THE BEAM BREAK-UP NUMERICAL SIMULATOR

Gil A. Travish

*Accelerator and Fusion Research Division
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720*

November, 1989

MASTER

THE BEAM BREAK-UP NUMERICAL SIMULATOR***Gül A. Travish***Lawrence Berkeley Laboratory, University of California, Berkeley, California 94720***ABSTRACT**

Beam Break-Up (BBU) is a severe constraint in accelerator design, limiting beam current and quality. The control of BBU has become the focus of much research in the design of the next generation collider, recirculating and linear induction accelerators and advanced accelerators. Determining the effect on BBU of modifications to cavities, the focusing elements or the beam is frequently beyond the ability of current analytic models. A computer code was written to address this problem. The Beam Break-Up Numerical Simulator (BBUNS) was designed to numerically solve for beam break-up (BBU) due to an arbitrary transverse wakefield. BBUNS was developed to be as user friendly as possible on the Cray computer series. The user is able to control all aspects of input and output by using a single command file. In addition, the wakefield is specified by the user and read in as a table. The program can model energy variations along and within the beam, focusing magnetic field profiles can be specified, and the graphical output can be tailored. In this note we discuss BBUNS, its structure and application. Included are detailed instructions, examples and a sample session of BBUNS. This program is available for distribution.

* Work supported by the Office of Energy Research, U.S. Dept of Energy, under Contract No. DE-AC03-76SF00098

TABLE OF CONTENTS

[1] INTRODUCTION	1
[2] BBU and WAKES.....	2
[3] THE CODE STRUCTURE.....	6
[3.a] The Input File.....	7
[3.b] The Main Code.....	9
[3.b.1] Code Timing.....	15
[3.c] The Output Code	15
[4] INSTRUCTIONS.....	18
[4.a] Input Command File	18
[4.a.1] The Text Category	22
[4.a.2] The General Category.....	22
[4.a.3] The Structure Category	23
[4.a.4] The Wake Category	23
[4.a.5] Pulse Category	24
[4.a.6] Energy Category.....	25
[4.a.7] Lambda-Beta Category	26
[4.b] Control of Output.....	26
[4.b.1] The General Command Revisited.....	27
[4.b.2] The Output Command Classification	28
[4.b.3] Multiple Frame Plots	31
[4.b.4] Three Dimensional Plots.....	31
[4.b.5] Logarithmic Plots	32
[4.c] Summary of Output Commands	33
[4.d] Error Handling.....	33
[4.e] Advanced Features	34
[4.e.1] Multiple Plot Sets	35
[4.e.2] Multiple Runs	36
[5] CONCLUSIONS.....	38
PROGRAM AVAILABILITY	39
ACKNOWLEDGEMENTS.....	40
REFERENCES AND NOTES	
APPENDIX: sample run	

TABLE OF FIGURES

Figure 1.	The two particle model.....	2
Figure 2.	Wakefield BBU in a beam	3
Figure 3.	A sample section of the beam pipe.....	4
Figure 4.	A sample cross-section.....	5
Figure 5.	The organization of the BBUNS code.....	6
Figure 6.	A few sample lines from an input file	8
Figure 7.	A flowchart of the main code	10
Figure 8.	The truncation length.....	14
Figure 9.	The discrete kick method.....	15
Figure 10.	A flowchart of the output code	17
Figure 11.	Anatomy of a command line for BBUNS.....	18
Figure 12.	Anatomy of an option line for BBUNS.....	19
Figure 13.	The relation between the command and option line...	20
Figure 14.	The two basic types of structures allowed.....	23
Figure 15.	The three types of initial beam offsets in BBUNS.....	25
Figure 16.	Anatomy of the output command classification	29
Figure 17.	Anatomy of a command line for a three dim. plot.....	32
Figure 18.	Sample lines from a two run input file.....	37

LIST OF TABLES

Table 1.	A list of notation used in this paper	4
Table 2.	A list of command classifications.....	20
Table 3.	A list of the additional commands.....	27
Table 4.	Devices, libraries and device values.....	27
Table 5.	Axis codes for use in the output command	30

INTRODUCTION [1]

Transverse beam break-up (BBU) has been addressed in many papers analytically,^{1,2,3} numerically^{4,5,6} and experimentally.^{7,8,9} BBU is an instability which plagues many current accelerators,¹⁰ and must be considered in all future designs.¹¹ BBU can lead to a degradation in beam quality, and “pulse shortening”, or loss of current due to the beam hitting the wall. Ultimately, BBU imposes stringent limits on current and makes problematic designs which, otherwise, would be quite promising.¹² In all cases, it is desirable to reduce BBU as much as possible.

The equations describing the transverse motion of the beam under the influence of the wakefield cannot be solved exactly and asymptotic solutions are available only for simple analytic models.¹³ For all but the most idealized conditions, these calculations require the use of computers. Employing a computer code also allows for calculations of a more physical nature. However, solving for the dynamics of BBU requires a great deal of computation: the longer the beam (pulse), and the longer the accelerator, the more intensive the calculation.

This paper describes a program which numerically solves the BBU dynamics due to an arbitrary transverse wakefield. The program has been designed to accommodate effects such as Landau¹⁴ and BNS damping,¹⁵ as well as realistic current variations along the macro-pulse. The wake must be pre-specified either analytically or by a numerical field solver.¹⁶

The organization of this paper is as follows. In the next section, the theory of cumulative BBU is discussed. The third section deals with the code: its capabilities, structure, input requirements and output format. The fourth section explains how to use the code. This is followed by a conclusion, a final section on where to obtain the code, and acknowledgements. At the end of the paper there is an Appendix which lists an input file along with a sample session and output from the code. ♦

BBU and WAKES [2]

A simple model of cumulative¹⁷ beam break-up due to wakefields is the two particle model with the particles traversing a structure on the beam-line. A particle traversing the structure excites RF fields which “ring” for some time after the test particle has left the cavity. A second test particle following behind will be deflected by the fields excited by the first particle (see Figure 1).

Figure 1. An off-axis particle traverses a pillbox cavity and excites wakefields. A second particle following the first one receives an impulse from the $\mathbf{v} \times \mathbf{B}$ force of the RF fields excited through the $J_z E_z$ coupling of the first.

If we now consider a beam of particles traversing several cavities, a more complete picture of wakefield BBU in induction linacs can be ascertained. The beam may be thought of as a stream of slices. Each slice will excite TM_{1n0} -like modes in the structure, coupling through the dipole moment of the axial current density to the axial electric field of the modes. The transverse B-field excited by the n th slice will deflect the $(n+1)$ th slice via the $\mathbf{v} \times \mathbf{B}$ force. The further off-axis a slice is, the greater its dipole-moment-induced fields in downstream cavities will be. These larger fields cause larger impulses and an instability ensues (see Figure 2).

Figure 2. Wakefield BBU in a beam. A beam can be considered as a stream of slices. The first slice excites a field which deflects the $n-1$ slices which follow. The second slice excites a field which deflects the $n-2$ slice to its rear, etc..

Large instabilities can lead to the beam scraping the beam pipe wall causing beam break-up or “pulse shortening”. Even small BBU growth is unwelcome since it increases the emittance.

In order to calculate beam dynamics due to BBU, the wakefield is introduced. The wakefield is the coupling of the RF fields in a particular structure to the motion of the beam centroid. The wakefield of a particular structure is then a measure of the effect a particle will have on a test-particle that follows it in this structure. More explicitly, the wakefield is the integral of the Lorentz force,¹⁸

$$\vec{W}(s) = \frac{1}{\Delta z Q} \int_{-\infty}^{\infty} dz \left[\vec{v} \times \vec{B}(z,t) \right] \Big|_{t = \frac{z+s}{c}} \quad (1)$$

where Q is the charge of the test particle, Δz is the length over which the structure's fields extend¹⁹, and the remaining notation is given in Table 1.

Table 1. A list of notation used in this paper. MKS units are listed in the third column.

γ	energy	[-]
z	structure coordinate	[m]
ρ	charge density	[C/m]
$W_{\perp}$	transverse wakefield	[V/C/m]
λ_{β}	betatron wavelength	[m]
k_{β}	betatron wave-number, $k_{\beta} = 2\pi/\lambda_{\beta}$	[m ⁻¹]
ξ	transverse displacement from pipe axis.	[m]

Although definitions of the transverse wakefield vary,²⁰ this paper and the computer code use a wakefield with units (in MKS) of Volts/Coulomb/meter/meter.²¹

Figure 3. A sample section of the beam pipe showing the use of the s and z coordinates. The z coordinate measures the distance down the accelerator. The s coordinate measures the distance along the beam starting from the beam head. The insert shows the ξ coordinate.

We may describe the motion of the beam center of mass driven by a transverse wake according to²²

$$\frac{d}{dz} \left[\gamma(s,z) \frac{d}{dz} \xi(s,z) \right] + k_\beta^2(s,z) \gamma(s,z) \xi(s,z) = \frac{e}{mc^2} \int_0^s ds' \rho(s') W_+(s'-s) \xi(s',z) \quad (2)$$

where, again, the notation is as in Table 1 and the geometry is depicted in Figure 3.²³ The equation implicitly assumes that the longitudinal position of the beam particles remain fixed relative to one another.²⁴

Each of the parameters in the equation may vary in s and z . Thus, variation of energy along the pulse may be used to study BNS damping.²⁵ In addition, each slice may be modelled by a number of macroparticles indexed by i (see Figure 4). In this way, energy variation within a slice may be used to study Landau (phase-mix) damping.²⁶

Figure 4. A sample cross-section, or “slice”, of the beam indicating the distribution of macro-particles within a slice.

To solve the BBU equation, BBUNS requires the user to specify the variation of γ and k_β in s and z . The wake and the initial conditions ξ , $d\xi/dz$ at $z=0$ are also specified by the user. BBUNS then integrates the initial value problem in z , numerically.²⁷

THE CODE STRUCTURE [3]

The code structure is outlined in the chart of Figure 5. There are two separate programs used: a calculation code, and an output code. In addition, there is an input (command) file and a wake file which must be supplied by the user. The user may alter the input file with any standard editor. This enables the user to rapidly make alterations to the input file without having to run an input code or go through many menus.

Figure 5. The organization of the BBUNS code. First, the input file is edited by the user. Next, the input file is read in by the main code. In turn, the main program generates an output file. This output file is used by the output program to generate tables and graphs.

The main code, *BBUNS*, is divided into several subroutines. Basically, *BBUNS* reads in the user's input file (*bbu.idat*) and the file which contains the wake potential²⁷ (*bbu.wake*) then it proceeds to solve the problem. Upon completion, it writes out a file containing the original inputs and the results (*bbu.odat*). The second program, *BBUNSOUT*, accepts the user's commands from the input file, reads

in the output file (*bbu.odat*) and then processes and formats the data for output (graphs, tables, etc.). Again, refer to Figure 5 for an outline of the program hierarchy. The input file, *bbu.idat*, is a standard Fortran formatted file, while the output file, *bbu.odat*, is unformatted. The formatted input file allows for alterations with any editor. The output file is much larger and thus the rapid processing ability of an unformatted file is used.

All code is written in Fortran and is able to run on either the Cray X-MP or the Cray 2 running under CTSS (other operating systems should work as well, but have not been tested). The next three sections detail the structure of the input file, the main code, and the output code.

THE INPUT FILE [3.A]

The input file enables the user to input the numerical parameters for each run²⁸ as well as specify the format and type of output. In fact, the input file is the user's only means of affecting the program execution. Neither the main code nor the output code requests user input during execution.

The format of the input file is best understood by examining a few sample lines of an input file. Figure 6 shows a few model input lines. For a complete input file see the Appendix.

Although there is a great deal of text in the sample, only the text to the left of the exclamation marks is important. The input file is formatted in a highly structured manner. The leftmost two characters indicate a *command classification*. If the classification is recognized by the program, then the rest of the line is analyzed based on the classification of that line. This enables the program to have more than one format of input. If the command is not recognized, as in the text "**Structure** parameters [1]" (the bold characters are examined first), then the line is simply ignored. This allows all non-command lines to be comments for the user's benefit.

Immediately after the command classification is a two digit *command number*. This command number tells the program what piece of data is being given. For instance, the first command line (the preceding lines are not commands) in the sample (Figure 4) starts with, $t>01$. This indicates that the line contains the data for the first datum of the command classification t (for text). The first datum of t , $t>01$, holds the text for the name of the particular run. A second example is the command line that begins with $g>04$. This indicates that the fourth datum of the g classification follows. In this case the program will recognize the command as specifying the number of particles to use.

```

Text data
-----
t>01      NAME=TEST1A      !Name of Run
t>02      DESCRIPTION=Example for instructions  !Description
t>03      OUTFILE=tw.odat  !Output file name

General parameters
-----
g>01i     NZPOINTS=14500   !NZPOINTS: number of Z points.      [-]
g>02i     IZSTEP=2        !IZSTEP:increment between Z steps saved [-]
g>03i     NPS=300         !NPS: number of s points to save      [-]
g>04i     NPART=1         !NPART: number of particles per slice  [-]

Structure parameters [1]
-----
s>02      ZMAX=90.00      !ZMAX: total length of structure    [Meters]
s>03      PIPERAD=7.62e-2 !The pipe radius                  [Meters]
s>01i     ICELOPT=1       !Cell Option number
----- Option #1 gives 1 kick per Cell
s>01/01   SPERIOD=0.5      ! The length between cavities    [Meters]
----- Option #2 gives kicks while in the gap
s>01/02   SPERIOD=1        ! Piece #2 of data for option #1  [Meters]
s>02/02   SGAP=1.0        ! Piece #1 of data for option #1  [Meters]

```

Figure 6. A few sample lines from an input file for the BBUNS code.

On the same line as a legitimate command classification and number is a *keyword* followed by an equal sign (=). The keyword (i.e. NPART in the last example) is an aid to the user. The program will ignore any text between the command number and the equal sign. The user could have typed any keyword,

```
q>04i  PARTICLES=1  !Number of particles per slice [-]
```

or equivalently,

```
q>04i  =1              !Number of particles per slice [-]
```

and the code would still read in the number of particles. The "i" which follows some of the command numbers is another optional aid to the user. It indicates that the input will be read as an integer (i.e. only an integral number of particles is allowed).

Not only is any text left of the equal sign, the keyword, ignored but so is any text right of the 20th column (40th for the τ classification, 60th for the ϕ classification). In other words, the user can place addition comments to the right of the data. In the example in figure 6, exclamation points are used to offset the comments from the data; however, the use of a delimiter such as the exclamation point is not required.

The code only considers the numbers or text which immediately follow the equal sign. Thus, in the above example, the "1" would be read in and assigned to a variable which holds the number of particles. The input file is an assembly of such lines and comments. All parameters necessary for a run are entered in a similar fashion. More detailed information on the input file will be given in section [4].

THE MAIN CODE [3.B]

A flowchart of the main code is given in Figure 7. On the left side of the figure are a series of rectangles. Each of the rectangles represents a subroutine. These subroutines are called once at the beginning of the run to initialize variables, fill static arrays, and read the input and wake file. On the right of the diagram are shaded areas which represent main loops of the code. The code has three types of loops: the main loop in z (along the accelerator), the sub-loops for the particles within each slice, and the loops in s (over all slices along the beam).

Figure 7. A flowchart of the main code, BBUNS. The shaded rectangles represent the main loops of the program. The rectangles represent subroutines.

In order to solve the equation for an arbitrary wakefield, it is necessary to divide the beam into slices. These slices, or macro-particles, are the “mesh” of the ‘s’ (beam) dimension. A mesh is also

applied to the 'z' (beam-line) dimension. The user is able to specify the number of mesh points along the beam-line. This is equivalent to specifying the number of points per λ_β . The mesh along the beam is determined by the wakefield data file. Since a variety of spacings are used by the various programs which generate wakefield data files, the BBUNS code must be flexible enough to accept any spacing. For simplicity, the program does not employ adaptive step sizes since this would require a great deal of interpolation. The wakefield data pairs must be equally spaced.

After reading in the wake file and performing the variable initializations, BBUNS begins executing the main loops required to solve the diffeo-integral equation at all the points along the beam mesh and beam-line mesh. There are two main steps necessary to accomplish this. The first step is computing the integral,

$$I(s) = \int_0^s ds' \rho(s') W_1(s'-s) \bar{\xi}(s', z) \quad (3)$$

where ξ is averaged over all the particles within a slice,

$$\bar{\xi}(s', z) = \sum_{i=1}^N \xi_i(s', z) \quad (4)$$

to obtain the force on particles within the beam slice at s. This integration is performed using a Gaussian quadrature algorithm.

The second step requires solving the second order differential equation,

$$\frac{d^2 \xi(s, z)}{dz^2} = \frac{e}{mc^2 \gamma(s, z)} I(s) - k_\beta^2 \xi(s, z) - \frac{d\xi(s, z)}{dz} \frac{d \ln \gamma(s, z)}{dz} \quad (5)$$

or equivalently, the two first order equations,

$$\begin{aligned} f(s, z) &= \frac{d\xi}{dz} \\ g(s, z) &= \frac{df(s, z)}{dz} = \frac{d^2 \xi}{dz^2} \end{aligned} \quad (6)$$

This is accomplished by use of a fourth order Runge-Kutta algorithm.

The Gaussian integration scheme considers a function f to be integrated on $[a,b]$. Gauss' formula (without remainder) is,

$$\int_a^b f(y) dy = \frac{b-a}{2} \sum_{i=1}^n w_i f(y_i) \quad (7)$$

with,

$$y_i = \left(\frac{b-a}{2}\right)x_i + \left(\frac{b+a}{2}\right) \quad (8)$$

where x_i is the i th root of the Legendre polynomial, $P_n(x)$, and,

$$w_i = \frac{2}{(1-x_i^2)} [P'_n(x_i)]^2 \quad (9)$$

is called the i th "weight"²⁹. The algorithm can be of arbitrary, n th, order. The implementation of this method is separated into three parts (and thus three subroutines). The first routine calculates the roots and weights. The second routine interpolates the function, f , between $[a,b]$ for the n sample points necessary. The interpolation is performed using a four-point Lagrangian method.³⁰ In order to solve for $I(s)$, three functions must be interpolated separately: $\rho(s)$, $\xi(s,z)$ and $W(s)$. Finally, the third routine performs the sum and returns the value of the integral.

The Runge-Kutta algorithm for advance in z is,^{31,32}

$$\begin{aligned} f_{n+1} &= f_n + \frac{1}{6} [k_1 + 2(k_2) + 2(k_3) + k_4] \\ g_{n+1} &= g_n + \frac{1}{6} [l_1 + 2(l_2) + 2(l_3) + l_4] \end{aligned} \quad (10)$$

with the coefficients given by

$$\begin{aligned}
k_1 &= hf_n(x,y,z) & l_1 &= hg_n(x,y,z) \\
k_2 &= hf_n(x+\frac{h}{2},y+\frac{k_1}{2},z+\frac{l_1}{2}) & l_2 &= hg_n(x+\frac{h}{2},y+\frac{k_1}{2},z+\frac{l_1}{2}) \\
k_3 &= hf_n(x+\frac{h}{2},y+\frac{k_2}{2},z+\frac{l_2}{2}) & l_3 &= hg_n(x+\frac{h}{2},y+\frac{k_2}{2},z+\frac{l_2}{2}) \\
k_4 &= hf_n(x+h,y+k_3,z+l_3) & l_4 &= hg_n(x+h,y+k_3,z+l_3)
\end{aligned} \tag{11}$$

and where h is a mesh size (in this case Δz).

This method solves for the $(n+1)$ th step (in z), given the n th step. The following formulas are the two first order equations obtained from the second order equation.

$$\begin{aligned}
f'_n &= \frac{d\xi_n}{dz} \\
g'_n &= \frac{df'_n}{dz} \\
&= \frac{e}{mc^2\gamma(s,z)} \int_0^s ds' \rho(s') W_{\perp}(s'-s) \bar{\xi}(s',z) - k_{\beta}^2 \gamma(s,z) \xi(s,z) - \frac{d\xi(s,z)}{dz} \frac{d\ln\gamma(s,z)}{dz}
\end{aligned} \tag{12}$$

The Runge-Kutta solver is divided into four routines, one routine for each pair of coefficients.

Beyond the inherent efficiency of the Runge-Kutta and Gaussian quadrature algorithms there are a number of features of the code which were designed to minimize run time. All attempts have been made to fully optimize and vectorize³³ the code. In addition, there are some user specifiable controls which can minimize run time. The first control allows the user to specify how far down the beam the effects of the wakefield will be calculated. This truncation length allows a user to include the physics while excluding much of the numerically intensive calculations. This is especially useful for long wakefields which decay rapidly or have a high frequency, low amplitude “ring”³⁴. By appropriately setting the truncation length, the user can have the code ignore the small contributions from the “tail” of the wakefield.³⁵ This enables the code to run much faster when the wakefield data contains more points than necessary for numerical accuracy. In the event that the user does not want to shorten the effect of the

wakefield, the truncation length simply becomes the lesser of either the length to which the wakefield is defined or the pulse length. Figure 8 shows the s and z coordinates along with the truncation length.

Figure 8. In addition of the s and z coordinates (previously described), there is a truncation length which allows the user to limit the effective length over which the wakefield induced forces act.

The most time saving feature of the code is also of physical significance. The integral, $I(s)$, represents a driving force. This driving force is only exerted on the beam slice while in, or very near, the accelerating gap.³⁶ On the scale of the accelerating structure the gap spacing is very small (see Figure 9). This enables, even requires, that the driving force only be applied to the beam for a very short distance in the structure.³⁷ This physical requirement is fortunate from a numerical stand point. It requires the code to calculate the integral only a few times per slice for each accelerating gap.³⁸ This *discrete kick* method reduces the code execution time and yields a more physical result. Another time-saving feature allows the user to specify that only every i th point of the wake be used. Thus, if the wake is specified by too fine a mesh in s , it may be “thinned out”.

Figure 9. The typical spacing (mesh) in z may be larger than the gap size of LIA's. A discrete kick is applied to the beam only when the beam is within a Δz of the gap. This produces a more physical result and saves computing time.

CODE TIMING [3.B.1]

The result of all these features is that the program is able to execute very quickly. An order of magnitude estimate of the cpu time follows from a simple formula,

$$t = \Delta t n_c n_t n_s \quad (13)$$

where n_c is the number of cavities, n_s is the number of mesh points in s , n_t is the number of mesh points within the truncation length, and Δt is a time constant whose current value is approximately 0.4 μsec . As an example, consider a typical run with 100 cavities, 2500 points in s and 1000 points along the wake (i.e. $n_t=1000$). This yields a run time of 100 seconds (unweighted CRU's³⁹). In addition there is always some overhead due to I/O and memory charges.

THE OUTPUT CODE [3.C]

The output code is responsible for producing human-readable information in the form of 2D and 3D plots, and tables. The graphics are produced using the DISSPLA[®] graphics library. The plots may be

sent to the screen for immediate viewing or to a graphics file for printing on paper, film or other media. The user has complete control of what the program will plot through commands specified in the input file. Any variable may be plotted against any other variable. There are three categories of plots. The first category allows a user to do simple 2D plots. The user simply specifies the y-axis variable, the x-axis variable, and optionally the plot value range. The second category allows the user to do a series of 2D plots. For instance, a user might want to plot γ vs. z at 10 points along s . This is easily done by specifying how many frames the program should plot. The third category allows for 3D plots.

In addition to being able to specifying what data the program should plot, the user has several “knobs” to adjust how the plots will look. These controls will be discussed in section [4].

The structure of the output code is rather simple. After reading in the output file (*bbu.idat*) that the main program generated, it scans the input (command) file for relevant commands. The program then proceeds to print out the tables and graphs as the user requested. The plots are generated using a general routine which accepts the x- and y-axis (and z-axis for 3D plots) data.

A simplified flowchart of the output code is given in Figure 10. The program uses a routine to choose the two or three arrays to plot according to the information read in from the input file. The subroutine then calls up the appropriate graphing routine (2D or 3D) to generate the plots. Depending on the user's choice, the graphs will either go directly to the screen or be saved in a graphics file.

The output code can be easily be re-executed to produce additional graphs and tables from an output file. Thus, if more output is required from a particular run, only the output program, and not the main program need be re-executed.

Figure 10. A flowchart of the output code, BBUNSOUT. The rectangles represent subroutines. The shaded area represents the graphing routines.

◇

INSTRUCTIONS [4]

The user's only way of sending input to the code is through the input command file. This simplifies what the user needs to know in order to run the code. By mastering the structure of the input file, the user masters all there is to using BBUNS. The next sub-section will detail the construction of input files.

INPUT COMMAND FILE [4.A]

The standard structure of a command line is given in figure 11.

Figure 11. Anatomy of a command line for BBUNS.

There are some variations to the standard command line. These will be detailed later. However, most command can be written in this format. Some command classifications have the first datum (i.e. 1>01) assigned to an option number. These options allow the user to pick from one of several mutually exclusive options. The user need only supply the additional data for the chosen option. To accomplish this, there is a standard option line (in addition to the standard command line) as shown in figure 12.

Figure 12. Anatomy of an option line for BBUNS.

The option lines correspond to option numbers given for each classification. For example, the user might want a λ_β profile that is due to a prescribed B-field. The user enters the following command line:

```
1>01      LAMOPT=2      !Lambda-Beta option
```

This tells the code to fill the λ_β array with option 2, solenoidal focusing in this case. Now, the user must also specify the strength of the B-field. This is done in an option command line:

```
1>01/02  BFIELD=0.2      !Focusing B-Field [Tesla]
```

This line specifies the B-field in Tesla. The option command numbers are simple to understand. The first number is the data piece (only one is needed for this option) while the second number corresponds to the option number. So, had the user wanted a constant λ_β , these lines would be used instead:

```
1>01      LAMOPT=1      !Lambda-Beta option
1>01/01  LAMDAB=1.25    !Constant Lambda-beta
```

Note the option command number changed to 01/01. The option number corresponds to the option selected (see figure 13).

```

1>01      LAMOPT=1                !Lambda-Beta option
1>01/01   LAMDAB=1.25              !Constant Lambda-beta
1>01/02   BFIELD=0.20              !Focusing B-Field in Tesla

```

If LAMOPT were = 2 then this
line would be used instead.

Figure 13. The relation between the command line and the option line.

These two type of command lines enable the user to specify virtually all the inputs and options for a given run. Table 2 gives a nearly complete list of command classifications, numbers and their function.

Table 2. A list of command classifications, numbers and their functions for use in the BBUNS input file.

t	01	name of run	[optional]
	02	description of run	[optional]
	03	output file name (new file)	[optional]
	04	wake file name (old file)	[optional]
	05	plot file name	[optional]
g	01	number of points along z	
	02	steps in z between saving of data	[optional]
	03	number of points along s to save	[optional]
	04	number of particles in each slice	
s	01	structure option number	
	--	option 1: one kick per cell	
	01/01	distance (in meters) between cavities	
	--	option 2: kicks while in gaps	
	01/02	distance (in meters) between gap centers	
	02/02	width of gaps	
	02	total length of structure	
w	01	wake processing option number	
	--	option 1: shift up non-causal data pairs	
	--	option 2: cut-off non-causal data pairs	
	02	points between wake file data used	[optional]
	03	truncation length (meters) of wake	[optional]
p	00	charge density, ρ , option number	
	--	option 1: constant density = current/c	
	--	option 2: exponential density with rise-time	
	01/02	rise time (in seconds)	

	02	initial beam displacement	,
	03	initial velocity ($d\xi/dz$)	
	06	beam current (in amps)	
	07	pulse length (in seconds)	
	09	pulse initial condition option number	
	--	option 1: offset beam by a constant	
	--	option 2: offset beam by a linear slope	
	10	final value of linear beam displacement	
	--	option 3: tickle the beam according to wake	
e	01	Energy (γ) profile option number	
	--	option 1: no variation along s	
	--	option 2: exponential with rise-time along s	
	01/02	rise time (in seconds)	
	02/02	$\Delta\gamma$ from head to tail of exponential	
	--	option 4: BNS and Landau damping	
	01/04	$\Delta\gamma_{\text{Landau}}$: Landau spread within each slice	
	02/04	Synchrotron wave number	
	03/04	$\Delta\gamma_{\text{BNS}}$: BNS spread along entire beam	
	02	Gamma "average" at start of structure	
	03	Gamma "average" at end of structure	
	07	Average gamma profile option number	
	--	option 1: linear gradient from e>02 to e>03	
	--	option 2: discrete kicks at each cavity	
l	01	betatron wave length, λ_β , option number	
	--	option 1: constant λ_β	
	01/01	value of constant λ_β	
	--	option 2: solenoidal focusing (in Tesla)	
	01/02	value of constant focusing B-Field	
	--	option 3: varying B-field [BBUNS modification]	

The input command file is simply composed of the available command lines. Some commands are optional. Also, only the options to be used need be specified. For instance, referring to figure 13 again, if option 2 is used for λ_β , then the option command line l>01/01 (for option 1) need not be specified.

The program will alert the user if a command line is not comprehensible; however, the program will not warn the user if any necessary information is not specified. Instead, the program will use default values, either values from the previous run or zero. It is the responsibility of the user to enter all the necessary parameters.

In practice, the user will edit an existing input file (either a template supplied with the code, or one of the user's own creation). Using an existing file assures that all critical data items are specified. The next few sub-sections discuss the built in options for each category.

THE TEXT CATEGORY [4.A.1]

All entries in this category are optional, that is they need not be specified. The name ($t>01$) and description ($t>02$) are for the user's benefit. The output ($t>03$), wake ($t>04$) and plot ($t>05$) file names all have default names: the default output filename is *bbu.odat*, the default wake file name is *bbu.wake* and the default plot file name is *bbu.plot*. There are two conditions on the file names: 1) The output file must not exist and 2) The wake file must exist.

There are no software restrictions on the type of text allowed. File names should be kept to eight (8) characters for full compatibility with the Cray. In addition, only forty characters **including the keyword and equal sign** will be read from each line.

THE GENERAL CATEGORY [4.A.2]

Only two pieces of data need be specified in this category: the number of mesh points along the structure ($g>01$) and the number of particles per slice ($g>04$). The other two options allow the user to control how many data points are saved (the default is the maximum allowed by the program).

The second datum ($g>02$) controls the number of points between successive stores of the *z* (structure) data. Thus, if 15 is specified, then every fifteenth point in *z* will be saved. The third datum ($g>03$) controls the total number of data points along the beam, in *s*, to save. Thus, if 250 is specified, then a total of 250 data points in *s* will be saved.

The general category is also used to alter the look of the graphs and request certain tables. This use will be discussed later.

THE STRUCTURE CATEGORY [4.A.3]

The first datum ($s>01$) in the structure category is the option number for how the structure should be modeled. The first option does not take into account the finite size of the accelerating gap. The only data required is the spacing between the gaps. The other option is to account for the gap size. In this case, both the spacing and the gap size are required. Refer to figure 14 for a graphical explanation. Note, sizes must be in meters since all inputs are in MKS.

Figure 14. The two basic types of structures allowed in BBUNS. On the left, the structure is composed of drift tubes and accelerating gaps which are infinitesimally wide. On the right, the size of the gap is taken to be finite.

The user also needs to specify the total length of the beam-line ($s>02$).

THE WAKE CATEGORY [4.A.4]

This category allows the user to control how the wake is conditioned. Some field solving programs generate non-causal ($s<0$) data points due to the use of a Gaussian profile for the exciting charge. The first datum is the wake option number ($w>01$). Option 1 takes the non-causal data points and shifts them up so that $s\geq 0$ for all points. Option 2 eliminates all data pairs with $s<0$. No data need be given to exploit these options.

The second datum ($w>02$) controls the density of wake points used. If n is the input then every n th data point will be used from the wake file. This feature is very important for reducing run times. Some field solving programs generate wakes with far too many data points (due to the numerical requirements of the field solver for a small Δs). BBUNS uses the mesh (the spacing of adjacent s points) from the wake file as the mesh for the beam. Thus, reducing the effective density of wake points read in, reduces the density of beam points and the run time required to solve the problem.

The third datum ($w>03$) controls the length of the wake used. If the user wants to truncate the wake to a length shorter than given in the wake data file, this option can be used. This is useful for eliminating high frequency, low amplitude ringing which may not contribute to BBU,⁴⁰ but does contribute to long run times.

PULSE CATEGORY [4.A.5]

The first datum ($p>01$) controls how the ρ array is filled. Option 1 is a simple constant density determined by the current entered in datum 6 ($p>06$). Option 2 allows for the density to have an exponential profile,

$$\rho(s) = I_0(1 - e^{-s/\tau_r}) \quad (14)$$

where τ_r is the rise time entered in option $p>01/02$.

The second datum ($p>02$) is the initial beam displacement. Typically, the entire beam is offset by one unit. Since the BBU growth is taken to be linear in ξ , the initial beam displacement is arbitrary. In addition to a constant offset of the beam, the user can request a linear slope (from the beam head to the tail) or the user can request that the beam be "tickled". The beam tickling option contours the offset of the beam to that of the wake. Datum 9 ($p>09$) controls which one of these options is used. If a sloped offset is desired, the final value of the offset should be specified in datum 10 ($p>10$). Figure 15 depicts

the three beam offset options. The third datum (p>03) sets the initial transverse "velocity". Typically this is set to zero.

Figure 15. The three types of initial beam offsets in BBUNS.

Finally, datum 7 (p>07) determines the pulse length (in seconds) of the beam.

ENERGY CATEGORY [4.A.6]

The first datum (e>01) determines the energy profile option. Option 1 gives a γ profile which is constant in s . Option 2 gives an exponential energy profile which is an attempt to model the effect of the finite rise time of the voltage in the injector:

$$\gamma(s) = \Delta\gamma \left(e^{-s/\tau_r} - \frac{1}{2} \right), \quad (15)$$

this requires that the rise time, τ_r , be specified in option e>01/02 and that $\Delta\gamma$ be specified in option e>02/02. Option 4 allows for Landau and BNS damping using the following formula:

$$\gamma(s) = \Delta\gamma_{\text{Landau}} [\text{Sin} (k_{\text{synch}}z + \phi_i)] + \Delta\gamma_{\text{BNS}} \left[\frac{s}{s_{\text{max}}} - \frac{1}{2} \right] \quad (16)$$

where the Landau spread in gamma, $\Delta\gamma_{\text{Landau}}$, is determined by e>01/04, the synchrotron wave number, k_{synch} , is determined by e>02/04 and the BNS spread in gamma, $\Delta\gamma_{\text{BNS}}$, is determined by e>03/04. In addition, ϕ_i is the phase of the i th particle⁴¹ and s_{max} is the pulse length.

The gamma profiles can also have a variation along z. In other words, The full gamma is described by,

$$\gamma(s,z) = \gamma(s) + \bar{\gamma}(z) \quad (17)$$

The average variation of gamma along z can either by a linear slope or a staircase function with a discrete “kick” in gamma at each cell. Datum 7 (e>07) determines which of two average gammas is used. A value of 1 chooses a linear slope and a value of 2 chooses discrete kicks. The second and third datums determine the initial and final values of the average gamma, respectively.

LAMBDA-BETA CATEGORY [4.A.7]

The first datum (1>01) determines how the betatron wavelength, λ_β , will be calculated. Option 1 allows for a constant λ_β with the value determined by e>01/01. Option 2 allows for modeling of solenoidal focussing with a constant B-field determined by e>01/02. If option 2 is used then

$$\lambda_\beta(s,z) = 4\pi \frac{mc}{e} \frac{\gamma(s,z)}{B} . \quad (18)$$

Option 3 allows for a B-field which varies along the structure. The B-field is determined by a user supplied routine.

CONTROL OF OUTPUT [4.B]

There are two categories used to control how and what is produced as output by the output program. The first is the general category as described in section [4.a.2]. The general category allows the user to control *how* the graphs will look and select some special graphs and tables. The other category, the output category, is used to choose *what* will be plotted.

THE GENERAL COMMAND REVISITED [4.B.1]

Table 3 lists the additional command numbers available under the general category for the output program. These commands are for the output program and are ignored by the input code.

Table 3. A list of the additional commands under the general classifications for use with the output program (BBUNSOUT).

<i>g</i>	20	Device type	
	21	Plot quality (1,2, or 3)	[optional]
	22	Major divisions along each axis	
	23	Data point marker number (0=none, <0=no line, valid [0,±19])	[optional]
	24	Grid lines on X-axis (#, 0=none)	[optional]
	25	Grid lines on Y-axis (#, 0=none)	[optional]
	26	Points between X-axis on 3D graph	
	27	Points between Y-axis on 3D graph	
	28	Page margin along X-axis (in inches)	
	29	Page margin along Y-axis (in inches)	
	30	X-axis view points for 3D graph	[optional]
	31	Y-axis view points for 3D graph	[optional]
	32	Z-axis view points for 3D graph	[optional]
	50	Camera type for TV80/FR80	[optional]
	60	Print text information table (1=yes)	[optional]
	70	Give a plot of the <i>full wake</i> (1=yes)	[optional]

The *g*>20 datum sets the type of device to which graphical output will be sent. The types of devices, graphical libraries used along with the required value of datum 20 are listed in table 4.

Table 4. Devices, libraries and device values for use in BBUNSOUT.

Device	Output	Number	Library (MFECC)
Tektronix 4014/4015	screen	1	disspla
FR80 graphics file	file	2	disspla
Tektronix 4010	screen	3	dissgks
Tektronix 4105	screen	4	dissgks
CGM meta-file	file	5	dissgks

Although GKS⁴² is the graphics standard of choice, the older MFECC library TV80/FR80 is available (under disspla) for those users who require it. Devices with output to the screen require a terminal with the appropriate graphics ability (or terminal emulation software). The

FR80 files may be displayed, printed or modified using the MFECC utility *netplot* while the CGM files require the utility *cgmview*.

Datum 21 ($g>21$) controls the quality of the plot. Quality refers to the look of the graphs. A value of 1 gives the plainest and thus the quickest plots. A value of 2 is suitable for most applications, it provides larger characters than a value of 1. A value of 3 is meant for final "publication" quality graphs and require a good deal of time to generate and print.

The function of the remainder of the commands should be apparent from table 3. Most of these inputs are optional and the casual user need not know them. Two important commands are controlled by datums 60 and 70. Datum 60 ($g>60$) controls whether a text list of general run information will be printed. A value of 1 implies that the information text will be printed (to screen). A variety of data is provided in the information text. A sample is given in the Appendix. Datum 70 ($g>70$) controls whether a plot of the full wake file is make. This allows the user to create a graph of the original wake file before BBUNS has done any processing to the data. If this command is given then the output program, BBUNSOUT, simply reads in the original wake file and produces a plot. Again, a value of 1 is a positive entry.

All commands entered in the input file will effect all the plots displayed.⁴³ Thus, if the user requests 10 grid lines on the x-axis (using $g>24$) then all plots will have 10 x-grid lines. In other words, graphical settings are global.

THE OUTPUT COMMAND CLASSIFICATION [4.B.2]

The general command classification controls how the graphs will look. To control what will be plotted, the user must employ the output command classification ($o>$). The basic format of the output classification is described in figure 16. The command format is a bit cumbersome to learn but is very powerful and gives the user complete control.

Figure 16. Anatomy of the output command classification for BBUNS.

Basically, the user instructs the program to plot data set YY versus data set XX where YY and XX are two digit codes for variables and are listed in table 5. The dependencies can be either s or z. An example will greatly clarify the use of this command type:

```
o>01 07vs01 !plot Lambda-Beta versus S
```

This is the most basic form of the output command, and also its most frequently used form. The example above will produce a plot of $\lambda_\beta(s,0)$ versus s (refer to Table 5 for codes). The plot number is given as 01, thus this will be the first plot.⁴⁴ The range of the plot is chosen by default to be the full range along s (i.e. $s=[0, S_{\max}]$) and at the first point in z (i.e. $z=0$). If, instead, the user wanted to plot $\lambda_\beta(s,z)$ versus s at some other point in z, say $z=10.5$ (meters), then the following command line could be used:

```
o>01 07vs01 @ z=10.5 !plot at z=10.5
```

The $z=10.5$ instructs the program to plot $\lambda_\beta(s,10.5)$ (for all s). Finally, the user might want to restrict the range of the plot. For instance, a plot of $\lambda_\beta(s,10.5)$ versus s plotted from $s=3.5$ to $s=4.4$ would be accomplished by using the following line:

```
o>01 07vs01 @ z=10.5 from s=3.5 to s=4.4
```

With this type of command, the user is unrestricted in what will appear on plots. Any data set listed in table 5 can be plotted against any other data set.

Table 5. Axis codes for use in the output command in BBUNSOUT. Any of these axis codes may be plotted against any other in this table.

Number	Data Set	Dimensions
1	s (beam coordinate in meters)	1
2	z (structure coordinate)	1
3	γ (Gamma)	2
4	ξ (Beam displacement)	2
5	$d\xi/dz$ (Beam velocity)	2
6	ρ (Charge density)	1
7	λ_β (Betatron wavelength)	2
8	Processed Wake	1
9	k_β (Betatron wavenumber)	2
10	Displacement amplitude	2
13	Maximum displacement	1

The only entry in table 5 that requires explanation is the Displacement amplitude (10). This data set is the envelope of the betatron oscillations. Explicitly,

$$\text{Amp}(s,z) = \sqrt{\xi^2 + \left(\frac{1}{k_\beta} \frac{d\xi}{dz} \right)^2}. \quad (19)$$

The data sets that are two dimensional (i.e. γ , λ_β , etc.) are handled in the following manner. The y-axis is assigned to the accelerator coordinate, z. The x-axis is assigned to the beam coordinate, s. Thus, to make a plot of $\gamma(z)$ versus $\xi(s)$ the user would enter the command below:

```
o>01 03vs04 !plot gamma versus zi
```

Conversely, to plot $\xi(z)$ versus $\gamma(s)$ the user would enter the following:

```
o>01 04vs03 !plot zi versus gamma
```

There is no way of plotting $\xi(s)$ versus $\gamma(z)$ or $\gamma(s)$ versus $\xi(z)$. That is, there is no way to have an s-dependence assigned to the y-axis or a z-dependence assigned to the x-axis for two dimensional data sets. But this is hardly a limitation since the plots can always be looked at side-ways. Of course, the user can always use the s (01) or z (02) data sets in either axis.

MULTIPLE FRAME PLOTS [4.B.3]

A feature beyond the single two dimensional plots explained in above allows the user to make a series of plots of a particular data pair at several points along either the s or z dependence. An example will be most illustrative. In order to make 10 plots of γ versus z at points equally spaced along the interval $s=(0,3.5]$, the user would have to enter 10 commands using the format explained in [4.b.2]. Instead, the user can simply enter the multiple frame command:

```
o>01  03vs02 @ f=10  from s=0  to s=3.5
```

This command form is very similar to that of the single frame command. The difference lies in the first dependence; an f , for frames, was used rather than an s or a z . The 10 after the f indicates that 10 frames (plots) are to be made between the given s range. The program would automatically generate graphs at $s=(0.35,0.70,\dots,3.5)$ for a total of 10 plots.

The multiple frame command should only be used when plotting a two dimensional data set on at least one of the axis. This is a trivial restriction since it makes no sense to have multiple plots of a variable with only one degree of freedom.

THREE DIMENSIONAL PLOTS [4.B.4]

The third form of the output command allows for three dimensional plots. The format is given in figure 17.

Figure 17. Anatomy of a command line for a three dimensional plot in BBUNSOUT.

There are two differences between the command line for a 3-D plot and that for a 2-D plot. The first is the use of the 3 for the first dependence (rather than an f, s or a z). The second is the use of the Z-axis code after the first equal sign. The Z axis codes are those codes (listed in table 5) which are assigned to two dimensional data sets. In addition, only data sets with a z-dependence should be used for the Y-axis and those with an s-dependence for the X-axis. Two examples of three dimensional plots will illustrate the above explanation.

```
o>01 02vs01 @ 3=04 ! A three-D plot
```

The previous example will produce a three dimensional plot of ξ versus z versus s. The next example produces a three dimensional plot of displacement amplitude versus z versus s from z=0 to z=6.0 for all s:

```
o>02 02vs01 @ 3=10 from z=0 to z=6.0
```

Notice the use of o>02 rather than o>01, this indicates that this will be the second plot made (assuming a first one exists).

LOGARITHMIC PLOTS [4.B.5]

Any of the three types of plots (2D, multiple frame, and 3D) can be made semi-logarithmic. Only the vertical axis can have a logarithmic scale (the Y-axis for 2D, the Z-axis for 3D). Semi-log plots are requested by using an l (lower case L) after the two digit plot number. Some examples are given below:

```
o>011  02vs01 @ 3=10  ! 3D semi-log plot
o>021  02vs04          ! 2D semi-log plot
o>031  01vs07 @ f=5    ! 10 plots semi-log
```

In the event that there are negative values in the log axis, the program will attempt to plot only the positive values (if any).

SUMMARY OF OUTPUT COMMANDS [4.C]

There are eight categories of input: text, general, structure, wake, pulse, energy, lambda-beta, and output. There are three formats of input. One for the first seven categories, one for the options and one for the output. The input file is an assembly of input command lines. The various commands, options and output codes are listed in tables 3 through 5. Basic rules which apply to the all input files are as follows:

- 1) A command line begins with a command classification which is composed of a letter and a greater than sign.
- 2) Any line which does not begin with a command classification will be ignored.
- 3) A line which does begin with a command classification will be interpreted as a command regardless of the contents of the remainder of the line.
- 4) The order of the command lines and options is irrelevant.

In practice, the user will begin with a supplied input file. This will serve as a template from which specific input files will be created.

ERROR HANDLING [4.D]

All efforts have been made to make BBUNS able to recover from errors. Both the main code and the output code display error messages in a similar fashion. Virtually all errors arise from incorrect input, incompatible parameters, or file problems. In designing the error handling of BBUNS two criterion were followed. The first is that any error or warning must be reported to the user both at run time and in the output (BBUNSOUT). The second criterion was that any

automatic correction made by the program should not alter the physics of a run.

Errors are reported on the screen as the error or warning is found during execution. Along with a description of the error, a possible cause and solution to the problem is reported. An example of a reported error is given below:

```
***** WARNING! There has been an ERROR!  
***** ERROR: Input file has an unknown command.  
***** ROUTINE: ReadErr  
***** ACTION: Run will continue. Record will be skipped.  
***** EFFECT: A piece of data may have been neglected.  
***** SOLUTION: Correct input file format. Check for a  
                  negative index!
```

All error lines begin with five asterisks to differentiate the error messages from other information. The first line simply alerts the user. The second line describes the error. The third line indicates where the error occurred. The fourth line explains what the program will do. The fifth line explain how the error will affect the program execution, and the sixth line proposes a solution to the user. Some errors, such as the one above, simply alert the user of an anomaly. Other errors report that the program has altered some parameter to enable the program to continue execution. A third type of error reports that a fatal error has occurred and that execution will stop.

Often error messages have additional lines. The additional lines provide information such as the filename in question and the value of a parameter before and after correction. The program will attempt to correct any numerical (non-physical) parameter which is not in order. For instance, if the user requests more data points to be saved than are calculated, the program will alert the user, and set the number of saved points to the maximum possible.

ADVANCED FEATURES [4.E]

[This section may be skipped without loss of continuity.]

In addition to the basic commands listed above there are two advanced features available to the user. The user should not employ

these commands until the basic commands have been mastered. The two commands allow the user to further customize the graphical output and to do multiple runs.

MULTIPLE PLOT SETS [4.E.1]

Often a user requires a set of graphs with particular features along with another set with different features. For instance, if a user wanted one set of plots with 10 grid lines on the x-axis and another set without any grid lines, the user would employ the ## command. The command works by altering the output program to stop reading the input file, generate the plots requested so far, then come back and continue reading. This allows the user to have commands which are interpreted in a particular order. This violates rule (4) given in section [4.I.6], but it gives the user greater control. Consider the example posed above. To accomplish this, the user might use the following input file segment:

```

.
.
.
g>24  xgrid=10      ! 10 grid lines on x axis
o>01  02vs01 @ 3=10 ! The first set of plots
o>02  02vs04        !
o>03l  01vs07 @ f=5  !
##      ! Tell computer to do plots given so far
g>24  xgrid=0       ! No grid lines now
o>01  02vs04        ! Now do more plots
.
.
.
```

Notice that the plot numbers before the ## command may be recycled. Had the ## command been omitted, then the program would have replaced the first g>24 command and the first o>01 plot request by the second pair. The ## command is only interpreted by the output program, BBUNSOUT. The main program, BBUNS, will ignore this command: there is no purpose for the command in the main program. The ## may be used an unlimited number of times (one for each graph is the most necessary).

MULTIPLE RUNS [4.E.2]

Whereas the `##` command allows the user to fully customize the graphical output, the `$$` command allows the user to execute multiple runs from one input file. This allows the user to specify several different sets of parameters to be executed in one session. The most useful application of this command is when a small set of parameters must be varied. Often the user might enter an entire set of parameters and then want to execute several runs with only the current, say, being varied. The `$$` command allows the user to do this.

There are three rules to be aware of when using the `$$` command:

- 1) All the inputs for a run before the `$$` will carry over to a run after the `$$`.
- 2) An input after the `$$` will replace the same input found before the `$$`.
- 3) A unique output file name must be specified for each run (i.e. after each `$$` a new output file name must be specified).

Rule 1 implies that the user need only re-specify those parameters to be changed in subsequent runs rather than having to re-specify all the parameters. An example is given in figure 18.

The `$$` command is also interpreted by the output code. Thus, all plot specifications are carried over from one run to the next. In this way, if a set of plots is desired for each run, only the first run needs to specify which plots to produce. The output program automatically selects a new unique plot file name (if the plots are to be sent to a file) for each run.


```

Text data
-----
t>01      NAME=TEST1A                      !Name of Run
t>02      DESCRIPTION=Example for instructions !Description
t>03      OUTFILE=run1.out                  !Output file name

General parameters
-----
g>01i     NZPOINTS=14500                   !NZPOINTS: number of Z points.      [-]
g>04i     NPART=1                          !NPART: number of particles per slice [-]

Structure parameters [1]
-----
s>01i     ICELOPT=1                       !Cell Option number
----- Option #1 gives 1 kick per Cell
s>01/01    SPERIOD=0.5                     !The length between cavities      [Meters]
.
.
.
o>01      02vs04
o>02      01vs10
.
.
.
$$                               !End of the first run
t>03      OUTFILE=run2.out                 !New output file name
g>04      NPART=10                         !This time run with 10 particles
o>02      02vs10                           !Change plot number 2
o>03      02vs01 @ 3=07                    !Add another plot

```

Figure 18. Sample lines from a two run input file for the BBUNS code. The \$\$ indicates that another run is to be specified. The second run (after the \$\$) modifies the output file name, the number of particles used in the second run as well as changing plot #2 and adding plot #3. All other parameters carry over from the first run.

The \$\$ may be used in conjunction with the ## command; however, caution must be taken to avoid unexpected results. The ## command will not carry over as other command do since it specifies that the lines are to be interpreted in order. If a run contains the ## command, then the next run (after the \$\$) will use the settings specified **after** the ## in the first run.◊

CONCLUSIONS [5]

The BBUNS code is designed to solve beam break-up due to wakefields. Its ability to accept any wake as a file allows BBUNS to do calculations on virtually any accelerating structure.

The program is designed to work in a user friendly manner on non-interactive super-computers such as the Cray 2. The user controls the program through the input command file. Once an input command file has been created, the user executes the main program, BBUNS. Then, once the calculations are complete, the user executes the output code BBUNSOUT to display the results.

The BBUNS code has been tested extensively. Initially, comparisons were made to analytic results.⁴⁵ Then, comparisons with published data were done.⁴⁶ Agreement was good in all cases.

The BBUNS code has also been applied to real physics problems. The first application was to the Two-Beam Accelerator (TBA).^{47,48,49} A second application was to the Advanced Test Accelerator (ATA), which has long been known to suffer from severe BBU (at higher currents and without ion channel guiding). Work is still being done to model Landau damping and to find the most realistic current and energy profiles. Future work includes studies of the SNOMADII cells which are in use at the Accelerator Research Center at LLNL.⁵⁰

PROGRAM AVAILABILITY

The BBUNS code is available for limited distribution. Any person or institution wishing to use BBUNS should write to the authors at the address below:

Collider Physics Group
MS 71-259
Lawrence Berkeley Lab
Berkeley, CA 94720

The only condition for use of the code is that users report all problems and complaints to the above address. Although the code has been tested, debugged and tailored for general use, there is always room for improvement. The code has the ability to support many new features, so suggestions about the code and this instruction manual are welcomed. No warranty expressed or implied is made as to the results or operation of the program. In addition, neither the code itself nor any part of the code may be used or incorporated in any commercial or copyrighted program.

ACKNOWLEDGEMENTS

The development of BBUNS required nearly two years of work. Any such project naturally has many contributors. I would like to mention just a few such people.

First and foremost the work of Andy Sessler should be acknowledged. He conceived of the code and guided its development throughout. David Whittum not only assisted in the writing of this document, but also in the creation of BBUNS. His contributions to this project have been most generous, and his influence is evidenced by the numerous References credited to him. Jeff Tennyson provided several solutions to seemingly unsolvable problems as well as providing a prototype for the input file. Jarvis Leung was the first (after the author) user of the BBUNS code. His patience and constructive criticism as well as original ideas for the program and this document were an immense help. Efrem Sternbach provided instruction on the use of the Cray. George Craig and John DeFord were always helpful and instrumental in providing many of the wake files used in BBUNS.

Unfortunately the list must be finite and we apologize to the people who should have been on the list, but are not. We extend our thank you.

In addition, we extend our appreciation to a number of institutions: The Magnetic Fusion Energy Computing Center and it's staff for technical support, the Beam Research group at LLNL for there support and input, the Lawrence Berkeley Lab (especially the computer division) for enabling this project to take place.

REFERENCES AND NOTES

- ¹R.H. Helm and G.A. Loew, "Beam Breakup," Linear Accelerators, ed. by P.M. Lapostolle and A. L. Septier (Amsterdam: North Holland Pub. Co., 1970)
- ²Y.Y. Lau, "Asymptotic Growth of Cumulative and Regenerative Beam Break-up Instabilities in Linear Accelerators," Naval Research Laboratory Memo Report 6237 (Washington, 1988).
- ³V.K. Neil, L.S. Hall, and B.K. Cooper, "Further Theoretical Studies of the Beam-Breakup Instability," Part. Acc., Vol. 9, 1979, pp. 217-222.
- ⁴D.G. Colombant and Y.Y. Lau, "Beam Breakup Instabilities in Linear Accelerators: Transition, Phase Mixing and Nonlinear Focusing," Appl. Phys. Lett., **53** (26), December 26, 1988.
- ⁵F. Selph and A. Sessler, "Transverse Wakefield Effects in the Two-Beam Accelerator," Lawrence Berkeley Laboratory AFRD paper number LBL-20083, (Berkeley, California, 1985).
- ⁶K. L. F. Bane, "Wakefield Effects in a Linear Collider," SLAC Pub No. 4169, December, 1986.
- ⁷G. J. Caporaso, "The Control of Beam Dynamics in High Energy Induction Linacs," 1986 Linear Accelerator Conference Proceedings, SLAC Pub-303.
- ⁸R.H. Helm, G.A. Loew, and W.K.H. Panofsky, The Stanford Two-mile Accelerator, ed. by R.B. Neal, *et al.* (Stanford, California), pp. 203-239.
- ⁹H. Figueroa, *et al.*, "Direct Measurement of Beam-Induced Fields in Accelerating Structures," Phys. Rev. Letters, **60**, 1988, pp.2144-2147.
- ¹⁰See "Electrostatic Channel Guiding: A Technological Breakthrough," Energy and Technology Review (ET&R) (LLNL, Livermore, California, March, 1985), pp. 1-11. And, "Generating Intense Electron Beams for Military Applications," ET&R, December, 1981, pp. 1-12.
- ¹¹A.M. Sessler, G.A. Travish and S.S. Yu, "Multibunch Operation of the 1 GeV Machine," Lawrence Livermore National Lab Beam Research Program Preprint UCRL-96712 (Livermore, California, 1987).
- ¹²For an example of BBU imposed design constraints see V. K. Neil, and D. H. Whittum, "Transverse Resistive Wall Instability in a High-power, Short-Wavelength Free-Electron Laser," Lawrence Livermore National Lab Beam Research Program Memo No. 88-19 (Livermore, California, 1988).

¹³In general, realistic current and energy variations are difficult to treat.

¹⁴Landau damping is often referred to as "phase mixing". Formally, BBU Landau damping is similar to the damping described by Landau. See L. Landau, "On the Vibrations of the Electronic Plasma," J. Phys. U.S.S.R. **10**, 25 (1946).

¹⁵BNS is an acronym for V. Balakin, A. Novokhatsky, and V. Smirnov three Soviet scientist who proposed a damping scheme for VLEPP. See V.E. Balakin, *et al.*, "VLEPP: Transverse Beam Dynamics," Proceedings of the 12th International Conference on High-Energy Accelerators, ed. by F.T. Cole and R. Donaldson (Batavia, Illinois: Fermi National Accelerator Laboratory, 1984), pp. 119-120.

¹⁶J. DeFord, G. Craig, R. McLeod, "The AMOS (Azimuthal Mode Simulation) Code," IEEE Particle Accelerator Conference Proceedings (Chicago, Illinois, 1989).

¹⁷For an explanation of the differences between cumulative and regenerative BBU see Ref. 2.

¹⁸The transverse E-field is not included in the lowest-order wakefield since it is $O(\xi^2)$ for TM modes.

¹⁹Since we restrict ourselves to cumulative BBU, fields are cut-off in the beam pipe, so that the integral is cut-off only after a few wavelengths. In other words, there is no "cross-talk" between cavities.

²⁰The transverse wake can be considered as a modal sum with the second mode ($m=1$) being the transverse dipole wake. Higher order transverse wake modes are usually not treated.

²¹For an explanation of the various formalisms used see, G. Craig and J. DeFord, "The Connection Between SLAC-DESY and LLNL Wakefield Formalisms," Lawrence Livermore National Lab Beam Research Memo 89-7.

²²See equation 4.1 in Ref. 6. Note, our (s,z) co-ordinates are (z,s), respectively, in the notation this reference.

²³We restrict ourselves to this focussing which does not couple the vertical and horizontal motion (e.g. quadrupole focusing).

²⁴The assumption that there is no longitudinal displacement of the beam particles relative to one another is a good one in the high beam energy limit. In other words, bunching (and other longitudinal effects) due to BBU is neglected.

²⁵See Ref. 15 and D.H. Whittum, "Asymptotic Growth of Transverse BBU with BNS Damping," Collider Physics Group Note 45, (Unpublished: 1989).

²⁶Landau damping refers to phase mix damping due to an energy spread within the beam slices. See Ref. 14.

²⁷The wake file is a simple text file that contains pairs of numbers. The first column contains the "s" values while the second contains the wake values. The list must have equally spaced "s" points. The program is able to deal with non-causal ($s < 0$) data points that may be generated by some field-solving codes. For more information, see section [4].

²⁸A "run" refers to the solution of a single problem based on a single set of parameters.

²⁹M. Abramowitz and I.A. Stegun, Handbook of Mathematical Functions (Toronto, Ontario: Dover, 1972), p.887.

³⁰*Ibid.*, p.878

³¹*Ibid.*, p.897

³²Note: the prime indicates differentiation with respect to z .

³³The program was designed to operate on the Cray computer series (X-MP or Cray 2). Vectorization allows the computer to perform many scalar operations at once. This results in a much faster run time for most codes.

³⁴G. Craig and J. Deford, "Diagonal Resonances and Trapped TE Modes," Lawrence Livermore National Lab Beam Research Program Research Memo 88-44 (Livermore, California, 1988).

³⁵It has been shown that at least in the ATA wakefield, the "trapped" ringing mode does not contribute significantly to beam break-up. See D.H. Whittum, "Beam Break-Up in the Two Beam Accelerator," IEEE Particle Accelerator Conference Proceedings (Chicago, Illinois, 1989).

³⁶See note 18.

³⁷It is possible to consider the force as averaged over the entire structure. However, this is less physical and much more CPU intensive. Yet, averaging is usually done for analytic work. For an exception to this, see Ref. 3.

³⁸Actually, most LIA gaps are so much smaller than the distances between the successive gaps, that the kick need only be applied once: Typical gap size $< 3\text{cm}$, typical cell spacing $\approx 50\text{cm}$.

³⁹CRU's are Cray Recharge Units and are roughly equivalent to minutes of CPU time.

⁴⁰Extensive runs have been done on the ATA cell which show that the ringing does not contribute significantly to the maximum growth of BBU; however, there are qualitative changes in the appearance of the beam displacement. See reference 35.

⁴¹The phase of the particles is fixed at equally spaced points on $[0, 2\pi]$ so as to produce an $\xi_{\text{average}} = 0$.

⁴²GKS is an acronym for Graphics Kernel System. It is one of the accepted standards for machine and language independent graphics primitives.

⁴³An exception to this rule is noted in section [4.E]

⁴⁴Plots may be numbered from 1 to 99 although more than 99 plots may be made per run by either using the ## command or the multiple frame option.

⁴⁵Asymptotic equations for linear wakes are well known. See Ref 6 for example.

⁴⁶B.B. Godfrey, "Computer Code Benchmark for the Beam Breakup Instability," Mission Research Co. Note MRC/ABQ-N-410, (1989).

⁴⁷A.M. Sessler, "New Particle Acceleration Techniques," Physics Today, January, 1988, pp.26-34.

⁴⁸D.B. Hopkins, *et al.*, "Elements of a Realistic 17 GHz FEL/TBA Design," Proceedings of the 1989 Workshop on Advanced Accelerator Concepts (Lake Arrowhead, California, 1983).

⁴⁹D.B. Hopkins and A.M. Sessler, "Status of the LBL/LLNL FEL Research for Two Beam Accelerator Applications," IEEE Particle Accelerator Conference Proceedings (Chicago, Illinois, 1989).

⁵⁰The **S**olid state **N**onlinear **M**agnetic **D**rive. See D. Goodman and J. Mangano, "New Concepts in Microwave Source Technology," SRL Report 01-Q-1989 (April 20, 1989).

APPENDIX: SAMPLE SESSION OF BBUNS

This appendix contains a complete sample input file for use with BBUNS. Following the input file is the text output from a run of BBUNS. Then, the output of BBUNSOUT (the results) is given.

Authors' comments are enclosed in boxes. For example,

This is a comment.

The boxed N/A refers to a line which is Not Applicable for this run. Some options were included in the input file text, but are not needed for the execution of this example. Such options are indicated by a boxed N/A. Text that would normally be entered by the user into the computer is indicated by **bold** characters.

Sample input file edited by user.

```
*****
*****  Input data file for BBUNS code                      *****
*****  Refer to instructions for further help              *****
*****  Below are some notes:                                *****
*****  1 - The first two characters are the command code (ex: a>) *****
*****  2 - The next two digits are the data index (example: a>xx) *****
*****  3 - If a hyphenated number is used then the first two digits *****
*****        are the data index and the next two digits are the *****
*****        option level (example: a>xx/yy)                *****
*****  4 - Beginning in column 11, a label=# format is used *****
*****        the label is garbage skipped over the number is read in *****
*****        based on the category and the data number.      *****
*****  5 - Anything after the 20th (40th for text)          *****
*****        column is info junk (units, etc)                *****
```

```

***** 6 - Any line not beginning with a command is ignored. *****
*****
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345

```

Text data

t>01	NAME=Example #1		!Name of Run
t>02	DESCRIPTION=Sample run for instructions		!Description
t>03	OUTFILE=bbu.out		!Output file name
t>04	WAKEFILE=b045wkt		!Wake file

General parameters

	I	I		
g>01i	NZPOINTS=10000		!NZPOINTS: number of Z points.	[-]
g>03i	NPS=300		!NPS: number of s points to save	[-]
g>04i	NPART=1		!NPART: number of particles per slice	[-]

Structure parameters [1]

s>01i	CELLOPTION=1	!Cell Option number	←	Infinitesimal Gap Width
----- Option #1 gives 1 kick per Cell				
s>01/01	SPERIOD=2.0	! The length between cavities		[Meters]
----- Option #2 gives kicks while in the gap				
s>01/02	SPERIOD=1	! Piece #2 of data for option #2		[Meters] ← N/A
s>02/02	SGAP=1.0	! Piece #1 of data for option #2		[Meters] ← N/A
s>02	ZMAX=100.00	!ZMAX: total length of structure		[Meters]

Wake processing options [2]

|--|--|--|

w>01i	WAKEOPTION=1	!Wake processing option	[-]	
-----		Option #1 Shifts non-causal (s<0) points up.		
-----		Option #2 Cuts off non-causal points (s<0)		N/A
w>02i	NWSTEP=3	!Spacing between wake points used	[-]	
w>03	STRUNC=5.0	!Truncation length	[Meters]	

Pulse parameters [3]

p>01i	RHOOPTION=1	!Charge density calculation option	[-]	
-----		Option #1 gives constant density		
-----		Option #2 requires rise time to be specified		N/A
p>01/02	RISETIME=5.0E-9	!Rise time in milli-sec	[Sec]	N/A
p>02	XIO=1.0	!Initial displacement	[Meters]	
p>03	DXIO=0.0	!Initial dx/dz	[-]	
p>06	CURRENT=5000	!Beam current	[Amps]	
p>07	PULSELEN=2.0e-8	!Pulse length in time	[Sec]	
p>09i	PULSEOPTION=1	!Pulse initial-condition option	[-]	
-----		Option #1 Gives a constant offset of the beam		
-----		Option #2 Gives a sloped offset from Xi0 to Xif below		N/A
p>10	Xif=0.5	!Final value of beam displacement @ Z=0	[-]	N/A
-----		Option #3 Tickles the beam: profile of beam like wake		N/A

Energy parameters [4]

e>01i	GAMMAOPTION=2	!Energy profile option	[-]	
-----		Option #1 has no s variation.		N/A
-----		Option #2 has an initial profile in gamma along s		
e>01/02	RISETIME=5.0e-9	!Risetime in nano-sec	[Sec]	
e>02/02	DGAM=5.0	!Magnitude of contour in gamma	[-]	
-----		Option #4 allows for landau and bns damping		N/A
e>01/04	dGamL=0.0	!Landau spread in Gamma	[-]	N/A

e>02/04	Ksynch=0.0	!Synchrotron wavenumber	[1/m]	←	N/A
e>03/04	dGamBNS=-2.0	!BNS spread in Gamma	[-]	←	N/A
e>02	Gamma0=5.00	!Initial gamma (option #1)	[-]		
e>03	GammaF=100.00	!Final gamma (option#1)	[-]		
e>07	GBarOPTION=1	!Average Gamma(z) option	[-]		
----- Option #1 is a linear gradient from Gamma0 to GammaF					
----- Option #2 is discrete kicks at each cavity				←	N/A

Lambda-Beta parameters [5]

l>01i	LAMBOPTION=2	!Lambda-Beta option	[-]		
----- Option #1 is a constant LambdaB				←	N/A
l>01/01	LAMBDAB=1.00	!Constant Lambda Beta	[Meters]	←	N/A
----- Option #2 is for solenoidal B-field and has an inverse gamma dependence					
l>01/02	BFIELD=0.20	!Constant focusing B-field	[Tesla]		
----- Option #3 allows for a varying B-field				←	N/A

Output parameters [6]

```

*****
***** There are two forms of the output command line: *****
***** 1. For output specifications, the g> command is used. *****
***** For clarity, they are placed under output parameters. *****
***** 2. For specifiers the command key if o>, the format is, *****
***** INPUT LINE FORMAT: *****
*****      AvsB @ Dep1=C      from Dep2=D      to Dep3=E *****
***** WHERE *****
***** A/B are the y/x axis dependencies. Options for A/B below: *****
***** *****
***** Dep1/Dep2 are the dependencies of A/B. *****
***** Option are s, z or f. The f option allows for multiple *****

```

```

*****      plots over the entire dependency      *****
*****      range.                                *****
*****
*****      2D EXAMPLES:                          *****
*****      *** o>01  04vs02                      *****
*****      (gives a plot of Xi vs. Z at S=SMAX (default) for all Z) *****
*****      *** o>02  04vs02 @ s=1.0              *****
*****      (gives a plot of Xi vs. Z at S=1.0 for all Z) *****
*****      *** o>03  10vs01 @ z=0.5              from s=0.0          to s=0.75 *****
*****      (gives a plot of Amp vs. s at z=Zmax/2 for s=(0,3*smax/4) *****
*****      MULTIPLE FRAMES EXAMPLE:               *****
*****      *** o>03  03vs02 @ f=10.              from z=2.3          to z=3.5 *****
*****      (gives 10 plots of Gamma vs Z at 10 points along s) *****
*****      3D EXAMPLE                             *****
*****      *** o>04  02vs01 @ 3=04                *****
*****      (gives a 3D plot of Xi vs. S and Z) *****
*****      NOTE: there should be NO spaces after the '=' *****
*****

```

```

*****
*****      WARNING: DIFFERENT DEVICES REQUIRE DIFFERENT LIBRARIES! *****
*****      DEVICE          TYPE          NUMBER          LIBRARY          *****
*****      Tek 4014/4015    TV80          1              disspla          *****
*****      FR80 file        TV80          2              disspla          *****
*****      Tek 4010         GKS           3              dissgks          *****
*****      Tek 4105         GKS           4              dissgks          *****
*****      CGM file         GKS           5              dissgks          *****
*****

```

```

g>20i      Device=2          !Output device type
g>21i      Quality=2         !1 low, 2 medium, 3 for publication (fancy)
g>22i      Divs=10           !Number of major divisions along axis
g>23i      Marker=0          !Marker number, 0=none,<0 no line, (Valid: 0,+/-19)

```

```

g>24i      iXGrid=0          !Number of grid lines on X-axis
g>25i      iYGrid=0          !Number of grid lines on Y-axis
g>26i      iXMesh=10         !Number of points between mesh lines on X-axis 3D
g>27i      iYMesh=10         !Number of points between mesh lines on Y-axis 3D
g>28       XMargin=1.5       !Page X margin
g>29       YMargin=1.5       !Page y margin
g>30       Xvu=-10.0         !X axis view point 3D
g>31       Yvu=-10.0         !Y axis view point 3D
g>32       Zvu=10.0         !Z axis view point 3D
g>50       icamera=9         !TV80/FR80 camera type
g>70i      FullWake=0        !Plot Full wake file (0 for NO, 1 for YES)
g>60i      INFO=1           !Print (text) general information? (0/1)

```

```

*****
*****      01> s           02> z           03> Gamma (energy) *****
*****      04> Xi          05> dXi/dz      06> Rho   (current) *****
*****      07> LambdaBeta 08> Processed Wake 09> K-beta (lambdaB)*****
*****      09> B-Field     10> Amp(Xi)      11> Time (iS,iZ) *****
*****      12> XiTime      13> XiMax        *****
*****

```

		@			
o>01	13vs02		←		Ximax vs. Z
o>02	07vs02 @ s=1.5		from z=2.0	to z=2.5	← LambdaB vs. Z
o>03	04vs01 @ f=3		←		Displacement vs. S (3)
o>04	04vs02 @ f=3		←		Displacement vs. Z (3)
o>05	02vs01 @ 3=4		←		3D plot of Displacement vs. Z vs. S
o>06	02vs01 @ 3=10		←		3D plot of Amplitude vs. Z vs. S

This is the end of the sample input file.

At this point the user has edited the input file (above), and would transfer the input file to the Cray (assuming it is not already there). Having done so, the user commands the computer to begin executing BBUNS.

bbuns

User types this to
begin executing BBUNS

BBUNS HAS BEGUN EXECUTION
Reading in data...

***** WARNING! There has been an ERROR!
***** PROBLEM: More output steps in Z have been requested than data steps.
***** ACTION: Run will continue: IZSTEP set to min allowed.
***** EFFECT: Maximum possible number of Z steps will be saved.
***** ROUTINE: CheckIn NUMBER: 4
***** SOLUTION: Correct error in input file: make IZSTEP > 1.

A Problem!

Input file read in successfully
RUN NAME: Example #1
DESCRIPTION: Sample run for instruc

Wake file read in successfully
Number of points from wake = 885
Only every 3th wake point was saved.

Number of Wake Points Used

***** WARNING! There has been an ERROR!
***** PROBLEM: S(1) [initial] is not zero.
***** ACTION: Run will continue.
***** EFFECT: S will be shifted over: possible inaccuracy.
***** ROUTINE: FillWake NUMBER: 10
***** SOLUTION: Alter bbu.wake
***** New S(1) = 0.
***** New S(NSw) = 5.96699996
***** New NSw = 885

A Problem!

NSPoints = 889

Number of Points Used along Beam

***** WARNING! There has been an ERROR!
***** PROBLEM: NPZ is greater than NPZMax.
***** ACTION: NPZ will be set to max possible.
***** EFFECT: Less points in Z will be saved.
***** ROUTINE: Init NUMBER: 18
***** SOLUTION: Increase NPZmax in bbunscom or increase IZSTEP in bbu.idat
***** Old NPZ = 10000
***** Old IZSTEP = 1
***** New NPZ = 295
***** New IZSTEP = 34

A Problem!

The beam was initially perturbed by 1.0
Index of Strunc = 741
Working...

Number of Points in Truncated "Wake"

Writing output file: DO NOT INTERRUPT...
RUN TIME information so far:


```
cpu time :41.806039
i/o time :1.148577
sys time :0.714837
mem time :1.468467
```

```
-----
Total    :45.13792
```

```
BBUNS CODE RUN COMPLETED SUCCESSFULLY
-----
```

all done

Cray responds with this

This is the end of the BBUNS execution.

The computer has finished executing BBUNS and has created the output file. The user now directs the computer to execute the output code BBUNSOUT. Since the user specified so, the computer will produce a text summary of the run, followed by some graphs.

bbunsout

User types this to begin
executing BBUNSOUT

```
BBUNSOUT HAS BEGUN EXECUTION
Reading in data...
```

```
PROGRAM: BBUNS (Output)
VERSION: May 26,1989
```

Info Table Starts Here

Output from BBUNS run:

RUN NAME : Example #1
DESCRIPTION: Sample run for instructions

DATA FILE NAME: bbu.out
WAKE FILE NAME: b045wkt
PLOT FILE NAME:

TOTAL RUN TIME :: 0.71808003333334 cru minutes
START of run Time:: 10:35:50 and Date:: 09/06/89
END of run Time :: 10:46:31 and Date:: 09/06/89
CURRENT Time :: 10:35:50 and Date:: 09/06/89

NOTE: ALL VALUES ARE APPROXIMATE DUE TO FINITE
NUMBER OF SAVED DATA POINTS.

NUMERICAL PARAMETERS:

NZPOINTS: Divisions in Z	=	10000
NPZ: Plot points in Z	=	294
NPS: Plot points in S	=	300
NSW: S points from wake file	=	885
NSPOINTS: Macro-particles used	=	889
NSTRUNC: Index of Strun	=	741
NGAUSORD: Order of integration scheme	=	4
DS	=	6.7500e-03
DZ	=	1.0001e-02
DZP (printing/plotting)	=	3.4003e-01

PHYSICAL SPECIFICATIONS

ZMAX: Length of the structure	=	1.0000e+02	[meters]
SPERIOD: Cell period	=	2.0000e+00	[meters]

CURRENT: constant current parameter	=	5.0000e+03	[Amps]
STRUNC: Truncation length of beam field	=	5.0000e+00	[meters]
TAU: Pulse length	=	2.0000e-08	[Sec]
GAMMA0: Gamma(z) start	=	5.0000e+00	[-]
GAMMAF: Gamma(z) end	=	1.0000e+02	[-]
BF: Constant Solenoidal B field	=	2.0000e-01	[Tesla]

DIAGNOSTICS:

Points per LambdaB beam head	=	8.4794e+01
Points per LambdaB beam tail	=	1.0984e+03
Points per LambdaB MINIMUM	=	8.4794e+01
Points per LambdaB MAXIMUM	=	1.1245e+03
Points per e-Fold at Beam Head	=	-2.6633e+08
Points per e-Fold at Beam Tail	=	-7.8303e+04
Points per e-fold MINIMUM	=	5.0974e+03
Points per e-fold MAXIMUM	=	4.8133e+03
Eta at Beam Head, Zmax	=	1.1226e+00
Eta at Beam Tail, Zmax	=	5.5328e+04

RESULTS:

Beam amplitude, Beam Head, Z=Zmax	=	9.9996e-01
Beam amplitude, Beam Tail, Z=Zmax	=	8.8012e-01
Beam amplitude MAXIMUM	=	7.9896e+00
Number of E-Folds MAXIMUM	=	1.9616e+00
Growth length (Lg)	=	5.0979e+01

Beam displacement MAXIMUM	=	7.1106e+00
---------------------------	---	------------

← Max Displacement

CALCULATED data sample:

Charge density at beam head	=	1.6678e-05
Charge density at beam tail	=	1.6678e-05
Charge density MINIMUM	=	1.6678e-05
Charge density MAXIMUM	=	1.6678e-05

Gamma at Z=0 (beam tail)	=	1.0268e+01	
Gamma at Z=Zmax (beam tail)	=	1.0492e+02	
Gamma MINIMUM	=	7.8248e+00	
Gamma MAXIMUM	=	1.0492e+02	
K beta at Z=0, S=Smax	=	5.7138e+00	
K beta at Z=Zmax, S=Smax	=	5.5918e-01	
K beta MINIMUM	=	5.5918e-01	
K beta MAXIMUM	=	7.4976e+00	
Wake MINIMUM	=	-4.6463e+13	
Wake MAXIMUM	=	1.6599e+14	
Current MINIMUM	=	5.0000e+03	[Amps]
Current MAXIMUM	=	5.0000e+03	[Amps]
Energy MINIMUM	=	3.9985e+00	[MeV]
Energy MAXIMUM	=	5.3612e+01	[MeV]
Energy gradient along Z	=	4.8545e-01	[MeV/m]
Energy gradient MAXIMUM	=	4.9614e-01	[MeV/m]
Lambda beta at Z=0, beam head	=	1.0997e+00	[meters]
Lambda beta at Z=Zmax, beam head	=	1.1236e+01	[meters]
Lambda beta MINIMUM	=	8.3802e-01	[meters]
Lambda beta MAXIMUM	=	1.1236e+01	[meters]

COMMENTS and WARNINGS:

← Messages from
Various Subroutines

-----|

Rho was filled by a constant charge density.
Gamma was filled with a rise time along S.
K-Beta was filled by an inverse gamma dependence.
Average Gamma(z) was a linear gradient.
50 cavities requested.
50 cavities kicked.
9999 calls made to CAVITY.
The beam was initially offset by a constant = 1.00
Only every 3th wake point was saved.

ERROR MESSAGES FROM MAIN CODE:

-----|
DESCRIPTION: More output steps in Z have been requested than data steps.
EFFECT: Maximum possible number of Z steps will be saved.

DESCRIPTION: S(1) [initial] is not zero.
EFFECT: S will be shifted over: possible inaccuracy.

DESCRIPTION: NPZ is greater than NPZMax.
EFFECT: Less points in Z will be saved.

-----|
Please wait... Plots are being generated.

Number of plots generated: 10

cpu time :15.472684
i/o time :3.047569
sys time :2.128736
mem time :1.096388

Total :21.745377

BBUNSOUT CODE RUN COMPLETED SUCCESSFULLY

all done

This is the end of the BBUNSOUT execution.

The user has now viewed the text summary, but still needs to view the graphics file (FR80 type) produced by BBUNSOUT. The user invokes a graphical utility, netplot, to view, manipulate and print the plots.

```
netplot f3plot0x view
```

User types this to view plots

The plots are given on the following pages. Note, the arrangement and size of the plots has been modified to fit the page. Actual output will not appear in this fashion. Finally, graphics quality should be substantially better than could be reproduced here.

In summary, a typical session with BBUNS might involve the following steps:

- 1) Edit input command file
- 2) Transfer input file to Cray (if not edited on Cray).
- 3) Execute BBUNS by typing, **bbuns**.
- 4) Execute BBUNSOUT by typing, **bbunsout**.
- 5) View text summary (on screen).
- 6) View graphics (on screen or from a file with a graphics utility such as **netplot**).

LAMBDA BETA VS. Z
AT Z = 1.4817

BBUNS RUN NAME: EXAMPLE RUN DATE: 8/15/97 DATA START: 10.000
DESCRIPTION: RUN TIME: 11.3550 DATA END: 11.355
SAMPLE RUN INSTRUCTIONS: 14/7/98 LAMBDA START: 1.480
CUT TIME: 10.000 LAMBDA END: 1.520

BEAM DISPLACEMENT VS. S
AT Z = 33.333

BBUNS RUN NAME: EXAMPLE RUN DATE: 8/15/97 DATA START: 20.000
DESCRIPTION: RUN TIME: 11.3550 DATA END: 11.355
SAMPLE RUN INSTRUCTIONS: 14/7/98 LAMBDA START: 1.480
CUT TIME: 10.000 LAMBDA END: 1.520

BEAM DISPLACEMENT VS. S
AT Z = 65.637

BBUNS RUN NAME: EXAMPLE RUN DATE: 8/15/97 DATA START: 70.000
DESCRIPTION: RUN TIME: 11.3550 DATA END: 11.355
SAMPLE RUN INSTRUCTIONS: 14/7/98 LAMBDA START: 1.480
CUT TIME: 10.000 LAMBDA END: 1.520

BEAM DISPLACEMENT VS. S
AT Z = 99.990

BBUNS RUN NAME: EXAMPLE RUN DATE: 8/15/97 DATA START: 80.000
DESCRIPTION: RUN TIME: 11.3550 DATA END: 11.355
SAMPLE RUN INSTRUCTIONS: 14/7/98 LAMBDA START: 1.480
CUT TIME: 10.000 LAMBDA END: 1.520

BEAM DISPLACEMENT VS. S
AT Z = 99.9810

BEAM DISPLACEMENT VS. Z
AT S = 1.8812

BEAM DISPLACEMENT VS. Z
AT S = 3.8882

BEAM DISPLACEMENT VS. Z
AT S = 5.8737

BEAM DISPLACEMENT VS. Z
At $S = 1.9912$

BEAM DISPLACEMENT VS. Z
At $S = 3.9842$

BEAM DISPLACEMENT VS. Z
At $S = 5.9737$

BEAM DISPLACEMENT VS. S VS. Z

AMPLITUDE VS. S VS. Z

[3]BUNS

RUN NAME: EXAMPLE #1
 RUN DATE: 09/06/89
 RUN TIME: 10:35:50
 DESCRIPTION: RUNS
 SAMPLE RUN FOR INSTRUCTION
 DATE: 09/07/89
 TIME: 15:59:28

GAMMA START: 0.0000
 GAMMA END: 0.0000
 LAMBDA START: 0.0000
 LAMBDA END: 0.0000

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. Neither the United States Government nor any agency thereof, nor The Regents of the University of California, 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 products process, or service by its 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, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or The Regents of the University of California and shall not be used for advertising or product endorsement purposes.

Available to DOE and DOE Contractors from the
Office of Scientific and Technical Information
P.O. Box 62, Oak Ridge, TN 37831
Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, VA 22161
Price: Printed Copy A04, Microfiche A01

Lawrence Berkeley Laboratory is an equal opportunity employer.