MASPfrom macro- to microparticles is first done separately for each of the six dimensions. In Figure 1.3, the marks on the horizontal axis represent the cell boundaries in a given dimension. We suppose that a macroparticle lands on one of the cells (not necessarily in the middle). Its position is shown by the red line. Assuming a gaussian distribution of scattering, the program superimposes a gaussian curve centred on the position of the macroparticle and with a standard deviation that can be determined by the user (typically 2,3 or 4 cell sizes). The program then uses the error function to calculate the area, or the probability that a particle from the macroparticle will land in that five-dimensional slice of the cell matrix. as shown in the figure. Although the gaussian curve extends to infinity in both directions, very small probabilities are ignored, and the probabilities in any axis are renormalised so that they sum to 1.

Once this process has been repeated for all six dimensions, the probability that a particle from the original macroparticle lands in a given cell is the product of the six probabilities from the individual dimensions. The expectations, (probabilities $\times$ size of the macroparticle) which are generally fractional at this stage, are then distributed according to these probabilities, making a microparticle in each cell. Each microparticle is initially placed in the centre of

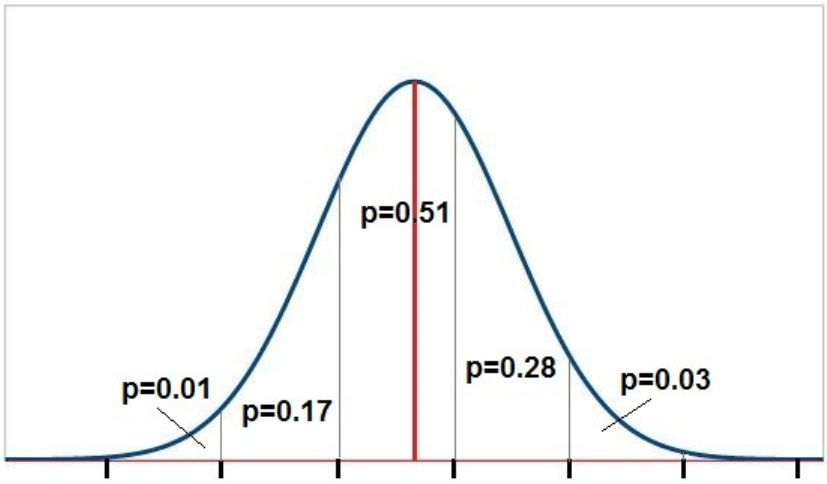

Figure 1.3: gaussian distribution

its cell. With a gaussian distribution, the largest microparticles will end up in the original target cell, but many others will be sent to adjacent cells. If two or more macroparticles are close together, a cell may have contributions from all of them. The expectations from multiple sources are simply added together.

Once all the macroparticles have been distributed, the microparticles within the cells are prepared for output. At this stage, each microparticle appears as an expection of a given size in the exact centre of its cell. Then, if the user has specified that noise be added, this takes place in two stages:

- First, each expectation is replaced by an integer using a Poisson distribution with the expectation as its parameter. If the size of the resulting microparticle is zero, it is discarded.

- Then, displacements are added to the positions in each of the axes. Consider a cell of size d (in any dimension). The starting position of the microparticle is the centre of the cell, but the position is then randomly shifted within a range $-d/(2\sqrt{N})$ to $d/(2\sqrt{N})$ where N is the integer size of the microparticle. The random numbers are taken from a rectangular distribution[1].

[1]This ensures that the microparticles model the statistical properties of an equivalent distribution of individual particles, according to [2]

Thus a microparticle with a large size will remain close to the centre of the cell, but a microparticle with a weight of one unit may be anywhere in the cell with equal probability.

Finally, the program carries out certain consistency checks:

- The total weight of all the microparticles should be equal to the weight of the macroparticles. In practice there will be a small loss because cells which lie at the extremities of the gaussian distribution and have very low expectations of containing a particle are ignored. The overall loss is reported as part of the output of **MASP**.
- If noise is present the output should show 'bunching' [2]. For a large sample of microparticles the characteristic curve should approximate to the Rayleigh distribution.

The conditioner can also do some auxiliary tasks:

- Provide data to draw histograms of the distribution of macroparticles in each dimension.
- Calculate the total number of particles in each three-dimensional *spatial* cell of the output, and so provide the spatial density.
- Provide data to draw the profile of the input pulse in each dimension, based on the particle flow at many points in time.
- Provide data to show the bunching characteristics of the microparticles.

In all cases, the conditioner generates files of data. The production of graphs of distributions and profiles is left to the user.

1.4 Input

The input to the conditioner is a file with a number of records, which all share the same format. Each record describes a macroparticle and must include values for each of the six dimensions, and for its weight. It may include other values (such as the Lorentz factor γ) but they are ignored by the conditioner.

Numerical values are given in 'scientific' format, with an argument and generally an exponent, such as 3.657388200e-8 . Values are separated by spaces, and each record (including the last!) is terminated by newline.

You are free to define the order of the components in each record. This is explained in the following section.

1.5 The Parameter File

The conditioner offers many options, which you can specify in the parameter file, which may have any name you choose.

Prepare the parameter file with a text editor such as *gedit*. Do not use a conventional word processor as it will add a wealth of 'invisible' characters to confuse the conditioner.

The parameter file consists of a number of lines, separated by newlines.

Any line which starts with a non-alphabetic character is taken as a comment, and is ignored. In the example parameter file we display further on, we use *****, but any other character, such as **+** or **$** would serve.

Each line, except for comments, starts with a keyword, which may be followed by one or more arguments, all separated by spaces.

The order of lines is immaterial.

The conditioner is not sensitive to the difference between upper and lower case, *except* where the argument is the name of a file.

If a keyword is omitted, the program will automatically supply a default.

In the rare event that *all* the lines are omitted, because all the default values are satisfactory, a parameter file **must** be made, even though it is empty or contains only comments.

1.5.1 Data description

The keyword **INPUT** gives the name of the file to be used as data.
An example is

INPUT electrons.txt

If the **INPUT** keyword is omitted, the default file name is **data.txt**

This program uses a record for each macroparticle with the following values:

- **X** is the X-coordinate of the macroparticle
- **Px** is the momentum of the macroparticle in the X direction
- **Y** is the Y-coordinate of the macroparticle
- **Py** is the momentum of the macroparticle in the Y direction
- **Z** is the Z-coordinate of the macroparticle
- **Pz** is the momentum of the macroparticle in the Z direction
- **E** is the size of the macroparticle (where *size* is the number of particles it contains. See section 1.2.)

Generator simulators may produce records for each macroparticle (of constant layout) in which this data might be in any order, and may include additional information which the conditioner program does not need. You can adapt the data file by using the **DATA** keyword.

DATA is followed by exactly 7 integers. The integer in the first place corresponds to the X-coordinate, that in the second place to Px, and so on. The *value* of each integer gives the position of that variable in the input record (again counting from 0).

Suppose that your input records contain values for X, Y, Z. Px, Py, Pz, X*Px, Y*Py, Z*Pz, γ, and E (in that order).

First, you must label these fields with the integers from 0 up, like this

X	Y	Z	Px	Py	Pz	X*Px	Y*Py	Z*Pz	γ	E
0	1	2	3	4	5	6	7	8	9	10

Next, identify the columns that the conditioner needs. In this case, it will be the set $\{0, 1, 2, 3, 4, 5, 10\}$. Colums 6, 7, 8 and 9 are not needed. Now write thes numbers after the **DATA** keyword, putting the position of your X column in the first place, of your Px column in the second, and so on. The result will be:

DATA 0 3 1 4 2 5 10

These numbers correspond to **X**, **Px**, **Y**, **Py**, **Z**, **Pz** and **E** in that order.

If the **DATA** keyword is not used, the default is

DATA 0 1 2 3 4 5 6.

The program expects all measurements to be in arbitrary units. Distances may be metres or nautical miles, and momenta in kg metres/sec or ton-knots. Outputs are in the same units as the inputs.

1.5.2 Tasks

The four tasks that the conditioner can do is each identified by its own keyword. Each keyword is followed by the name of a file used to record the results:

- **MICROPARTICLES** is the main task, which converts macroparticles into microparticles. It is the most laborious of all the the tasks, and the only one which really needs a multiprocessor machine. The output file can be large (typically over 1 gigaByte). An example is:

  MICROPARTICLES results.txt

  The others are:

- **HISTOGRAM** : This builds histograms of the distribution of the data projected onto each dimension. To simplify handling by a spreadsheet, the file name should have the extension **.csv** An example is

  HISTOGRAM datadistributions.csv

- **WEIGHTS** This task generates the the number of elementary particles in each of the three dimensional spatial cells of the six-dimensional space. Momenta are ignored in this context. An example is

  WEIGHTS weights.txt

- **BUNCHING** This option generates data to check that the microparticles show suitable bunching characteristics. An example is:

  BUNCHING characteristc.csv

If any one or more tasks are specified in the parameter file, the conditioner will carry out only these tasks. But if *none* is given, the conditioner will do *all* of them, using the default file names

Task	**Default output file**
MICROPARTICLES	results.txt
HISTOGRAM	projection.csv
WEIGHTS	weights.txt
BUNCHING	bunching.csv

1.5.3 Dimension parameters

Each of the six dimensions has properties that can be defined in the parameter file:

- **Cell number**. This is the number of cells that span the range of this dimension. For example, if the X dimension is given a resolution of 100, then the 6-dimensional space will have 100 cells in the X dimension, equally distributed between the highest amd lowest X-values in the data.

- **Width**. This controls the width of the scattering process in that dimension. For a scattering parameter of 1, assuming a gaussian distribution, the some 99.6% of all the charge will be distributed among cells which are three cell-widths or less distant from the macroparticle position. A scattering parameter of 2 will spread the distribution twice as wide. Note that if the resolution is 1,there will be only one cell in that dimension, and the only sensible value for the scatter is zero.

- **Profile**. The default scattering pattern is assumed to be gaussian, but you can specify others. Three other patterns are included as an illustration. Thus the **TOPHAT** pattern will distribute microparticles evenly within a certain distance of the aiming point, with none falling outside. A **BLANCMANGE** distribution is like an inverted parabola. A **TRIFLE** distribution, with a minimum in the centre, ensures that particles are distributed away from the aiming point. Ways of defining other distributions are described in the appendix. [2]

[2] These distributions are not generated by any mathematical function, but defined purely by arrays of numbers.

Each dimension parameter starts with a keyword which is one of

X, Y, Z, PX, PY or Pz

This is followed by the resolution, the scatter parameter, and the name of the scattering pattern. An example is

PZ 500 1.2 TOPHAT

The defaults for each of these six parameters are

Dimension	Cell number	Width	Profile
X	10	1.0	GAUSSIAN
PX	10	1.0	GAUSSIAN
Y	10	1.0	GAUSSIAN
PY	10	1.0	GAUSSIAN
Z	1000	1.0	GAUSSIAN
PZ	100	1.0	GAUSSIAN

1.5.4 Other parameters

RANGE

This controls the scattering of particles under the gaussian assumption. If **RANGE** is set to 3, then only cells up to three standard deviations from the aiming point are taken into consideration, and this gives a small loss of charge - about 0.4%. If **RANGE** is set to 4, then the loss of charge is less, but the program takes longer to run. The line can take the form

RANGE 4

The default value is

RANGE 3

It is best not to specify values of 5 or more for the **RANGE** parameter, as this will greatly increase the processing time for the calculation.

THRESHOLD

When a macroparticle is scattered, the weight expectation arriving at any cell is often much less than 1. In principle this value should be kept, as other bursts may contribute to the same cell, but in six dimensions, the number of cells within three (or four) cells from the aiming point is huge. The **THRESHOLD** keyword defines a level below which the weight expectation on a cell is ignored. Experiments show that with a spread of 4 and a threshold of 0.001 the total loss of weight is small.

An example of the **THRESHOLD** parameter is

THRESHOLD 0.05.

If you need a quick and dirty analysis, set

RANGE 3
THRESHOLD 0.1

NONOISE

If thie keyword **NONOISE** is included, noise is **not** added to the position of the microparticles. The default is for noise to be added.

1.6 Output

This section explains the outputs of the electron conditioner in some detail.

1.6.1 MICROPARTICLES

Each microparticle generated by the system corresponds to a non-empty cell in the six-dimensional space. It appears in the output file as a record with seven real numbers, or if the addition of noise has been specified, six real numbers and a positive integer, all separated by spaces.

The real numbers are the coordinates of the particle, in the default order

X, Px, Y, Py, Z, Pz

The final value is the expectation or weight of the microparticle. If necessary, the order of the spacial coordinates can be changed by the **OUTPUT** parameter, which is described below.

A sample set of output records (with added noise) is shown below.

```
-1.666481e-06 1.249562e-22 1.201243e-06 -4.008479e-23 2.186326e-05 1.379387e-19   1
-2.909566e-06 1.628031e-22 2.980604e-06 -2.165251e-23 2.185996e-05 1.380008e-19   1
-1.784693e-06 1.338876e-22 2.239557e-06 -2.384489e-23 2.185941e-05 1.377288e-19   2
-2.668413e-06 1.548629e-22 1.983784e-06 -4.120813e-23 2.186211e-05 1.377122e-19   1
-2.149306e-06 1.460538e-22 2.121886e-06 -2.435864e-23 2.185499e-05 1.372505e-19  22
-1.853056e-06 1.558197e-22 1.876072e-06 -1.713046e-23 2.185779e-05 1.372342e-19  18
-1.601554e-06 1.644343e-22 1.854537e-06 -3.037348e-23 2.185903e-05 1.372200e-19   4
-2.137594e-06 1.554611e-22 1.787428e-06 -1.347176e-23 2.185721e-05 1.372709e-19   5
-2.282024e-06 1.720266e-22 1.501850e-06 -1.402764e-23 2.184841e-05 1.372844e-19   1
```

Note that the number of records in this output file may be large. The order of records is random.

The keyword **BINARY** offers the option of writing the details of each macroparticle as a binary record. If there is no noise adjustment it will consist of s seven *floats*; otherwise six *floats* and one *integer*. This will make the output file about 40% shorter and give an increase in speed. On the other hand, the

internal formats of numbers may not be the same between different computers, which will make the binary file unusable. Do not use the binary option unless you are certain that internal formats are compatible [3].

To put the coordinates in a different order, use the statement **OUTPUT**. The keyword is followed by six integers, which have the following meanings:

X	Px	Y	Py	Z	Pz	Weight
0	1	2	3	4	5	6

Follow the **OUTPUT** keyword with these numbers in the order in which you wante the coordinates.

To give a somewhat quirky example, example, the statement

OUTPUT 6 2 5 1 4 0 3

will output each record in the order

Weight Y Pz Px Z X Py

1.6.2 HISTOGRAM

The program finds the lowest and highest values for each dimension. Then it divides this range into 100 equal parts, and counts the number of records falling into each part.

The output is a file of integers arranged into six columns and 100 rows. The columns normally refer to the dimensions in the order

X, Y, Z, PX, PY, Pz

but this can be changed if the **OUTPUT** command is included in the parameter file.

When this file is loaded into a spreadsheet, a graph drawn on each column will show the distribution of values in the corresponding dimension. Figure 1.4 shows the distribution of the X and Z axes for a typical data set.

1.6.3 WEIGHTS

If the momentum dimensions are ignored, each microparticle generated by the system can be placed in a cell in the three-dimensional grid based on the X, Y and Z dimensions. The output of this task is a file that includes one record for each non-empty spatial cell, and gives the position of its centre, and the total number of particles it contains, in the default order

X, Y, Z, weight.

[3] Think : Big-endian vs. Little-endian

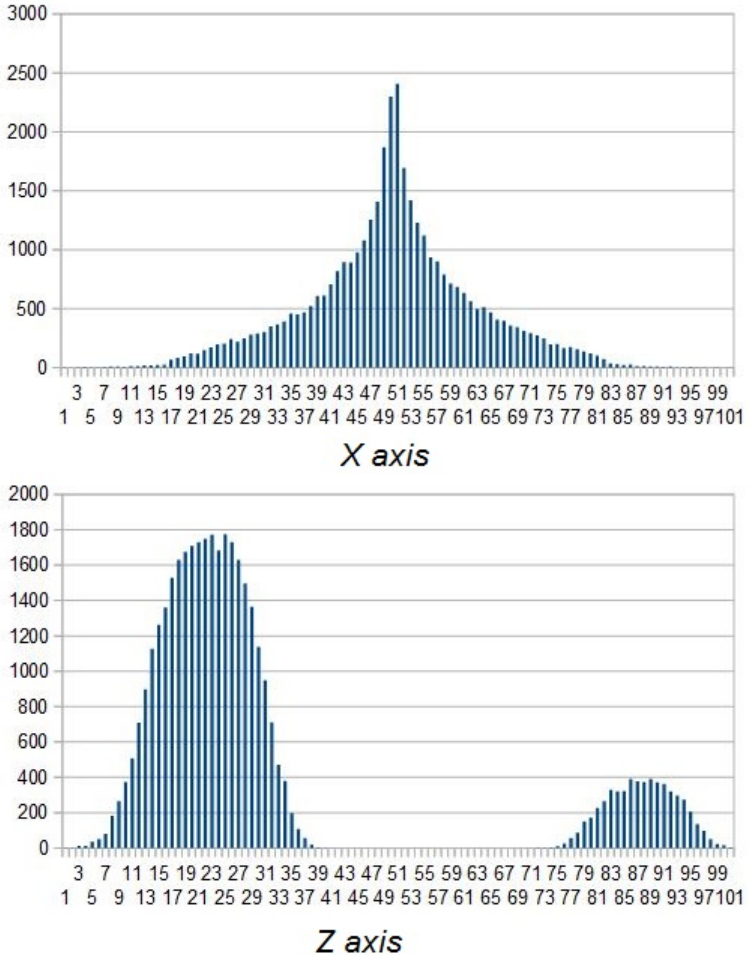

Figure 1.4: Distributions of values in the X and Z axes

For example:

```
5.82111e-07,  -1.51176e-06,   2.06924e-05,  107
5.82111e-07,  -1.51176e-06,   2.07836e-05,   64
5.82111e-07,  -1.51176e-06,   2.07392e-05,   54
5.82111e-07,  -1.51176e-06,   2.07158e-05,   77
5.82111e-07,  -1.51176e-06,   2.07626e-05,   48
5.82111e-07,  -1.51176e-06,   2.08094e-05,   79
5.82111e-07,  -1.51176e-06,   2.08328e-05,  145
5.82111e-07,  -1.51176e-06,   2.08562e-05,  169
```

The order can be changed by the OUTPUT statement (the same one as applies to the microparticle output).

1.6.4 BUNCHING

Bunching, a measure of the non-uniformity of particle flow, is an important characteristic of any stream of microparticles. In [2] each simulated particle in a very long stream of Z particles is assigned an integer size N from a Poisson distribution based on a given expectancy, and then displaced in the direction of travel by a random amount proportional to $\pm 1/(2\sqrt{N})$. This displacement is called *shot noise* [4].

The particles are divided into Z/P groups o fG particles each (G is arbitrary, but might typically be 10 or 12). Then a quantity $|b|$ is calculated for each group, as follows:

Consider a snapshot of a group of particles , and suppose that in the absence of noise the first one is at position zero, and the others are spaced one unit apart. Now the *actual* position of particle j in the stream will be $j + noise_j$, and its weight N_j. For each group the method calculates

$$b = \sum_{k=0}^{G-1} N_k \times \exp(i \times (k + noise_k) \times 2\pi/G) \tag{1.1}$$

b is complex, but $|b|$ is real. The method finds the average value of $|b|$, called $< |b| >$, and forms the ratio $|b|/ < |b| >$ for each group. The resulting numbers form a close approximation to the Rayleigh distribution [3], as shown in figure 1.6. Likewise, the values of $|b|^2/ < |b| >^2$ give a negative exponential distribution.

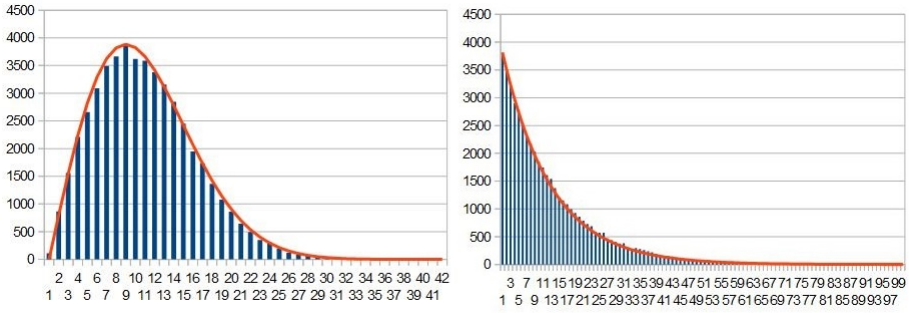

Figure 1.5: Ideal bunching characteristics

The **BUNCHING** option takes a sample of the microparticles, in vectors aligned to the Z axis, and carries out the same calculation. The outcome can be seen by examining the appropriate output file. In contrast to the ideal situation, the data is extremely noisy, many periods are empty or contain only one macroparticle. Typical results are shown in figure 1.6:

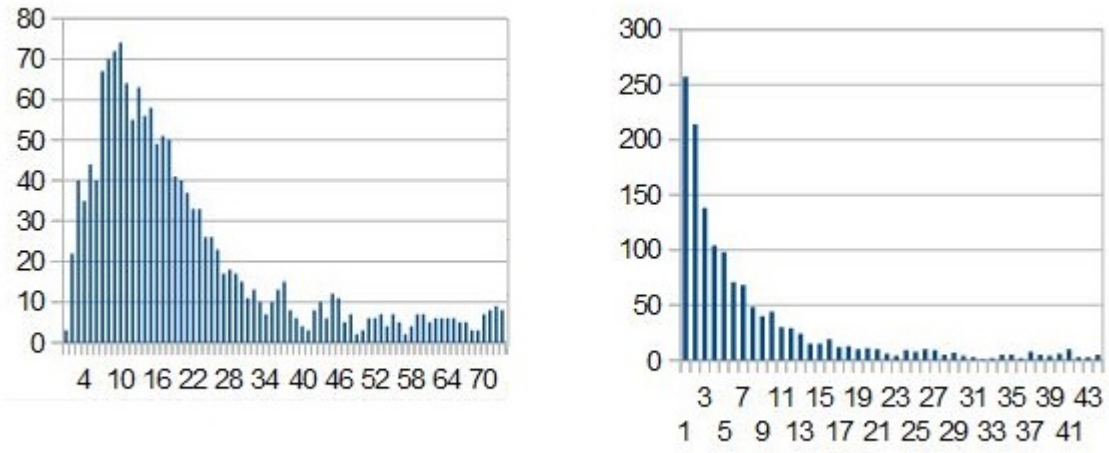

Figure 1.6: Practical bunching characteristics

1.7 Sample parameter file

```
* This is a typical parameter file.
INPUT mydatafile.txt
DATA 3 4 5 0 1 2 8
HISTOGRAM shapes.csv
MICROPARTICLES myresults.txt
OUTPUT 3 4 5 0 1 2
WEIGHTS myweights.txt
BUNCHING bun.csv
X 10 1.0 GAUSSIAN
Y 10 0.5 TOPHAT
PX 1 0 GAUSSIAN
PY 1 0 GAUSSIAN
Z 1000 1.0 GAUSSIAN
PZ 500 1.0 TOPHAT
RANGE 3
THRESHOLD 0.005
* In this case, noise addition is needed
*NONOISE (This is a comment!)
BINARY
```

Note that:

- These lines may be placed in any order
- Blank lines are accepted
- Any (or all) the lines may be omitted. Omission implies that the defaults will be used.

1.8 Practical Details

MASP is distributed as a set of source files:

masp.h
maspstructures.h
MPIstuff.c
masplibrary.c
maspparams.c
maspdatainput.c
maspweight.c
maspbunch.c
masptree.c
maspcheck.c
masphisto.c
maspcheckdatainput.c
maspparticles.c
masp.c
TOPHAT
BLANCMANGE
TRIFLE

Load these files into your home directory, or whichever directory you intend to use for **MASP**. Put your parameter file, and the data file into the same directory.

There are two separate programs, **masp** and **maspcheck**. **maspcheck** will run a preliminary check on your parameter file.

1.8.1 The check program

If your data needs to be processed by a multi-processor machine, your job will probably have to compete for resources and spend a long time in a queue. If the parameter file has any errors, the job will be aborted as soon as it is started. You may not be happy with this outcome.

To avoid this risk, we recommend using the check program, **maspcheck**. This program should have access to the parameter file and to the data set.

The check program starts by analysing the parameter file. If no errors are detected, the check program reads the data and estimates the resources you will need to specify for the main program.

To run the check program, type the following command lines:

```
gcc maspcheck.c -o zz -lm
./zz  parameter.file
```

where **parameter.file** is the name of your parameter file.

The check program uses few resources, and you can run it on a personal computer (provided you supply the parameter file and the data file in the same directory). When the program terminates, look at **monitor.txt** for a list of errors (if any) and an estimate of the resources needed to run **masp**, using the same parameters and data.

To run **masp**, you will probably need a *job script*. The sample below is correct for Archie-West and similar multiprocessor machines, but may differ in detail in other environments:

```
# ***************Sample job script*****************
module load mpi/gcc/openmpi/1.6
mpicc masp.cpp -o qq.out -lm
export   PROCS_ON_EACH_NODE = 12
#   Export env variables and keep current working directory
#$ -V -cwd
#  Associate with my project
#$ -P identity.prj
#  Select parallel environment and number of queue slots
#$ -pe mpi-verbose 3
#  Combine STDOUT/STDERR
#$ -j y
#  Specify output file
#$ -o out.\$ JOB_ID
#  Add run-time indication
#$ -ac runtime="5m"
export NCORES = `expr$ PROCS_ON_EACH_NODE \* NSLOTS`
$  export OMPI_MCA_btl=openib,self
mpirun -np $NCORES qq.out
#  ********************End job script **************
```

You need to make three alterations:

- In line 9, replace "identity.prj" with your own project identity
- In line 11, replace "3" by the number of modules you need. **maspcheck** will g ive you an estimate of the number of cores; divide by 12 and round up.
- in line 17, replace "5m" by the time estimated by **maspcheck**

Give your job script a name with the extension **.sh** (for example **feljob.sh** and start the job with the command line

`feljob.sh`

Warning: If any of your named files (possibly from a previous run) are open in any other program such as a spreadsheet, this program will fail. Delete the other program (or close the file) and try again.

During the run, interesting information is sent to a file called **monitor.txt**. Its contents include error reports for the monitor file (if any) , a complete list of parameters, including default values, the number of input records, and the volume of each spatial cell.

1.9 Estimating run time

The run time for **MASP** depends on so many factors (apart from queueing time if your machine is heavily loaded) that that only approximate estimates can be made. Here we report a number of measurements which may be of assistance.

The 'reference run' was made with a set of 38035 macroparticle records, and the following parameter file:

```
* Reference parameter file
RANGE  4
THRESHOLD 0.01\\
INPUT electrons.txt\\

MICROPARTICLES microparticles.txt\\
WEIGHTS weights.txt\\
X 10 1 GAUSSIAN\\
PX 10 1 GAUSSIAN\\
Y  10 2 GAUSSIAN\\
PY 10 1 GAUSSIAN\\
Z 1000 1 GAUSSIAN\\
PZ 100 1 GAUSSIAN\\
```

Using 12 cores, this file took 108.3 seconds to process its data.
Further factors to be taken into account include:

- The overall time is roughly proportional to the number of records processed. Note that you should specify enough cores not to exceed 3200 records *per core*.
- Reducing the **THRESHOLD** by a factor of 100 (to 0.0001) will roughly double the computation time.
- The times taken by the **HISTOGRAM**, **WEIGHTS** and **BUNCHING** are all neglegible and need not be taken into account.

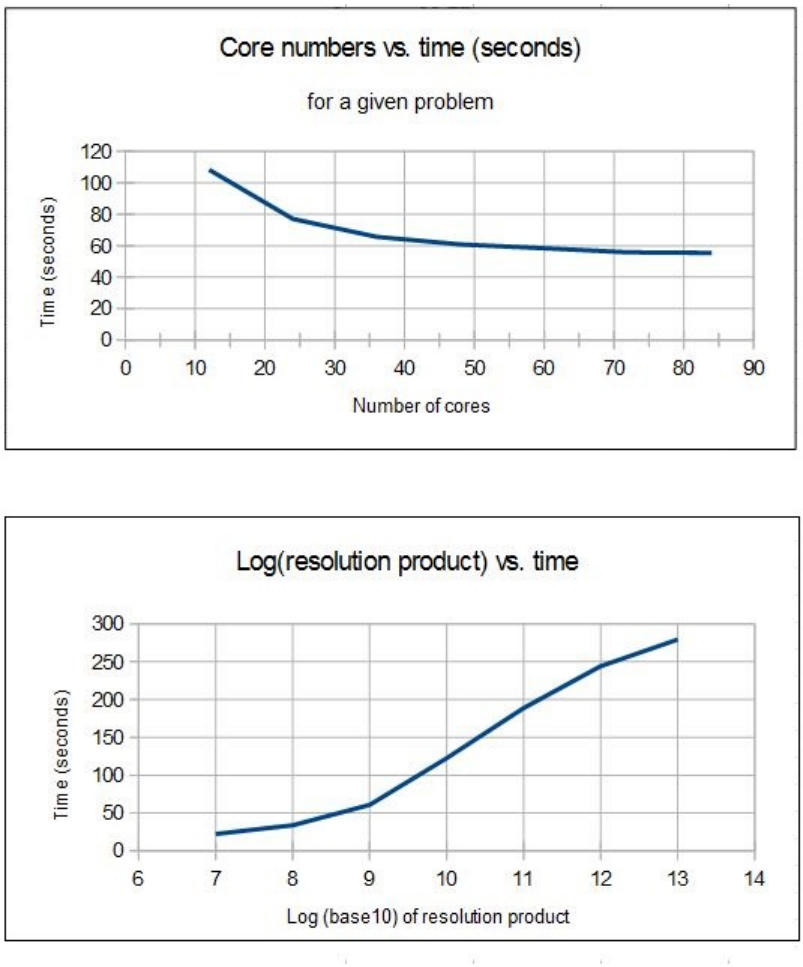

Figure 1.7: Timing characteriscs

- Figure1.7 (upper) shows how the time needed for the reference parameter file changes with the number of cores allocated. There is no point in using too many cores.
- Figure1. 7 (lower) shows how the time varies with the resolution. We have plotted to time against the log_{10} of the product of the resolutions in all six dimensions, so that (for example) the set of resolutions **100,10,100,10,100,10** gives a product of 10^9 and a processing time of about 60 seconds. All the measurements for this chart were taken using

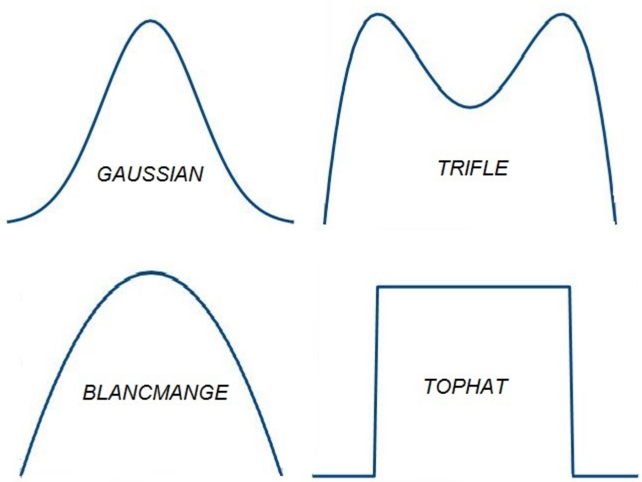

Figure 1.8: Scattering patterns

24 cores.

1.10 Appendix : Defining scattering patterns

You don't need to read this section if gaussian scattering is all you wish to use

A scattering pattern is defined as a sequence of numbers that define the height of the scattering function at equally spaced intervals on the x-axis.
The scales are not defined, as the program will automatically assume that the x-axis ranges from -3 to +3, cells of your defined resolution, and adjust the heights, so that the total area under the curve is 1.

There are four pre-defined scattering patterns, shown in figure 1.5. The numbers should be placed in a file with the name you choose for the pattern (and no extension). The first number in the file gives the number of values that follow. Here are the contents of file TOPHAT:

101
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 The gaussian distribution pattern follows the standard mathematical definition. The other patterns are defined only by sets of numbers, and the same would be true of any other patterns the user may use.

1.11 Index of Keywords

BINARY	1.7.1	
BLANCMANGE	1.6.3	
BUNCHING	1.6.2	1.7.5
DATA	1.6.1	
Dimension parameters	1.6.3	
HISTOGRAM	1.6.2	1.7.2
INPUT	1.6.1	
MICROPARTICLES	1.6.2	1.7.1
NONOISE	1.6.4	
OUTPUT	1.7.1	
Px, Py, Pz	1.63	
RANGE	1.64	
THRESHOLD	1.64	
TOPHAT	1.63	
TRIFLE	1.63	
WEIGHTS	1.62	1.72
X, Y, Z	1.63	

Bibliography

[1] R.P.Cameron, S.M. Barnett and A.M.Yao
Discriminatory optical force for chiral molecules
New Journal of Physics 16 (2014) 013020

[2] B.J.W. McNeil, M.W. Poole and G.R.M.Robb
Unified model of electron beam shot noise and coherent spontaneous emission in the helical wiggler free electron laser
Physical Review Special Topics - Accelerators and Beams, Vol 6, 070701 (2003)

[3] Wikipedia
Wikipedia article on Rayleigh Distribution

[4] W. Schottky
ber spontane Stromschwankungen in verschiedenen Elektrizittsleitern
Annalen der Physik (in German) 57: 541567. (1918).