sLHC Project Note 0008

2010-01-13

Author.Jean.Baptiste.Lallement@cern.ch

A Program to Generate a Particle Distribution from Emittance Measurements

D. S. Bouma, J. Stovall, JB Lallement

Summary

We have written a program to generate a particle distribution based on emittance measurements in *x-x'* and *y-y'*. The accuracy of this program has been tested using real and constructed emittance measurements. Based on these tests, the distribution generated by the program can be used to accurately simulate the beam in multi-particle tracking codes, as an alternative to a Gaussian or uniform distribution.

1. Introduction

The purpose of the genBeam program is to improve the accuracy of accelerator simulations in predicting the behaviour of a beam and managing its halo. It will do this by providing the simulations with a particle distribution based on real measurements. The program uses the measurements taken by slit-and-grid or Allison-type emittance scanners to generate an approximate distribution of the original beam by creating a histogram having the resolution of the measurements and then fitting randomly generated sets of coordinates of macroparticles that are consistent with the histogram. It writes the resultant particle coordinates to a beam data file which can be read by the PATH Manager particle-tracking software (1). This same output file can also be made compatible with the program TRACEWIN (2).

2. Program Use

The genBeam user manual contains instructions on using the program, and a detailed discussion of resolving the errors printed by the program at run-time.

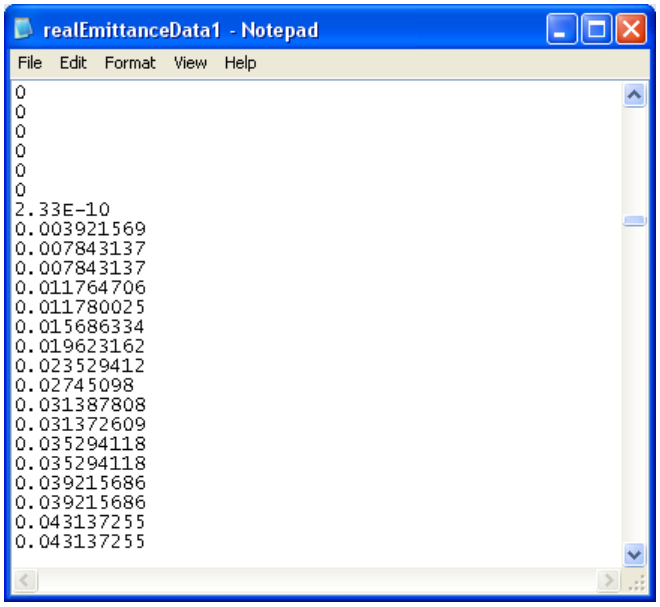
2.1. Installation

The genBeam program is an MS-DOS executable file. It can be installed in any directory; however, its input files must be found in the same directory from which it runs. Its output files will be saved to the same directory.

2.2. Command-line arguments and input files

The genBeam program requires certain command-line arguments to run. Because of this, it must be accessed through Command Prompt (*Start* > *Run* > *cmd*). When the MS-DOS window appears, type the command `cd path_to_directory` to navigate to the directory where genBeam is installed. The current version of genBeam is v1.0, so to run the program, type the command `genBeam1 [name of x-x' emittance data file] [name of y-y' emittance data file] [number`

`of macroparticles to be generated] [name of output file]`. The final argument is optional, but recommended. By default the program will save to the output file generatedBeam.txt, overwriting anything already in that file. There should be no spaces in any of the directory names or file names.



The input files must be in the .txt format, and the name of the output file must include the .txt extension. The input files must be in a very specific and simplified format, as follows. They must not contain any data about the x-position or the x'-position, but only the signal value taken by the detector at that position. The program infers the x and x' positions from the detector parameters entered at run-time. Each data element should be separated by a space, a tab, or a new line (in the example shown to the left, each element is on a new line). The example is a text file consisting of the column of measured signals from the detector, copied from the .csv file which contained that column, with each measurement next to its corresponding values of x and x'. It is worth noting that this file could correspond to x-x' or y-y'; only the file name can indicate the plane. One crucial assumption made by the code in the case of the input file is that, supposing the slit takes n steps and therefore n measurements for each value of x', the first n data elements are taken for varying x and constant x'. Then, x' takes one step, and the next n data elements are again taken for varying x and the new constant x'. If they are reversed, with x' varying before x varies, MS Excel can sort the values first according to x' then according to x to put them in the correct order.

2.3. Parameters handled at run-time

In addition to its command-line arguments, the program requires a series of parameters at run-time to perform its calculations. Each time the program is run, it saves some of these parameters for use during the next run if, for example, the user chooses to use the same emittance measurements with a different number of particles, or chooses to align the distribution centroid with the coordinate axis. These parameters are saved to the file "lastUsedParams.txt", which is accordingly a filename reserved for use by the genBeam program.
These used and saved parameters are:

- Maximum and minimum values of x, x', y, and y', in meters and radians. These are usually found in the original emittance file (often of type .csv or .xls).
- Step size taken by the slit, in meters, in the case of x and y, and the angular increment, in radians, in the case of x' and y': these can be deduced from the original emittance file.
- Reference momentum of the original beam, which the program calculates from the energy that the user enters at run-time. The program assumes that all the particles are at the same energy.

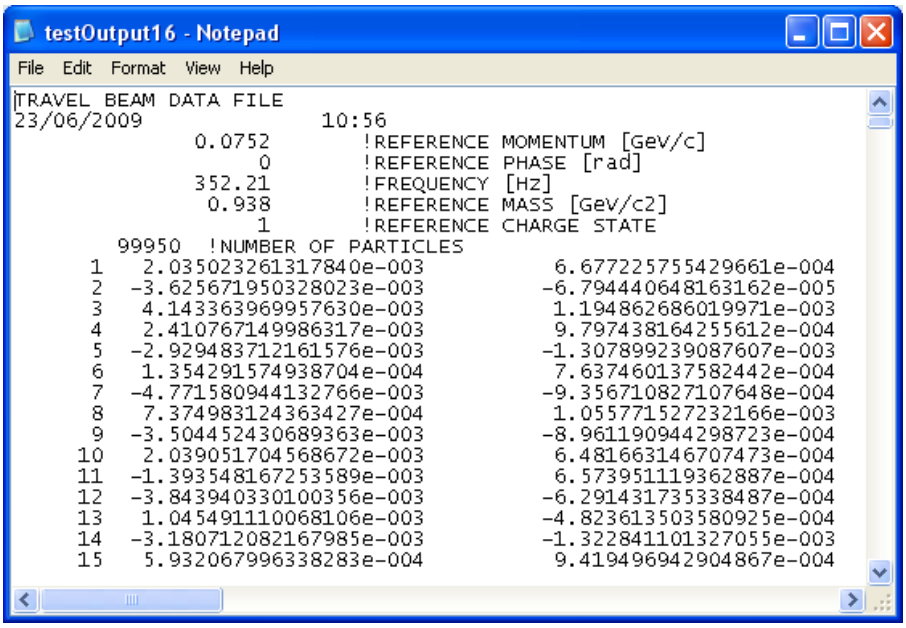
- Reference phase of the original beam: this is the phase relative to which the phase of each individual particle is measured. The program generates a uniform distribution of phase for all the particles on a 360º interval.
- Frequency: this refers to the RF frequency of the original beam if it is not continuous.
- Reference mass: this is the rest mass of a single particle in the original beam.
- Charge state: this is the charge on a single particle in the beam (+1 for protons, -1 for H^-, etc.).

The program requires the user to enter four more settings, which are not saved between runs. The first of these refers to the data in the *x* and *y* emittance data files: has the electronic noise already been removed? If not, the program will zero all measurements below a predefined threshold that varies according to the total amount of current (or number of particles) measured by the detector.

The second and third questions refer to the distribution of macroparticles that the program will generate: although we have not yet calculated any specific correlation in the cross-planes (*x*-*y*′, *y*-*x*′, *x*′-*y*′), the program has an option to ensure that the particles it generates fit inside upright ellipses in these three planes. The program also has an option to ensure that the particles fit in a four-dimensional hyper-ellipsoid that comprises the *x*, *x*′, *y* and *y*′ values of each coordinate.

The third setting is applied after the program has generated all the particles. It determines whether or not the particles in the beam data file created by the program are centred with respect to the mean values of *x*, *x*′, *y* and *y*′. In terms of this option, it should be noted that the beam can also be centred in PATH and the off-axis beam may be more useful to begin with.

2.4. Output beam data file



The genBeam program writes the six coordinates of the particles that it generates to a data file of type .txt that can be read by PATH Manager. This file can be made compatible with TRACEWIN software, or opened in Notepad or MS Excel. It is tabulated such that MS Excel can read it into rows and columns. The file contains a header that includes the reference values for momentum P_s, phase φ_s, RF frequency, mass *m*, and charge state Q, followed by 10 columns. The columns contain the particle number and its coordinates *x*, *x*′, *y*, *y*′, φ-φ_s, P-P_s, 0, Q and *m*. The example to

the left shows the header and the first 3 columns of the file. The program assumes that the momentum, charge, and mass will be the same for all the particles.

2.5. Error messages printed by genBeam

The genBeam user manual contains a much more detailed discussion on interpreting and resolving the error messages printed by the program. These error messages are printed as follows: `Error – [name of .cpp file] [error number]: [error message]`. The errors can be the result of one of two things. The file from which the program is trying to read or to which it is trying to write is open, and needs to be closed before it can continue, or the detector parameters that were entered do not match the data that the program attempted to load. Whenever one of these errors appears, if the program does not abort, the user should terminate the program by pressing ctrl + C (`^C` will be printed in the Command Prompt window) and begin again. If, during the last run, the parameters from the previous run were reused, it is recommended that the user check the input files and re-enter the detector and beam parameters to ensure that they are correct. The user should also make sure that the input and output files are closed, and that the file "lastUsedParams.txt", which is reserved by the system, has not been altered.

3. Program Design

The genBeam program was developed in the C++ programming language using Dev-C++ software (version 4.9.9.2). Its design was based on the ISISPrm2 program written by J. Stovall in the FORTRAN language. This section of the note discusses in some detail the assumptions the code makes, their implications, and the algorithms the code uses. We have attempted to make this section independent of programming language.

3.1. Assumptions made by the code

The genBeam program makes two major assumptions, one regarding the data file that it reads, and one regarding the phase and energy coordinates of the particles that it will generate. Any discussion of the histograms will be mentioned in terms of x and x' but since the y and y' histogram works the same way, will also apply to y and y'.

First we assume that, in the data file, if the detector slit took n different measurements for x at each value of x', the program expects the first n measurements to correspond to varying values of x at a constant x'. Therefore, if one were to picture the data as a matrix, with each column representing a constant x and each row representing a constant x', the program reads the matrix row by row, beginning in the top left corner. In the context of this assumption, it is worth noting that the program assumes that the steps both in x and x' are of uniform size.

Secondly, the program assumes that all the particles have the same energy (as found from the momentum) and are found uniformly distributed over 360^{o}. This concerns only the behaviour of the beam in the phase-energy plane, but would be an assumption worth correcting in the future.

3.2. Algorithm used by the code

Since C++ can be used as an object-oriented language, the genBeam program has two associated classes. The first of these is called origBeam. It is used primarily as a data structure, to read the emittance measurement files and to create normalized histograms of the emittance data. One origBeam object has space for an x-x' histogram and a y-y' histogram. The origBeam class also has a function to filter these histograms if the electronic noise has not already been removed. The second associated class is called randomNumGen, and, as its name implies, is a pseudo-random number generator that will generate random numbers, using the system time as a seed.

After processing the command line arguments, the program loads the beam parameters (beam energy, reference phase, frequency, mass, and charge state) and detector parameters (maximum and minimum values of, and steps in, x, x', y and y'). It reads these from the keyboard or from the file "lastUsedParams.txt", which has been saved from the previous run. The parameters entered from the keyboard are used in calculations to determine the internal parameters of the program. Since the emittance data exists in histogram bins of a sort, the minimum and maximum x and x' are indicators of the mean value of the bins at the outermost edges of the detector. Accordingly, if we define the range in x or x' to be the magnitude of the coordinate space covered by a slit or the magnitude of the angular divergence detected by a grid, accordingly, the range of x can be calculated by

$$\text{range} = \text{max} - \text{min} + \text{step}$$

where max and min are, respectively, the maximum and minimum values in the detector, and step is the step size between one x or x' measurement and the next. Once it has calculated the range, the program goes on to calculate the total number of steps in the range, also known as the number of histogram bins. From this it can determine how much space to allocate for the data that it will read from the file. It uses the formula

$$\text{bins} = \frac{\text{range}}{\text{step}}$$

to determine how many bins to allocate in x and x'. Also, since the program takes energy as input from the keyboard but PATH manager uses momentum, it calculates the momentum of the beam from the energy using the relativistic formula, where p is measured in GeV/c, W in GeV, and m_0 in GeV/c^2 (3)

$$cp = \sqrt{W(2m_0c^2 + W)}$$

The way in which the program allocates the bins and subsequently reads from the input file into requires the input file to be in a very specific format. The histogram is represented as a vector of vectors, a two-dimensional vector, and can be pictured as a matrix. The code uses a syntax to refer to a specific element of the matrix is matrix[column][row], and the syntax used to refer to a specific element in the histogram is histogram[x bin number][x' bin number]. If the order of the bin numbers is reversed, the program may attempt to access a location in memory that does not exist.

Once the code has read in the beam and detector parameters, it creates an origBeam object which contains the x-x' and y-y' distributions. The distributions are then

normalized such to sum of the contents of each distribution, and the program creates the histograms from which it will reconstruct the particle distribution by multiplying each element in each distribution by the number of particles that was entered in the command line rounded to an integer value. Because of the rounding, the program may generate slightly greater or fewer particles than specified.

Following the creation of the normalized histograms, the program asks the user whether the background noise was already filtered out of the data. If not, the origBeam object examines the original distribution and zeroes any bins containing less than 1% of the data, in order to provide a rough idea of where the maximum and minimum significant values of x and x' are found. If the background noise was already filtered out, the program continues to the next step.

At this point, the program asks the user if the reconstructed beam should fit in upright ellipses in the cross-planes (x-y', y-x', x'-y'). This sets a flag that determines whether the program should perform a comparison later on, when it is generating the particles. The program automatically fits the reconstructed beam to an upright ellipse in real space (x-y).

It should be noted that the method of fitting the generated particle coordinates to ellipses is less than ideal. The principle behind it is to exclude particles that do not fit within a certain specified ellipse. The program currently uses the filtered data to find maximum values of x, x', y and y' and uses these to define upright ellipses according to the one of the equations

$$\frac{x^2}{x_{max}^2} + \frac{y^2}{y_{max}^2} = 1$$
$$\frac{x'^2}{x'^2_{max}} + \frac{y'^2}{y'^2_{max}} = 1$$
$$\frac{x^2}{x^2_{max}} + \frac{y'^2}{y'^2_{max}} = 1$$

This method results in a vague ellipse traced by the outer ring of particles, with fairly distinct concentric rectangles in it corresponding to regions with a denser distribution of particles. The maximum values of x, x', y and y' are found by examining the filtered distribution as if it were uniform, finding its centroid, and finding the points furthest from the centroid. We have attempted to ameliorate the ellipse fit by adding another option which permits the user to fit the particles to a four-dimensional hyperellipsoid using the equation

$$\frac{x^2}{x_{max}^2} + \frac{y^2}{y_{max}^2} + \frac{x'^2}{x'^2_{max}} + \frac{y'^2}{y'^2_{max}} = 1$$

Once these parameters are entered, the program uses a randomNumGen object to generate random numbers on the x and x' ranges for x and x'. It calculates the indices in the histogram to which these values would correspond using the following algorithm:

- x Index $= \frac{x \cdot \text{number of bins in } x}{x \text{ range}}$

- In the unlikely event that the random number generated is equal to x range, this will cause the program to attempt to access a location in memory that does not exist (segmentation fault). Thus, this number is put in the bin containing the highest values.
- This same algorithm applies to x', y and y'.

After calculating the histogram indices for x and x', the program checks if the histogram bin specified by those indices has space left in it. In this version, for a histogram bin to have space, the value in it must be greater than 0. A histogram bin with a value of 0 has already been filled, if it had space for any particles to begin with. If the x-x' histogram bin has no space, the program goes back and generates new values of x and x'. However, if the x-x' histogram bin has space available, the program repeats the process of generating values for y and y', and of checking if there is space in the y-y' histogram. After finding a combination of coordinates that can be accommodated by the bins in the two histograms, the code checks to see if the x and y coordinates fit in the defined upright ellipse and, if appropriate, the x and y', y and x', and x' and y' coordinates fit in upright ellipses in those planes, and the four coordinates in the hyperellipsoid.

If any one of these conditions is not fulfilled, the program throws out the four coordinates, and starts over at generating x and x' coordinates. If all the conditions are fulfilled, however, then the program adds the coordinates of the particle to the internal vector in which they are being stored (precisely, it is a two-dimensional vector of type double values). Each element, which is itself a vector, in this two-dimensional vector represents one particle. The index of the element is one less than the particle number, and the elements inside each element of the two-dimensional vector are, in order, the x, x', y, y' coordinates, the phase with respect to the reference phase, the momentum with respect to the reference momentum, the charge on the particle, and the mass. Note that the charge and mass are the same for every particle generated, and the momentum with respect to the reference momentum is hard-coded into the program as 0, since the program assumes that all the particles have identical momentum.

Once the coordinates of the particle have been added to the internal vector, the program marks the appropriate x-x' and y-y' histogram bins as having one particle added to them, and increments a counter which tracks the number of particles generated.

It repeats the loop of generating coordinates, checking if they fit, and adding them to the internal vector until either the specified number of particles is reached or, more commonly, either the x-x' or y-y' histogram has filled up, so every value in the histogram is equal to 0. This causes any further particles generated to be rejected. As a result of these exit conditions, the total number of particles generated may be lower than the number of particles that the user requested in the command line.

Once all the particles have been generated, the program asks the user if the particles in the beam data file should be aligned according to their mean in x, x', y, and y'. This aligns the beam centroid, calculated from the particles actually generated, with the coordinate (accelerator) axis. If the user selects yes, then the program subtracts the

appropriate mean from each coordinate in the internal vector. If the user says no, the program does nothing.

The program then prints the beam data file header and all the data, in the format specified by PATH Manager, to the output file. Finally, it calculates ε_{rms}, $\varepsilon_{rms,n}$, and $\varepsilon_{99\%}$. It calculates the geometric rms emittance using the formula

$$\varepsilon_{rms} = \sqrt{\overline{x^2} * \overline{x'^2} - \overline{xx'}^2}$$

and multiplies it by the relativistic $\beta\gamma$ to normalize it. The program finds the relativistic factors β and γ using the formulae

$$\beta = \frac{p}{\sqrt{c^2 m^2 + p^2}}$$

$$\gamma = \frac{1}{\sqrt{1 - \beta^2}}$$

where p is the reference momentum, m is the rest mass, and c is the speed of light (which, since the momentum is in units of GeV/c and the mass in units of GeV/c^2, is 1 in this calculation.). To find the 99% emittance, the program calculates the Twiss parameters a, β, and γ, in each plane, using the formulae (4)

$$\beta = \frac{\overline{x^2}}{\varepsilon_{rms}}$$

$$\gamma = \frac{\overline{x'^2}}{\varepsilon_{rms}}$$

$$\alpha = \pm\sqrt{\beta\gamma - 1}$$

To determine if a is positive or negative, the program looks at whether the values of x' is positive or negative at the maximum value of x. If x' is positive (i.e. the beam is defocused in x), then a is negative. If x' is negative (i.e. the beam is focused in x), then a is positive. Using the parameters, it calculates an emittance ellipse for every single particle in the internal vector and stores its value in a vector of type double. It then sorts this vector and eliminates the highest 1% of values, and returns the maximum value to give the 99% emittance.

4. Program Testing

The genBeam program was tested for functionality as it was developed, using a Gaussian distribution created in MS Excel and an x-x' emittance measurement from Linac 2. After the program was completed, it was further tested using emittance measurements created from beam data. These later tests used a MATLAB routine to sort the particles from the original beam data into histogram bins comparable to emittance measurements (5) (6). Then genBeam was used to generate a particle distribution, from those emittance measurements, to compare with the original beam data. The primary purpose of these tests was to ensure that the program functions correctly, given an input file in the correct format and the corresponding parameters, and, to a lesser extent, to see how close the reconstructed beam comes to the original.

The output beam data files were first tested with PATH Manager, to ensure that the files were printed in the correct format. PATH read the files as anticipated and returned the same emittance values (rms and normalized rms) as genBeam. The only

way to verify genBeam's calculation of $\varepsilon_{99\%}$ was to note that it falls between the total and 95.4% emittances calculated by PATH. Also, the x-x' and y-y' coordinates were graphed in scatter plots in MS Excel to show that the distribution in these planes was similar to what was shown in PATH. Scatter plots of x-y, x-y', y-x' and x'-y' were also made in Excel, and the rectangles within the elliptical distributions were first observed at that point.

Figure 4.i: Excel surface graph of an emittance measurement from Linac2.

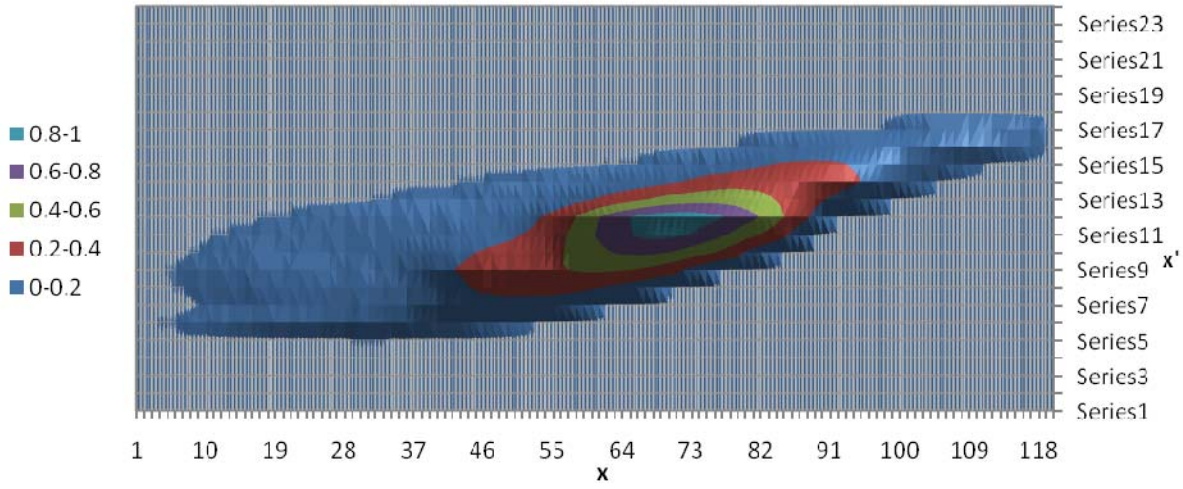


Fig. 4.i, above, shows a contour plot of the emittance data from Linac 2 used in earlier testing. In this test, since no y-y' data was available, this measurement was used for both planes. The slit step size, in this case, was 0.1mm, and the angular increment was 0.3mrad. The x values ranged from -5.95mm to 5.95mm and the x' values ranged from -3.45mrad to 3.45mrad. Consequently, there were 120 histogram bins in x and 24 in x'. The axis labels in fig. 4.i represent the bin numbers. The resultant generated particle distribution is shown in fig. 4.ii, below. The normalized rms emittance ellipse is shown on fig. 4.ii; PATH's calculation and genBeam's calculation return the same value for both the geometric and the normalized rms emittances.

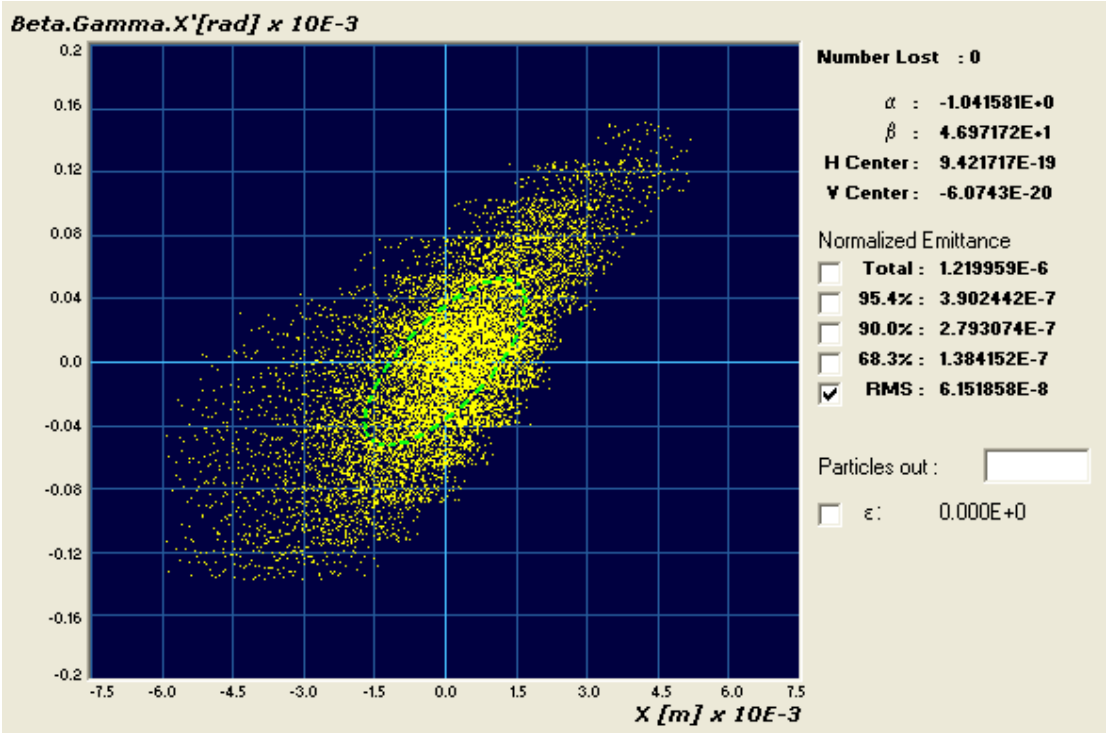


Figure 4.ii: PATH display of generated particle distribution from Linac2 measurement

Different numbers of macroparticles were tested in the command line arguments, ranging from 100 particles to 100,000 particles. These tests found that, in general, for fewer than 1,000 particles, the number of particles generated varied significantly from the number requested. The variation ranged between 49 and 58 fewer particles than requested, using the distribution shown in fig. 4.i. A similar number of particles was missed when the program was requested to generate 5,000 particles, but as a percentage of the total it is less significant than when the request was 1,000 particles. The variation in the number of particles generated depends on the distribution as well as the total number of particles, and it is assumed that the program will be used to generate large numbers of particles. These tests also showed that the program may take several minutes to generate 100,000 particles in the requested distribution, and that, for greater numbers of particles, the resolution of the original emittance measurement becomes more apparent in the generated particle distribution. The generated distribution shown in fig. 4.ii has only 10,000 particles, and the grain of the original measurement is pronounced. The time taken by the program to generate the particle distribution depends also on whether the ellipse or hyperellipsoid constraints are in effect.

Table 4.i: Parameters used in generating an emittance measurement from beam data

	x [mm]	x' [mrad]	y [mm]	y' [mrad]
max. value	-2.3	-36	-6.3	-16
min. value	2.3	36	5.5	16
size of increment	0.1	1	0.1	1

The later tests had mixed results in their comparison of the reconstructed beam to the original beam; some of the results are shown following. Using the parameters in table

4.i, the original beam data was sorted into 47 histogram bins in x, 73 in x', 119 in y, and 33 in y' (5). This resolution of the histogram is not from a real measurement; it is based on the beam data. In general, the reconstruction of the emittance measurements in x-x' and y-y' was very close to the original, while the upright ellipse-fits in real space and the cross-planes were markedly more rectangular than the original. This is shown in figs. 4.iii and 4.iv, below (5). The graphs in each figure are, clockwise from top left: x-x', y-y', x-y, x'-y'. The difference in intensity between the two beams, shown by fewer red particles at the centre of the reconstructed beam, is a result of using only 50,000 particles in the simulation, while the original beam contained 87,671 particles. Beyond this difference, the original and resultant beams in x-x' and y-y' are close to identical. The major differences in shape occur in real space and in the cross-planes. In the original, the normalized emittance in x-x' is 0.2597 π.mm.mrad, and in y-y' the normalized emittance is 0.2760 π.mm.mrad. The normalized emittance of the generated beam, by contrast, is 0.2670 π.mm.mrad in x-x' (2.8% greater than the original) and 0.2856 π.mm.mrad in y-y' (3.4% greater than the original). These emittances were all calculated in TRACEWIN.

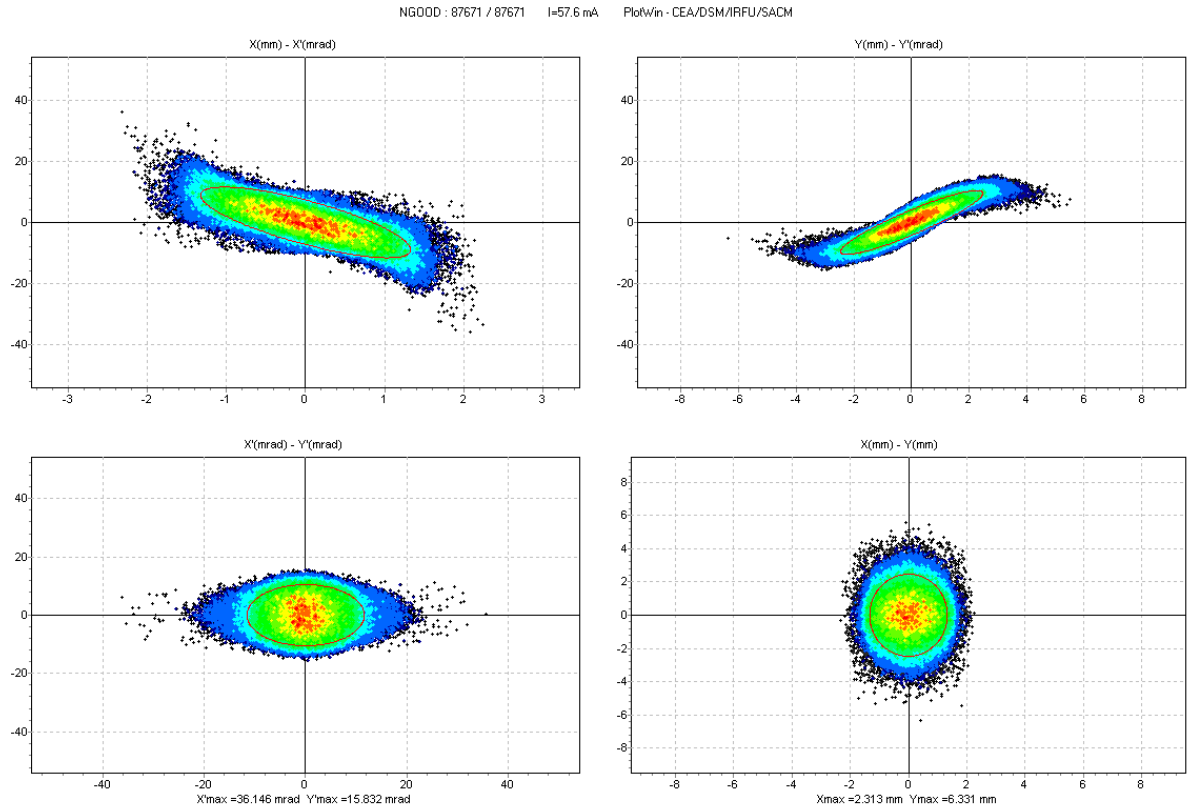


Figure 4.iii: Original beam data used in one of the later tests

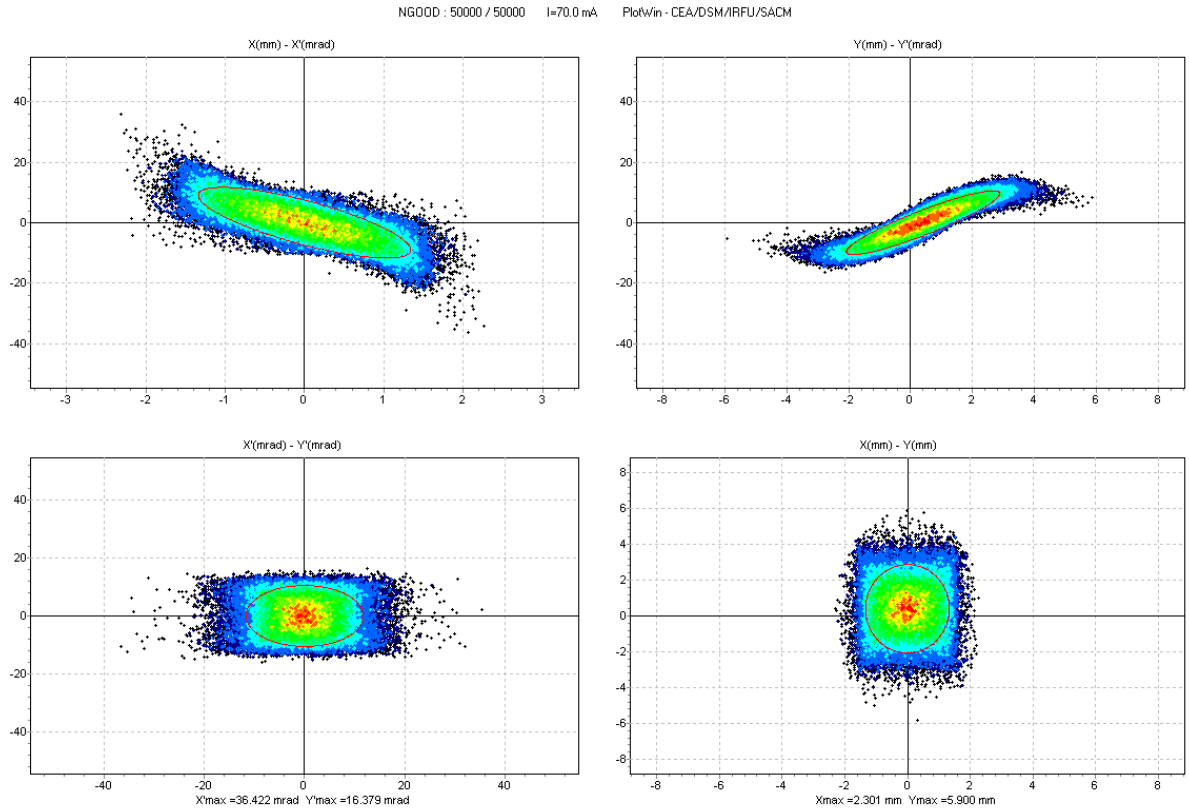


Figure 4.iv: Generated beam from data in fig. 4.iii

5. Further Development

The genBeam program has some potential for improvement. The most significant place in which the program could be changed is in terms of the phase-energy plane. Presently, that plane is handled on two assumptions. However, the code could be modified to handle a bunched beam, in the event that longitudinal data becomes available, for example, following an RFQ. This could involve adding a third histogram matrix to keep track of phase and energy, or perhaps something different, depending on the form of the longitudinal data.

6. Conclusion

We have written and tested a program that will generate a particle distribution based on emittance measurements. It was tested with an emittance measurement from Linac2 and with emittance measurements created by generating a histogram from beam data. These later tests showed that the emittance of the particle distribution generated by genBeam differed about 3% from the emittance of the original particle distribution, before it was put in a histogram. From these tests, we have concluded that this program is useful for generating a particle distribution from real emittance measurements to be used in particle-tracking codes, in place of a uniform or Gaussian distribution for the beam.

7. Bibliography

1. **Perrin, A., Amand, A.F.** *Travel v4.06, user manual.* s.l. : CERN, 2003.

2. *CEA Saclay Codes Review*. **Duperrier, R., Pichoff, N., Uriot, D.** Amsterdam : s.n., 2002. ICCS Conference.
3. **Lapostolle, P. and Weiss, M.** *Formulae and Procedures Useful for the Design of Linear Accelerators.* s.l. : CERN, 2000. CERN-PS-2000-001.
4. **Rusthoi, K.R. Crandall and D.P.** *TRACE3-D Documentation.* s.l. : Los Alamos National Laboratory. LA-UR-97-886 (third edition, May 1997).
5. **Lallement, J.-B.** private communication.
6. **Sargsyan, E.** private communication.