User's Manual for `elegant`

Program Version 14.8
Advanced Photon Source
Michael Borland

September 24, 2002

1 Highlights of What's New in Version 14.8

Here is a summary of what's changed since version 14.7.1. For details and references, see the specific entries in the manual.

One notable change is that `elegant` now runs on MAC OS-X, thanks to the efforts of Robert Soliday (ANL). Also thanks to Bob's efforts, `elegant` is now distributed under an Open Source license.

1.1 New Elements

- Added `SCRIPT` element to allow transforming the beam with an external script. For example, the external script might run the `shower` program to simulate interaction with matter.
- Added `WIGGLER` element for damping wiggler simulation. Does not include edge focusing and has had limited testing.

1.2 Improvements and Bug Fixes for Elements

- Added fiducialization to `MATR` element to resolve problems with matrices that have nonzero R_{n5} or T_{n5m} elements. This bug was found by Paul Emma (SLAC).
- The `SREFFECTS` element (which does synchrotron radiation effects in tracking) had several bug fixes.
  - Fixed problems with quantum excitation when the element was placed in a region with high dispersion.
  - Fixed problems with quantum excitation when the of `FRACTION` parameter was used.
- Fixed bugs and added features to `IBSCATTER` (intrabeam scattering):
  - Added `SMOOTH` parameter to allow selecting a nonrandom simulation of IBS (simply does inflation of the emittances according to the growth factor).
  - Added `FACTOR` parameter to allow multiplying IBS effects by a chosen factor.
  - Fixed problem with energy scattering that caused false exponential growth in energy spread for low-energy beams.

- Fixed problem with `tlimit` implementation for `CLEAN` element that caused incorrect results for low-energy beams.
- In prior versions, when tracking a circular machine with CSR in bends and drifts, the first CSR drift would not be handled properly. This has been fixed.
- Added `LINEARIZE` control for `CSRCSBEND` and `CSRDRIFT`, which allows using purely linear transport between CSR wake kicks.
- Added low-pass filter capability for `CSRCSBEND` element. This allows applying a low-pass filter to the density histogram, as a form of noise control. The filter is also applied to the `CSRDRIFT` elements downstream of each `CSRCSBEND` element.
- Added renormalization of density histograms after smoothing and filtering in `CSRCSBEND` and `CSRDRIFT`. This enforces conservation of charge and prevents negative values of the particle density.
- Added `LINEARIZE` control for `RFCA` and `RFCW`, which results in a linear dependence of energy gain on phase deviation.

1.3 New Commands

- Added `semaphores` command to provide better semaphore functionality. Previously, one could use the `semaphore_file` entry on the `run_setup` command to cause `elegant` to create a file when the run terminated successfully. The `semaphores` command improves on this, allowing specification of files that mark the start and finish of a run. The `semaphore_file` entry on the `run_setup` command is still functional.

1.4 Modified Commands

- For `bunched_beam`, added a new beam type, "halo(gaussian)", which permits generating only the halo particles from a gaussian beam. This entailed rewriting the beam generation subroutines, so you will find that the exact particle coordinates generated for most beam types are different in this version of `elegant`.
- For `slice_analysis`, added `final_values_only` control. If set, this directs `elegant` to perform slice analysis only at the end of the beamline. These final values are also now available for use in optimization expressions (regardless of whether `final_values_only` is set).
- For `load_parameters`, added `include_type_pattern` and `exclude_type_pattern` controls. These allow providing a list of patterns (strings containing wildcards) to use in selecting what element types will be loaded from a file.

1.5 Other New Features and Bug Fixes

- Particle output files from the `WATCH` element now include the Charge parameter.
- Symbol `sTotal` is now available in optimization equations, providing the total arc length of the system.

- Parameters `DtPercXX` added to `final` output file (from `run_setup`), providing percentile locations for time coordinates.
- Fixed problem with setting of the `optimized` parameter in the `final` output file at the end of optimization.

1.6 Known Bugs and Problems

- Use of the perturbed matrix for chromaticity correction does not work.
- Twiss output contains entries for the higher-order dispersion and chromaticity as well as the tune shifts with amplitude, *even when these are not calculated*, which is potentially misleading.
- Loading parameters when using element division doesn't work unless element lengths, bending angles, and edge angles are excluded from the load.

2 Introduction

`elegant` stands for "ELEctron Generation ANd Tracking," a somewhat out-of-date description of a fully 6D accelerator program that now does much more than generate particle distributions and track them. `elegant`, written entirely in the C programming language[1], uses a variant of the MAD[2] input format to describe accelerators, which may be either transport lines, circular machines, or a combination thereof. Program execution is driven by commands in a namelist format.

This document describes the features available in `elegant`, listing the commands and their arguments. The differences between `elegant` and MAD formats for describing accelerators are listed. A series of examples of `elegant` input and output are given. Finally, appendices are included describing the post-processing programs.

2.1 Program Philosophy

For all its complexity, `elegant` is not a stand-alone program. For example, most of the output is not human-readable, and `elegant` itself has no graphics capabilities. These tasks are handled by a suite of post-processing programs that serve both `elegant` and other physics programs. These programs, collectively known as the SDDS Toolkit[8, 9], provide sophisticated data analysis and display capabilities. They also serve to prepare input for `elegant`, supporting multi-stage simulation.

Setting up for an `elegant` run thus involves more than creating input files for `elegant` per se. A complicated run will typically involve creation of a post-processing command file that processes `elegant` output and puts it in the most useful form, typically a series of graphs. Users thus have the full power of the SDDS Toolkit, the resident command interpreter (e.g., the UNIX shell), and their favorite scripting language (e.g., Tcl/Tk) at their disposal. The idea is that instead of continually rewriting the physics code to, for example, make another type of graph or squeeze another item into a crowded table, one should allow the user to tailor the output to his specific needs using a set of generic post-processing programs. This approach has been quite successful, and is believed particularly suited to the constantly changing needs of research.

Unlike many other programs, `elegant` allows one to make a single run simulating an arbitrary number of randomizations or variations of an accelerator. By using the SDDS toolkit to postprocess the data, the user's postprocessing time and effort do not depend on how many random seeds or situations are chosen. Hence, instead of doing a few simulations with a few seed numbers or values,

the user can simulate hundreds or even thousands of instances of one accelerator to get an accurate representation of the statistics or dependence on parameters, with no more work invested than in doing a few simulations.

In addition, complex simulations such as start-to-end jitter simulations[11] and top-up tracking[12] can be performed involving hundreds or thousands of runs, with input created by scripts depending on the SDDS toolkit. These simulations make use of concurrent computing on about 20 workstation using the Distributed Queueing System[10]. Clearly, use of automated postprocessing tools greatly increases the scale and sophistication of simulations possible. This stands in stark contrast to the current trend toward graphical user interfaces, which virtually force an inefficient one-job, one-computer, manual postprocessing way of working.

2.2 Capabilities of `elegant`

`elegant` started as a tracking code, and it is still well-suited to this task. `elegant` tracks in the 6-dimensional phase space $(x, x', y, y', s, \delta)$, where x (y) is the horizontal (vertical) transverse coordinate, primed quantities are slopes, s is the *total* distance traveled, and δ is the fractional momentum deviation[3]. Note that these quantities are commonly referred to as (x, xp, y, yp, s, dp) in the namelists, accelerator element parameters, and output files. ("dp" is admittedly confusing—it is supposed to remind the user of $\Delta P/P_o$. Sometimes this quantity is referred to as "delta.")

Tracking may be performed using matrices (of selectable order), canonical kick elements, numerically integrated elements, or any combination thereof. For most elements, second-order matrices are available; matrix concatenation can be done to any order up to third. Canonical kick elements are available for bending magnets, quadrupoles, sextupoles, and higher-order multipoles; all of these elements also support optional classical synchrotron radiation losses. Among the numerically integrated elements available are extended-fringe-field bending magnets and traveling-wave accelerators. A number of hybrid elements exist that have first-order transport with exact time dependence, e.g., RF cavities. Several elements support simulation of collective effects, such as wakefields and intra-beam scattering. Some of the more unusual elements available are third-order alpha-magnets[4, 5], time-dependent kicker magnets, voltage-ramped RF cavities, beam scrapers, and beam-analysis "screens."

A wide variety of output is available from tracking, including centroid and sigma-matrix output along the accelerator. In addition to tracking internally generated particle distributions, `elegant` can track distributions stored in external files, which can either be generated by other programs or by previous `elegant` runs. Because `elegant` uses SDDS format for reading in and writing out particle coordinates, it is relatively easy to interface `elegant` to other programs using files that can also be used with SDDS to do post-processing for the programs.

`elegant` allows the addition of random errors to virtually any parameter of any accelerator element. One can correct the orbit (or trajectory), tunes, and chromaticity after adding errors, then compute Twiss parameters, track, or perform a number of other operations.

In addition to randomly perturbing accelerator elements, `elegant` allows one to systematically vary any number of elements in a multi-dimensional grid. As before, one can track or do other computations for each point on the grid. This is a very useful feature for the simulation of experiments, e.g., emittance measurements involving beam-size measurements during variation of one or more quadrupoles[6].

Like many accelerator codes, `elegant` does accelerator optimization. While `elegant` is not designed to replace matching codes like MAD, it will fit the first-order matrix and beta functions. Of more interest is the ability to optimize results of tracking using a user-supplied function of

the final beam and transport parameters. This permits solution of a wide variety of problems, from matching a kicker bump in the presence of nonlinearities to optimizing dynamic aperture by adjusting sextupoles.

`elegant` provides several methods for determining accelerator aperture, whether dynamic or physical. One may do straightforward tracking of an ensemble of particles that occupies at uniform grid in (x, y) space. A more efficient variant of this procedure involves tracking a series of constant-x lines of particles with fixed y values, with elimination of any given y value whenever a stable particle is found. Finally, one may use a single-particle search method that can locate the aperture for a series of y values, to a predefined resolution in x.

In addition to using analytical expressions for the transport matrices, `elegant` supports computation of the first-order matrix and linear optics properties of a circular machine based on tracking. A common application of this is to compute the tune and beta-function variation with momentum offset by single-turn tracking of a series of particles. This is much more efficient than, for example, tracking and performing FFTs (though `elegant` will do this also). This both tests analytical expressions for the chromaticity and allows computations using accelerator elements for which such expressions do not exist (e.g., a numerically integrated bending magnet with extended fringe fields).

A common application of random error simulations is to set tolerances on magnet strength and alignment relative to the correctability of the closed orbit. A more efficient way to do these calculations is to use correct-orbit amplification factors[6]. `elegant` the computes amplification factors and functions for corrected and uncorrected orbits and trajectories pertaining to any element that produces an orbit or trajectory distortion. It simultaneously computes the amplification functions for the steering magnets, in order to determine how strong the steering magnets will need to be.

3 Fiducialization in `elegant`

In some tracking codes, there is a "fiducial particle" that is assumed to travel along the ideal trajectory or orbit, with the ideal momentum, and at the ideal phase. There is no fiducial particle in `elegant`. Instead, *elements* are fiducialized in `elegant`. Fiducializing an element means determining the momentum and arrival time (or phase) of the reference particle. This is particularly important in simulations of linacs.

If the reference momentum does not change and no time-dependent elements are involved, then fiducialization is irrelevant. All elements are fiducialized at the central momentum defined in `run_setup`.

A number of commands have parameters for controlling fiducialization:

- The `always_change_p0` parameter of `run_setup` causes `elegant` to re-establish the central momentum after each element when fiducializing. This may be more convenient than setting the `CHANGE_P0` parameter on the elements themselves. However, it can have unexpected consequences, such as changing the central momentum to match changes in beam momentum due to synchrotron radiation.

- `run_control` has three parameters that affect fiducialization, which come into play when multi-step runs are made. Typically, these are runs that involve variation of elements, addition of errors, or loading of multiple sets of parameters.

  - `reset_rf_for_each_step` — If nonzero, the rf phases are re-established for each beam tracked. If this is 1 (the default), the time reference is discarded after each bunch is

tracked. This means that bunch-to-bunch phasing errors due to time-of-flight differences would be lost.

  - `first_is_fiducial` — The first bunch seen is taken to establish the fiducial phases and momentum profile. If one is simulating, for example, successive beams in a fixed accelerator, this should be set to 1. Otherwise, the momentum reference is discarded after each bunch is tracked.
  - `restrict_fiducialization` — If nonzero, then momentum profile fiducialization occurs only after elements that are known to possibily change the momentum. It would not occur, for example, after a scraper that changes the average beam momentum by removing a low-momentum tail.

- The `bunched_beam` command has a `first_is_fiducial` parameter that is convenient for use with the `first_is_fiducial` mode established by `run_control`. If nonzero, this parameter causes `elegant` to generate a first bunch with only one particle. This is very useful if one wants to track with many particles but doesn't want to waste time fidicializing with a many-particle bunch.

4 Namelist Command Dictionary

The main input file for an `elegant` run consists of a series of namelists, which function as commands. Most of the namelists direct `elegant` to set up to run in a certain way. A few are "action" commands that begin the actual simulation. FORTRAN programmers should note that, unlike FORTRAN namelists, these namelists need not come in a predefined order; `elegant` is able to detect which namelist is next in the file and react appropriately.

Each namelist has a number of variables associated with it, which are used to control details of the run. These variables come in three data types: (1) `long`, for the C long integer type. (2) `double`, for the C double-precision floating point type. (3) `STRING`, for a character string enclosed in double quotation marks. All variables have default values, which are listed on the following pages. `STRING` variables often have a default value listed as `NULL`, which means no data; this is quite different from the value "", which is a zero-length character string. `long` variables are often used as logical flags, with a zero value indicating false and a non-zero value indicating true.

On the following pages the reader will find individual descriptions of each of the namelist commands and their variables. Each description contains a sequence of the form

```
&<namelist-name>
    <variable-type> <variable-name> = <default-value>;
    .
    .
    .
&end
```

This summarizes the parameters of the namelist. Note, however, that the namelists are invoked in the form

```
&<namelist-name>
    [<variable-name> = <value> ,]
    [<array-name>[<index>] = <value> [,<value> ...] ,]
        .
```

.
.
&end

The square-brackets enclose an optional component. Not all namelists require variables to be given– the defaults may be sufficient. However, if a variable name is given, it must have a value. Values for STRING variables must be enclosed in double quotation marks. Values for double variables may be in floating-point, exponential, or integer format (exponential format uses the 'e' character to introduce the exponent).

Array variables take a list of values, with the first value being placed in the slot indicated by the subscript. As in C, the first slot of the array has subscript 0, *not* 1. The namelist processor does not check to ensure that one does not put elements into nonexistent slots beyond the end of the array; doing so may cause the processor to hang up or crash.

Wildcards are allowed in a number of places in elegant and the SDDS Toolkit. The wildcard format is very similar to that used in UNIX:

- * — stands for any number of characters, including none.
- ? — stands for any single character.
- [<list-of-characters>] — stands for any single character from the list. The list may include ranges, such as a-z, which includes all characters between and including 'a' and 'z' in the ASCII character table.

The special characters *, ?, [, and] are entered literally by preceeding the character by a backslash (e.g., *).

In many places where a filename is required in an elegant namelist, the user may supply a so-called "incomplete" filename. An incomplete filename has the sequence "%s" imbedded in it, for which is substituted the "rootname." The rootname is by default the filename (less the extension) of the lattice file. The most common use of this feature is to cause elegant to create names for all output files that share a common filename but differ in their extensions. Post-processing can be greatly simplified by adopting this naming convention, particularly if one consistently uses the same extension for the same type of output. Recommended filename extensions are given in the lists below.

When elegant reads a namelist command, one of its first actions is to print the namelist back to the standard output. This printout includes all the variables in the namelist and their values. Occasionally, the user may see a variable listed in the printout that is not in this manual. These are often obsolete and are retained only for backward compatibility, or else associated with a feature that is not fully supported. Use of such "undocumented features" is discouraged.

elegant supports substitution of fields in namelists using the commandline macro option. This permits making runs with altered parameters without editing the input file. Macros inside the input file have one of two forms: <tag> or $tag. To perform substitution, use the syntax

elegant *inputfile* -macro=*tag1*=*value1*[,*tag2*=*value2*...]

When using this feature, it is important to substitute the value of rootname (in run_setup) so that one can get a new set of output files (assuming use of the suggested "%s" field in all the output file names). One may give the macro option any number of times, or combine all substitutions in one option.

4.1 alter_elements

- type: action command.
- function: modify the value of a parameter for one or more elements

```
&alter_elements
        STRING name = NULL;
        STRING item = NULL;
        STRING type = NULL;
        STRING exclude = NULL;
        double value = 0;
        STRING string_value = NULL;
        long differential = 0;
        long multiplicative = 0;
        long verbose = 0;
        long allow_missing_parameters = 0;
&end
```

- `name` — A possibly-wildcard-containing string giving the names of the elements to alter. If not specified, then one must specify `type`.
- `item` — The name of the parameter to alter.
- `type` — A possibly-wildcard-containing string giving the names of element *types* to alter. May be specified with `name` or by itself.
- `exclude` — A possibly-wildcard-containing string giving the names of elements to excluded from alteration.
- `value`, `string_value` — The new value for the parameter. Use `string_value` only if the parameter takes a character string as its value.
- `differential` — If nonzero, the new value is the predefined value of the parameter plus the quantity given with `value`.
- `multiplicative` — If nonozero, the new given value is the predefined value of the parameter times the quantity given with `value`.
- `verbose` — If nonzero, information is printed to the standard output describing what elements are changed.
- `allow_missing_parameters` — If nonzero, then it is not an error if an element does not have the parameter named with `item`. Normally, such an occurence is an error and terminates the program.

4.2 amplification_factors

- type: action command.
- function: compute corrected and uncorrected orbit amplification factors and functions.

```
&amplification_factors
    STRING output = NULL;
    STRING uncorrected_orbit_function = NULL;
    STRING corrected_orbit_function = NULL;
    STRING kick_function = NULL;
    STRING name = NULL;
    STRING type = NULL;
    STRING item = NULL;
    STRING plane = NULL;
    double change = 1e-3;
    long number_to_do = -1;
    double maximum_z = 0;
&end
```

- `output` — The (incomplete) name of a file for text output. Recommended value: "%s.af".
- `uncorrected_orbit_function` — The (incomplete) name of a file for an SDDS-format output of the uncorrected-orbit amplification function. Recommended value: "%s.uof".
- `corrected_orbit_function` — The (incomplete) name of a file for an SDDS-format output of the corrected-orbit amplification function. Recommended value: "%s.cof".
- `kick_function` — The (incomplete) name of a file for an SDDS-format output of the kick amplification function. Recommended value: "%s.kaf".
- `name` — The optionally wildcarded name of the orbit-perturbing elements.
- `type` — The optional type name of the the orbit-perturbing elements.
- `item` — The parameter of the elements producing the orbit.
- `plane` — The plane ("h" or "v") to examine.
- `change` — The parameter change to use in computing the amplification.
- `number_to_do` — The number of elements to perturb.
- `maximum_z` — The maximum z coordinate of the elements to perturb.

4.3 analyze_map

- type: action command.
- function: find the approximate first-order matrix and related quantities for an accelerator by tracking.

```
&analyze_map
    STRING output = NULL;
    double delta_x = 1e-6;
    double delta_xp = 1e-6;
    double delta_y = 1e-6;
    double delta_yp = 1e-6;
    double delta_s  = 1e-6;
    double delta_dp = 1e-6;
    long center_on_orbit = 0;
    long verbosity = 0;
&end
```

- `output` — The (incomplete) name of a file for SDDS output.
  - Recommended value: "%s.ana".
  - File contents: A series of dumps, each consisting of a single data point containing the centroid offsets for a single turn, the single-turn R matrix, the matched Twiss parameters, tunes, and dispersion functions.
- `delta_X` — The amount by which to change the quantity X in computing the derivatives that give the matrix elements.
- `center_on_orbit` — A flag directing the expansion to be made about the closed orbit instead of the design orbit.
- `verbosity` — The larger this value, the more output is printed during computations.

4.4 bunched_beam

- type: setup command.
- function: set up for tracking of particle coordinates with various distributions.

```
&bunched_beam
    STRING bunch = NULL;
    long n_particles_per_bunch = 1;
    double time_start = 0;
    STRING matched_to_cell = NULL;
    double emit_x  = 0;
    double emit_nx  = 0;
    double beta_x  = 1.0;
    double alpha_x = 0.0;
    double eta_x   = 0.0;
    double etap_x  = 0.0;
    double emit_y  = 0;
    double emit_ny  = 0;
    double beta_y  = 1.0;
    double alpha_y = 0.0;
    double eta_y   = 0.0;
    double etap_y  = 0.0;
    long use_twiss_command_values = 0;
    double Po = 0.0;
    double sigma_dp = 0.0;
    double sigma_s = 0.0;
    double dp_s_coupling = 0;
    double emit_z = 0;
    double beta_z = 0;
    double alpha_z = 0;
    double momentum_chirp = 0;
    long one_random_bunch = 1;
    long symmetrize = 0;
    long halton_sequence[3] = {0, 0, 0};
    long halton_radix[6] = {0, 0, 0, 0, 0, 0};
    long randomize_order[3] = {0, 0, 0};
    long limit_invariants = 0;
    long limit_in_4d = 0;
    long enforce_rms_values[3] = {0, 0, 0};
    double distribution_cutoff[3] = {2, 2, 2};
    STRING distribution_type[3] = {"gaussian","gaussian","gaussian"};
    double centroid[6] = {0.0, 0.0, 0.0, 0.0, 0.0, 0.0};
    long first_is_fiducial = 0;
    long save_initial_coordinates = 1;
&end
```

- `bunch` — The (incomplete) name of an SDDS file to which the phase-space coordinates of the bunches are to be written. Recommended value: "%s.bun".

- `n_particles_per_bunch` — Number of particles in each bunch.

- `time_start` — The central value of the time coordinate for the bunch.

- `matched_to_cell` — The name of a beamline from which the Twiss parameters of the bunch are to be computed.

- `emit_X` — RMS emittance for the X plane.

- `emit_nX` — RMS normalized emittance for the X plane. Ignored if `emit_X` is nonzero.

- `beta_X`, `alpha_X`, `eta_X`, `etap_X` — Twiss parameters for the X plane.

- `use_twiss_command_values` — If nonzero, then the values for β, α, η, and η' are taken from the `twiss_output` command. It is an error if no `twiss_output` command has been given.

- `Po` — Central momentum of the bunch.

- `sigma_dp`, `sigma_s` — Fractional momentum spread, δ, and bunch length. Note that `sigma_s` is actually the length in $\beta_z * c * t$, so that for $\beta_z << 1$ the length of the bunch in time will be greater than one might expect.

- `dp_s_coupling` — Specifies the coupling between s and δ, defined as $\langle s\delta \rangle / (\sigma_s \sigma_\delta)$.

- `emit_z`, `beta_z`, `alpha_z` — Provide another way to specify the longitudinal phase space, either separately from or in combination with `sigma_dp`, `sigma_s`, and `dp_s_coupling`.

  Basically, which values `elegant` uses depends on what one sets to nonzero values. If one sets emit_z, then sigma_dp, sigma_s, and dp_s_coupling are ignored. If one doesn't set emit_z, then `elegant` uses sigma_dp and sigma_s; it additionally uses alpha_z if it is nonzero, otherwise it uses dp_s_coupling. For reference, the relationship between them is $C = \frac{\Sigma_{56}}{\sqrt{\Sigma_{55}\Sigma_{66}}} = -\frac{\alpha}{\sqrt{1+\alpha^2}}$. Note that to impart a chirp that results in compression for $R_{56} < 0$ (e.g., a normal four-dipole chicane), one must have $\alpha_z < 0$ or $C > 0$.

- `momentum_chirp` — Permits imparting an additional momentum chirp to the beam, in units of 1/m. E.g., a value of 1 indicates that a 1mm long bunch has a linear variation in momentum of 0.1% from end-to-end. A positive chirp is needed to provide compression of a bunch with an ordinary $R_{56} < 0$ four-dipole chicane.

- `one_random_bunch` — If non-zero, then only one random particle distribution is generated. Otherwise, a new distribution will be generated for every simulation step.

- `enforce_rms_values[3]` — Flags, one for each plane, indicating whether to force the distribution to have the specified RMS properties.

- `distribution_cutoff[3]` — Distribution cutoff parameters for each plane. For gaussian distributions, this is the number of sigmas to use. For other distributions, this number simply multiplies the sizes. This is potentially confusing and hence it is suggested that the distribution cutoff be set to 1 for nongaussian beams.

- `distribution_type[3]` — Distribution type for each plane. May be "gaussian", "hard-edge", "uniform-ellipse", "shell", "dynamic-aperture", "line", "halo(gaussian)".

  For the transverse plane, the interpretation of the emittance is different for the different beam types. For gaussian beams, the emittances are rms values. For all other types, $\sqrt{\epsilon * \beta}$ times the distribution cutoff defines the edge of the beam in position space, while $\sqrt{\epsilon * (1 + \alpha^2)/\beta}$ times the distribution cutoff defines the edge of the beam in slope space.

  A hard-edge beam is a uniformly-filled parallelogram in phase space. A uniform-ellipse beam is a uniformly-filled ellipse in phase space. A shell beam is a hollow ellipse in phase space. A dynamic aperture beam has zero slope and uniform spacing in position coordinates. A line beam is a line in phase space. A "halo(gaussian)" beam is the part of the gaussian distribution *beyond* the distribution cutoff.

- `limit_invariants` — If non-zero, the distribution cutoffs are applied to the invariants, rather than to the coordinates. This is useful for gaussian beams when the distribution cutoff is small.

- `limit_in_4d` — If non-zero, then the transverse distribution is taken to be a 4-d gaussian or uniform distribution. One of these must be chosen using the `distribution_type` control. It must be the same for x and y. This is useful, for example, if you want to make a cylindrically symmetric beam.

- `symmetrize` — If non-zero, the distribution is symmetric under changes of sign in the coordinates. Automatically results in a zero centroid for all coordinates.

- `halton_sequence[3]` and `halton_radix[6]` — This provides a "quiet-start" feature by choosing Halton sequences in place of random number generation. There are three new variables that control this feature. `halton_sequence` is an array of three flags that permit turning on Halton sequence generation for the horizontal, vertical, or longitudinal planes. For example, `halton_sequence[0] = 3*1` will turn on Halton sequences for all three planes, while `halton_sequence[2] = 1`, will turn it on for the longitudinal plane only.

  `halton_radix` is an array of six integers that permit giving the radix for each sequence (i.e., x, x', y, y', t, p). Each radix must be a prime number. One should never use the same prime for two sequences, unless one randomizes the order of the sequences relative to each other (see the next item). If these are left at zero, then elegant chooses values that eliminate phase-space banding to some extent. The user is cautioned to plot all coordinate combinations for the initial phase space to ensure that no unacceptable banding is present.

  A suggested way to use Halton sequences is to set `halton_radix[0] = 2, 3, 2, 3, 2, 3` and to set `randomize_order[0] = 2, 2, 2,`. This avoids banding that may result from choosing larger radix values.

- `randomize_order[3]` — Allows randomizing the order of assigned coordinates for the pairs (x, x'), (y, y'), and (t,p). 0 means no randomization; 1 means randomize (x, x', y, y', t, p) values independently, which destroys any x-x', y-y', and t-p correlations; 2 means randomize (x, x'), (y, y'), and (t, p) in pair-wise fashion. This is used with Halton sequences to remove banding. It is suggested that that the user employ `sddsanalyzebeam` to verify that the beam properties when randomization is used.

- `centroid[6]` — Centroid offsets for each of the six coordinates.

- `first_is_fiducial` — Specifies that the first beam generated shall be a single particle beam, which is suitable for fiducialization. See the section on "Fiducialization in `elegant`" for more discussion.

- `save_initial_coordinates` — A flag that, if set, results in saving initial coordinates of tracked particles in memory. This is the default behavior. If unset, the initial coordinates are not saved, but are regenerated each time they are needed. This is more memory efficient and is useful for tracking very large numbers of particles.

4.5 chromaticity

- type: setup command.
- function: set up for chromaticity correction.

```
&chromaticity
    STRING sextupoles = NULL;
    double dnux_dp = 0;
    double dnuy_dp = 0;
    double sextupole_tweek = 1e-3;
    double correction_fraction = 0.9;
    long n_iterations = 5;
    double tolerance = 0;
    STRING strength_log = NULL;
    long change_defined_values = 0;
    double strength_limit = 0;
    long use_perturbed_matrix = 0;
&end
```

- `sextupoles` — List of names of elements to use to correct the chromaticities.
- `dnux_dp`, `dnuy_dp` — Desired chromaticity values.
- `sextupole_tweek` — Amount by which to tweak the sextupoles to compute derivatives of chromaticities with respect to sextupole strength. [The word "tweak" is misspelled "tweek" in the code.]
- `correction_fraction` — Fraction of the correction to apply at each iteration. In some cases, correction is unstable at this number should be reduced.
- `n_iterations` — Number of iterations of the correction to perform.
- `tolerance` — Stop iterating when chromaticities are within this value of the desired values.
- `strength_log` — The (incomplete) name of an SDDS file to which the sextupole strengths will be written. Recommended value: "%s.ssl".
- `change_defined_values` — Changes the defined values of the sextupole strengths. This means that when the lattice is saved (using `save_lattice`), the sextupoles will have the corrected values. This would be used for correcting the chromaticity of a design lattice, for example, but not for correcting chromaticity of a perturbed lattice.
- `strength_limit` — Limit on the absolute value of sextupole strength (K_2). '
- `use_perturbed_matrix` — If nonzero, requests use of the perturbed correction matrix in performing correction. For difficult lattices with large errors, this may be necessary to obtain correction. In general, it is not necessary and only slows the simulation.

4.6 closed_orbit

- type: setup command.
- function: set up for computation of the closed orbit.

```
&closed_orbit
    STRING output = NULL;
    long output_monitors_only = 0;
    long start_from_centroid = 1;
    double closed_orbit_accuracy = 1e-12;
    long closed_orbit_iterations = 10;
    double iteration_fraction = 1;
    long fixed_length = 0;
    long start_from_recirc = 0;
    long verbosity = 0;
&end
```

- `output` — The (incomplete) name of an SDDS file to which the closed orbits will be written. Recommended value: "%s.clo".
- `output_monitors_only` — If non-zero, indicates that the closed orbit output should include only the data at the locations of the beam-position monitors.
- `start_from_centroid` — A flag indicating whether to force the computation to start from the centroids of the beam distribution.
- `closed_orbit_accuracy` — The desired accuracy of the closed orbit, in terms of the difference between the start and end coordinates, in meters.
- `closed_orbit_iterations` — The number of iterations to take in finding the closed orbit.
- `iteration_fraction` — Fraction of computed change that is used each iteration. For lattices that are very nonlinear or close to unstable, a number less than 1 can be helpful. Otherwise, it only slows the simulation.
- `fixed_length` — A flag indicating whether to find a closed orbit with the same length as the design orbit by changing the momentum offset.
- `start_from_recirc` — A flag indicating whether to compute the closed orbit from the recirculation (`recirc`) element in the beamline. In general, if one has a recirculation element, one should give this flag.
- `verbosity` — A larger value results in more printouts during the computations.

4.7 correct

- type: setup command.
- function: set up for correction of the trajectory or closed orbit.

```
&correct
    STRING mode = "trajectory";
    STRING method = "global";
    STRING trajectory_output = NULL;
    STRING corrector_output = NULL;
    STRING statistics = NULL;
    double corrector_tweek[2] = {1e-3, 1e-3};
    double corrector_limit[2] = {0, 0};
    double correction_fraction[2] = {1, 1};
    double correction_accuracy[2] = {1e-6, 1e-6};
    double bpm_noise[2] = {0, 0};
    double bpm_noise_cutoff[2] = {1.0, 1.0};
    STRING bpm_noise_distribution[2] = {"uniform", "uniform"};
    long verbose = 1;
    long fixed_length = 0;
    long fixed_length_matrix = 0;
    long n_xy_cycles = 1;
    long n_iterations = 1;
    long prezero_correctors = 1;
    long track_before_and_after = 0;
    long start_from_centroid = 1;
    long use_actual_beam = 0;
    double closed_orbit_accuracy = 1e-12;
    long closed_orbit_iterations = 10;
    double closed_orbit_iteration_fraction = 1;
    long use_perturbed_matrix = 0;
&end
```

- `mode` — Either “trajectory” or “orbit”, indicating correction of a trajectory or a closed orbit.
- `method` — For trajectories, may be “one-to-one” or “global”. For closed orbit, must be “global”.
- `trajectory_output` — The (incomplete) name of an SDDS file to which the trajectories or orbits will be written. Recommended value: “%s.traj” or “%s.orb”.
- `corrector_output` — The (incomplete) name of an SDDS file to which information about the final corrector strengths will be written. Recommended value: “%s.cor”.
- `statistics` — The (incomplete) name of an SDDS file to which statistical information about the trajectories (or orbits) and corrector strengths will be written. Recommended value: “%s.scor”.

- `corrector_tweek[2]` — The amount by which to change the correctors in order to compute correction coefficients. [The word "tweak" is misspelled "tweek" in the code.]
- `corrector_limit[2]` — The maximum strength allowed for a corrector.
- `correction_fraction[2]` — The fraction of the computed correction strength to actually use for any one iteration.
- `correction_accuracy[2]` — The desired accuracy of the correction in terms of the RMS BPM values.
- `bpm_noise[2]` — The BPM noise level.
- `bpm_noise_cutoff[2]` — Cutoff values for the random distributions of BPM noise.
- `bpm_noise_distribution[2]` — May be either "gaussian", "uniform", or "plus_or_minus".
- `verbose` — If non-zero, information about the correction is printed during computations.
- `fixed_length` — Indicates that the closed orbit length should be kept the same as the design orbit length by changing the momentum offset of the beam.
- `fixed_length_matrix` — Indicates that for fixed-length orbit correction, the fixed-length matrix should be computed and used. This will improve convergence but isn't always needed.
- `n_xy_cycles` — Number of times to alternate between correcting the x and y planes.
- `n_iterations` — Number of iterations of the correction for each x/y cycle.
- `prezero_correctors` — Flag indicating whether to set the correctors to zero before starting.
- `track_before_and_after` — Flag indicating whether tracking should be done both before and after correction.
- `start_from_centroid` — Flag indicating that correction should start from the beam centroid. For orbit correction, only the beam momentum centroid is relevant.
- `use_actual_beam` — Flag indicating that correction should employ tracking of the beam distribution rather than a single particle.
- `closed_orbit_accuracy` — Accuracy of closed orbit computation.
- `closed_orbit_iterations` — Number of iterations of closed orbit computation.
- `closed_orbit_iteration_fraction` — Fraction of change in closed orbit to use at each iteration.
- `use_perturbed_matrix` — If nonzero, specifies that prior to each correction `elegant` shall recompute the response matrix. This is useful if the lattice is changing significantly between corrections.

4.8 correction_matrix_output

- type: setup/action command.
- function: provide output of the orbit/trajectory correction matrix.

```
&correction_matrix_output
    STRING response[2] = NULL, NULL;
    STRING inverse[2] = NULL, NULL;
    long KnL_units = 0;
    long BnL_units = 0;
    long output_at_each_step = 0;
    long output_before_tune_correction = 0;
    long fixed_length = 0;
&end
```

- `response` — Array of (incomplete) filenames for SDDS output of the x and y response matrices. Recommnended values: “
- `inverse` — Array of (incomplete) filenames for SDDS output of the x and y inverse response matrices. Recommnended values: “
- `KnL_units` — Flag that, if set, indicates use of “units” of m/K0L rather than m/rad. This results in a sign change for the horizontal data.
- `BnL_units` — Flag that, if set, indicates use of “units” of m/(T*m) rather than m/rad. This is useful for linac work in that the responses are automatically scaled with beam momentum.
- `output_at_each_step` — Flag that, if set, specifies output of the data at each simulation step. By default, the data is output immediately for the defined lattice.
- `output_before_tune_correction` — Flag that, if set, specifies that when `output_at_each_step` is set, that output shall occur prior to correcting the tunes.
- `fixed_length` — Flag that, if set, specifies output of the fixed-path-length matrix.

4.9 correct_tunes

- type: setup command.
- function: set up for correction of the tunes.

```
&correct_tunes
    STRING quadrupoles = NULL;
    double tune_x = 0;
    double tune_y = 0;
    long n_iterations = 5;
    double correction_fraction = 0.9;
    double tolerance = 0;
    long step_up_interval = 0;
    double max_correction_fraction = 0.9;
    double delta_correction_fraction = 0.1;
    STRING strength_log = NULL;
    long change_defined_values = 0;
    long use_perturbed_matrix = 0;
&end
```

- `quadrupoles` — List of names of quadrupoles to be used. Only two may be given.
- `tune_x`, `tune_y` — Desired x and y tune values. If not given, the desired values are assumed to be the unperturbed tunes.
- `n_iterations` — The number of iterations of the correction to perform.
- `correction_fraction` — The fraction of the correction to apply at each iteration.
- `tolerance` — When both tunes are within this value of the desired tunes, the iteration is stopped.
- `step_up_interval` — Interval between increases in the correction fraction.
- `max_correction_fraction` — Maximum correction fraction to allow.
- `delta_correction_fraction` — Change in correction fraction after each `step_up_interval` steps.
- `strength_log` — The (incomplete) name of a SDDS file to which the quadrupole strengths will be written as correction proceeds. Recommended value: "%s.qst".
- `change_defined_values` — Changes the defined values of the quadrupole strengths. This means that when the lattice is saved (using `save_lattice`), the quadrupoles will have the corrected values. This would be used for correcting the tunes of a design lattice, for example, but not for correcting tunes of a perturbed lattice.
- `use_perturbed_matrix` — If nonzero, requests use of the perturbed correction matrix in performing correction. For difficult lattices with large errors, this may be necessary to obtain correction. In general, it is not necessary and only slows the simulation.

4.10 divide_elements

- type: setup command.
- function: define how to subdivide certain beamline elements.
- notes:
  - Any number of these commands may be given.
  - Not effective unless given prior to `run_setup`.
  - The `element_divisions` field in `run_setup` provides a simpler, but less flexible, method of performing element division. At present, these element types may be divided: `QUAD`, `SBEN`, `RBEN`, `DRIF`, `SEXT`.
  - Only effective if given prior to the `run_setup` command.
- warnings:
  - Using `save_lattice` and element divisions together will produce an incorrect lattice file.
  - Element subdivision may produce unexpected results when used with `load_parameters` or parameters saved via the `parameter` entry of the `run_setup` command. If you wish to load parameters while doing element divisions or if you wish to load parameters from a run that had element divisions in effect, you should not load length data for any elements that are (or were) split. The name and item pattern features of `load_parameters` are helpful in restricting what is loaded.

```
&divide_elements
    STRING name = NULL;
    STRING type = NULL;
    STRING exclude = NULL;
    long divisions = 0;
    double maximum_length = 0;
    long clear = 0;
&end
```

- `name` — A possibly wildcard-containing string specifying the elements to which this specification applies.
- `type` — A possibly wildcard-containing string specifying the element types to which this specification applies.
- `exclude` — A possibily wildcard-containing string specifying elements to be excluded from the specification.
- `divisions` — The number of times to subdivide the specified elements. If zero, then `maximum_length` should be nonzero.
- `maximum_length` — The maximum length of a slice. This is usually preferrable to specifying the number of divisions, particularly when the elements divided may be of different lengths. If zero, then `divisions` should be nonzero.
- `clear` — If nonzero, all prior division specifications are deleted.

4.11 error_element

- type: setup command.
- function: assert a random error defintion for the accelerator.

```
&error_element
    STRING name = NULL;
    STRING element_type = NULL;
    STRING item = NULL;
    STRING type = "gaussian";
    double amplitude = 0.0;
    double cutoff = 3.0;
    long bind = 1;
    long bind_number = 0;
    long post_correction = 0;
    long fractional = 0;
    long additive = 1;
    STRING after = NULL;
    STRING before = NULL;
&end
```

- `name` — The possibly wildcarded name of the elements for which errors are being specified.
- `element_type` — An optional, possibly wildcarded string giving the type of elements to which the errors should be applied. E.g., `element_type=*MON*` would match all beam position monitors. If this item is given, then `name` may be left blank.
- `item` — The parameter of the elements to which the error pertains.
- `type` — The type of random distribution to use. May be one of "uniform", "gaussian", or "plus_or_minus". A "plus_or_minus" error is equal in magnitude to the amplitude given, with the sign randomly chosen.
- `amplitude` — The amplitude of the errors.
- `cutoff` — The cutoff for the random distribution.
- `bind`, `bind_number` — If `bind_number` is positive, then a positive value of `bind` indicates that `bind_number` successive elements having the same name will have the same error value. If `bind` is negative, then the sign of the error is alternated between successive elements.
- `post_correction` — A flag indicating whether the errors should be added after orbit, tune, and chromaticity correction.
- `fractional` — A flag indicating whether the errors are fractional, in which case the amplitude refers to the amplitude of the fractional error.
- `additive` — A flag indicating that the errors should be added to the prior value of the parameter. If zero, then the errors replace the prior value of the parameter.

- `after` — The name of an element. If given, the error is applied only to elements that follow the named element in the beamline.
- `before` — The name of an element. If given, the error is applied only to elements that precede the named element in the beamline.

4.12 error_control

- type: setup command
- function: overall control of random errors.

```
&error_control
    long clear_error_settings = 1;
    long summarize_error_settings = 0;
    STRING error_log = NULL;
&end
```

- `clear_error_settings` — Clear all previous error settings.
- `summarize_error_settings` — Summarize current error settings.
- `error_log` — The (incomplete) name of a SDDS file to which error values will be written. Recommended value: "%s.erl".

4.13 find_aperture

- type: action command.
- function: find the aperture in (x, y) space for an accelerator.

```
&find_aperture
    STRING output = NULL;
    STRING boundary = NULL;
    STRING mode = "many-particle";
    double xmin = -0.1;
    double xmax =  0.1;
    double ymin =  0.0;
    double ymax =  0.1;
    long nx  = 21;
    long ny  = 11;
    long n_splits = 0;
    double split_fraction = 0.5;
    double desired_resolution = 0.01;
    long verbosity = 0;
    long assume_nonincreasing = 0;
&end
```

- `output` — The (incomplete) name of an SDDS file to send output to. Recommended value: "%s.aper".
- `boundary` — The (incomplete) name of an SDDS file for the boundary points of the aperture search. Recommended value: "%s.bnd".
- `xmin`, `xmax`, `ymin`, `ymax` — Region of the aperture search.
- `mode` — May be "many-particle" or "single-particle". Many-particle searching is much faster, but does not allow interval splitting to search for the aperture boundary.
- `nx` — Number of x values to take in initial search.
- `ny` — Number of y values to take in search.
- `n_splits` — If positive, the number of times to do interval splitting. Interval splitting refers to searching between the original grid points in order to refine the results.
- `split_fraction` — If interval splitting is done, how the interval is split.
- `desired_resolution` — If interval splitting is done, fraction of `xmax-xmin` to which to resolve the aperture.
- `assume_nonincreasing` — If this variable is non-zero, the search assumes that the aperture at $y + \text{sign}(y) * \Delta y$ is no larger than that at y. This results in tracking of fewer particles but may give a pessimistic result.
- `verbosity` — A larger value results in more printouts during computations.

4.14 floor_coordinates

- type: action command.
- function: compute floor coordinates for an accelerator.

```
&floor_coordinates
    STRING filename = NULL;
    double X0 = 0.0;
    double Z0 = 0.0;
    double theta0 = 0.0;
    long include_vertices = 0;
    long vertices_only = 0;
    long magnet_centers = 0;
&end
```

- `filename` — The (incomplete) name of an SDDS file to send output to. Recommended value: "%s.flr".
- `X0`, `Z0`, `theta0` — Initial X, Z, and angle coordinate of the
- `include_vertices` — Flag that, if set, specifies including in the output the coordinates of the vertices of bending magnets.
- `vertices_only` — Flag that, if set, specifies that output will contain only the coordinates of the vertices of bending magnets.
- `magnet_centers` — Flag that, if set, specifies that output will contain the coordinates of the centers of all magnets. By default, the coordinates of the downstream end are given.

4.15 link_control

- type: setup command.
- function: overall control of element parameter links.

```
&link_control
    long clear_links = 1;
    long summarize_links = 0;
    long verbosity = 0;
&end
```

- `clear_links` — Clear all previously set links.
- `summarize_links` — Summarize all current set links.
- `verbosity` — A larger value results in more output during computations.

4.16 link_elements

- type: setup command.
- function: assert a link between parameters of accelerator elements.

```
&link_elements
    STRING target = NULL;
    STRING exclude = NULL;
    STRING item = NULL;
    STRING source = NULL;
    STRING source_position = "before";
    STRING mode = "dynamic";
    STRING equation = NULL;
&end
```

- `target` — The name of the elements to be modified by the link. May contain wild-cards.
- `exclude` — Wildcard sequence to match to element names. If a match is found, the element is excluded from the link.
- `item` — The parameter that will be modified.
- `source` — The name of the elements to be linked to.
- `source_position` — May be one of "before", "after", "adjacent", "nearest", or "same-occurrence".
- `mode` — May be either "dynamic" or "static". A dynamic link is asserted whenever the source is changed (during correction, for example). A static link is asserted only when an error or variation is imparted to the source, and at the end of correction.
- `equation` — An `rpn` equation for the new item value in terms of the item values for the source. The prior value of the item is on the top of the stack. To refer to the source parameter values, use the name of the parameters. These names must appear in capital letters.

4.17 load_parameters

- type: setup command.
- function: load parameters for elements from an SDDS file.

```
&load_parameters
        STRING filename = NULL;
        STRING include_name_pattern = NULL;
        STRING exclude_name_pattern = NULL;
        STRING include_item_pattern = NULL;
        STRING exclude_item_pattern = NULL;
        STRING include_type_pattern = NULL;
        STRING exclude_type_pattern = NULL;
        long change_defined_values = 0;
        long clear_settings = 0;
        long allow_missing_elements = 0;
long force_occurence_data = 0;
        long verbose = 0;
&end
```

- `filename` — Name (possibly containing the "of SDDS file from which to take data. The file must contain some of the following columns:
  - ElementName — Required string column. The name of the element to change.
  - ElementParameter — Required string column. The name of the parameter of the element to change.
  - ParameterValue — Optional double column. If given, gives value of the parameter named in ElementParameter for element named in ElementName.
  - ParameterValueString — Optional string column. If ParameterValue is not present, then this column must be present. The string data will be scanned, if necessary, to obtain a value for the parameter.
  - ParameterMode — Optional string column. If given, for each row the value must be one of "absolute", "differential", "ignore", or "fractional". The meaning of these modes is as follows: absolute mode means the given value is used as the new value for the parameter; differential mode means the given value is added to the existing value for the parameter; ignore mode means the value is ignored; fractional mode means the existing value is increased by the product of the given value and the existing value (i.e., the given value is a fractional change).

  Unless `change_defined_values` is set, successive pages of the file are used for successive steps of the simulation. Several `elegant` commands generate output that may be used (on a subsequent run) with `load_parameters`; among these are the tune and chromaticity correction commands and the `run_setup` command (parameters output).

- `include_name_pattern`, `exclude_name_pattern` — Wildcard patterns to be used in selecting, respectively, which elements to include and which to exclude from loading.

- `include_item_pattern`, `exclude_item_pattern` — Wildcard patterns to be used in selecting, respectively, which items (i.e., which element parameters) to include and which to exclude from loading.

- `include_type_pattern`, `exclude_type_pattern` — Wildcard patterns to be used in selecting, respectively, which element types (e.g., QUAD, DRIFT) to include and which to exclude from loading.

- `change_defined_values` — Changes the defined values of the parameters. This means that when the lattice is saved (using `save_lattice`), the parameters will have the altered values. Also, if one wants to alter the values for all steps of the simulation, one must set this flag.

  Note that the `ElementOccurence` data is normally ignored if `change_defined_values` is nonzero. This is because there is only one definition of each element, even if it is used multiple times. This behavior can be altered with the next control.

- `force_occurence_data` — If set, then occurence data is used even in `change_defined_values` mode.

- `clear_settings` — If set, clear all settings and files being used for loading parameters.

- `allow_missing_elements` — If set, allow elements in the file that are not in the lattice. In this case, the nonapplicable data is simply ignored.

- `verbose` — If set, provide informational printouts about changes to parameters.

4.18 matrix_output

- type: setup/action command.
- function: generate matrix output, or set up to do so later.

```
&matrix_output
    STRING printout = NULL;
    long printout_order = 1;
    long full_matrix_only = 0;
    STRING SDDS_output = NULL;
    long SDDS_output_order = 1;
    STRING SDDS_output_match = NULL;
    long output_at_each_step = 0;
    STRING start_from = NULL;
    long start_from_occurrence = 1;
&end
```

- `printout` — The (incomplete) name of a file to which the matrix output will be printed (as text). Recommended value: "%s.mpr".
- `printout_order` — The order to which the matrix is printed.
- `full_matrix_only` — A flag indicating that only the matrix of the entire accelerator is to be output.
- `SDDS_output` — The (incomplete) name of an SDDS file to which the matrix will be written. Recommended value: "%s.mat".
- `SDDS_output_order` — The order to which the matrix is output in SDDS format.
- `SDDS_output_match` — A wildcard string which element names must match in order for data to appear in the SDDS output file.
- `output_at_each_step` — A flag indicating whether matrix output is desired at every simulation step.
- `start_from` — The optional name of the accelerator element from which to begin concatenation and output.
- `start_from_occurrence` — If `start_from` is not NULL, the number of the occurrence of the named element from which to start.

4.19 optimize

- type: action command.
- function: perform optimization.
- note: on UNIX systems, the user may press Control-C to force `elegant` to terminate optimization and proceed as if optimization had converged. (To genuinely terminate the run during optimization press Control-C twice.) This is very useful if one wants to get a look at the partially optimized result. If one uses parameter saving (`run_setup`) or `save_lattice` one can make a new run that starts from the optimized result.

```
&optimize
     long summarize_setup = 0;
&end
```

- `summarize_setup` — A flag indicating, if set, that a summary of the optimization parameters should be printed.

4.20 optimization_constraint

- type: setup command.
- function: define a constraint for optimization.
- note: This command is not recommended. It is better to put constraints into the optimization equation (via the `equation` parameter of `optimization_setup` or via `optimization_term`). The reason is that the hard constraints imposed by `optimization_constraint` may make it more difficult for the optimizer to converge.

```
&optimization_constraint
    STRING quantity = NULL;
    double lower = 0;
    double upper = 0;
&end
```

- `quantity` — The quantity to be constrained, given as an `rpn` expression in terms of the optimization variables, the optimization covariables, and and the "final" parameters (see the entry for `run_setup` for the last of these). The optimization (co)variables are referred to as `<element-name>.<parameter-name>`, in all capital letters.
- `lower`, `upper` — The lower and upper limits allowed for the expression.

4.21 optimization_covariable

- type: setup command.
- function: define an element parameter to be varied as a function of optimization parameters.

```
&optimization_covariable
    STRING name = NULL;
    STRING item = NULL;
    STRING equation = NULL;
&end
```

- `name` — The name of the element.
- `item` — The parameter of the element to be changed.
- `equation` — An `rpn` equation for the value of the parameter in terms of the values of any parameters of any optimization variable. These latter appear in the equation in the form `<element-name>.<parameter-name>`, in all capital letters.

4.22 optimization_setup

- type: setup command.
- function: define overall optimization parameters and methods.

```
&optimization_setup
    STRING equation = NULL;
    STRING mode = "minimize";
    STRING method = "simplex";
    double tolerance = -0.01;
    double target = 0;
    long soft_failure = 1;
    long n_passes = 2;
    long n_evaluations = 500;
    long matrix_order = 1;
    STRING log_file = NULL;
    long output_sparsing_factor = 0;
&end
```

- `equation` — An `rpn` equation for the optimization function, expressed in terms of any parameters of any optimization variables and the "final" parameters of the beam (as recorded in the `final` output file available in the `run_setup` namelist). The optimization variables or covariables may appear in the equation in the form `<element-name>.<parameter-name>`, all in capital letters. Data from MARK elements with FITPOINT=1 may be used via symbols of the form `<element-name>#<occurrence-number>.<parameter-name>`, where `<parameter-name>` can be a Twiss parameter name (if the twiss command was given), a floor coordinate name (if the floor command was given, or a beam-size or centroid name. The parameter names are the same as those used in the corresponding output files. Beam sizes and centroids are from tracking the particle distribution.

  If the twiss command was given, one may also refer to statistics of Twiss parameters in the form `<statistic>.<parameter-name>`, where `<statistic>` is either `min` or `max`. One may also use the symbols `nux`, `dnux/dp`, (and corresponding symbols for y), `alphac`, and `alphac2`. These are the tune, chromaticity, and first- and second- order momentum compaction factors. The final values of the Twiss parameters are referred to simply as `betax`, `etax`, etc.

  If the twiss command was given and radiation integral computation was requested, one may use `ex0` and `Sdelta0` for the equilibrium emittance and momentum spread, plus `J<plane>` and `tau<plane>` for the damping partition and damping time, where `<plane>` is `x`, `y`, or `delta`.

  If the `floor_coordinates` command was given, one may use `X`, `Z`, and `theta` to refer to the final values of the floor coordinates.

  If the `sasefel` command was given, one may use variables of the form `SASE.<property>`, where `<property>` is one of `gainLength`, `saturationLength`, `saturationPower`, or `lightWavelength`.

  Finally, one may use any of the names from the "final" output file (see `run_setup`), e.g., `Sx` (x beamsize) or `eny` (y normalized emittance). These refer to tracked properties of the beam.

The equation may be left blank, in which case the user must give one or more `optimization_term` commands. These use the same symbols, of course.

- `mode` — May be either "minimize" or "maximize".
- `method` — May be one of "simplex", "grid", and "sample".
- `tolerance` — The convergence criterion for the optimization, with a negative value indicating a fractional criterion.
- `target` — The value which, if reached, results in immediate termination of the optimization, whether it has converged or not.
- `soft_failure` — A flag indicating, if set, that failure of an optimization pass should not result in termination of the optimization.
- `n_passes` — The number of optimization passes made to achieve convergence ("simplex" only).
- `n_evaluations` — The number of allowed evaluations of the optimization function. If simplex optimization is used, this is the number of allowed evaluations per pass.
- `matrix_order` — Specifies the highest order of matrix elements that should be available for fitting. Elements up to third order are available for the terminal point of the beamline, and up to secod order for interior fit points. Names for first-, second-, and third-order elements are of the form `R`*ij*, `T`*ijk*, and `U`*ijkl*.
- `log_file` — A file to which progress reports will be written as optimization proceeds. For SDDS data, use the `final` output file from the `run_setup` namelist.
- `output_sparsing_factor` — If set to a value larger than 0, results in sparsing of output to the "final" file (see `run_setup`). This can make a significant difference in the optimization speed.

4.23 optimization_term

- type: setup command.
- function: define optimization equation via individual terms

```
&optimization_
    STRING term = NULL;
&end
```

- `term` — An `rpn` expression giving one term to be optimized. If more than one `optimization_term` command is given, then the terms are added. The advantage of using this command over giving an equation via `optimization_setup` is that `elegant` will report the value of each term as it performs the optimization (if a `log_file` is given to `optimization_setup`). This permits determination of which terms are causing problems for the optimization.

4.24 optimization_variable

- type: setup command.
- function: defines a parameter of an element to be used in optimization.

```
&optimization_variable
    STRING name = NULL;
    STRING item = NULL;
    double lower_limit = 0;
    double upper_limit = 0;
    double step_size = 1;
&end
```

- `name` — The name of the element.
- `item` — The parameter of the element to be varied.
- `lower_limit`, `upper_limit` — The lower and upper limits allowed for the parameter. If these are equal, the range of the parameter is unlimited.
- `step_size` — The initial step size ("simplex" optimization) or the grid size in this dimension ("grid" or "sample" optimization).

4.25 print_dictionary

- type: action command.
- function: print dictionary of supported accelerator elements.

```
&print_dictionary
    STRING filename = NULL;
&end
```

- `filename` — The name of a text file to which the dictionary will be printed.

4.26 rpn_expression

- type: action/setup command.
- function: pass an expression directly to the rpn submodule for execution.

```
&rpn_expression
    STRING expression = NULL;
&end
```

- `expression` — An `rpn` expression. This expression is executed immediately and can be used, for example, to read in `rpn` commands from a file or store values in `rpn` memories.

4.27 run_control

- type: setup command.
- function: set up the number of simulation steps and passes.

```
&run_control
    long n_steps = 1;
    double bunch_frequency = 0;
    long n_indices = 0;
    long n_passes = 1;
    long reset_rf_for_each_step = 1;
    long first_is_fiducial = 0;
    long restrict_fiducialization = 0;
&end
```

- `n_steps` — The number of separate repetitions of the action implied by the next action command. If random errors are defined, this is also the number of separate error ensembles.
- `bunch_frequency` — The frequency to use in calculating the time delay between repetitions.
- `n_indices` — The number of looping indices for which to expect definitions in subsequent `vary_element` commands. If nonzero, then `n_steps` is ignored.
- `n_passes` — The number of passes to make through the beamline per repetition.
- `reset_rf_for_each_step` — If nonzero, the rf phases are established anew for each bunch tracked. Should be zero to simulate phase and timing jitter.
- `first_is_fiducial` — If nonzero, the first bunch seen is taken to establish the reference phases and momentum profile. If zero, each bunch is treated as a new fiducializing bunch.
- `restrict_fiducialization` — If nonzero, then momentum profile fiducialization occurs only after elements that are intended change the momentum, such as rf cavities. If zero, then each element is fiducialized to the average momentum of the beam.

4.28 run_setup

- type: setup command.
- function: set global parameters of the simulation and define primary input and output files.

```
&run_setup
    STRING lattice = NULL;
    STRING use_beamline = NULL;
    STRING rootname = NULL;
    STRING output = NULL;
    STRING centroid = NULL;
    STRING sigma = NULL;
    STRING final = NULL;
    STRING acceptance = NULL;
    STRING losses = NULL;
    STRING magnets = NULL;
    STRING semaphore_file = NULL;
    STRING parameters = NULL;
    long combine_bunch_statistics = 0;
    long wrap_around = 1;
    long default_order = 2;
    long concat_order = 0;
    long print_statistics = 0;
    long random_number_seed = 987654321;
    long correction_iterations = 1;
    double p_central = 0.0;
    STRING expand_for = NULL;
    long tracking_updates = 1;
    long echo_lattice = 0;
    STRING search_path = NULL;
    long element_divisions = 0;
&end
```

- `lattice` — Name of the lattice definition file.
- `echo_lattice` — If nonzero, the lattice input is echoed to the standard output as the lattice is parsed. This can help detect certain problems with the lattice that cause `elegant` to crash.
- `use_beamline` — Name of the beamline to use.
- `rootname` — Filename fragment used in forming complete names from incomplete filenames. By default, the filename minus extension of the input file is used.
- `output` — The (incomplete) name of an SDDS file into which final phase-space coordinates will be written. Recommended value: "%s.out".
- `centroid` — The (incomplete) name of an SDDS file into which beam centroids as a function of s will be written. Recommended value: "%s.cen".

- `sigma` — The (incomplete) name of an SDDS file into which the beam sigma matrix as a function of z will be written. Recommended value: "%s.sig".
- `final` — The (incomplete) name of an SDDS file into which final beam and transport parameters will be written. Recommended value: "%s.fin".
- `acceptance` — The (incomplete) name of an SDDS file into which the initial coordinates of transmitted particles will be written. Recommended value: "%s.acc".
- `losses` — The (incomplete) name of an SDDS file into which information on lost particles will be written. Recommended value: "%s.lost".
- `magnets` — The (incomplete) name of an SDDS file into which a magnet layout representation will be written. Recommended value: "%s.mag".
- `semaphore_file` — The (incomplete) name of file that will be created just before exit from the program, but only if no errors occured. If the file exists, it is deleted. This file can be used to record the fact that the run completed without error.
- `parameters` — The (incomplete) name of an SDDS file into which parameters of accelerator elements are written.
- `combine_bunch_statistics` — A flag indicating whether to combine statistical information for all simulation steps. If non-zero, then the `sigma` and `centroid` data will be combined over all simulation steps.
- `wrap_around` — A flag indicating whether the z coordinate should wrap-around or increase monotonically in multipass simulations.
- `default_order` — The default order of transfer matrices used for elements having matrices.
- `concat_order` — If non-zero, the order of matrix concatenation used.
- `print_statistics` — A flag indicating whether to print information as each element is tracked.
- `random_number_seed` — A seed for the random number generators. If zero, a seed will be generated from the system clock.
- `correction_iterations` — Number of iterations of tune and chromaticity correction.
- `p_central` — Central momentum of the beamline, about which expansions are done.
- `expand_for` — Name of an SDDS file containing particle information, from which the central momentum will be set. The file contents are the same as required for `elegant` input with the `sdds_beam` namelist.
- `tracking_updates` — A flag indicating whether to print summary information about tracking.
- `search_path` — Specify a list of pathnames in which to look for input files, including lattice files, wakefield input, particle input, etc. This allows storing common input files in a convenient location without having to put the location into every filename.

- `element_divisions` — Specify how many pieces to split elements into. Only certain elements (basically, those with a matrix) are split. Results in creation of `element_divisions` new elements having the same name as each split element.

4.29 sasefel

- type: setup/action command.
- function: set parameters for computation of SASE FEL gain and other properties.

```
&sasefel
    STRING output = NULL;
    STRING model = "Ming Xie";
    double beta = 0;
    double undulator_K = 3.1;
    double undulator_period = 0.033;
    double slice_fraction = 0.0;
    long n_slices = 0;
&end
```

- `output` — The (incomplete) filename of an SDDS file to which output will be written.
- `model` — The name of the FEL model used. At present, only one model is supported; the "Ming-Xie" model is based on the simple parametrization M. Xie[13].
- `beta` — The value of the beta function, in meters.
- `undulator_K` — The K parameter of the undulator.
- `undulator_period` — The undulator period, in meters.
- `slice_fraction`, `n_slices` — The fraction of beam beam contained by each analysis slice and the number of such slices. By default, no slice analysis is done. Instead, the beam is analyzed only as a whole. If `slice_fraction`*`n_slices` is less than 1, then the slice analysis is centered on the median of the time distribution. E.g., if `n_slices`=1 and `slice_fraction`=0.1, then the central 10% of the beam would be analyzed. More typically, one gives values such that `slice_fraction`*`n_slices` is equal to 1, so that every part of the beam is analyzed. There are separate values in the output file for each slice, plus the whole-beam and slice-averaged results.

4.30 save_lattice

- type: action command.
- function: save the current accelerator element and beamline definitions.

```
&save_lattice
    STRING filename = NULL;
&end
```

- `filename` — The (incomplete) name of a file to which the element and beamline definitions will be written. Recommended value: "%s.new".

4.31 sdds_beam

- type: setup command.
- function: set up for tracking of particle coordinates stored in an SDDS file.

```
&sdds_beam
    STRING input = NULL;
    STRING iput_list = NULL;
    STRING input_type = "elegant";
    long n_particles_per_ring = 0;
    STRING selection_parameter = NULL;
    STRING selection_string = NULL;
    long one_random_bunch = 0;
    long reuse_bunch = 0;
    long prebunched = 0;
    long sample_interval = 1;
    long n_tables_to_skip = 0;
    long center_transversely = 0;
    long center_arrival_time = 0;
    double sample_fraction = 1;
    double p_lower = 0.0;
    double p_upper = 0.0;
    long save_initial_coordinates = 1;
    long reverse_t_sign = 0;
&end
```

- `input` — Name of an SDDS file containing coordinates of input particles.
- `input_type` — May be "elegant" or "spiffe", indicating the name of the program that wrote the input file. The expected data quantities for the different types are:
  - `elegant`: (x, xp, y, yp, t, p), where x and y are in meters, xp = x′ and xp = y′ are dimensionless, t is in seconds, and p = $\beta\gamma$ is the dimensionless momentum.
  - `spiffe`: (r, z, pr, pz, t), where r and z are in meters, pr = $\beta_r\gamma$, pz = $\beta_z\gamma$, and t is in seconds.
- `n_particles_per_ring` — For `spiffe` data, gives the number of particles to generate for each ring of charge.
- `selection_parameter` — The name of a parameter in the SDDS file to be used for selection of pages of data.
- `selection_string` — The value of the `selection_parameter` selection parameter required for a page to be used. E.g., if one has a file from the `shower` program containing positrons, electrons, and photons, one might want to select only the positrons.
- `one_random_bunch` — A flag indicating whether, for `spiffe` data, a new random distribution should be calculated for each step of the simulation.

- `prebunched` — A flag indicating, if zero, that the entire file is one "bunch," and otherwise that each page in the file is a different bunch.
- `sample_interval` — If non-zero, only every `sample_interval`$^{\text{th}}$ particle is used.
- `n_tables_to_skip` — Number of SDDS pages to skip at the beginning of the file.
- `center_transversely` — If non-zero, the transverse centroids of the distribution are made to be zero.
- `center_arrival_time` — If non-zero, the mean arrival time of particles at the start of the accelerator is set to zero.
- `sample_fraction` — If non-unity, the randomly selected fraction of the distribution to use.
- `p_lower`, `p_upper` — If different, the lower and upper limit on $\beta\gamma$ of particles to use.
- `save_initial_coordinates` — A flag that, if set, results in saving initial coordinates of tracked particles in memory. This is the default behavior. If unset, the initial coordinates are not saved, but are reread from disk each time they are needed. This is more memory efficient and is useful for tracking very large numbers of particles.

4.32 semaphores

- type: setup command.
- function: set up names for semaphore files, which are used to mark the start and end of program execution.

```
&semaphores
        STRING started = ``%s.started'';
        STRING done    = ``%s.done'';
&end
```

- `started` — Gives the (incomplete) filename of a file to create when a valid `run_setup` command is given.
- `done` — Gives the (incomplete) filename of a file to create when the program exits without error. If the file exists, it is deleted when a valid `run_setup` command is given.

4.33 slice_analysis

- type: setup command.
- function: set parameters for slice analysis of the beam along a beamline. Also, results in placing the final slice analysis (at the end of the beamline) in symbols for use in optimization equations. The names of the symbols are the same as the names of the columns in the output file.

```
&slice_analysis
STRING output = NULL;
long n_slices = 0;
double s_start = 0;
double s_end = 1e300;
long final_values_only = 0;
&end
```

- `output` — The (incomplete) filename of the output file. Recommended value is “
- `n_slices` — Number of slices to use.
- `s_start`, `s_end` — Position in beamline at which to start and stop performing slice analysis.
- `final_values_only` — If nonzero, then slice quantities are computed only at the end of the beamline.

4.34 subprocess

- type: action command.
- function: execute a system command in a shell.

```
&subprocess
    STRING command = NULL;
&end
```

- `command` — The text of the command to execute. The command may use the sequence “A literal “

4.35 steering_element

- type: setup command.
- function: setup for use of a given parameter of a given element as a steering corrector.
- N.B.: any use of this command disables the built-in definition of HKICK, VKICK, and HVKICK elements as steering elements. For trajectory correction, this facility works without any effort by the user. It will not work for orbit correction unless the user does two things: First, all correction elements for each plane must be the same. Second, the gain must be less than the ratio of the angle kick to unit parameter change for the element.

```
&steering_element
    STRING name = NULL;
    STRING element_type = NULL;
    STRING item = NULL;
    STRING plane = "h";
    double tweek = 1e-3;
    double limit = 0;
&end
```

- `name` — Optional: the (possibly wild-carded) name of the element to add to the steering list. If not given, then `element_type` must be given.
- `element_type` — Optional: the (possibly wild-carded) name of the element type to add to the steering list. If not given, then `name` must be given.
- `item` — The parameter of the element to be varied.
- `plane` — May be either "h" or "v", for horizontal or vertical correction.
- `tweek` — The amount by which to change the item to compute the steering strength.
- `limit` — The maximum allowed absolute value of the item.

4.36 transmute_elements

- type: setup command.
- function: Changes the type of selected elements, which may be used to turn off unneeded diagnostics and speed up tracking when concatenation is being used.
- notes:
  - Any number of these commands may be given.
  - Not effective unless given prior to `run_setup`.
  - The only property of the original element that is preserved is the length. For example, transmuting a SBEN into a CSBEN will not have the expected result.

```
&transmute_elements
STRING name = NULL,
STRING type = NULL,
STRING exclude = NULL,
        STRING new_type = "DRIF",
        long clear = 0;
&end
```

- `name` — Possibily wild-card containing string specifying the elements to which the transmutation specification is to be applied.
- `type` — Possibily wild-card containing string specifying the element types to which the transmutation specification is to be applied.
- `exclude` — Possibily wild-card containing string specifying elements to be excluded from the specified transmutation. Does not affect elements transmuted due to other specifications.
- `new_type` — Type into which specified elements will be transmuted.
- `clear` — If nonzero, all prior transmutation specifications are deleted.

4.37 twiss_analysis

- type: setup command.
- function: analyze Twiss parameters within a user-defined region for purposes of optimization.

```
&twiss_analysis
        STRING start_name = NULL;
        STRING end_name = NULL;
        double s_start = -1;
        double s_end = -1;
        STRING tag = NULL;
        long clear = 0;
&end
```

- `start_name` — Name of the element at which to start analysis. If the element occurs more than once, the first occurrence is used.
- `end_name` — Name of the element at which to end analysis. If the element occurs more than once, the first occurrence is used.
- `s_start` — Position (in meters) at which to start analysis.
- `s_end` — Position (in meters) at which to end analysis.
- `tag` — Name prefix for quantities computed by the analysis. The quantity names will have the form *tag.statistic.quantity*, where *statistic* is one of `min`, `max`, and `ave`, and *quantity* is one of `betax`, `betay`, `etax`, and `etay`. E.g., if *tag* is `region1`, then one could use expressions like `region1.max.betax` in optimization.
- `clear` — If nonzero, all previously defined analysis regions are deleted.

4.38 twiss_output

- type: action/setup command.
- function: compute and output Twiss parameters, or set up to do so.

```
&twiss_output
    STRING filename = NULL;
    long matched = 1;
    long output_at_each_step = 0;
    long output_before_tune_correction = 0;
    long final_values_only = 0;
    long statistics = 0;
    long radiation_integrals = 0;
    long concat_order = 3;
    double beta_x = 1;
    double alpha_x = 0;
    double eta_x = 0;
    double etap_x = 0;
    double beta_y = 1;
    double alpha_y = 0;
    double eta_y = 0;
    double etap_y = 0;
    STRING reference_file = NULL;
    STRING reference_element = NULL;
    long reference_element_occurrence = 0;
&end
```

- `filename` — The (incomplete) name of an SDDS file to which the Twiss parameters will be written. Recommended value: "%s.twi".
- `matched` — A flag indicating, if set, that the periodic or matched Twiss parameters should be found.
- `output_at_each_step` — A flag indicating, if set, that output is desired at each step of the simulation.
- `output_before_tune_correction` — A flag indicating, if set, that output is desired both before and after tune correction.
- `final_values_only` — A flag indicating, if set, that only the final values of the Twiss parameters should be output, and not the parameters as a function of s.
- `statistics` — A flag indicating, if set, that minimum, maximum, and average values of Twiss parameters should be computed and included in output.
- `radiation_integrals` — A flag indicating, if set, that radiation integrals should be computed and included in output.

- `beta_X`, `alpha_X`, `eta_X`, `etap_X` — If `matched` is zero, the initial values for the X plane.
- `concat_order` — Order of matrix concatenation to use for determining matrix for computation of Twiss parameters. Using a lower order will result in inaccuracy for nonlinear lattices with orbits. However, if the orbit is zero, it is much faster to use `concat_order=1`.
- `reference_file` — If given, the name of a file from which twiss parameter data will be taken to give the starting values. Ignored if `matched` is nonzero. The file should have the beta and alpha functions with the same names as the file created by this command.
- `reference_element` — Element in `reference_file` at which to take the twiss parameter values. If not given, the values at the last element in `reference_file` are used.
- `reference_element_occurrence` — Ignored if `reference_element` is not given. Otherwise, the occurence number of `reference_element` to use. If 0, the last occurence is used.

4.39 track

- type: action command.
- function: track particles.

```
&track
    long center_on_orbit = 0;
    long center_momentum_also = 1;
    long soft_failure = 1;
    long use_linear_chromatic_matrix = 0;
    long longitudinal_ring_only = 0;
&end
```

- `center_on_orbit` — A flag indicating whether to center the beam transverse coordinates on the closed orbit before tracking.
- `center_momentum_also` — A flag indicating whether to center the momentum coordinate also.
- `soft_failure` — If there is an error during tracking (e.g., a failure of orbit correction), continue to produce file output. This creates essentially empty slots in the files corresponding to the failed steps.
- `use_linear_chromatic_matrix` — For each particle, a first-order matrix is computed for the particular momentum offset of the particle using the linear chromaticity and linear dependence of the beta functions on momentum.
- `longitudinal_ring_only` — Tracks longitudinal coordinates only for a ring.

4.40 tune_shift_with_amplitude

- type: setup command.
- function: prepare for computation of tune shifts with amplitude.
- note: must be given prior to the `twiss_output` command.
- method: tune shifts with amplitude are computed via tracking a series of particles at different amplitudes. NAFF is used to determine the tunes from the tracking data. It is the user's responsbility to optimize the parameters to ensure that results are reasonable. Using tracking to determine tune shifts is more accurate than analytical methods as it includes multi-turn effects that are important in some rings (e.g., the APS).

```
&tune_shift_with_amplitude
    long turns = 1000;
    double x0 = 1e-10;
    double y0 = 1e-10;
    double x1 = 1e-6;
    double y1 = 1e-6;
    long use_concatenation = 1;
&end
```

- `turns` — The number of turns to track. If zero, then the concatenated matrix is used instead of tracking. This doesn't work well with all lattices. The order of the concatenated matrix is given by the `concat_order` control in `twiss_output`.
- `x0`, `y0` — The initial x and y amplitudes to use for determining the small-amplitude tunes.
- `x1`, `y1` — The initial x and y amplitudes to user for determining the tune shifts. These values should be small enough to ensure linearity in the tune shift. If the difference in tune is greater than 0.01, `elegant` will automatically reduce these initial values and iterate.
- `use_concatenation` — If nonzero, then tracks with the concatenated matrix instead of element-by-element. The order of the concatenated matrix is given by the `concat_order` control in `twiss_output`. The user should experiment with this option to see if the results are reliable for a particular lattice.

4.41 vary_element

- type: setup command.
- function: define an index and/or tie a parameter of an element to it.

```
&vary_element
    long index_number = 0;
    long index_limit = 0;
    STRING name = NULL;
    STRING item = NULL;
    double initial = 0;
    double final = 0;
    long differential = 0;
    long multiplicative = 0;
    long geometric = 0;
    STRING enumeration_file = NULL;
    STRING enumeration_column = NULL;
&end
```

- `index_number` — A non-negative integer giving the number of the index.
- `index_limit` — A positive integer giving the number of values the index will take. Must be given if this `index_number` has not been listed in a previous `vary_element` command, unless `enumeration_file` is given.
- `name` — The name of an element.
- `item` — The parameter of the element to vary.
- `initial`, `final` — The initial and final values of the parameter.
- `enumeration_file` — Name of an SDDS file giving values for the item.
- `enumeration_column` — Column of the SDDS file giving the values.
- `differential` — If nonzero, the initial and final values are taken as offsets from the predefined value of the parameter.
- `multiplicative` — If nonzero, the initial and final values are taken as multipliers to be applied to the predefined value of the parameter in order to obtain the actual initial and final values.
- `geometric` — If nonzero, then variation is geometric rather than arithmetic.

5 Specialized Tools for Use with `elegant`

A number of specialized programs are available that work with `elegant`. Most are SDDS-compliant, so they will also work with any program that reads or writes appropriate SDDS data. These programs will be made available in Version 14.3Beta. The following is a brief description of each program. Full descriptions are available on subsequent pages.

- `elegant2genesis` — This program performs slice analysis of particle output files, which are suitable for use with the SDDS-compliant APS version of GENESIS[14]. This program is part of the SDDS toolkit. See the SDDS toolkit manual for documentation.

- `haissinski` — Computes the steady-state longitudinal distribution in an electron storage ring. Requires as input a file containing the Twiss parameters around the ring, such as that provided by the `twiss_output` command. (Program by L. Emery)

- `ibsEmittance` — Computes the transverse and longitudinal emittances of a beam in an electron storage ring, resulting from the combination of quantum excitation, damping, and intra-beam scattering. Requires as input a file containing the Twiss parameters, such as that provided by the `twiss_output` command. (Program by L. Emery)

- `madto` — Translates an `elegant`-style lattice file (or a MAD file, with some restrictions) into formats accepted by other programs, such as PARMELA, PATPET, PATRICIA, TRANSPORT, and XORBIT. Will also generate an SDDS file containing lattice data.

- `sddsanalyzebeam` — Analyzes a beam of macro-particles and produces an SDDS file containing beam moments, emittances, equivalent beta functions, etc. The beam file is of the type written by `elegant` using the `output` field of the `run_setup` command, or the WATCH element.

- `sddsemitmeas` — Analyzes quadrupole scan emittance measurement data. Accepts a file containing the transport matrix for each point and measured beam sizes. The file may, for example, be the file produced by the `final` field of the `run_setup` command. The quadrupole scan can be executed inside of `elegant` using `vary_elements`.

- `sddsmatchtwiss` — Transforms a beam of macro-particles to match to given beta functions and dispersion. The beam file is of the type written by `elegant` using the `output` field of the `run_setup` command, or the WATCH element.

- `sddssampledist` — This program allows creating particle distributions from user-designed distribution functions. It is thus a more flexible alternative to `bunched_beam`. This program is part of the SDDS toolkit. See the SDDS toolkit manual for documentation.

5.1 haissinski

- **description:**
- **examples:**
- **synopsis:**
- **files:**
- **switches:**
- **author:** L. Emery, ANL/APS.

5.2 ibsEmittance

- **description:**
- **examples:**
- **synopsis:**
- **files:**
- **switches:**
- **author:** L. Emery, ANL/APS.

5.3 madto

- **description:** `madto` translates an `elegant`-style (or a MAD file, with some restrictions) into formats accepted by other programs, such as PARMELA, PATPET, PATRICIA, TRANSPORT, and XORBIT. Will also generate an SDDS file containing lattice data.

- **examples:** The following command would translate the `elegant` lattice file `lattice.lte` into a TRANSPORT lattice file with 10mm quadrupole aperture and 5mm sextupole aperture, at an energy of 1.5 GeV.

```
madto lattice.lte lattice.trin -transport=10,5,1.5
```

- **synopsis:**

`madto` *inputfile outputfile* {`-patricia | -patpet |`
`-transport[=`*quadAper(mm)*`,`*sextAper(mm)*`,`*p(GeV/c)*`] |`
`-parmela[=`*quadAper(mm)*`,`*sextAper(mm)*`,`*p(GeV/c)*`] | -sdds[=`*p(GeV/c)*`] | -xorbit}`
`[-angle_tolerance=`*value*`] [-flip_k_signs] [-magnets=`*filename*`] [-header=`*filename*`]`
`[-ender=`*filename*`]`

- **files:**
  - *inputfile* — An `elegant`-style lattice file.
  - *outputfile* — A file containing lattice data in the chosen format.

- **switches:**
  - **-patricia** — Provide data for the program PATRICIA.
  - **-patpet** — Provide data for the program PATPET, a merging of the programs PATRICIA and PETROS.
  - **-transport**[=*quadAper(mm)*,*sextAper(mm)*,*p(GeV/c)*] — Provide data for the program TRANSPORT (original style). One may give apertures for the quadrupoles and sextupoles, as well as the beam momentum in GeV/c.
  - **-parmela**[=*quadAper(mm)*,*sextAper(mm)*,*p(GeV/c)*] — Provide data for the program PARMELA. One may give apertures for the quadrupoles and sextupoles, as well as the beam momentum in GeV/c.
  - **-sdds**[=*p(GeV/c)*] — Provide data in SDDS form. One may give the beam momentum in GeV/c.
  - **-angle_tolerance=***value* — PATPET and PATRICIA only allow sector and rectangular bends. This tolerance, in radians, determines how far from sector or rectangular a bend definition may be and still get processed.
  - **-flip_k_signs** — Changes the signs of all quadrupoles.
  - **-magnets=***filename* — Results in output of an additional SDDS file with the magnet layout. This is the same file that would be generated by the `magnets` field of the `run_setup` command in `elegant`.

  - **-header=***filename*, **-ender=***filename* — Allow specification of files to be prepended and appended to the lattice output. For example, if additional commands are required prior to the lattice definition to set up the run, they would be put in the **header** file. If additional commands are needed after the lattice definition to initiate processing, they would be put in the `ender` file.

- **author:** M. Borland, ANL/APS.

5.4 sddsanalyzebeam

- **description:** `sddsanalyzebeam` analyzes a beam of macro-particles and produces an SDDS file containing beam moments, emittances, equivalent beta functions, etc. The beam file is of the type written by `elegant` using the `output` field of the `run_setup` command, or the WATCH element.

- **examples:**

```
sddsanalyzebeam run.out run.analysis
```

- **synopsis:**

```
sddsanalyzebeam [-pipe=[input][,output]] [inputfile] [outputfile] [-nowarnings]
```

- **files:**

  - *inputfile* — An SDDS file containing the columns `x`, `xp`, `y`, `yp`, `t`, and `p`, giving the six phase-space coordinates for a set of macroparticles. This file can be produced from `elegant`, for example, using the `output` field of the `run_setup` command, the `bunch` field of the `bunched_beam` command, or the WATCH element in coordinate mode.
  - *outputfile* — An SDDS file containing columns giving moments, emittances, equivalent Twiss parameters, and so on, for the macro-particles. Each row of this file corresponds to a page of the input file. The names and meanings of the columns are identical to what is used for `elegant`'s `final` output file from the `run_setup` command. The file from `elegant`, however, stores the results as parameters instead of columns; to convert `outputfile` to that convention, use the SDDS toolkit program `sddsexpand`.

- **switches:**

  - `pipe` — The standard SDDS Toolkit pipe option.
  - `nowarnings` — Suppressses warning messages.

- **author:** M. Borland, ANL/APS.

5.5 sddsemitmeas

- **description:**

  `sddsemitmeas` analyzes quadrupole scan emittance measurement data. It accepts a file containing the transport matrix for each data point and measured beam sizes. Because `sddsemitmeas` uses the matrix rather than a thin-lens model, it can analyze data from arbitrarily complex scans, involving, for example, multiple thick-lens quadrupoles.

  The matrix data can be prepared using `elegant`. For example, the `vary_element` command can be used to vary one or more quadrupoles. In addition, the beam size data may be prepared using `elegant`, to allow simulation of emittance measurements.

  `sddsemitmeas` will perform error analysis using a Monte Carlo technique. A user-specified number of random error sets are generated and added to all measurements. Analysis is performed for each error set. Statistics over all the error sets provide most likely values and error bars.

- **examples:**

```
elegant quadScan.ele sddscollapse quadScan.fin -pipe=out
| sddsxref -pipe=in quadScan.data -take=SigmaX,SigmaY
| sddsemitmeas -pipe=in emitResults.sdds
```

- **synopsis:**

```
sddsemitmeas [inputfile] [outputfile] [-pipe=[input][,output]]
[-sigmaData=xName,yName] [-energySpread=fractionalRMSValue]
[-errorLevel=valueInm,[{gaussian,nSigmas | uniform}]] [-nErrorSets=number]
[-seed=integer] [-limitMode=resolution | zero[,reject]
[-deviationLimit=xLevelm,yLevelm] [-resolution=xResolutionm,yResolutionm]
[-uncertaintyFraction=xValue,yValue] [-fixedUncertainty=xValuem,yValuem]
[-findUncertainties] [-minimumUncertainty=xValuem,yValuem]
[-constantWeighting] [-verbosity=level]
```

- **files:**

  - *inputfile* — An SDDS file containing one or more pages with columns named R*ij*, where *ij* is 11, 12, 33, and 34. These give elements of the horizontal and vertical transport matrices from the beginning of a system to the observation point. The sigma matrix inferred will be that for the beginning of the system. Typically, one starts with the `final` file from the `run_setup` command in `elegant`, and collapses it using `sddscollapse`. Each page of *inputfile* corresponds to a different emittance measurement.

    If energy spread is included (`-energySpread` option), the file must also contain columns named `etax` and `etay`, giving the horizontal and vertical dispersion at the observation point. These may be added to the `final` file using `sddsprocess` (to get the final values as parameters) and `sddsxref` (to transfer the parameters), prior to using `sddscollapse`.

    In addition to this data, *inputfile* must also contain columns giving the rms beam sizes in x and y. The user supplies the names of the columns using the `-sigmaData` option.

These columns may be from `elegant` (e.g., `Sx` and `Sy`), if one wants to simulate an emittance measurement. Note that the theory behind the emittance measurement is strictly correct only for true RMS beamsize measurements. Use of FWHM or some other measure will give unreliable results.

– *outputfile* — A file containing one page for each page of *inputfile*. The parameters of *outputfile* give the measured geometric rms emittance, sigma matrix, and Twiss parameters of the beam in the horizontal and vertical planes. If error sets were requested (using `-nErrorSets`), then there are also parameters giving the error bars ("sigma's") of the measured values.

The columns of *outputfile* contain various quantities depending on the mode. For files generated with `elegant` and no error sets, it contains the measured and fit beam size data, along with the strength of one of the varied quadrupoles. In other cases, less data may be present.

- **switches:**

  – `-sigmaData=`*xName*`,`*yName* — Supplies the names of the columns in *inputfile* from which the x and y rms beam sizes are to be taken. Default values are `Sx` and `Sy`, which are the data provided by `elegant`.

  – `-energySpread=`*fractionalRMSValue* — Supplies the fractional rms energy spread of the beam. If given, the *inputfile* must contain dispersion data, as described above.

  – `-errorLevel=`*valueInm*`,[gaussian,`*nSigmas* `| uniform]` — Supplies the standard deviation of random errors to be added to the measured beam sizes for Monte Carlo error analysis. This control should not be confused with the controls over the individual data point uncertainties; the latter are used for purposes of weighting the fits only. Supposedly, the error levels and the uncertainties would be the same, however.

  – `-nErrorSets=`*number* — The number of sets of random errors to generate and add to the measurements. Each error set is used to perturb the original measurement data. The results are analyzed separately for each error set, then combined to give means and error bars.

  – `-seed=integer` — Seed for the random number generator. Recommend a large, positive, odd integer less than 2^31. If no seed is given or if the given seed is negative, then a seed is generated from the system clock.

  – `-resolution=`*xResolutionm*`,`*yResolutionm* — The resolution of the beam size measurements, in meters. These values are subtracted in quadrature from the measured beam sizes to obtain the true beam sizes.

  – `-limitMode=resolution | zero[,reject]` — If measured or perturbed beam sizes are less than the resolution or less than zero, then errors will result. One can use this option to limit minimum beam size values or reject points. In general, if one has to do this the measurement is probably bad.

  – `-deviationLimit=`*xLevelm*`,`*yLevelm* — Specifies the maximum deviation, in meters, from the fit that data points may have and still be included. An initial fit is performed for each randomized set or the raw data, as appropriate. Outliers are then removed and the fit is repeated.

  – `-uncertaintyFraction=`*xValue*`,`*yValue* — Specifies that uncertainties in individual measurements should determined as the given fractions of the measured values. Generally not realistic.

- `-fixedUncertainty=`*xValuem*`,`*yValuem* — Specifies that uncertainties in individual measurements should be identical, at the values given (in meters).
- `-findUncertainties` — Specifies that uncertainties in individual measurements should be deduced from an initial unweighted fit.
- `-minimumUncertainty=`*xValuem*`,`*yValuem* — Specifies that uncertainties may not be smaller than the given values, in meters.
- `-constantWeighting` — Specifies uncertainties such that all points are given equal weight in the fit.
- `-verbosity=`*level* — Higher values of *level* result in more informational printouts as the program runs.

- **author:** M. Borland, ANL/APS.

5.6 sddsmatchtwiss

- **description:** `sddsmatchtwiss` transforms a beam of macro-particles to match to given beta functions and dispersion. This can be useful in taking macro-particle data from one simulation and using it in another. For example, a beam file from PARMELA could be given the right beta functions for use with a specific lattice in an `elegant` run, saving the trouble of rematching to join the two simulations. Similarly, a beam from `elegant` could be matched into an FEL simulation.

- **examples:**

```
sddsmatchtwiss elegantBeam.out FELBeam.in -xPlane=beta=1.0,alpha=-0.2
-yPlane=beta=0.5,alpha=0.2
```

- **synopsis:**

`sddsmatchtwiss [-pipe=[input][,output]]` *inputfile outputfile*
`[-xPlane=[beta=`*meters*`,alpha=`*value*`][,etaValue=`*meters*`][,etaSlope=`*value*`]]`
`[-yPlane=[beta=`*meters*`,alpha=`*value*`][,etaValue=`*meters*`][,etaSlope=`*value*`]]`
`[-nowarnings]`

- **files:**

inputfile is an SDDS file containing one or more pages of data giving the phase-space coordinates of macro particles. The macro particle data is stored in columns named `x`, `xp`, `y`, `yp`, and `p`. The units are those used by `elegant` for the `output` file from `run_setup`, the `bunch` file from `bunched_beam`, and the coordinate-mode output from the `WATCH` element. The data from these columns is used together with the commandline arguments to produce new values for these columns; the new values are delivered to `outputfile`. Other columns may be present in `inputfile`; if so, they are passed to *outputfile* unchanged.

- **switches:**

  - `-xPlane=[beta=`*meters*`,alpha=`*value*`][,etaValue=`*meters*`][,etaSlope=`*value*`]` — Specifies the desired parameters for the beam in the horizontal plane. `beta` and `alpha` give β and $\alpha = -\frac{1}{2}\frac{\partial\beta}{\partial s}$; they must both be given or both be omitted. `etaValue` and `etaSlope` give the dispersion, η, and its slope, $\frac{\partial\eta}{\partial s}$.
  - `-yPlane=[beta=`*meters*`,alpha=`*value*`][,etaValue=`*meters*`][,etaSlope=`*value*`]` — Same as `-xPlane`, except for the vertical plane.
  - `-nowarnings` — Suppresses warning messages.

- **author:** M. Borland, ANL/APS.

6 Accelerator and Element Description

As mentioned in the introduction, `elegant` uses a variant of the MAD input format for describing accelerators. With some exceptions, the accelerator description for one program can be read by the other with no modification. Among the differences:

- `elegant` does not support the use of MAD-style equations to compute the value of a quantity. The `link_element` namelist command can be used for this purpose, and is actually more flexible than the method used by MAD. Also, `rpn`-style equations may be given in double-quotes; these are evaluated once only when the lattice is parsed.
- `elegant` does not support substitution of parameters in beamline definitions.
- `elegant` contains many elements that MAD does not have, such as kick elements, wake fields, and numerically integrated elements.
- The length of an input line is not limited to 80 characters in `elegant`, as it is in MAD. However, for compatibility, any lattice created by `elegant` will conform to this limit.

`elegant`'s `print_dictionary` command allows the user to obtain a list of names and short descriptions of all accelerator elements recognized by the program, along with the names, units, types, and default values of all parameters of each element. The present output of this command is listed in the next section. The reader is referred to the MAD manual[2] for details on sign conventions for angles, focusing strength, and so forth.

7 Element Dictionary

7.1 ALPH

An alpha magnet implemented as a matrix, up to 3rd order. PART is used to split the magnet into halves. XS<n> and DP<n> allow momentum filtration at the midpoint.

Parameter Name	Units	Type	Default	Description
XMAX	*M*	double	0.0	size of alpha
XS1	*M*	double	0.0	inner scraper position
XS2	*M*	double	0.0	outer scraper position
DP1		double	-1	inner scraper momentum deviation
DP2		double	1	outer scraper momentum deviation
XPUCK	*M*	double	-1	position of scraper puck
WIDTHPUCK	*M*	double	0.0	size of scraper puck
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
TILT		double	0.0	rotation about incoming longitudinal axis
PART		long	0	0=full, 1=first half, 2=second half
ORDER		long	0	matrix order [1,3]

7.2 BMAPXY

A map of Bx and By vs x and y.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
STRENGTH	$NULL$	double	0.0	factor by which to multiply field
ACCURACY	$NULL$	double	0.0	integration accuracy
METHOD	$NULL$	STRING	NULL	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta
FILENAME	$NULL$	STRING	NULL	name of file containing columns (x, y, Fx, Fy) giving normalized field (Fx, Fy) vs (x, y)

7.3 BUMPER

A time-dependent uniform-field rectangular kicker magnet with no fringe effects. The waveform is in mpl format, with time in seconds and amplitude normalized to 1.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
ANGLE	*RAD*	double	0.0	kick angle
TILT	*RAD*	double	0.0	rotation about longitudinal axis
TIME_OFFSET	*S*	double	0.0	time offset of waveform
PERIODIC		long	0	is waveform periodic?
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
FIRE_ON_PASS		long	0	pass number to fire on
WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving kick factor vs time
SPATIAL_DEPENDENCE		STRING	NULL	rpn function of x and y giving spatial dependence of kick

7.4 CENTER

An element that centers the beam transversely on the ideal trajectory.

Parameter Name	Units	Type	Default	Description
X		long	1	center x coordinates?
XP		long	1	center x' coordinates?
Y		long	1	center y coordinates?
YP		long	1	center y' coordinates?
ONCE_ONLY		long	0	compute centering offsets for first beam only, apply to all?

7.5 CEPL

A numerically-integrated linearly-ramped electric field deflector.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
RAMP_TIME	*S*	double	1e-09	time to ramp to full strenth
TIME_OFFSET	*S*	double	0.0	offset of ramp-start time
VOLTAGE	*V*	double	0.0	maximum voltage between plates due to ramp
GAP	*M*	double	0.01	gap between plates
STATIC_VOLTAGE	*V*	double	0.0	static component of voltage
TILT	*RAD*	double	0.0	rotation about longitudinal axis
ACCURACY		double	0.0001	integration accuracy
X_MAX	*M*	double	0.0	x half-aperture
Y_MAX	*M*	double	0.0	y half-aperture
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of steps (for nonadaptive integration)
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, nonadaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")

7.6 CHARGE

An element to establish the total charge of a beam. Active on first pass only. If given, overrides all charge specifications on other elements.

Parameter Name	Units	Type	Default	Description
TOTAL	C	double	0.0	total charge in beam
PER_PARTICLE	C	double	0.0	charge per macroparticle

7.7 CLEAN

Cleans the beam by removing outlier particles.

Parameter Name	Units	Type	Default	Description
MODE		STRING	stdeviation	stdeviation, absdeviation, or absvalue
XLIMIT		double	0.0	Limit for x
XPLIMIT		double	0.0	Limit for x'
YLIMIT		double	0.0	Limit for y
YPLIMIT		double	0.0	Limit for y'
TLIMIT		double	0.0	Limit for t
DELTALIMIT		double	0.0	Limit for (p-p0)/p0

7.8 CSBEND

A canonical kick sector dipole magnet.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric quadrupole strength
K2	$1/M^3$	double	0.0	geometric sextupole strength
K3	$1/M^3$	double	0.0	geometric octupole strength
K4	$1/M^4$	double	0.0	geometric decapole strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
H1	$1/M$	double	0.0	entrance pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
ETILT		double	0.0	error rotation about incoming longitudinal axis
N_KICKS		long	4	number of kicks
NONLINEAR		long	1	include nonlinear field components?
SYNCH_RAD		long	0	include classical synchrotron radiation?
EDGE1_EFFECTS		long	1	include entrace edge effects?
EDGE2_EFFECTS		long	1	include exit edge effects?
EDGE_ORDER		long	1	order to which to include edge effects
INTEGRATION_ORDER		long	2	integration order (2 or 4)
EDGE1_KICK_LIMIT		double	-1	maximum kick entrance edge can deliver

CSBEND continued

A canonical kick sector dipole magnet.

Parameter Name	Units	Type	Default	Description
EDGE2_KICK_LIMIT		double	-1	maximum kick exit edge can deliver
KICK_LIMIT_SCALING		long	0	scale maximum edge kick with FSE?
USE_BN		long	0	use B<n> instead of K<n>?
B1	$1/M$	double	0.0	K1 = B1*rho, where rho is bend radius
B2	$1/M^2$	double	0.0	K2 = B2*rho
B3	$1/M^3$	double	0.0	K3 = B3*rho
B4	$1/M^4$	double	0.0	K4 = B4*rho
ISR		long	0	include incoherent synchrotron radiation (scattering)?

7.9 CSRCSBEND

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric quadrupole strength
K2	$1/M^3$	double	0.0	geometric sextupole strength
K3	$1/M^3$	double	0.0	geometric octupole strength
K4	$1/M^4$	double	0.0	geometric decapole strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
H1	$1/M$	double	0.0	entrance pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
ETILT		double	0.0	error rotation about incoming longitudinal axis
N_KICKS		long	4	number of kicks
NONLINEAR		long	1	include nonlinear field components?
LINEARIZE		long	0	use linear matrix instead of symplectic integrator?
SYNCH_RAD		long	0	include classical synchrotron radiation?
EDGE1_EFFECTS		long	1	include entrace edge effects?
EDGE2_EFFECTS		long	1	include exit edge effects?
EDGE_ORDER		long	1	order to which to include edge effects
INTEGRATION_ORDER		long	2	integration order (2 or 4)

CSRCSBEND continued

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Parameter Name	Units	Type	Default	Description
BINS		long	0	number of bins for CSR wake
BIN_ONCE		long	0	bin only at the start of the dipole?
BIN_RANGE_FACTOR		double	1.2	Factor by which to increase the range of histogram compared to total bunch length. Large value eliminates binning problems in CSRDRIFTs.
SG_HALFWIDTH		long	0	Savitzky-Golay filter half-width for smoothing current histogram
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing current histogram
SGDERIV_HALFWIDTH		long	0	Savitzky-Golay filter half-width for taking derivative of current histogram
SGDERIV_ORDER		long	1	Savitzky-Golay filter order for taking derivative of current histogram
TRAPAZOID_INTEGRATION		long	1	Select whether to use trapazoid-rule integration (default) or a simple sum.
OUTPUT_FILE		STRING	NULL	output file for CSR wakes
OUTPUT_INTERVAL		long	1	interval (in kicks) of output to OUTPUT_FILE
OUTPUT_LAST_WAKE_ONLY		long	0	output final wake only?
STEADY_STATE		long	0	use steady-state wake equations?
USE_BN		long	0	use B<n> instead of K<n>?
B1	$1/M$	double	0.0	K1 = B1*rho, where rho is bend radius
B2	$1/M^2$	double	0.0	K2 = B2*rho
B3	$1/M^3$	double	0.0	K3 = B3*rho
B4	$1/M^4$	double	0.0	K4 = B4*rho
ISR		long	0	include incoherent synchrotron radiation (scattering)?

CSRCSBEND continued

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.

Parameter Name	Units	Type	Default	Description
CSR		long	1	enable CSR computations?
BLOCK_CSR		long	0	block CSR from entering CSRDRIFT?
DERBENEV_CRITERION_MODE		STRING	disable	disable, evaluate, or enforce
PARTICLE_OUTPUT_FILE		STRING	NULL	name of file for phase-space output
PARTICLE_OUTPUT_INTERVAL		long	0	interval (in kicks) of output to PARTICLE_OUTPUT_FILE
SLICE_ANALYSIS_INTERVAL		long	0	interval (in kicks) of output to slice analysis file (from slice_analysis command)
HIGH_FREQUENCY_CUTOFF0		double	-1	Spatial frequency at which smoothing filter begins. If not positive, no frequency filter smoothing is done. Frequency is in units of Nyquist (0.5/binsize).
HIGH_FREQUENCY_CUTOFF1		double	-1	Spatial frequency at which smoothing filter is 0. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.

For a discussion of the method behind this element, see M. Borland, "Simple method for particle tracking with coherent synchrotron radiation," Phys. Rev. ST Accel. Beams 4, 070701 (2001).

Recommendations for using this element. The default values for this element are not the best ones to use. They are retained only for consistency through upgrades. In using this element, it is recommended to have 50 to 100 k particle in the simulation. Setting `BINS=600` and `SG_HALFWIDTH=1` is also recommended to allow resolution of fine structure in the beam and to avoid excessive smoothing. It is strongly suggested that the user vary these parameters and view the histogram output to verify that the longitudinal distribution is well represented by the histograms (use `OUTPUT_FILE` to obtain the histograms). For LCLS simulations, we find that the above parameters give essentially the same results as obtained with 500 k particles and up to 3000 bins.

In order to verify that the 1D approximation is valid, the user should also set `DERBENEV_CRITERION_MODE = ''evaluate''` and view the data in `OUTPUT_FILE`. Generally, the criterion should be much less than 1.

7.10 CSRDRIFT

A follow-on element for CSRCSBEND that applies the CSR wake over a drift.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
ATTENUATION_LENGTH	*M*	double	0.0	exponential attenuation length for wake
DZ		double	0.0	interval between kicks
N_KICKS		long	1	number of kicks (if DZ is zero)
SPREAD		long	0	use spreading function?
USE_OVERTAKING_LENGTH		long	0	use overtaking length for ATTENUATION_LENGTH?
OL_MULTIPLIER		double	1	factor by which to multiply the overtaking length to get the attenuation length
USE_SALDIN54		long	0	Use Saldin et al eq. 54 (NIM A 398 (1997) 373-394 for decay vs z?
SALDIN54POINTS		long	1000	Number of values of position inside bunch to average for Saldin eq 54.
CSR		long	1	do CSR calcuations
SALDIN54NORM_MODE		STRING	peak	peak or first
SPREAD_MODE		STRING	full	full, simple, or radiation-only
WAVELENGTH_MODE		STRING	sigmaz	sigmaz or peak-to-peak
BUNCHLENGTH_MODE		STRING	68-percentile	rms, 68-percentile, or 90-percentile
SALDIN54_OUTPUT		STRING	NULL	Filename for output of CSR intensity vs. z as computed using Saldin eq 54.
USE_STUPAKOV		long	0	Use treatment from G. Stupakov's note of 9/12/2001?
STUPAKOV_OUTPUT		STRING	NULL	Filename for output of CSR wake vs. s as computed using Stupakov's equations.
STUPAKOV_OUTPUT_INTERVAL		long	1	Interval (in kicks) between output of Stupakov wakes.
SLICE_ANALYSIS_INTERVAL		long	0	interval (in kicks) of output to slice analysis file (from slice_analysis command)

CSRDRIFT continued

A follow-on element for CSRCSBEND that applies the CSR wake over a drift.

Parameter Name	Units	Type	Default	Description
LINEARIZE		long	0	use linear optics for drift pieces?

This element has a number of models for simulation of CSR in drift spaces following CSRCSBEND elements. Note that all models allow support splitting the drift into multiple CSRDRIFT elements. One can also have intervening elements like quadrupoles, as often happens in chicanes. The CSR effects inside such intervening elements are applied in the CSRDRIFT downstream of the element.

For a discussion of some of the methods behind this element, see M. Borland, "Simple method for particle tracking with coherent synchrotron radiation," Phys. Rev. ST Accel. Beams 4, 070701 (2001).

N.B.: by default, this element uses 1 CSR kick (N_KICKS=1) at the center of the drift. This is usually not a good choice. I usually use the DZ parameter instead of N_KICKS, and set it to something like 0.01 (meters). The user should vary this parameter to assess how small it needs to be.

The models are as following, in order of decreasing sophistication and accuracy:

- G. Stupakov's extension of Saldin et al. Set USE_STUPAKOV=1. The most advanced model at present is based on a private communication from G. Stupakov (SLAC), which extends equation 87 of the one-dimensional treatment of Saldin et al. (NIM A 398 (1997) 373-394) to include the post-dipole region. This model includes not only the attenuation of the CSR as one proceeds along the drift, but also the change in the shape of the "wake."

  This model has the most sophisticated treatment for intervening elements of any of the models. For example, if you have a sequence `CSRCSBEND-CSRDRIFT-CSRDRIFT` and compare it with the sequence `CSRCSBEND-CSRDRIFT-DRIFT -CSRDRIFT`, keeping the total drift length constant, you'll find no change in the CSR-induced energy modulation. The model back-propagates to the beginning of the intervening element and performs the CSR computations starting from there.

  This is the slowest model to run. It uses the same binning and smoothing parameters as the upstream CSRCSBEND. If run time is a problem, compare it to the other models and use only if you get different answers.

- M. Borland's model based on Saldin et al. equations 53 and 54. Set USE_SALDIN54=1. This model computes the fall-off of the CSR wake from the work of Saldin and coworkers, as described in the reference above. It does not compute the change in the shape of the wake. The fall-off is computed approximately as well, based on the fall-off for a rectangular current distribution. The length of this rectangular bunch is taken to be twice the bunch length computed according to the BUNCHLENGTH_MODE parameter (see below). If your bunch is nearly rectangular, then you probably want BUNCHLENGTH_MODE of "90-percentile".

- Exponential attenuation of a CSR wake with unchanging shape. There are two options here. First, you can provide the attenuation length yourself, using the ATTENUATION_LENGTH parameter. Second, you can set USE_OVERTAKING_LENGTH=1 and let `elegant` compute

the overtaking length for use as the attenuation length. In addition, you can multiply this result by a factor if you wish, using the OL_MULTIPLIER parameter.

- Beam-spreading model. This model is not recommended. It is based on the seemingly plausible idea that CSR spreads out just like any synchrotron radiation, thus decreasing the intensity. The model doesn't reproduce experiments.

The "Saldin 54" and "overtaking-length" models rely on computation of the bunch length, which is controlled with the BUNCHLENGTH_MODE parameter. Nominally, one should use the true RMS, but when the beam has temporal spikes, it isn't always clear that this is the best choice. The choices are "rms", "68-percentile", and "90-percentile". The last two imply using half the length determined from the given percentile in place of the rms bunch length. I usually use 68-percentile, which is the default.

7.11 DRIF

A drift space implemented as a matrix, up to 2nd order

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
ORDER		long	0	matrix order

7.12 ECOL

An elliptical collimator.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
X_MAX	M	double	0.0	half-axis in x
Y_MAX	M	double	0.0	half-axis in y
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment

7.13 ELSE

Not implemented.

Parameter Name	Units	Type	Default	Description

7.14 ENERGY

An element that matches the central momentum to the beam momentum, or changes the central momentum or energy to a specified value.

Parameter Name	Units	Type	Default	Description
CENTRAL_ENERGY	MC^2	double	0.0	desired central gamma
CENTRAL_MOMENTUM	MC	double	0.0	desired central beta*gamma
MATCH_BEAMLINE		long	0	if nonzero, beamline reference momentum is set to beam average momentum
MATCH_PARTICLES		long	0	if nonzero, beam average momentum is set to beamline reference momentum

7.15 FMULT

Multipole kick element with coefficient input from an SDDS file.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
TILT	*RAD*	double	0.0	rotation about longitudinal axis
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
FSE		double	0.0	fractional strength error
N_KICKS		long	1	number of kicks
SYNCH_RAD		long	0	include classical synchrotron radiation?
FILENAME		STRING	NULL	name of file containing multipole data

This element simulates a multipole element using a 4th-order sympletic integration. Specification of the multipole strength is through an SDDS file. The file is expected to contain a single page of data with the following elements:

1. An integer column named `order` giving the order of the multipole. The order is defined as $(N_{poles} - 2)/2$, so a quadrupole has order 1, a sextupole has order 2, and so on.

2. A floating point column named `KnL` giving the integrated strength of the multipole, $K_n L$, where n is the order. The units are $1/m^(n-1)$.

To specify skew multipoles, you must use a separate `FMULT` element and set the `TILT` parameter. This is not very convenient and should be changed in the future. The `MULT` element is also available, which allows the same functionality without an external file.

7.16 HISTOGRAM

Request for histograms of particle coordinates to be output to SDDS file.

Parameter Name	Units	Type	Default	Description
FILENAME		STRING		filename for histogram output
INTERVAL		long	1	interval in passes between output
START_PASS		long	0	starting pass for output
BINS		long	50	number of bins
FIXED_BIN_SIZE		long	0	if nonzero, bin size is fixed at given value
X_DATA		long	1	histogram x and x'?
Y_DATA		long	1	histogram y and y'?
LONGIT_DATA		long	1	histogram t and p?
BIN_SIZE_FACTOR		double	1	multiply computed bin size by this factor before histogramming

7.17 HKICK

A horizontal steering dipole implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KICK	RAD	double	0.0	kick strength
TILT	RAD	double	0.0	rotation about longitudinal axis
B2	$1/M^2$	double	0.0	normalized sextupole strength (kick = KICK*(1+B2*x$\hat{2}$))
CALIBRATION		double	1	strength multiplier
EDGE_EFFECTS		long	0	include edge effects?
ORDER		long	0	matrix order
STEERING		long	1	use for steering?

7.18 HMON

A horizontal position monitor, accepting a rpn equation for the readout as a function of the actual position (x).

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
WEIGHT		double	1	weight in correction
TILT		double	0.0	rotation about longitudinal axis
CALIBRATION		double	1	calibration factor for readout
ORDER		long	0	matrix order
READOUT		STRING	NULL	rpn expression for readout (actual position supplied in variable x)

7.19 IBSCATTER

A simulation of intra-beam scattering.

Parameter Name	Units	Type	Default	Description
COUPLING		double	1	x-y coupling
FACTOR		double	1	factor by which to multiply growth rates before using
CHARGE	C	double	0.0	beam charge (or use CHARGE element)
DO_X		long	1	do x-plane scattering?
DO_Y		long	1	do y-plane scattering?
DO_Z		long	1	do z-plane scattering?
SMOOTH		long	0	Use smooth method instead of random numbers?

7.20 KICKER

A combined horizontal-vertical steering magnet implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
HKICK	RAD	double	0.0	x kick angle
VKICK	RAD	double	0.0	y kick angle
TILT	RAD	double	0.0	rotation about longitudinal axis
B2	$1/M^2$	double	0.0	normalized sextupole strength (e.g., kick = KICK*(1+B2*x$\hat{2}$))
HCALIBRATION		double	1	factor applied to obtain x kick
VCALIBRATION		double	1	factor applied to obtain y kick
EDGE_EFFECTS		long	0	include edge effects?
ORDER		long	0	matrix order
STEERING		long	1	use for steering?

7.21 KPOLY

A thin kick element with polynomial dependence on the coordinates in one plane.

Parameter Name	Units	Type	Default	Description
COEFFICIENT	M^{-ORDER}	double	0.0	coefficient of polynomial
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FACTOR		double	1	additional factor to apply
ORDER		long	0	order of polynomial
PLANE		STRING	x	plane to kick (x, y)

7.22 KQUAD

A canonical kick quadrupole, which differs from the MULT element with ORDER=1 in that it can be used for tune correction.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K1	$1/M^2$	double	0.0	geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
BORE	M	double	0.0	bore radius
B	T	double	0.0	pole tip field (used if bore nonzero)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE	M	double	0.0	fractional strength error
N_KICKS		long	4	number of kicks
SYNCH_RAD		long	0	include classical synchrotron radiation?
SYSTEMATIC_MULTIPOLES		STRING	NULL	input file for systematic multipoles
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles
INTEGRATION_ORDER		long	4	integration order (2 or 4)

This element simulates a quadrupole using a kick method based on symplectic integration. The user specifies the number of kicks and the order of the integration. For computation of twiss parameters and response matrices, this element is treated like a standard thick-lens quadrupole; i.e., the number of kicks and the integration order become irrelevant.
Specification of systematic and random multipole errors is supported through the `SYSTEMATIC_MULTIPOLES` and `RANDOM_MULTIPOLES` fields. These fields give the names of SDDS files that supply the multipole data. The files are expected to contain a single page of data with the following elements:

1. Floating point parameter `referenceRadius` giving the reference radius for the multipole data.

2. An integer column named `order` giving the order of the multipole. The order is defined as $(N_{poles} - 2)/2$, so a quadrupole has order 1, a sextupole has order 2, and so on.

3. Floating point columns `an` and `bn` giving the values for the normal and skew multipole strengths, respectively. These are defined as a fraction of the main field strength measured at the reference radius, R: $a_n = \frac{K_n r^n/n!}{K_m r^m/m!}$, where $m = 1$ is the order of the main field and n is the order of the error multipole. A similar relationship holds for the skew multipoles. For random multipoles, the values are interpreted as rms values for the distribution.

7.23 KSBEND

A kick bending magnet which is NOT canonical, but is better than a 2nd order matrix implementation. Recommend using CSBEND instead.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric quadrupole strength
K2	$1/M^3$	double	0.0	geometric sextupole strength
K3	$1/M^3$	double	0.0	geometric octupole strength
K4	$1/M^4$	double	0.0	geometric decapole strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
H1	$1/M$	double	0.0	entrance pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE		double	0.0	fractional strength error
ETILT		double	0.0	error rotation about incoming longitudinal axis
N_KICKS		long	4	number of kicks
NONLINEAR		long	1	include nonlinear field components?
SYNCH_RAD		long	0	include classical synchrotron radiation?
EDGE1_EFFECTS		long	1	include entrace edge effects?
EDGE2_EFFECTS		long	1	include exit edge effects?
EDGE_ORDER		long	1	edge matrix order
PARAXIAL		long	0	use paraxial approximation?
TRANSPORT		long	0	use (incorrect) TRANSPORT equations for T436 of edge?
METHOD		STRING	modified-midpoint	integration method (modified-midpoint, leap-frog

7.24 KSEXT

A canonical kick sextupole, which differs from the MULT element with ORDER=2 in that it can be used for chromaticity correction.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K2	$1/M^3$	double	0.0	geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
BORE	M	double	0.0	bore radius
B	T	double	0.0	field at pole tip (used if bore nonzero)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE	M	double	0.0	fractional strength error
N_KICKS		long	4	number of kicks
SYNCH_RAD		long	0	include classical synchrotron radiation?
SYSTEMATIC_MULTIPOLES		STRING	NULL	input file for systematic multipoles
RANDOM_MULTIPOLES		STRING	NULL	input file for random multipoles
INTEGRATION_ORDER		long	4	integration order (2 or 4)

This element simulates a sextupole using a kick method based on symplectic integration. The user specifies the number of kicks and the order of the integration. For computation of twiss parameters, chromaticities, and response matrices, this element is treated like a standard thick-lens sextuupole; i.e., the number of kicks and the integration order become irrelevant.
Specification of systematic and random multipole errors is supported through the `SYSTEMATIC_MULTIPOLES` and `RANDOM_MULTIPOLES` fields. These fields give the names of SDDS files that supply the multipole data. The files are expected to contain a single page of data with the following elements:

1. Floating point parameter `referenceRadius` giving the reference radius for the multipole data.

2. An integer column named `order` giving the order of the multipole. The order is defined as $(N_{poles} - 2)/2$, so a quadrupole has order 1, a sextupole has order 2, and so on.

3. Floating point columns `an` and `bn` giving the values for the normal and skew multipole strengths, respectively. These are defined as a fraction of the main field strength measured at the reference radius, R: $a_n = \frac{K_n r^n/n!}{K_m r^m/m!}$, where $m = 2$ is the order of the main field and n is the order of the error multipole. A similar relationship holds for the skew multipoles. For random multipoles, the values are interpreted as rms values for the distribution.

7.25 MAGNIFY

An element that allows multiplication of phase-space coordinates of all particles by constants.

Parameter Name	Units	Type	Default	Description
MX		double	1	factor for x coordinates
MXP		double	1	factor for x' coordinates
MY		double	1	factor for y coordinates
MYP		double	1	factor for y' coordinates
MS		double	1	factor for s coordinates
MDP		double	1	factor for (p-pCentral)/pCentral

7.26 MALIGN

A misalignment of the beam, implemented as a zero-order matrix.

Parameter Name	Units	Type	Default	Description
DXP		double	0.0	delta x'
DYP		double	0.0	delta y'
DX	M	double	0.0	delta x
DY	M	double	0.0	delta y
DZ	M	double	0.0	delta z
DT	S	double	0.0	delta t
DP		double	0.0	delta p/pCentral
DE		double	0.0	delta gamma/gammaCentral
ON_PASS		long	-1	pass on which to apply
FORCE_MODIFY_MATRIX		long	0	modify the matrix even if on_pass>=0

7.27 MAPSOLENOID

A numerically-integrated solenoid specified as a map of (Bz, Br) vs (z, r).

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
ETILT	*RAD*	double	0.0	misalignment
EYAW	*RAD*	double	0.0	misalignment
EPITCH	*RAD*	double	0.0	misalignment
N_STEPS		long	100	number of steps (for nonadaptive integration)
INPUTFILE		STRING	NULL	SDDS file containing (Br, Bz) vs (r, z). Each page should have values for a fixed r.
RCOLUMN		STRING	NULL	column containing r values
ZCOLUMN		STRING	NULL	column containing z values
BRCOLUMN		STRING	NULL	column containing Br values
BZCOLUMN		STRING	NULL	column containing Bz values
FACTOR		double	0.0001	factor by which to multiply fields in file
BXUNIFORM		double	0.0	uniform horizontal field to superimpose on solenoid field
BYUNIFORM		double	0.0	uniform vertical field to superimpose on solenoid field
LUNIFORM		double	0.0	length of uniform field superimposed on solenoid field
ACCURACY		double	0.0001	integration accuracy
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)

7.28 MARK

A marker, equivalent to a zero-length drift space.

Parameter Name	Units	Type	Default	Description
FITPOINT		long	0	supply Twiss parameters, moments, floor coordinates for optimization?

7.29 MATR

Explicit matrix input from a text file, in the format written by the print_matrix command.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
FILENAME		STRING		input file
ORDER		long	1	matrix order

The input file for this element uses a simple text format. It is identical to the output in the `printout` file generated by the tt matrix_output command. For example, for a 1st-order matrix, the file would have the following appearance:
description: *C1 C2 C3 C4 C5 C6*
R1: *R11 R12 R13 R14 R15 R16*
R2: *R21 R22 R23 R24 R25 R26*
R3: *R31 R32 R33 R34 R35 R36*
R4: *R41 R42 R43 R44 R45 R46*
R5: *R51 R52 R53 R54 R55 R56*
R6: *R61 R62 R63 R64 R65 R66*

Items in normal type must be entered exactly as shown, whereas those in italics must be provided by the user. The colons are important! For this particular example, one would set `ORDER=1` in the `MATR` definition. In general, the *Ci* are zero, except for *C5*, which is usually equal to the length of the element (which must be specified with the `L` parameter in the `MATR` definition).

7.30 MATTER

A Coulomb-scattering and energy-absorbing element simulating material in the beam path.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
XO	M	double	0.0	radiation length
ELASTIC		long	0	elastic scattering?
ENERGY_STRAGGLE		long	0	use simple-minded energy straggling model?
Z		long	0	Atomic number
A	AMU	double	0.0	Atomic mass
RHO	KG/M^3	double	0.0	Density
PLIMIT		double	0.05	Probability cutoff for each slice

This element is based on section 3.3.1 of the *Handbook of Accelerator Physics and Engineering*, specifically, the subsections **Single Coulomb scattering of spin-$\frac{1}{2}$ particles**, **Multiple Coulomb scattering through small angles**, and **Radiation length**. There are two aspects to this element: scattering and energy loss.

Scattering. The multiple Coulomb scattering formula is used whenever the thickness of the material is greater than $0.001X_o$, where X_o is the radiation length. (Note that this is inaccurate for materials thicker than $100X_o$.) For this regime, the user need only specify the material thickness (L) and the radiation length (XO).

For materials thinner than $0.001X_o$, the user must specify additional parameters, namely, the atomic number (Z), atomic mass (A), and mass density (RHO) of the material. Note that the density is given in units of kG/m^3. (Multiply by 10^3 to convert g/cm^3 to kG/m^3.) In addition, the simulation parameter PLIMIT may be modified.

To understand this parameter, one must understand how `elegant` simulates the thin materials. First, it computes the expected number of scattering events per particle, $E = \sigma_T nL = \frac{K_1\pi^3 nL}{K_2^2+K_2*\pi^2}$, where n is the number density of the material, L is the thickness of the material, $K_1 = (\frac{2Zr_e}{\beta^2\gamma})^2$, and $K_2 = \frac{\alpha^2 Z^{\frac{2}{3}}}{\beta\gamma}$, with r_e the classical electron radius and α the fine structure constant. The material is then broken into N slices, where $N = E/P_{limit}$. For each slice, each simulation particle has a probability E/N of scattering. If scattering occurs, the location within the slice is computed using a uniform distribution over the slice thickness.

For each scatter that occurs, the scattering angle, θ is computed using the cumulative probability distribution $F(\theta > \theta_o) = \frac{K_2(\pi^2-\theta_o^2)}{\pi^2(K_2+\theta_o^2)}$. This can be solved for θ_o, giving $\theta_o = \sqrt{\frac{(1-F)K_2\pi^2}{K_2+F\pi^2}}$. For each scatter, F is chosen from a uniform random distribution on $[0, 1]$.

Energy loss. Energy loss simulation is very simple. The energy loss per unit distance traveled, x, is $\frac{d}{E}dx = -E/X_o$. Hence, in traveling through a material of thickness L, the energy of each particle is transformed from E to Ee^{-L/X_o}.

Energy straggling. This refers to variation in the energy lost by particles. The model used by `elegant` is *very*, *very* crude. It assumes that the standard deviation of the energy loss is equal to half the mean energy loss. This is an overestimate, we think, and is provided to give an upper

bound on the effects of energy straggling until a real model can be developed. Note one obvious problem with this: if you split a MATTER element of length L into two pieces of length L/2, the total energy loss will not not change, but the induced energy spread will be about 30% lower, due to addition in quadrature.

7.31 MAXAMP

A collimating element that sets the maximum transmitted particle amplitudes for all following elements, until the next MAXAMP.

Parameter Name	Units	Type	Default	Description
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
ELLIPTICAL		long	0	is aperture elliptical?

7.32 MODRF

A first-order matrix RF cavity with exact phase dependence, plus optional amplitude and phase modulation.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
VOLT	V	double	0.0	nominal voltage
PHASE	DEG	double	0.0	nominal phase
FREQ	Hz	double	500000000	nominal frequency
Q		double	0.0	cavity Q
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
AMMAG		double	0.0	magnitude of amplitude modulation
AMPHASE	DEG	double	0.0	phase of amplitude modulation
AMFREQ	Hz	double	0.0	frequency of amplitude modulation
PMMAG	DEG	double	0.0	magnitude of phase modulation
PMPHASE	DEG	double	0.0	phase of phase modulation
PMFREQ	Hz	double	0.0	frequency of phase modulation
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)

7.33 MONI

A two-plane position monitor, accepting two rpn equations for the readouts as a function of the actual positions (x and y).

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
WEIGHT		double	1	weight in correction
TILT		double	0.0	rotation about longitudinal axis
XCALIBRATION		double	1	calibration factor for x readout
YCALIBRATION		double	1	calibration factor for y readout
ORDER		long	0	matrix order
XREADOUT		STRING	NULL	rpn expression for x readout (actual position supplied in variables x, y
YREADOUT		STRING	NULL	rpn expression for y readout (actual position supplied in variables x, y

7.34 MULT

A canonical kick multipole.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KNL	$M^{(1-ORDER)}$	double	0.0	integrated geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
BORE	M	double	0.0	bore radius
BNL	TM	double	0.0	integrated field at pole tip, used if BORE nonzero
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FACTOR		double	1	factor by which to multiply strength
ORDER		long	1	multipole order
N_KICKS		long	4	number of kicks
SYNCH_RAD		long	0	include classical synchrotron radiation?

7.35 NIBEND

A numerically-integrated dipole magnet with various extended-fringe-field models.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	arc length
ANGLE	*RAD*	double	0.0	bending angle
E1	*RAD*	double	0.0	entrance edge angle
E2	*RAD*	double	0.0	exit edge angle
TILT		double	0.0	rotation about incoming longitudinal axis
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
FINT		double	0.5	edge-field integral
HGAP	*M*	double	0.0	half-gap between poles
FP1	*M*	double	10	fringe parameter (tanh model)
FP2	*M*	double	1	not used
FSE		double	0.0	fractional strength error
ETILT		double	0.0	error rotation about incoming longitudinal axis
ACCURACY		double	0.0001	integration accuracy
MODEL		STRING	linear	fringe model (hard-edge, linear, cubic-spline, tanh, quintic
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta
SYNCH_RAD		long	0	include classical synchrotron radiation?

7.36 NISEPT

A numerically-integrated dipole magnet with a Cartesian gradient.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
E1	RAD	double	0.0	entrance edge angle
B1	$1/M$	double	0.0	normalized gradient (K1=B1*L/ANGLE)
Q1REF	M	double	0.0	distance from septum at which bending radius is L/ANGLE
FLEN	M	double	0.0	fringe field length
ACCURACY		double	0.0001	integration accuracy
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta
MODEL		STRING	linear	fringe model (hard-edge, linear, cubic-spline, tanh, quintic

7.37 OCTU

Not implemented–use the MULT element.

Parameter Name	Units	Type	Default	Description

7.38 PEPPOT

A pepper-pot plate.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
RADII	*M*	double	0.0	hole radius
TRANSMISSION		double	0.0	transmission of material
TILT	*RAD*	double	0.0	rotation about longitudinal axis
THETA_RMS	*RAD*	double	0.0	rms scattering from material
N_HOLES		long	0	number of holes

7.39 PFILTER

An element for energy and momentum filtration.

Parameter Name	Units	Type	Default	Description
DELTALIMIT		double	-1	maximum fractional momentum deviation
LOWERFRACTION		double	0.0	fraction of lowest-momentum particles to remove
UPPERFRACTION		double	0.0	fraction of highest-momentum particles to remove
FIXPLIMITS		long	0	fix the limits in p from LOWERFRACTION and UPPERFRACTION applied to first beam
BEAMCENTERED		long	0	if nonzero, center for DELTALIMIT is average beam momentum

7.40 QUAD

A quadrupole implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K1	$1/M^2$	double	0.0	geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
FFRINGE		double	0.0	fraction of length occupied by linear fringe region
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE	M	double	0.0	fractional strength error
ORDER		long	0	matrix order

7.41 QUFRINGE

An element consisting of a linearly increasing or decreasing quadrupole field.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K1	$1/M^2$	double	0.0	peak geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE	M	double	0.0	fractional strength error
DIRECTION		long	0	1=entrance, -1=exit
ORDER		long	0	matrix order

7.42 RAMPP

A momentum-ramping element that changes the central momentum according to a mpl format file of the momentum factor vs time in seconds.

Parameter Name	Units	Type	Default	Description
WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving momentum factor vs time

7.43 RAMPRF

A voltage-ramped RF cavity, implemented like RFCA. The voltage ramp pattern is given by a mpl-format file of the voltage factor vs time in seconds.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
VOLT	V	double	0.0	nominal voltage
PHASE	DEG	double	0.0	nominal phase
FREQ	Hz	double	500000000	nominal frequency
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
VOLT_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving voltage waveform factor vs time
PHASE_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving phase offset vs time (requires FREQ_WAVEFORM)
FREQ_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving frequency factor vs time (requires PHASE_WAVEFORM)
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)

7.44 RBEN

A rectangular dipole, implemented as a SBEND with edge angles.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric focusing strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
K2	$1/M^3$	double	0.0	geometric sextupole strength
H1	$1/M$	double	0.0	entrace pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
DX	M	double	0.0	misaligment of entrance
DY	M	double	0.0	misalignment of entrace
DZ	M	double	0.0	misalignment of entrance
FSE		double	0.0	fractional strength error
ETILT	RAD	double	0.0	error rotation about incoming longitudinal axis
EDGE1_EFFECTS		long	1	include entrace edge effects?
EDGE2_EFFECTS		long	1	include exit edge effects?
ORDER		long	0	matrix order
EDGE_ORDER		long	0	edge matrix order
TRANSPORT		long	0	use (incorrect) TRANSPORT equations for T436 of edge?
USE_BN		long	0	use B1 and B2 instead of K1 and K2 values?
B1	$1/M$	double	0.0	K1 = B1*rho, where rho is bend radius
B2	$1/M^2$	double	0.0	K2 = B2*rho

7.45 RCOL

A rectangular collimator.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
X_MAX	M	double	0.0	half-width in x
Y_MAX	M	double	0.0	half-width in y
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment

7.46 RECIRC

An element that defines the point to which particles recirculate in multi-pass tracking

Parameter Name	Units	Type	Default	Description
I_RECIRC_ELEMENT		long	0	

7.47 REFLECT

Reflects the beam back on itself, which is useful for multiple beamline matching.

Parameter Name	Units	Type	Default	Description
DUMMY		long	0	

7.48 REMCOR

An element to remove correlations from the tracked beam to simulate certain types of correction.

Parameter Name	Units	Type	Default	Description
X		long	1	remove correlations in x?
XP		long	1	remove correlations in x'?
Y		long	1	remove correlations in y?
YP		long	1	remove correlations in y'?
WITH		long	6	coordinate to remove correlations with (1,2,3,4,5,6)=(x,x',y,y',s,dP/Po)
ONCE_ONLY		long	0	compute correction only for first beam, apply to all?

7.49 RFCA

A first-order matrix RF cavity with exact phase dependence.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
VOLT	V	double	0.0	peak voltage
PHASE	DEG	double	0.0	phase
FREQ	Hz	double	500000000	frequency
Q		double	0.0	cavity Q
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
CHANGE_P0		long	0	does cavity change central momentum?
CHANGE_T		long	0	not recommended
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)
END1_FOCUS		long	0	include focusing at entrance?
END2_FOCUS		long	0	include focusing at exit?
BODY_FOCUS_MODEL		STRING	NULL	None (default) or SRS (simplified Rosenzweig/Serafini for standing wave)
N_KICKS		long	1	number of kicks to use. Set to zero for matrix method.
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
T_REFERENCE	S	double	-1	arrival time of reference particle
LINEARIZE		long	0	Linearize phase dependence?

The body-focusing model is based on Rosenzweig and Serafini, Phys. Rev. E 49 (2), 1599. As suggested by N. Towne (NSLS), I simplified this to assume a pure pi-mode standing wave.

7.50 RFCW

A combination of RFCA, WAKE, and TRWAKE.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
CELL_LENGTH	M	double	0.0	cell length (used to scale wakes)
VOLT	V	double	0.0	voltage
PHASE	DEG	double	0.0	phase
FREQ	Hz	double	500000000	frequency
Q		double	0.0	cavity Q
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
CHANGE_P0		long	0	does element change central momentum?
CHANGE_T		long	0	not recommended
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)
END1_FOCUS		long	0	include focusing at entrance?
END2_FOCUS		long	0	include focusing at exit?
BODY_FOCUS_MODEL		STRING	NULL	None (default) or SRS (simplified Rosenzweig/Serafini for standing wave)
N_KICKS		long	1	number of kicks to use. Set to zero for matrix method.
WAKEFILE		STRING	NULL	name of file containing Green functions
ZWAKEFILE		STRING	NULL	if WAKEFILE=NULL, optional name of file containing longitudinal Green function
TRWAKEFILE		STRING	NULL	if WAKEFILE=NULL, optional name of file containing transverse Green functions
TCOLUMN		STRING	NULL	column containing time data
WXCOLUMN		STRING	NULL	column containing x Green function
WYCOLUMN		STRING	NULL	column containing y Green function

A combination of RFCA, WAKE, and TRWAKE.

Parameter Name	Units	Type	Default	Description
WZCOLUMN		STRING	NULL	column containing longitudinal Green function
N_BINS		long	0	number of bins for current histogram
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	smooth current histogram?
SG_HALFWIDTH		long	4	Savitzky-Golay filter half-width for smoothing
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
LINEARIZE		long	0	Linearize phase dependence?

This element is a combination of the RFCA, WAKE, and TRWAKE elements. As such, it provides combined simulation of an rf cavity with longitudinal and transverse wakes.

For the wakes, the input files and their interpretation are identical to WAKE and TRWAKE, except that the transverse and longitudinal wakes are interpreted as the wakes for a single cell of length given by the CELL_LENGTH parameter.

Users should read the entries for WAKE, TRWAKE, and RFCA for more details on this element.

7.51 RFDF

A simple traveling-wave (beta=1) deflecting RF cavity.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
PHASE	*DEG*	double	0.0	phase
TILT	*RAD*	double	0.0	rotation about longitudinal axis
FREQUENCY	*HZ*	double	2856000000	frequency
VOLTAGE	*V*	double	0.0	voltage
TIME_OFFSET	*S*	double	0.0	time offset (adds to phase)
N_KICKS		long	1	number of kicks
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)

7.52 RFMODE

A simulation of a beam-driven TM monopole mode of a RF cavity.

Parameter Name	Units	Type	Default	Description
RA	*Ohm*	double	0.0	shunt impedance
RS	*Ohm*	double	0.0	shunt impedance (Ra=2*Rs)
Q		double	0.0	cavity Q
FREQ	*Hz*	double	0.0	frequency
CHARGE	*C*	double	0.0	beam charge (or use CHARGE element)
INITIAL_V	*V*	double	0.0	initial voltage
INITIAL_PHASE	*RAD*	double	0.0	initial phase
INITIAL_T	*S*	double	0.0	time at which INITIAL_V and INITIAL_PHASE held
BETA		double	0.0	normalized load impedance
BIN_SIZE	*S*	double	0.0	bin size for current histogram (use 0 for autosize)
N_BINS		long	20	number of bins for current histogram
PRELOAD		long	0	preload cavity with steady-state field
PRELOAD_FACTOR		double	1	multiply preloaded field by this value
RIGID_UNTIL_PASS		long	0	don't affect the beam until this pass
DETUNED_UNTIL_PASS		long	0	cavity is completely detuned until this pass
SAMPLE_INTERVAL		long	1	passes between output to RECORD file
RECORD		STRING	NULL	output file for cavity fields
SINGLE_PASS		long	0	if nonzero, don't accumulate field from pass to pass
PASS_INTERVAL		long	1	interval in passes at which to apply PASS_INTERVAL times the field (may increase speed)

7.53 RFTMEZ0

A TM-mode RF cavity specified by the on-axis Ez field.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
FREQUENCY	*HZ*	double	2856000000	frequency
PHASE	*RAD*	double	0.0	phase
EZ_PEAK	*V*	double	0.0	Peak on-axis longitudinal electric field
TIME_OFFSET	*S*	double	0.0	time offset (adds to phase)
PHASE_REFERENCE		long	0	phase reference number (to link to other time-dependent elements)
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
ETILT	*RAD*	double	0.0	misalignment
EYAW	*RAD*	double	0.0	misalignment
EPITCH	*RAD*	double	0.0	misalignment
N_STEPS		long	100	number of steps (for nonadaptive integration)
RADIAL_ORDER		long	1	highest order in off-axis expansion
CHANGE_P0		long	0	does element change central momentum?
INPUTFILE		STRING	NULL	file containing Ez vs z at r=0
ZCOLUMN		STRING	NULL	column containing z values
EZCOLUMN		STRING	NULL	column containing Ez values
SOLENOID_FILE		STRING	NULL	file containing map of Bz and Br vs z and r. Each page contains values for a single r.
SOLENOID_ZCOLUMN		STRING	NULL	column containing z values for solenoid map.
SOLENOID_RCOLUMN		STRING	NULL	column containing r values for solenoid map.
SOLENOID_BZCOLUMN		STRING	NULL	column containing Bz values for solenoid map.
SOLENOID_BRCOLUMN		STRING	NULL	column containing Br values for solenoid map.

RFTMEZ0 continued

A TM-mode RF cavity specified by the on-axis Ez field.

Parameter Name	Units	Type	Default	Description
SOLENOID_FACTOR		double	1	factor by which to multiply solenoid fields.
ACCURACY		double	0.0001	integration accuracy
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")

7.54 RMDF

A linearly-ramped electric field deflector, using an approximate analytical solution FOR LOW ENERGY PARTICLES.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
TILT	*RAD*	double	0.0	rotation about longitudinal axis
RAMP_TIME	*S*	double	1e-09	length of ramp
VOLTAGE	*V*	double	0.0	full voltage
GAP	*M*	double	0.01	gap between plates
TIME_OFFSET	*S*	double	0.0	time offset of ramp start
N_SECTIONS		long	10	number of sections
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment

7.55 ROTATE

An element that rotates the beam coordinates about the longitudinal axis.

Parameter Name	Units	Type	Default	Description
TILT	*RAD*	double	0.0	rotation about longitudinal axis

7.56 SAMPLE

An element that reduces the number of particles in the beam by interval-based or random sampling.

Parameter Name	Units	Type	Default	Description
FRACTION		double	1	fraction to keep
INTERVAL		long	1	interval between sampled particles

7.57 SBEN

A sector dipole implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	arc length
ANGLE	RAD	double	0.0	bend angle
K1	$1/M^2$	double	0.0	geometric focusing strength
E1	RAD	double	0.0	entrance edge angle
E2	RAD	double	0.0	exit edge angle
TILT	RAD	double	0.0	rotation about incoming longitudinal axis
K2	$1/M^3$	double	0.0	geometric sextupole strength
H1	$1/M$	double	0.0	entrace pole-face curvature
H2	$1/M$	double	0.0	exit pole-face curvature
HGAP	M	double	0.0	half-gap between poles
FINT		double	0.5	edge-field integral
DX	M	double	0.0	misaligment of entrance
DY	M	double	0.0	misalignment of entrace
DZ	M	double	0.0	misalignment of entrance
FSE		double	0.0	fractional strength error
ETILT	RAD	double	0.0	error rotation about incoming longitudinal axis
EDGE1_EFFECTS		long	1	include entrace edge effects?
EDGE2_EFFECTS		long	1	include exit edge effects?
ORDER		long	0	matrix order
EDGE_ORDER		long	0	edge matrix order
TRANSPORT		long	0	use (incorrect) TRANSPORT equations for T436 of edge?
USE_BN		long	0	use B1 and B2 instead of K1 and K2 values?
B1	$1/M$	double	0.0	K1 = B1*rho, where rho is bend radius
B2	$1/M^2$	double	0.0	K2 = B2*rho

7.58 SCATTER

A scattering element to add gaussian random numbers to particle coordinates.

Parameter Name	Units	Type	Default	Description
X	*M*	double	0.0	rms scattering level for x
XP	*M*	double	0.0	rms scattering level for x'
Y	*M*	double	0.0	rms scattering level for y
YP	*M*	double	0.0	rms scattering level for y'
DP	*M*	double	0.0	rms scattering level for (p-pCentral)/pCentral

7.59 SCRAPER

A collimating element that sticks into the beam from one side only. The directions 0, 1, 2, and 3 are from +x, +y, -x, and -y, respectively.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
POSITION	*M*	double	0.0	position of edge
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
XO	*M*	double	0.0	radiation length
INSERT_FROM		STRING	NULL	direction from which inserted (+x, -x, +y, -y
ELASTIC		long	0	elastic scattering?
DIRECTION		long	-1	obsolete

7.60 SCRIPT

An element that allows transforming the beam using an external script.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	Length to be used for matrix-based operations such as twiss parameter computation.
COMMAND		STRING	NULL	SDDS-compliant command to apply to the beam. Use the sequence %i to represent the input filename and %o to represent the output filename.
ROOTNAME		STRING	NULL	Rootname for use in naming input and output files. %s may be used to represent the run rootname.
INPUT_EXTENSION		STRING	in	Extension for the script input file.
OUTPUT_EXTENSION		STRING	out	Extension for the script output file.
KEEP_FILES		long	0	If nonzero, then script input and output files are not deleted after use. By default, they are deleted.
NP0		double	0.0	User-defined numerical parameter for command substitution for sequence %np0
NP1		double	0.0	User-defined numerical parameter for command substitution for sequence %np1
NP2		double	0.0	User-defined numerical parameter for command substitution for sequence %np2
NP3		double	0.0	User-defined numerical parameter for command substitution for sequence %np3
NP4		double	0.0	User-defined numerical parameter for command substitution for sequence %np4
NP5		double	0.0	User-defined numerical parameter for command substitution for sequence %np5

SCRIPT continued

An element that allows transforming the beam using an external script.

Parameter Name	Units	Type	Default	Description
NP6		double	0.0	User-defined numerical parameter for command substitution for sequence %np6
NP7		double	0.0	User-defined numerical parameter for command substitution for sequence %np7
NP8		double	0.0	User-defined numerical parameter for command substitution for sequence %np8
NP9		double	0.0	User-defined numerical parameter for command substitution for sequence %np9
SP0		STRING	NULL	User-defined string parameter for command substitution for sequence %sp0
SP1		STRING	NULL	User-defined string parameter for command substitution for sequence %sp1
SP2		STRING	NULL	User-defined string parameter for command substitution for sequence %sp2
SP3		STRING	NULL	User-defined string parameter for command substitution for sequence %sp3
SP4		STRING	NULL	User-defined string parameter for command substitution for sequence %sp4
SP5		STRING	NULL	User-defined string parameter for command substitution for sequence %sp5
SP6		STRING	NULL	User-defined string parameter for command substitution for sequence %sp6
SP7		STRING	NULL	User-defined string parameter for command substitution for sequence %sp7

SCRIPT continued

An element that allows transforming the beam using an external script.

Parameter Name	Units	Type	Default	Description
SP8		STRING	NULL	User-defined string parameter for command substitution for sequence %sp8
SP9		STRING	NULL	User-defined string parameter for command substitution for sequence %sp9

7.61 SEXT

A sextupole implemented as a matrix, up to 2nd order

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K2	$1/M^3$	double	0.0	geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
FSE	M	double	0.0	fractional strength error
ORDER		long	0	matrix order

7.62 SOLE

A solenoid implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KS	RAD/M	double	0.0	geometric strength
B	T	double	0.0	field strength (used if KS is zero)
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
DZ	M	double	0.0	misalignment
ORDER		long	0	matrix order

7.63 SREFFECTS

Simulation of synchrotron radiation effects (damping and quantum excitation).

Parameter Name	Units	Type	Default	Description
JX		double	1	x damping partition number
JY		double	1	y damping partition number
JDELTA		double	2	momentum damping partition number
EXREF	m	double	0.0	reference equilibrium x emittance
EYREF	m	double	0.0	reference equilibrium y emittance
SDELTAREF	m	double	0.0	reference equilibrium fractional momentum spread
DDELTAREF		double	0.0	reference fractional momentum loss (per turn)
PREF	$m_e c$	double	0.0	reference momentum (to which other reference values pertain)
COUPLING		double	0.0	x-y coupling
FRACTION		double	1	fraction of implied SR effect to simulate with each instance
DAMPING		long	1	include damping?
QEXCITATION		long	1	include quantum excitation?
LOSSES		long	1	include average losses?

This element allows simulation of synchrotron radiation effects in a lumped fashion for quick, approximate results. There are two ways to set up the element: explicit initialization or automatic initialization.

In explicit initialization, the user supplies the quantities `EXREF`, `EYREF`, `SDELTAREF`, `DDELTAPREF`, and `PREF`. These are, respectively, the reference values for the x-plane emittance, y-plane emittance, fractional momentum spread, energy loss per turn, and momentum. The first four values pertain to the reference momentum. `JX`, `JY`, and `JDELTA` may also be given, although the defaults work for typical lattices.

In automatic initialization, the user turns on the radiation integral feature in `twiss_output`, causing `elegant` to automatically compute the above quantities. The `COUPLING` parameter can be used to change the partitioning of quantum excitation between the horizontal and vertical planes.

7.64 STRAY

A stray field element with local and global components. Global components are defined relative to the initial beamline direction. ** Not correct if there are tilts in the beamline. **

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
LBX	T	double	0.0	local Bx
LBY	T	double	0.0	local By
GBX	T	double	0.0	global Bx
GBY	T	double	0.0	global By
GBZ	T	double	0.0	global Bz
ORDER		long	0	matrix order

7.65 TMCF

A numerically-integrated accelerating TM RF cavity with spatially-constant fields.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
FREQUENCY	*HZ*	double	2856000000	frequency
PHASE	*S*	double	0.0	phase
TIME_OFFSET	*S*	double	0.0	time offset (adds to phase)
RADIAL_OFFSET	*M*	double	1	not recommended
TILT	*RAD*	double	0.0	rotation about longitudinal axis
ER	*V*	double	0.0	radial electric field
BPHI	*T*	double	0.0	azimuthal magnetic field
EZ	*V*	double	0.0	longitudinal electric field
ACCURACY		double	0.0001	integration accuracy
X_MAX	*M*	double	0.0	x half-aperture
Y_MAX	*M*	double	0.0	y half-aperture
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of steps (for nonadaptive integration)
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, nonadaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")

7.66 TRCOUNT

An element that defines the point from which transmission calculations are made.

Parameter Name	Units	Type	Default	Description
DUMMY		long	0	

7.67 TRFMODE

A simulation of a beam-driven TM dipole mode of a RF cavity.

Parameter Name	Units	Type	Default	Description
RA	Ohm	double	0.0	shunt impedance
RS	Ohm	double	0.0	shunt impedance (Ra=2*Rs)
Q		double	0.0	cavity Q
FREQ	Hz	double	0.0	frequency
CHARGE	C	double	0.0	beam charge (or use CHARGE element)
BETA		double	0.0	normalized load impedance
BIN_SIZE	S	double	0.0	bin size for current histogram (use 0 for autosize)
N_BINS		long	20	number of bins for current histogram
PLANE		STRING	both	x, y, or both
SINGLE_PASS		long	0	if nonzero, don't accumulate field from pass to pass
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment

7.68 TRWAKE

Transverse wake specified as a function of time lag behind the particle.

Parameter Name	Units	Type	Default	Description
INPUTFILE		STRING	NULL	name of file giving Green functions
TCOLUMN		STRING	NULL	column in INPUTFILE containing time data
WXCOLUMN		STRING	NULL	column in INPUTFILE containing x Green function
WYCOLUMN		STRING	NULL	column in INPUTFILE containing y Green function
CHARGE	C	double	0.0	beam charge (or use CHARGE element)
FACTOR	C	double	1	factor to multiply wake by
N_BINS		long	128	number of bins for current histogram
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	smooth current histogram?
SG_HALFWIDTH		long	4	Savitzky-Golay filter half-width for smoothing
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignmnet

The input file for this element gives the transverse-wake Green functions, $W_x(t)$ and $W_y(t)$, versus time behind the particle. The units of the wakes are V/C/m, so this element simulates the integrated wake of some structure (e.g., a cell or series of cells). If you have, for example, the wake for a cell and you need the wake for N cells, then you may use the `FACTOR` parameter to make the appropriate multiplication. The values of the time coordinate should begin at 0 and be equi-spaced. A positive value of time represents the distance behind the exciting particle.

Use of the `CHARGE` parameter on the `TRWAKE` element is disparaged. It is preferred to use the `CHARGE` element as part of your beamline to define the charge.

Setting the `N_BINS` paramater to 0 is recommended. This results in auto-scaling of the number of bins to accomodate the beam. The bin size is fixed by the spacing of the time points in the wake.

The default degree of smoothing (`SG_HALFWIDTH=4`) may be excessive. It is suggested that users vary this parameter to verify that results are reliable if smoothing is employed (`SMOOTHING=1`).

7.69 TUBEND

A special rectangular bend element for top-up backtracking.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	arc length
ANGLE	*RAD*	double	0.0	bend angle
FSE		double	0.0	fractional strength error
OFFSET		double	0.0	horizontal offset of magnet center from arc center
MAGNET_WIDTH		double	0.0	horizontal width of the magnet pole
MAGNET_ANGLE		double	0.0	angle that the magnet was designed for

7.70 TWISS

Sets Twiss parameter values.

Parameter Name	Units	Type	Default	Description
BETAX	M	double	1	horizontal beta function
BETAY	M	double	1	vertical beta function
ALPHAX		double	0.0	horizontal alpha function
ALPHAY		double	0.0	vertical alpha function
FROM_BEAM		long	0	compute correction from tracked beam properties instead of Twiss parameters?
ONCE_ONLY		long	0	compute correction only for first beam or input twiss parameters, apply to all?

7.71 TWLA

A numerically-integrated first-space-harmonic traveling-wave linear accelerator.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
FREQUENCY	HZ	double	2856000000	frequency
PHASE	RAD	double	0.0	phase
TIME_OFFSET	S	double	0.0	time offset (adds to phase)
EZ	V/M	double	0.0	electric field
B_SOLENOID	T	double	0.0	solenoid field
ACCURACY		double	0.0001	integration accuracy
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
BETA_WAVE		double	1	(phase velocity)/c
ALPHA	$1/M$	double	0.0	field attenuation factor
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of steps (for nonadaptive integration)
FOCUSSING		long	1	include focusing effects?
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, nonadaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
CHANGE_P0		long	0	does element change central momentum?

7.72 TWMTA

A numerically-integrated traveling-wave muffin-tin accelerator.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
FREQUENCY	HZ	double	2856000000	frequency
PHASE	RAD	double	0.0	phase
EZ	V/M	double	0.0	electric field
ACCURACY		double	0.0001	integration accuracy
X_MAX	M	double	0.0	x half-aperture
Y_MAX	M	double	0.0	y half-aperture
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
KX	$1/M$	double	0.0	horizontal wave number
BETA_WAVE		double	1	(phase velocity)/c
BSOL		double	0.0	solenoid field
ALPHA	$1/M$	double	0.0	field attenuation factor
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of kicks
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")

7.73 TWPL

A numerically-integrated traveling-wave stripline deflector.

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
RAMP_TIME	*S*	double	1e-09	time to ramp to full strenth
TIME_OFFSET	*S*	double	0.0	offset of ramp-start time
VOLTAGE	*V*	double	0.0	maximum voltage between plates due to ramp
GAP	*M*	double	0.01	gap between plates
STATIC_VOLTAGE	*V*	double	0.0	static component of voltage
TILT	*RAD*	double	0.0	rotation about longitudinal axis
ACCURACY		double	0.0001	integration accuracy
X_MAX	*M*	double	0.0	x half-aperture
Y_MAX	*M*	double	0.0	y half-aperture
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
N_STEPS		long	100	number of steps (for nonadaptive integration)
METHOD		STRING	runge-kutta	integration method (runge-kutta, bulirsch-stoer, nonadaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")

7.74 VKICK

A vertical steering dipole implemented as a matrix, up to 2nd order.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KICK	RAD	double	0.0	kick strength
TILT	RAD	double	0.0	rotation about longitudinal axis
B2	$1/M^2$	double	0.0	normalized sextupole strength (kick = KICK*(1+B2*y$\hat{2}$))
CALIBRATION		double	1	strength multiplier
EDGE_EFFECTS		long	0	include edge effects?
ORDER		long	0	matrix order
STEERING		long	1	use for steering?

7.75 VMON

A vertical position monitor, accepting a rpn equation for the readout as a function of the actual position (y).

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
WEIGHT		double	1	weight in correction
TILT		double	0.0	rotation about longitudinal axis
CALIBRATION		double	1	calibration factor for readout
ORDER		long	0	matrix order
READOUT		STRING	NULL	rpn expression for readout (actual position supplied in variable y)

7.76 WAKE

Longitudinal wake specified as a function of time lag behind the particle.

Parameter Name	Units	Type	Default	Description
INPUTFILE		STRING	NULL	name of file giving Green function
TCOLUMN		STRING	NULL	column in INPUTFILE containing time data
WCOLUMN		STRING	NULL	column in INPUTFILE containing Green function
CHARGE	C	double	0.0	beam charge (or use CHARGE element)
FACTOR	C	double	1	factor to multiply wake by
N_BINS		long	128	number of bins for current histogram
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	smooth current histogram?
SG_HALFWIDTH		long	4	Savitzky-Golay filter half-width for smoothing
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
CHANGE_P0		long	0	change central momentum?
ALLOW_LONG_BEAM		long	0	allow beam longer than wake data?

The input file for this element gives the longitudinal Green function, $W(t)$ versus time behind the particle. The units of the wake are V/C, so this element simulates the integrated wake of some structure (e.g., a cell or series of cells). If you have, for example, the wake for a cell and you need the wake for N cells, then you may use the `FACTOR` parameter to make the appropriate multiplication. The values of the time coordinate should begin at 0 and be equi-spaced. A positive value of time represents the distance behind the exciting particle.

Use of the `CHARGE` parameter on the `WAKE` element is disparaged. It is preferred to use the `CHARGE` element as part of your beamline to define the charge.

Setting the `N_BINS` paramater to 0 is recommended. This results in auto-scaling of the number of bins to accomodate the beam. The bin size is fixed by the spacing of the time points in the wake.

The default degree of smoothing (`SG_HALFWIDTH=4`) may be excessive. It is suggested that users vary this parameter to verify that results are reliable if smoothing is employed (`SMOOTHING=1`).

7.77 WATCH

A beam property/motion monitor–allowed modes are centroid, parameter, coordinate, and fft.

Parameter Name	Units	Type	Default	Description
FRACTION		double	1	fraction of particles to dump (coordinate mode)
INTERVAL		long	1	interval for data output (in turns)
START_PASS		long	0	pass on which to start
FILENAME		STRING		output filename
LABEL		STRING		output label
MODE		STRING	coordinates	coordinate, parameter, or centroid
X_DATA		long	1	include x data in coordinate mode?
Y_DATA		long	1	include y data in coordinate mode?
LONGIT_DATA		long	1	include longitudinal data in coordinate mode?
EXCLUDE_SLOPES		long	0	exclude slopes in coordinate mode?
FLUSH_INTERVAL		long	0	file flushing interval (parameter or centroid mode)

7.78 WIGGLER

A wiggler or undulator for damping or excitation of the beam. Does not include focusing effects.

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
RADIUS	M	double	0.0	peak bending radius
POLES		long	0	number of wiggler poles

7.79 ZLONGIT

A simulation of a single-pass broad-band or functionally specified longitudinal impedance.

Parameter Name	Units	Type	Default	Description
CHARGE	C	double	0.0	beam charge (or use CHARGE element)
BROAD_BAND		long	0	broad-band impedance?
RA	Ohm	double	0.0	shunt impedance
RS	Ohm	double	0.0	shunt impedance (Ra=2*Rs)
Q		double	0.0	cavity Q
FREQ	Hz	double	0.0	frequency (BROAD_BAND=1)
ZREAL		STRING	NULL	<filename>=<x>+<y> form specification of input file giving real part of impedance vs f (BROAD_BAND=0)
ZIMAG		STRING	NULL	<filename>=<x>+<y> form specification of input file giving imaginary part of impedance vs f (BROAD_BAND=0)
BIN_SIZE	S	double	0.0	bin size for current histogram (use 0 for autosize)
N_BINS		long	128	number of bins for current histogram
WAKES		STRING	NULL	filename for output of wake
WAKE_INTERVAL		long	1	interval in passes at which to output wake
AREA_WEIGHT		long	0	use area-weighting in assigning charge to histogram?
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	smooth current histogram?
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
SG_HALFWIDTH		long	4	Savitzky-Golay filter halfwidth for smoothing

7.80 ZTRANSVERSE

A simulation of a single-pass broad-band or functionally-specified transverse dipole impedance.

Parameter Name	Units	Type	Default	Description
CHARGE	C	double	0.0	beam charge (or use CHARGE element)
BROAD_BAND		long	0	broad-band impedance?
RS	Ohm	double	0.0	shunt impedance (Ra=2*Rs)
Q		double	0.0	cavity Q
FREQ	Hz	double	0.0	frequency (BROAD_BAND=1)
INPUTFILE		STRING	NULL	name of file giving impedance (BROAD_BAND=0)
FREQCOLUMN		STRING	NULL	column in INPUTFILE containing frequency
ZXREAL		STRING	NULL	column in INPUTFILE containing real impedance for x plane
ZXIMAG		STRING	NULL	column in INPUTFILE containing imaginary impedance for x plane
ZYREAL		STRING	NULL	column in INPUTFILE containing real impedance for y plane
ZYIMAG		STRING	NULL	column in INPUTFILE containing imaginary impedance for y plane
BIN_SIZE	S	double	0.0	bin size for current histogram (use 0 for autosize)
INTERPOLATE		long	0	interpolate wake?
N_BINS		long	128	number of bins for current histogram
SMOOTHING		long	0	smooth current histogram?
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
SG_HALFWIDTH		long	4	Savitzky-Golay filter halfwidth for smoothing
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment

8 Examples

Example runs and post-processing files are included along with the distribution of `elegant`. These are drawn from the author's research and all concern various aspects of the Argonne Positron Accumulator Ring (PAR) and its injection and ejection lines (LTP and PTB, respectively).

The examples are intended to demonstrate program capabilities with minimal work on the user's part. Each demo is invoked using a command (a C-shell script) that can both run `elegant` and post-process the output. After running the demo, the output can be viewed again without rerunning `elegant` by invoking the command with the word `review` added to the command line. Including the word `hardcopy` on the command line results in the graphs being sent to the default printer, which is assumed to accept Postscript.

The post-processing is typically handled by a lower-level script that is called from the demo script. These lower-level scripts are good models for the creation of customized scripts for user applications.

1. `par10h*` — These files provide a demonstration of Twiss parameter computation, tracking, element variation, and map analysis. The lattice is defined with kick elements, which are used for all tracking. After computation of the Twiss parameters for the PAR[6], a series of particles are tracked with different initial x coordinates. Finally, the tunes and Twiss parameters are computed by tracking; they are very close to the analytical values. The post-processing commands make phase-space plots and plots of FFTs of the motion, showing that the motion becomes chaotic at the stability limit. To execute this demo, type the command `par10h`.

2. `par_sympl*` — These files provide a demonstration of the symplecticity of tracking with `elegant` kick elements. A single large-amplitude particle is tracked for 2^{14} turns. The invariant J_x is then computed and plotted as a function of turn number. To execute this demo, type the command `par_sympl`. The post-processing takes quite some time because of the very large number of points.

3. `par_chrom*` — These files provide a demonstration of computing chromaticity and other parameters as a function of momentum offset using map analysis. The lattice is the same as `par10h.lte`, except all of the elements are implemented using second-order matrices. Hence, the chromaticity from tracking should be nearly identical to the analytical results computed by the `twiss_output` command, which it is. To run this demonstration, enter `par_chrom`. The reader may wish to try this demo again using `ksbend`, `csbend`, or `nibend` elements in place of the `sbend` elements, and `kquad` (`ksext`) elements in place of the `quad` (`sext`) elements.

4. `par_damp*` — These files provide a demonstration of damping partition calculation using single turn tracking with synchrotron radiation. The expected value of the longitudinal damping partition for PAR is $J_\delta = 1.758$. The user may edit the lattice file, `par_damp.lte`, to invoke a different element for the dipole magnet. In particular, definitions for numerically integrated dipoles with extended fringe-fields are present. To execute this demo, type the command `par_damp`.

5. `par_dynap*` — These files provide a demonstration of dynamic aperture runs for a series of randomized machines. Also exhibited here are orbit, tune, and chromaticity correction. The post-processing commands make a plot of the dynamic apertures with the physical aperture superimposed. (The `orbcorr_plots` script can also be used to plot orbit correction information.) To execute this demo, type the command `par_dynap`. The lattice has been stripped

down so that only a few of the more significant multipoles are present. Also, fictitious extra sextupoles have been added to compensate the lack of second-order edge terms in the bending magnets (these would result in nonsymplectic tracking if included). Still, the running time is many hours.

6. `ejoptk*` — These files provide a demonstration of the optimization of a multi-turn ejection bump for PAR, using a time-dependent kicker waveform (formed from two cubic splines). After optimization, the lattice is tracked with a realistic beam distribution to verify good transmission and show the centroid position vs z over three turns. To execute this demo, type the command `ejoptk`.

7. `ltp_te*` — These files provide a demonstration of transport line simulation. The Linac-to-PAR transport line is simulated with errors and trajectory correction to predict the transmission losses and the steering error at the exit of the septum. The trajectory correction uses tracking of a beam distribution, which is slower than tracking the centroid, but which produces better results in the presence of the large momentum spread. The reader may wish to verify this by turning off this feature and running the simulation again. To execute this demo, type the command `ltp_te`. The running time for this demo is quite long.

9 The rpn Calculator

The program `rpn` is a Reverse Polish Notation programmable scientific calculator written in C. It is incorporated as a subprogram into `elegant`, and a number of the SDDS programs. It also exists as a command-line program, `rpnl`, which executes its command-line arguments as `rpn` operations and prints the result before exiting. Use of `rpn` in any of these modes is extremely straightforward. Use of the program in its stand-alone form is the best way to gain familiarity with it. Once one has entered `rpn`, entering "help" will produce a list of the available operators with brief summaries of their function. Also, the `rpn` definitions file `rpn.defns`, distributed with `elegant`, gives examples of most `rpn` operation types.

Like all RPN calculators, `rpn` uses stacks. In particular, it has a numeric stack, a logical stack, and a string stack. Items are pushed onto the numeric stack whenever a number-token is entered, or whenever an operation concludes that has a number as its result; items are popped from this stack by operations that require numeric arguments. Items are pushed onto the logical stack whenever a logical expression is evaluated; they are popped from this stack by use of logical operations that require logical arguments (e.g., logical ANDing), or by conditional branch instructions. Items enclosed in double quotes are pushed onto the string stack; items are popped from this stack by use of operations that require string arguments (e.g., formatted printing).

`rpn` supports user-defined memories and functions. To create a user-defined memory, one simply stores a value into the name, as in "1 sto unity"; the memory is created automatically when `rpn` detects that it does not already exist. To create a user-defined function, enter the "udf" command; `rpn` will prompt for the function name and the text that forms the function body. To invoke a UDF, simply type the name.

A file containing `rpn` commands can be executed by pushing the filename onto the string stack and invoking the "@" operator. `rpn` supports more general file I/O through the use of functions that mimic the standard C I/O routines. Files are identified by integer unit numbers, with units 0 and 1 being permanently assigned to the terminal input and terminal output, respectively.

References

[1] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, Englewood Cliffs, N.J., second edition, 1988.

[2] H. Grote, F. C. Iselin, "The MAD Program–Version 8.1," CERN/SL/90-13(AP), June 1991.

[3] K. L. Brown, R. V. Servranckx, "First- and Second-Order Charged Particle Optics," SLAC-PUB-3381, July 1984.

[4] M. Borland, "A High-Brightness Thermionic Microwave Electron Gun," SLAC-Report-402, February 1991, Stanford University Ph.D. Thesis.

[5] H. A. Enge, "Achromatic Mirror for Ion Beams," Rev. Sci. Inst., 34(4), 1963.

[6] M. Borland, private communication.

[7] W. H. Press, *et al*, *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1988.

[8] M. Borland, "A Self-Describing File Protocol for Simulation Integration and Shared Postprocessors," Proc. 1995 PAC, May 1-5, 1995, Dallas, Texas, pp. 2184-2186 (1996).

[9] M. Borland, "A Universal Postprocessing Toolkit for Accelerator Simulation and Data Analysis," Proc. 1998 ICAP Conference, Sept. 14-18, 1998, Monterey, California, to be published.

[10] T. P. Green, "Research Toward a Heterogeneous Networked Computer Cluster: The Distributed Queuing System Version 3.0," SCRI Technical Publication, 1994.

[11] M. Borland *et al*, "Start-to-End Jitter Simulation of the LCLS," Proceedings of the 2001 Particle Accelerator Conference, Chicago, 2001.

[12] M. Borland and L. Emery, "Tracking Studies of Top-Up Safety for the Advanced Photon Source,", Proceedings of the 1999 Particle Accelerator Conference, New York, 1999, pg 2319-2321.

[13] M. Xie, "Free ElectronLaser Driven by SLAC LINAC".

[14] S. Reiche, *NIM* A 429 (1999) 242.