User's Manual for `elegant`

Program Version 19.1
Advanced Photon Source
Michael Borland

August 14, 2008

Note: another source of help for `elegant` is the on-line forum. Users are encouraged to join and participate.

1 Highlights of What's New in Version 19.1

Here is a summary of what's changed since release 19.0. For details and references, see the specific entries in the manual. Several minor bugs were fixed. There were also several significant feature enhancements, including RADIA-style kick maps and the ability to simulate particles other than electrons. A new program for computing the shielded CSR impedance is also included.

N.B. There was a change to the `SCRIPT` *element that is not by default backward compatible. See below for details.*

1.1 Known Bugs and Problems

- Twiss output contains entries for the higher-order dispersion, tune shifts with amplitude, higher-order chromaticity, and tune spreads due to chromaticity and amplitude *even when these are not calculated*, which is potentially misleading. The values are zero when the calculation is not requested.

- Computation of closed orbits and Twiss parameters will not always include the effects of synchrotron radiation losses when these are imposed using `SREFFECTS` elements. See the documentation for `SREFFECTS` for details.

- The file created with the parameters field of `run_setup` does not contain any non-numerical parameters of the lattice.

- When `transmute_elements` is used to turn a sextupole into a quadrupole, several problems are seen. Some of the quadrupole parameters are filled with garbage values. When `alter_elements` or `save_lattice` are subsequently used, `elegant` crashes. These problems were reported by V. Sajaev (ANL).

1.2 Bug Fixes for Elements

- The `CSRDRIFT` element was excluded from element division. This has been corrected. Y. Ding (SLAC) reported this.

- The `SCRIPT` element would be treated as a drift space if matrix concatenation was requested in `run_setup`. W. Guo (BNL) reported problem. This has been changed, so that `elegant` now by default uses tracking to determine the matrix. *As reported by C. Yao (ANL), this may impact the operation of some runs, but can be turned off using the* `DRIFT_MATRIX` *parameter.*

1.3 Bug Fixes for Commands

- The `twiss_output` command would fail to generate Twiss parameters for elements upstream of a recirculation element. This would prevent the use of `SREFFECTS` and `SCMULT` elements. M. Cornacchia (SLAC) reported this.

- A new mode (`lines_only=1`) was added to `tune_shift_with_amplitude` that allows turning off the two-dimensional fitting. V. Sajaev (ANL) reported that the two-dimensional fitting may be unreliable, giving $\partial\nu_x/\partial A_y$ significantly different from $\partial\nu_y/\partial A_x$.

1.4 New and Modified Elements

- A new element, `KQUSE`, was added. It combines quadrupole and sextupole fields. V. Sajaev (ANL) requested this.

- A new element, `UKICKMAP`, was added. It provides simulation of undulators using kick maps [27]. A script (`km2sdds`) is provide to translate the RADIA[28] output into SDDS for use by `elegant`. This was requested by W. Guo (BNL), who also assisted with the implementation and debugging.

- The `SCRIPT` element previously was treated as a drift space for the purpose of Twiss parameter computations. In this release, the linear matrix of the element is determined by tracking. This was suggested by W. Guo (BNL).

- The `WATCH` and `HISTOGRAM` elements now incorporate a feature allowing the element occurrence number to be substituted into the output filename. This can be helpful in creating a set of histograms or particle dumps with sequentially-numbered filenames. See the documentation for those elements for details.

- The `CWIGGLER`, `CSBEND`, `CSRCSBEND`, `KQUAD`, and `KSEXT` elements now have a parameter (`ISR1PART`) that allows turning off ISR when a single particle is being tracked (e.g., for an explicit fiducialization step).

- The `ILMATRIX` element now has a `TILT` parameter. A. Petrenko (FNAL) pointed out that this element didn't include x-y coupling. This parameter may help in that regard.

1.5 New and Modified Commands

- A new command, `change_particle`, was added that allows tracking particles other than electrons. Predefined types are positrons, protons, and muons. Others may be defined in terms of charge and mass ratios to the electron. This was requested by H. Sayed and Y. Roblin (TJNAF). Y. Roblin also helped with testing.

- A new command, `insert_elements`, was added by A. Xiao (ANL). It allows inserting elements into a lattice at many locations. This is a convenient way to add a large number of identical elements (e.g., markers, monitors) to a lattice without modifying the lattice file.

- The `frequency_map` command now computes the diffusion, defined as $\log_{10}(\Delta\nu_x^2 + \Delta\nu_y^2)$.
- The `matrix_output` command now allows outputting the individual *on-trajectory* matrices of the elements, rather than the just concatenated matrix of the beamline. This was requested by N. Sereno (ANL).
- The `magnets` output from `run_setup` now has a column giving the element type (`ElementType`).

1.6 Changes to Related Programs and Files

- A new program, `csrImpedance`, was added. This program, written by Y. Wang, H. Shang, and M. Borland (ANL), computes the shielded steady-state CSR impedance in a form that can be immediately used with `elegant`'s `ZLONGIT` element.
- A new script, `km2sdds`, was added. This script translates kick maps into SDDS format for use by `elegant`.
- A new script, `astra2elegant`, was added. This script converts ASTRA particle data to a form that `elegant` accepts.
- A new script, `elegant2astra`, was added. This script converts `elegant` particle data to a form that ASTRA accepts.

2 Credits

Contributors to `elegant` include M. Borland, W. Guo, R. Soliday, V. Sajaev, Y. Wang, Y. Wu, and A. Xiao. Contributors to related programs include M. Borland, R. Dejus, L. Emery, H. Shang, and Y. Wang. R. Soliday is responsible for multi-platform builds and distribution. Of course, we also appreciate the many suggestions, comments, and bug reports from users.

3 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.

3.1 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.

In passing, we note another "philosophical" point about `elegant`, namely, the goal of complete backward compatibility. We consider it unacceptable if a new version of the program gives different answers than an old version, unless the old version was wrong. Hence, there are sometimes less-than-ideal default settings in `elegant`, incorrect spelling of parameters, etc., that are never fixed, because doing so would break old input files. It helps to read the manual pages carefully for the more complex features to ensure that the defaults are understood and appropriate.

3.2 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 $(\mathrm{x}, \mathrm{x}', \mathrm{y}, \mathrm{y}', \mathrm{s}, \delta)$, where x (y) is the horizontal (vertical) transverse coordinate, primed quantities are slopes, s is the *total, equivalent* 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 $\Delta\mathrm{P}/\mathrm{P_o}$. 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. Some of the more unusual elements available are third-order alpha-magnets[5, 4], time-dependent kicker magnets, voltage-ramped RF cavities, beam scrapers, and

beam-analysis "screens."

Several elements support simulation of collective effects, such as short-range wakefields, resonator impedances, intra-beam scattering, coherent synchrotron radiation, and the longitudinal space charge impedance.

A wide variety of output is available from tracking, including centroid and sigma-matrix output along the accelerator, phase space output at arbitrary locations, turn-by-turn moments at arbitrary locations, histograms of particle coordinates, coordinates of lost particles, and initial coordinates of transmitted particles. 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. `elegant` makes it easy to evaluate a large number of ensembles ("seeds") in a single run. Alternatively, different ensembles can be readily run of different CPUs and the SDDS output files combined.

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. It will optimize a user defined function of the transfer matrix elements (up to third-order), beta functions, tunes, chromaticities, radiation integrals, natural emittance, floor coordinates, beam moments, etc. It also has the ability to optimize results of tracking using a user-supplied function of the beam parameters at one or more locations. 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. One may also invoke a search procedure that finds the aperture boundary. A related feature is the ability to determine the frequency map for an accelerator, to help identify aperture-limiting resonances.

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.

4 Digression on the Longitudinal Coordinate Definition

A word is in order about the definition of s, which we've described as the *total, equivalent* distance traveled. First, by *total* distance we mean that s is *not* measured relative to the bunch center or a fiducial particle. It is entirely a property of the individual particle and its path through the accelerator.

To explain what we mean by *equivalent* distance, note that the relationship between s and arrival time t at the observation point is, for each particle, $s = \beta ct$, where βc is the instantaneous velocity of the particle. Whenever a particle's velocity changes, `elegant` recomputes s to ensure that this relationship holds. s is thus the "equivalent" distance the particle would have traveled at the present velocity to arrive at the observation point at the given time. This book-keeping is required because `elegant` was originally a matrix-only code using s as the longitudinal coordinate.

Users should keep the meaning of s in mind when viewing statistics for s, for example, in the `sigma` or watch point output files. A quantity like `Ss` is literally the rms spread in s. It is *not* defined as $\sigma_t/(\langle\beta\rangle c)$. A nonrelativistic beam with velocity spread will show no change in `Ss` in a drift space, because the distance traveled is the same for all particles.

5 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.

6 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.

`elegant` also allows execution of commands in the shell as part of evaluation of a namelist field. To invoke this, one encloses the commandline string in curly braces. E.g.,

```
betax = "{sdds2stream -parameter=betaxFinal data.twi}"
```

(Note that the quotes are also required.) In this example, `betax` is assigned the value of the parameter `betaxFinal` from the file `data.twi`. Frequently, the commandline RPN calculator, `rpnl` is also used in this way, for example

```
betax = "{rpnl 8 pi / 2 /}"
```

assigns the value $8/(2\pi)$ to `betax`. One possible pitfall with using `rpnl` in this fashion is interpretation of the multiplication symbol (`*`) as a file wildcard by the shell. For this reason, the alternate multiplication operator `mult` is preferred, e.g.,

```
betax = "{rpnl 8 pi mult}"
```

rather than

```
betax = "{rpnl 8 pi *}"
```

We used the program `rpnl` in these examples because it is perhaps familiar. However, versions 17.4 and later allow direct evaluation of RPN expressions in commands whenever parentheses are used to delimite a sequence. For example,

```
betax = "(8 2 / pi /)"
```

(Note that the quotes are also required.) The advantages of this method are speed (no subprocess is needed), lack of intermediate interpretation by the shell, and persistence of the stack and variables. So, for example, one might use

```
betax = "(8 2 / pi / sto betax0)"
betay = "(betax0)"
```

Another advantage is the ability to mix subcommands and rpn expressions, as in

```
betax = "({sdds2stream -parameter=betaxFinal data.twi} 2 /)"
```

would assign to `betax` half the value of the parameter `betaxFinal` from the file `data.twi`.

6.1 Table of elegant commands and their functions

Table 1: Table of elegant commands and their functions.

alter_elements6.2	Change an element parameter from the command file.
amplification_factors6.3	Compute orbit amplification functions.
analyze_map6.4	Determine first-order matrix from tracking.
aperture_data6.5	Define aperture using an SDDS file.
bunched_beam6.6	Set up beam generation.
change_particle6.7	Change the type of particle. Default is electron.
chromaticity6.8	Correct the chromaticity.
closed_orbit6.9	Compute the closed orbit.
correct6.10	Correct the orbit or trajectory.
correction_matrix_output6.11	Obtain orbit/trajectory correction matrix in a file.
correct_tunes6.12	Correct the tunes.
coupled_twiss_output6.13	Compute and output coupled twiss parameters.
divide_elements6.14	Specify division of elements into pieces.
error_element6.15	Define errors for a set of elements.
error_control6.16	Set up and control error generation process.
find_aperture6.17	Determine the transverse (e.g., dynamic) aperture.
floor_coordinates6.18	Compute and output floor coordinates.
frequency_map6.19	Compute and output frequency map.
insert_sceffects6.20	Insert elements into the lattice at many places.
insert_sceffects6.21	Insert space charge kick elements.
linear_chromatic_tracking_setup6.22	Set up for fast tracking with chromatic effects.
link_control6.23	Control linking of element parameters.
link_elements6.24	Define link between parameters of two elements.
load_parameters6.25	Load element parameters from SDDS file.
matrix_output6.26	Output transfer matrix along beamline.
moments_output6.27	Compute coupled beam moments, with radiation option.
momentum_aperture6.28	Determine s-dependent momentum aperture.

Table 2: Table of `elegant` commands and their functions.

`optimize`6.29	Execute an optimization.
`optimization_constraint`6.30	Define a constraint for optimization.
`optimization_covariable`6.31	Define a dependent parameter for optimization.
`optimization_setup`6.32	Perform initial optimization setup.
`optimization_term`6.33	Define a term of penalty function.
`optimization_variable`6.34	Define an optimization variable.
`print_dictionary`6.35	Print the element dictionary.
`rpn_expression`6.36	Execute an expression in the rpn interpreter.
`rpn_load`6.37	Load values from SDDS file into rpn interpreter.
`run_control`6.38	Set up simulation steps and passes.
`run_setup`6.39	Define global simulation parameters and output files.
`sasefel`6.40	Evaluate SASE FEL gain etc.
`save_lattice`6.41	Save new lattice file.
`sdds_beam`6.42	Define loading of particles from SDDS file.
`semaphores`6.43	Define file semaphores for start/end of run.
`slice_analysis`6.44	Perform slice analysis along beamline.
`subprocess`6.45	Execute a command in the shell.
`steering_element`6.46	Define element parameters as steering correctors.
`transmute_elements`6.47	Transmute elements from one type to another.
`twiss_analysis`6.48	Define subset of beamline for twiss parameter analysis.
`twiss_output`6.49	Set up twiss parameter computation.
`track`6.50	Execute tracking of particles and other operations.
`tune_shift_with_amplitude`6.51	Compute tune shifts with amplitude.
`vary_element`6.52	Vary element parameters in loops.

6.2 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;
        long start_occurence = 0;
        long end_occurence = 0;
        double s_start = -1;
        double s_end = -1;
        STRING before = NULL;
        STRING after = NULL;
&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.

- `start_occurence`, `end_occurence` — If nonzero, these give the starting and ending occurence numbers of elements that will be altered. N.B.: if wildcards are used, occurence number counting is for each set of identically-named elements separately, rather than for the sequence of matched elements.
- `s_start`, `s_end` — If non-negative, these give the gaving and ending position limits for the end-of-element locations of elements to be altered.
- `after` — The name of an element. If given, the alteration is applied only to elements that follow the named element in the beamline.
- `before` — The name of an element. If given, the alteration is applied only to elements that precede the named element in the beamline.

6.3 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.

6.4 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.

6.5 aperture_data

- type: setup command.
- function: specify a file from which to take x and y aperture data vs s.

```
&aperture_data
        STRING input = NULL;
        long periodic = 1;
        long persistent = 0;
        long disable = 0;
&end
```

- `input` — Name of SDDS file supplying the aperture data. The following columns are all required, in double or float type, with units of `m` (meters).
  1. `s` — Distance along the central trajectory.
  2. `xHalfAperture` — Half aperture in the horizontal.
  3. `yHalfAperture` — Half aperture in the vertical.
  4. `xCenter` — Center of the aperture in the horizontal.
  5. `yCenter` — Center of the aperture in the vertical.
- `periodic` — If non-zero, the aperture is a periodic function of `s`, with period equal to the range of the data.
- `persistent` — If non-zero, the aperture data persists across subsequent `run_setup` commands. By default, the aperture data is forgotten when a new `run_setup` command is seen.
- `disable` — If non-zero, the command is ignored.

6.6 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 (except dynamic aperture), 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.

  The exception is "dynamic-aperture" distribution type. In this case, the cutoff value is the number of grid points in the dimension in question.

- `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.

6.7 chromaticity

- type: action command.
- function: change the particle type from the default value of "electron."
- N.B.: this feature has had limited testing, mostly to verify that electron tracking is not impacted by the implementation. Please use with caution and be alert for suspicious results.

```
&change_particle
    STRING name = "electron";
    double mass_ratio = 0;
    double charge_ratio = 0;
&end
```

- `name` — The name of the particle to use. Possible values are `electron`, `positron`, `proton`, `muon`, and `custom`.
- `mass_ratio`, `charge_ratio` — If the particle name is "custom," these parameters specify the mass and charge of the particle relative to the electron. E.g., for an anti-proton, one would use a mass ratio of 1836.18 and a charge ratio of 1.

6.8 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.

6.9 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;
    long start_from_dp_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.
- `start_from_dp_centroid` — A flag indicating whether to force the computation to use the momentum centroid of the beam for the closed orbit. This can allow computing the closed orbit for an off-momentum beam, then starting the beam on that orbit using the `offset_by_orbit` or `center_on_orbit` parameters of the `track` command. In contrast to the `start_from_centroid`, this command doesn't force the algorithm to start from the beam transverse centroids.
- `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.

6.10 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 minimum_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", "one-to-best", "one-to-next", "thread", or "global". "One-to-one" and "one-to-next" are the same: steering is performed by pairing one corrector with the next downstream BPM. "One-to-best" attempts to find a BPM with a large response to each corrector. "Thread" does corrector sweeps to work the beam through a beamline with apertures; it is quite slow. "Global" simply uses the global response matrix; it is the best choice if the trajectory is not lost on an aperture. 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 for transport lines. [The word "tweak" is misspelled "tweek" in the code.] The default value, 1 mrad, may be too large for systems with small apertures. If you get an error message about "tracking failed for test particle," try decreasing this value.
- `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.
- `minimum_cycles` — The minimum number of x-y cycles to perform, even if the correction does not improve.
- `n_iterations` — Number of iterations of the correction in each plane 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. This is valid for trajectory correction only.
- `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.

6.11 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. Recommended values: "%s.hrm" and "%s.vrm".
- `inverse` — Array of (incomplete) filenames for SDDS output of the x and y inverse response matrices. Recommended values: "%s.hirm" and "%s.virm".
- `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.

6.12 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.

6.13 coupled_twiss_output

- type: setup/action command.
- function: set up or execute computation of coupled twiss parameters and beam sizes

```
&coupled_twiss_output
    STRING filename = NULL;
    long output_at_each_step = 0;
    long emittances_from_twiss_command = 1;
    double emit_x = 0;
    double emittance_ratio = 0.01;
    double sigma_dp = 0;
    long calculate_3d_coupling = 1;
    long verbosity = 0;
    long concat_order = 2;
&end
```

- `filename` — The (incomplete) name of the SDDS file to which coupled twiss parameters and beam sizes will be written. Suggested value: "%s.ctwi".
- `output_at_each_step` — If nonzero, then this is a setup command and results in computations occurring for each simulation step (e.g., for each perturbed machine if errors are included). If zero, then this is an action command and computations are done immediately (e.g., for the unperturbed machine).
- `emittances_from_twiss_command` — If nonzero, then the values of the horizontal emittance and the momentum spread are taken from the uncoupled computation done with the `twiss_output` command. In this case, the user must issue a `twiss_output` command prior to the `coupled_twiss_output`. If zero, then the values of the horizontal emittance and the momentum spread are taken from the parameters `emit_x` and `sigma_dp`, respectively. `\verb|emit_x|`— Gives the horizontal emittance, if `emittances_from_twiss_command=0`.
- `emittance_ratio` — Gives the ratio of the x and y emittances. Used to determine the vertical emittance from the horizontal emittance. Note that the computation is not self-consistent. I.e., the user is free to enter any emittance ratio desired, whether it is consistent with the machine optics or now.
- `sigma_dp` — Gives the momentum spread, if `emittances_from_twiss_command=0`.

This feature was added to `elegant` using code supplied by V. Sajaev, based on Ripkin's method. The code computes the coupled lattice functions, then uses the supplied emittance, emittance ratio, and momentum spread to compute the beam sizes, bunch length (if rf is included), and beam tilt.

6.14 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.

6.15 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;
    longn bind_across_names = 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`, `bind_across_names` — These parameters control "binding" of errors among elements, which means assigning the same error contribution to several elements. This occurs if `bind` is nonzero, **which it is by default**! If `bind` is negative, then the sign of the error will alternate between successive elements. `bind_number` can be used to limit the number of elements bound together. In particular, 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. Finally, by default, `elegant` only binds the errors of objects having the same name, even if they are assigned errors by the same `error_element` command (i.e., through a wildcard `name`). If `bind_across_names` is nonzero, then binding is done even for elements with different names.

- `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.

6.16 error_control

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

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

- `clear_error_settings` — Clear all previous error settings.
- `summarize_error_settings` — Summarize current error settings.
- `no_errors_for_first_step` — If non-zero, then there will be no errors for the first step. This can be useful for fiducialization of phase and momentum profiles.
- `error_log` — The (incomplete) name of a SDDS file to which error values will be written. Recommended value: "%s.erl".
- `error_factor` — A value by which to multiply the error amplitudes in all `error` commands.

6.17 find_aperture

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

```
&find_aperture
    STRING output = NULL;
    STRING search_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 assume_nonincreasing = 0;
    long verbosity = 0;
    long offset_by_orbit = 0;
&end
```

- `output` — The (incomplete) name of an SDDS file to send output to. Recommended value: "%s.aper".
- `mode` — May be "many-particle", "single-particle", "one-line", "three-line", "five-line", "seven-line", "nine-line", or "eleven-line". Many-particle searching is much faster than single-particle, but does not allow interval splitting to search for the aperture boundary. Both "many-particle" and "single-particle" modes involve searching from the outside inward, which improves speed but may result in including islands.

  The n-line modes avoid this by searching form the origin outward. Of these, the one-line and three-line modes are special: one-line mode searches the line from the origin to (x_{max}, y_{max}). three-line mode searches this line, plus the lines from the origin to $(x_{max}, 0)$ and $(0, y_{max})$. For n-line modes with $n > 3$, n lines are explored from $(0, 0)$ to $(x_{max} * sin(\theta), y_{max} * cos(\theta))$, where θ takes values from $-pi/2$ to $\pi/2$. In these modes, the output file contains a parameter called "Area," which gives the area of the dynamic aperture.
- `search_output` — The (incomplete) name of an SDDS file for output of detailed information on each tracked particle (single-particle mode only). Recommended value: "%s.apso".
- `boundary` — The (incomplete) name of an SDDS file for the boundary points of the aperture search. Recommended value: "%s.bnd". Valid for many- and single-particle modes.
- `xmin`, `xmax`, `ymin`, `ymax` — Region of the aperture search. The minimum values are relevant only for many- and single-particle modes.

- `nx` — For many- and single-particle modes, the number of x values to take in initial search. For line modes, this determines the initial x and y step sizes via $\Delta x = x_{max}/n_x$ and $\Delta y = y_{max}/n_x$.
- `ny` — For many- and single-particle modes, the number of y values to take in search. Ignored for line modes.
- `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. This is done only for single-particle and line modes.
- `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. Ignored for all but single-particle mode.
- `assume_nonincreasing` — If this variable is non-zero, the search assumes that the aperture at $\mathrm{y} + \mathrm{sign(y)} * \Delta\mathrm{y}$ is no larger than that at y. This results in tracking of fewer particles but may give a pessimistic result. Used only for single- and multi-particle modes.
- `offset_by_orbit` — A flag indicating whether to offset the transverse beam coordinates by the closed orbit before tracking. The default value is zero for backward compatibility, but the recommended value is 1.
- `verbosity` — A larger value results in more printouts during computations.

6.18 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 beamline.
- `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.

6.19 frequency_map

- type: action command. The number of turns tracked is set by the `run_control` command.
- function: compute frequency map from tracking

```
&frequency_map
    STRING output = NULL;
    double xmin = -0.1;
    double xmax = 0.1;
    double ymin = 1e-6;
    double ymax = 0.1;
    long nx = 21;
    long ny = 21;
    long verbosity = 1;
    long include_changes = 0;
&end
```

- `output` — The (incomplete) name of an SDDS file to send output to. Recommended value: "%s.fma".
- `xmin`, `xmax` — Limits of grid of initial x coordinates for tracking.
- `ymin`, `ymax` — Limits of grid of initial y coordinates for tracking. `ymin` should typically be a small, positive value so that there is some betatron oscillation from which to get the tune.
- `nx` — Number of values of x coordinate in the grid.
- `ny` — Number of values of y coordinate in the grid.
- `verbosity` — If nonzero, prints possibly useful information while running.
- `include_changes` — If nonozero, then computes not only the tunes, but also the changes in the tunes. It is recommended to leave this at the default value of 0, since the resolution of the tunes is higher then.

6.20 insert_elements

- type: setup command.
- function: Insert elements into a lattice at many locations. This is a convenient way to add elements to a lattice without modifying the lattice file.

```
&insert_elements
        STRING name = NULL;
        STRING type = NULL;
        STRING exclude = NULL;
        long skip = 0;
        long disable = 0;
        long add_at_end = 0;
        STRING element_def = NULL;
        long verbose = 1;
&end
```

- `name` — Possibily wild-card containing string specifying the name of the elements after which to insert the new element.
- `type` — Possibily wild-card containing string specifying the type of the elements after which to insert the new element.
- `exclude` — Possibily wild-card containing string specifying the name of elements to be excluded from the specified insertion.
- `skip` — If nonzero, the element is inserted at every n^{th} element.
- `disable` — If nonzero, the command is ignored.
- `add_at_end` — If nonzero, the element is also added at the end of the beamline.
- `element_def` — The definition of the element, just as it would be entered in the lattice file.
- `verbose` — A non-zero value results in printing of informational messages to the terminal.

6.21 insert_sceffects

- type: setup command.
- function: set up for transverse space charge calculation. **N.B.: This feature is not designed for space charge calculations in guns or linacs. It is only intended for simulating space charge in rings.**

```
&insert_sceffects
        STRING name = NULL;
        STRING type = NULL;
        STRING exclude = NULL;
        long disable = 0;
        long clear = 0;
        STRING element_prefix = "MYSC";
        long skip = 0;
        long vertical = 0;
        long horizontal = 0;
        long nonlinear = 0;
&end
```

- `name` — Possibily wild-card containing string specifying the name of the elements after which to insert the space charge kick element.
- `type` — Possibily wild-card containing string specifying the type of the elements after which to insert the space charge kick element.
- `exclude` — Possibily wild-card containing string specifying the name of elements to be excluded from the insertion of the space charge kick element.
- `disable` — If nonzero, the command is ignored.
- `clear` — If nonzero, all prior space charge insertions are deleted.
- `element_prefix` — Name under which the space charge kick will appear in the beamline.
- `skip` — If nonzero, the given number of insertion locations are skipped. If zero, only one space charge kick is inserted at the end of beamline.
- `vertical`, `horizontal`, `nonlinear` — If non-zero, then space charge is included in the plane in question.

Note: Some users use "*matched_to_cell*" or "*use_twiss_command_values*" inside "*bunched_beam*" command. Those options will erase SCMULT assignments along the beamline. In this case, user has to re-calculate twiss using "*twiss_output*" just before tracking.

6.22 linear_chromatic_tracking_setup

- type: setup command.
- function: define chromatic variation of beta functions, tunes, etc. for using in fast linear-chromatic tracking

```
&linear_chromatic_tracking_setup
    double nux[4] = {-1, 0, 0, 0};
    double betax[2] = {1.0, 0.0};
    double alphax[2] = {0.0, 0.0};
    double etax[2] = {0.0, 0.0};
    double etapx[2] = {0.0, 0.0};
    double nuy[4] = {-1, 0, 0, 0};
    double betay[2] = {1.0, 0.0};
    double alphay[2] = {0.0, 0.0};
    double etay[2] = {0.0, 0.0};
    double etapy[2] = {0.0, 0.0};
    double alphac[2] = {0.0, 0.0};
&end
```

- `nux` — Provide the horizontal tune plus its first three chromatic derivatives, i.e., $\partial\nu_x/\partial\delta$, $\partial^2\nu_x/\partial\delta^2$, and $\partial^3\nu_x/\partial\delta^3$.
- `betax` — Provide the horizontal beta function plus its chromatic derivative.
- `alphax` — Provide the horizontal alpha function plus its chromatic derivative.
- `etax` — Provide the first- and second-order horizontal dispersion: $\eta_x = \eta_x[0] + \eta_x[1]\,\delta$.
- `etapx` — Provide the first- and second-order horizontal dispersion slope.
- `alphac` — Provide the first and second-order momentum compaction. N.B: if you are tracking with an rf cavity, be sure that your lattice length equal to the actual circumference. See the example below.

An example of a good lattice for use with this feature is the following:

```
! Actual length of the ring
d0: drift,l=30.6667
! Actual rf voltage and frequency
rf: rfca,l=0,volt=30e3,phase=180,freq="c_mks 30.6667 /"
w1: watch,filename="%s.w1",mode="centroid"
ring: line=(d0,rf,w1)
```

Any additional elements (e.g,. wakes or impedances) should follow d0. No other elements represented by a matrix should be present!

An alternative for using this command is to use `ILMATRIX` elements in the lattice definition. This is more flexible, since one can include tune shifts with amplitude.

6.23 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.

6.24 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.

6.25 load_parameters

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

```
&load_parameters
        STRING filename = NULL;
        STRING filename_list = 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 allow_missing_parameters = 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).

- `filename_list` — A list of filenames, which may be given in place of `filename`. If used, each file in the list is treated as if it was separately supplied with an individual `load_parameters` command.

- `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.

- `allow_missing_parameters` — If set, it is not an error if any element in the lattice lacks a parameter that exists in the file.

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

6.26 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;
    long individual_matrices = 0;
    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.
- `individual_matrices` — If non-zero, the matrices in the SDDS file are the individual *on-trajectory* matrices of the elements, rather than the concatenated matrix of the beamline.
- `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.

6.27 moments_output

- type: action/setup command.
- function: compute periodic or non-periodic beam moments, optionally including radiation

```
&moments_output
    STRING filename = NULL;
    long output_at_each_step = 0;
    long output_before_tune_correction = 0;
    long final_values_only = 0;
    long verbosity = 0;
    long matched = 1;
    long radiation = 1;
    long n_slices = 10;
    double emit_x = 0;
    double beta_x = 0;
    double alpha_x = 0;
    double eta_x = 0;
    double etap_x = 0;
    double emit_y = 0;
    double beta_y = 0;
    double alpha_y = 0;
    double eta_y = 0;
    double etap_y = 0;
    double emit_z = 0;
    double beta_z = 0;
    double alpha_z = 0;
&end
```

- `filename` — The (incomplete) name of a file to which the moments results will be written. Recommended value: "%s.mom".
- `output_at_each_step` — A flag indicating, if set, that computations and/or 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.
- `verbosity` — Larger numbers result in an increasing amount of informational output to the standard output stream.
- `matched` — A flag indicating, if set, that the periodic or matched moments should be found.
- `radiation` — A flag indicating, if set, that synchrotron radiation effects should be included.

- `n_slices` — The number of slices into which to cut individual dipoles, quadrupoles, and sextuoples for computations. 10 has been found to work for all rings tested, but users are advised to ensure it is sufficient for their cases.

- `emit_x`, `beta_x`, `alpha_x`, `eta_x`, `etap_x`, and related quantities for `y` and `z` — If `matched=0`, then these specify the starting beam ellipses in all three planes.

This command performs several functions. In the most basic form, it propagates beam moments, i.e., the 6x6 sigma matrix, from the beginning to the end of a transport line, including coupling from rotated elements or offset sextupoles. This can be performed with or without synchrotron radiation effects in dipoles, quadrupoles, and sextupoles. (*NB: radiation effects are not yet included for wigglers!*) These computations include the evolution of the trajectory due to errors and (if included) synchrotron radiation.

If desired, the command will instead compute the periodic beam moments. In this case, the user must include an appropriate rf cavity in the lattice in order to get valid results. (By "appropriate rf cavity" we mean that it must have the right voltage, frequency, and phase to support stored beam.) It is also suggested that the user compute the closed orbit using `closed_orbit` so that the computations are performed on the closed orbit.

The results of moments computation may be subjected to optimization using values at marker elements. See the documentation for `MARK` for more details.

6.28 momentum_aperture

- type: action command.
- function: determine momentum aperture as a function of position in the lattice by tracking

```
&momentum_aperture
    STRING output = NULL;
    double x_initial = 0;
    double y_initial = 0;
    double delta_negative_limit = -0.10;
    double delta_positive_limit = 0.10;
    double delta_step_size = 0.01;
    long steps_back = 1;
    long splits = 2;
    long split_step_divisor = 10;
    long skip_elements = 0;
    long process_elements = 2147483647;
    double s_start = 0;
    double s_end = DBL_MAX;
    STRING include_name_pattern = NULL;
    long fiducialize = 0;
    long verbosity = 1;
&end
```

- `output` — The (incomplete) name of a file to which the momentum aperture results will be written. Recommended value: "%s.mmap".
- `x_initial`, `y_initial` — The initial x and y coordinate values for tracking. It is essential that `y_initial` be nonzero if one wants to see losses due to vertical resonances.
- `delta_negative_limit`, `delta_positive_limit` — Limiting values of scans in the negative and positive directions. Scanning begins at 0 and moves to to these values, but never beyond.
- `delta_step_size` — Initial size of steps in δ. This should be fairly large in order to save time.
- `steps_back` — Number of steps to back up after a particle is lost, relative to the last surviving δ, before continuing with a smaller step size. If this is set to zero, there is a risk of finding a too-large momentum aperture (a stable island).
- `splits` — Number of times to split the step size in order to refine the location of the maximum surviving momentum offsets. When a particle is lost, the algorithm steps back to a momentum offset where a particle survived, subdivides the step size, and continues searching.
- `split_step_divisor` — Factor by which to subdivide the step size for each split.
- `skip_elements` — Number of elements to skip before starting to compute momentum apertures.

- `process_elements` — Number of elements for which to compute momentum aperture.
- `s_start`, `s_end` — Limiting s coordinates of the elements from which tracking will start. The default values will exclude no elements.
- `include_name_pattern` — If given, tracking will start only at the entrance to elements that match the given wildcard pattern.
- `fiducialize` — If given, an initially on-energy particle is tracked before the momentum aperture search begins, in order to fiducialize the reference momentum. This is useful if there are synchrotron radiation losses or energy gain due to cavities in the system.
- `verbosity` — Larger values result in more detailed printouts as calculations proceed. Mostly for debugging.

The idea for this command is from M. Belgroune *et al.*, "Refined Tracking Procedure for the SOLEIL Energy Acceptance Calculation," Proceedings of PAC 2003, p 896, as implemented for TRACYII. In particular, the energy aperture as a function of position around the ring is determined by tracking. Starting at the beginning of the lattice and working downstream, particles are tracked starting from the exit of each selected element. The betatron coordinates are initially zero (or very small), while the momentum deviation is gradually increased until loss of the particle is observed. This defines the momentum aperture at that location.

In `elegant` version 19.0 and later, the algorithm is as follows. For simplicity in wording, we'll assume the momentum deviations are positive values, although the method is applied separately for negative values as well:

1. Start with $\delta = 0$, i.e., zero momentum offset.
2. Track a particle to see if it gets lost. If so, proceed to step 4.
3. Increase δ by step size $\Delta\delta$ and return to step 2.
4. If no splitting steps remain, proceed to the next step. Otherwise:
   (a) Change δ to $delta_s - s_b\Delta\delta.$, where δ_s is the largest δ for which the particle survived, and s_b is the `steps_back` parameter.
   (b) Divide the step size by `split_step_divisor` to get a new step size $\Delta\delta$.
   (c) Set $\delta = \delta + \Delta\delta$.
   (d) Decrement the "splits remaining" counter by 1.
   (e) Continue from step 2.
5. Stop. The momentum aperture is δ_s

This command can be used for both rings and transport lines. For rings it is most appropriate to have an rf cavity (i.e., an `RFCA` element) in the lattice. One should also include radiation loss using either of two methods:

1. `SREFFECTS` element, with `QEXCITATION=0`. To set up this element more easily, one can include a `twiss_output` command with `radiation_integrals=1`.
2. Use `CSBEND` and `KQUAD` elements with `SYNCH_RAD=1` and `ISR=0`.

When including radiation loss, one must be certain to set the phase of the rf cavity correctly. It is also a good idea to track for several synchrotron oscillation periods.

6.29 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.

6.30 optimization_constraint

- type: setup command.
- function: define a constraint for optimization.
- N.B.: This command is *not recommended*. It is *far* 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. See the discussion of the `selt` and `segt` macros in the manual entry to `optimization_setup`.

```
&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.

6.31 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;
    long disable = 0;
&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. The original values of all variables and covariable may be accessed via names like `<element-name>.<parameter-name>0`.
- `disable` — If nonzero, the covariable is ignored.

6.32 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 n_restarts = 0;
    long matrix_order = 1;
    STRING log_file = NULL;
    long output_sparsing_factor = 0;
    long balance_terms = 0;
    double restart_worst_term_factor = 1;
    long restart_worst_terms = 1;
    long verbose = 1;
    long balance_terms = 0;
    double simplex_divisor = 3;
    double simplex_pass_range_factor = 1;
    long include_simplex_1d_scans = 1;
    long start_from_simplex_vertex1 = 0;
    long restart_random_numbers = 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. In addition, initial values of variables and covariables are available in the form `<element-name>.<parameter-name>0`.

  Data from MARK elements with `FITPOINT=1` and from beam position monitors with `CO_FITPOINT=1` may be used via symbols of the form *elementName*#*occurenceNum*.*parameterName*. See the documentation for the `MARK`, `MONI`, `HMON`, and `VMON` elements for detailed discussion and listing.

  If response matrix calculation is requested, response matrix values are available in variables with names *Plane*`R`_*bpmName*#*occurence*_*corrName*#*occurence*.*corrParam*, where *Plane* is `H` (horizontal) or `V` (vertical) and *corrParam* is the parameter of the corrector used for changing the orbit (e.g., `HKICK` or `VKICK` for a `KICKER` element).

  If the twiss command was given with `output_at_each_step=1`, 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 with output_at_each_step=1 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.

The coupling integral and emittance ratio due to x-y coupling may be accessed using the symbols couplingIntegral and emittanceRatio. See section 3.1.4.4 of [19].

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.

There are several rpn functions that are useful in constructing a good optimization equation. These are "soft-edge" greater-than, less-than, and not-equal functions, which have the names segt, selt, and sene, respectively. The usage is as follows:

- *V1 V2 T* segt. Returns a nonzero value if and only if value *V1* is greater than *V2*. The returned value is $((V_1 - V_2)/T)^2$. Typically used to constraint a quantity from above. E.g., to limit the maximum horizontal beta function to 10m with a tolerance of $T = 0.1m$, one would use max.betax 10 .1 segt.
- *V1 V2 T* selt. Returns a nonzero value if and only if value *V1* is less than value *V2*. The returned value is $((V_1 - V_2)/T)^2$. Typically used to constrain a value from below. E.g., to limit a beta function to greater than 3 m with a tolerance of 0.1 m, one would use betax 3 .1 selt.
- *V1 V2 T* sene. Returns a nonzero value if and only if *V1* and *V2* differ by more than *tol*. If $V_1 > V_2$, returns $((V_1 - (V_2 + T))/T)^2$. If $V_2 > V_1$, returns $((V_2 - (V_1 + T))/T)^2$.

- mode — May be either "minimize" or "maximize".
- method — May be one of "simplex", "grid", "powell", "randomwalk", "randomsample", or "sample". Recommended methods are "simplex" and "randomwalk". The latter is very useful when the lattice is unstable or nearly so.
- 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_evaluations` — The number of allowed evaluations of the optimization function. If simplex optimization is used, this is the number of allowed evaluations per pass.

- `n_passes` — The number of optimization passes made to achieve convergence ("simplex" only). A pass ends (roughly) when the number of evaluations is completed or the function doesn't change within the tolerance. A new pass involves starting the optimization again using step sizes determined from the range of the simplex and the factor `simplex_pass_factor`.

- `n_restarts` — The number of complete restarts of the optimization (simplex only). This is an additional loop around the `n_passes` loop. The difference is that a restart involves using the optimized result but the original step sizes. It is highly recommended that this feature be used if convergence problems are seen.

- `restart_worst_term_factor`, `restart_worst_terms` — Often when there are convergence problems, it is because a few terms are causing difficulty. Convergence can often be obtained by *increasing* the weighting of these terms. If `restart_worst_term_factor` is positive, then `elegant` will multiply the weight of the `restart_worst_terms` largest terms by this factor at the beginning of a restart.

- `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.

- `balance_terms` — If nonzero, then all terms of the optimization expression have their weights adjusted so they make equal contributions to the penalty function. This can help prevent optimization of a single term at the expense of others. It is performed only for the initial value of the optimization function.

- `simplex_divisor` — The factor by which simplex step sizes are changed as the optimization algorithm searches for a valid initial simplex.

- `simplex_pass_range_factor` — When starting a new pass, the simplex optimizer takes the range over the previous simplex of each variable times this factor as the starting step size for that variable. This can be useful if the optimization brings the system close to an instability. In such a case, the simplex routine may have trouble constructing an initial simplex if the range of the variables is large. Setting this control to a value less than 1 may help.

- `include_simplex_1d_scans` — If nonzero, optimizer performs single-variable scans prior to starting simplex optimization. This is usually a good idea, but in some cases it will cause problems. For example, if your design is on the edge of being unstable, you may get some many errors from the initial steps that the single-variable optimizer can't continue. Disabling the single-variable scans will sometimes solve this.

- `start_from_simplex_vertex1` — If nonzero, optimizer uses the initial simplex vertex as the starting point for each new 1d scan. Otherwise, it uses the result of the previous scan.
- `restart_random_numbers` — If nonzero, the random number generators used by `elegant` are reset for each evaluation of the optimization function. This is valuable if one is optimizing tracking results that involve random processes (e.g., ISR or scattering).

6.33 optimization_term

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

```
&optimization_term
    STRING term = NULL;
    double weight = 1.0;
    STRING field_string = NULL;
    long field_initial_value = 0;
    long field_final_value = 0;
    long field_interval = 1;
&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.

  Please see the entry for `equation` under `optimization_setup` for details on designing optimization terms.

- `weight` — The weight to assign to this term. If zero, the term is ignored.
- `field_string`, `field_initial_value`, `field_final_value`, `field_interval` — These parameters are used to perform substitution of a series of values into the string given by `term`. This can be used to make an identical constraint at a number of instances of the same marker. For example, to constraint `Cx` to zero at instances 1, 3, 5, ..., 39, of marker `M1`, one could use

  ```
  &optimization_term
    term = "M1#@.Cx sqr",
    field_string = @,
    field_initial_value = 1, field_final_value = 39, field_interval = 2
  &end
  ```

6.34 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;
    long disable = 0;
&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).
- `disable` — If nonzero, the covariable is ignored.

6.35 print_dictionary

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

```
&print_dictionary
    STRING filename = NULL;
    long SDDS_form = 0;
&end
```

- `filename` — The name of a file to which the dictionary will be written. By default, the output is in LaTeXformat.
- `SDDS_form` — If non-zero, then the output is in SDDS format.

6.36 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.

6.37 rpn_load

- type: action/setup command.
- function: load data from SDDS file into RPN variables.

```
&rpn_load
    STRING tag = NULL;
    STRING filename = NULL;
    STRING match_column = NULL;
    STRING match_column_value = NULL;
    long matching_row_number = -1;
    STRING match_parameter = NULL;
    STRING match_parameter_value = NULL;
    long use_row = -1;
    long use_page = -1;
&end
```

This command is used to facilitate multi-stage optimization runs by allowing convenient loading of data from SDDS files into RPN variables. For example, one may match the final Twiss parameters of a lattice to the parameters stored in an SDDS file from a different run.

- `tag` — A string that will be pre-pended to the names of all the numerical columns in the file in order to create RPN variable names. E.g., if the input file was from the `twiss_output` command and `tag = tw1` was given, then RPN variables `tw1.betax`, `tw1.alphax`, etc. would be used.
- verb—filename— — The (incomplete) name of the SDDS file from which to read data. By default, data is taken from all columns from the last row of the last page of the file. This default behavior can be altered using one or more of the following parameters:
  - `match_column` — The name of a string column to use in selecting the row from which data will be taken.
  - `match_column_value` — The value that the column named by `match_column` must have to be selected from the file. By default, the last row with a matching value is used.
  - `matching_row_number` — If a nonnegative value is given, then the `matching_row_number`th matching row is selected (0 is the first row, 1 the second, etc). Otherwise, the last match row is used. Ignored if `match_column` is not given.
  - `match_parameter` — The name of a string parameter to use in selecting the page from which data will be taken.
  - `match_parameter_value` — The value that the parameter named by `match_parameter` must have to be selected from the file. By default, the last page with a matching value is used.
  - `use_row` — If nonnegative, specifies the row number to use, starting at 0 for the first row. Ignored if `match_column` is given.
  - `use_page` — If nonnegative, specifies the page number to use, starting at 1 for the first page. Takes precedence over `\match_parameter` if both are given.

6.38 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.

6.39 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 final_pass = 0;
    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;
    double p_central_mev = 0.0;
    long always_change_p0 = 0;
    STRING expand_for = NULL;
    long tracking_updates = 1;
    long echo_lattice = 0;
    STRING search_path = NULL;
    long element_divisions = 0;
    long load_balancing_on = 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". N.B.: confusion sometimes occurs about some of the quantities related to the `s` coordinate in this file. Please see Section 4 above.
- `final` — The (incomplete) name of an SDDS file into which final beam and transport parameters will be written. Recommended value: "%s.fin". N.B.: confusion sometimes occurs about some of the quantities related to the `s` coordinate in this file. Please see Section 4 above.
- `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. N.B.: this file does not contain any non-numerical parameters of the lattice. Hence, it is not a complete description of the settings of the lattice.
- `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 s coordinate should wrap-around or increase monotonically in multipass simulations. If zero, then the centroid and sigma data is computed for each turn with the s coordinate increasing continuously.
- `final_pass` — A flag indicating whether the centroid and sigma output should be computed only from the data from the final pass. By default, the statistics include data from all passes.
- `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. This is $\beta\gamma$.
- `p_central_mev` — Central momentum of the beamline in MeV/c, about which expansions are done. Ignored if `p_central` is nonzero.
- `always_change_p0` — If nonzero, then `elegant` will match the reference momentum to the beam momentum after each element. For example, in a beamline with radiation losses, one might want to adjust downstream magnets to match the energy of the incoming beam.
- `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.
- `load_balancing_on` — If non-zero, load-balancing is performed for parallel mode. This can result in non-deterministic results if the load-balancing is different on two otherwise identical runs. Load-balancing variations may occur in heterogeneous clusters, clusters with multiple users, or for other reasons. In such situations, turning off load balancing can be useful if, for example, one is performing parameter scans and wishes to eliminate spurious sources of variation.

6.40 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.

6.41 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".

6.42 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. If this file is to be generated by the user, the expected units string in the column definitions should be "m", "s", and "mbenc" for meters, seconds and the dimensionless momentum, respectively.
  - `spiffe`: (r, z, pr, pz, pphi, t), where r and z are in meters, pr = $\beta_r\gamma$, pz = $\beta_z\gamma$, $p_\phi = \omega r\gamma/c$, and t is in seconds. If this file is to be generated by the user use the units strings described above.
- `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`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.

6.43 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.

6.44 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 "%s.slan".
- `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.

6.45 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 "

6.46 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.

6.47 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 disable = 0;
        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.
- `disable` — If nonzero, the command is ignored.
- `clear` — If nonzero, all prior transmutation specifications are deleted.

6.48 twiss_analysis

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

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

- `match_name` — Optional wildcard string to match to element names for selection of elements to inculde in the analysis.
- `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`, `etay`, `alphax`, `alphay`, `etaxp`, and `etayp`. 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.

6.49 twiss_output

- type: action/setup command.
- function: compute and output uncoupled 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;
    long higher_order_chromaticity = 0;
    long higher_order_chromaticity_points = 5;
    double higher_order_chromaticity_range = 4e-4;
    double chromatic_tune_spread_half_range = 0;
    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;
    long cavities_are_drifts_if_matched = 1;
&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. *N.B.: Radiation integral computation is not correct for systems with vertical bending, nor does it take into account coupling.*
- `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 and/or momentum errors. However, for on-momentum conditions with zero orbit, it is much faster to use `concat_order=1`.
- `higher_order_chromaticity` — If nonzero, requests computation of the second- and third-order chromaticity. To obtain reliable values, the user should use `concat_order=3` in this namelist and the highest available order for all beamline elements. `elegant` computes the higher-order chromaticity by finding the trace of off-momentum matrices obtained by concantenation of the matrix for `higher_order_chromaticity_points` values of δ over the full range `higher_order_chromaticity_range`.
- `chromatic_tune_spread_half_range` — Half range of δ for which the chromatic tune spread is computed. The results are available in for optimization and in the twiss output file under the names `nuxChromUpper`, `nuxChromLower`, and similarly for the y plane. This computation uses the chromaticities.
- `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.
- `cavities_are_drifts_if_matched` — By default, if `matched=1`, `elegant` treats rf cavities as drift spaces, allowing the user to have a cavity in the ring definition without it affecting the lattice functions. By setting `cavities_are_drifts_if_matched=0`, one can force `elegant` to use the actual matrix for the rf cavity. The differences between the results are generally small, but the default behavior disagrees with the results of `moments_output`.

The output file from this command contains the following columns, giving values of quantities at the exit of each element, unless otherwise noted.

- `s` — The arc length.
- `ElementName` — The name of the element.
- `ElementType` — The type name of the element.
- `betax` and `betay` — The horizontal and vertical beta functions.

- `alphax` and `alphay` — The horizontal and vertical alpha functions, where $\alpha = -\frac{d\beta}{2ds}$.
- `psix` and `psiy` — The horizontal and vertical betatron phase advance in radians.
- `etax` and `etay` — The horizontal and vertical dispersion functions.
- `etaxp` and `etayp` — The slopes of the horizontal and vertical dispersion functions.
- `xAperture` and `yAperture` — The horizontal and vertical apertures. If undefined, will have a value of 10m. If the beam trajectory is non-zero, then the aperture will be changed (usually reduced) accordingly. Hence, these are best understood as the `effective` apertures. They are used in determining the horizontal and vertical acceptance parameters, `Ax` and `Ay`.
- `pCentral0` — The central momentum ($\beta\gamma$) at the **entrance** to the element.
- `dI`n — Contribution to radiation integral `I`n. Radiation integrals take account of horizontal bending only.

The output file contains the following parameters. Note that chromatic quantities depend on the order settings of the individual elements, the default order (in `run_setup`), and the concatenation order given in the `twiss_output` command. These quantities pertain to the end of the lattice or to the lattice as a whole.

- `nux` and `nuy` — The horizontal and vertical tunes.
- `dnux/dp` and `dnuy/dp` — The horizontal and vertical chromaticities, defined as $d\nu/d\delta$.
- `dnux/dp2` and `dnuy/dp2` — The horizontal and vertical 2nd-order chromaticities, defined as $d^2\nu/d\delta^2$. Will be zero if `higher_order_chromaticity` is zero.
- `dnux/dp3` and `dnuy/dp3` — The horizontal and vertical 3rd-order chromaticities, defined as $d^3\nu/d\delta^3$. Will be zero if `higher_order_chromaticity` is zero.
- `Ax` and `Ay` — The horizontal and vertical acceptance. These will be zero if no apertures are defined.
- `dbetax/dp` and `dbetay/dp` — Chromatic derivatives of the horizontal and vertical beta functions, defined as $\frac{d\beta}{d\delta}$.
- `etax2`, `etax3`, `etay2`, `etay3` — Higher order dispersion in the horizontal and vertical planes. For example, for the horizontal plane, the closed orbit at the end of the lattice depends on δ according to $x = \eta_x\delta + \eta_{x2}\delta^2 + \eta_{x3}\delta^3$. This differs from the chromaticity expansion, which is given in terms of successive derivatives of $\nu(\delta)$.
- `dnux/dAx`, `dnux/dAy`, `dnuy/dAx`, `dnuy/dAy` — Tune shifts with amplitude, where amplitude is defined as $A_q = (1 + \alpha_q)q^2/\beta_q$, with $q = x$ or $q = y$. These will be zero unless the `tune_shift_with_amplitude` command is given.
- `alphac`, `alphac2` — First- and second-order momentum compaction. The path length is $s = s_o + \alpha_c\delta + \alpha_{c2}\delta^2$.
- `couplingIntegral`, `couplingDelta`, and `emittanceRatio` — These quantities are defined in section 3.1.4.4 of [19]. The computations include tilted quadrupoles, vertical orbit in sextupoles, vertical sextupole displacement, and solenoids. Note that the emittance ratio *does not* include the effect of vertical dispersion.

- `I`n — The nth radiation integral.
- `taux`, `tauy`, `taudelta` — Radiation damping times for x, y, and δ.
- `Jx`, `Jy`, `Jdelta` — Damping partition factors for x, y, and δ.
- `ex0`, `enx0` — Horizontal equilibrium geometric and normalized emittances.
- `Sdelta0` — Equilibrium fractional rms energy spread.
- `U0` — Energy loss per turn.

N.B.: the higher-order dispersion and higher-order chromaticity are computed using the concatenated third-order matrix. However, `elegant` only has third-order matrices for three elements: alpha magnets, quadrupoles, and sextupoles. This may be acceptable if any dipoles (for example) have large bending radius. Users who are concerned about these effects should perform off-energy tracking using canonical elements (i.e., CSBEND, KQUAD, KSEXT, and MULT), which include energy dependence to all orders.

Also, note that by default all elements are computed to second order only. You must change the `default\_order` parameter on `run\_setup` to `3` in order to use the third-order matrices for alpha magnets, quadrupoles, and sextupoles. You may also use the `ORDER` parameter on individual element definitions.

6.50 track

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

```
&track
    long center_on_orbit = 0;
    long center_momentum_also = 1;
    long offset_by_orbit = 0;
    long offset_momentum_also = 1;
    long soft_failure = 1;
    long use_linear_chromatic_matrix = 0;
    long longitudinal_ring_only = 0;
    long stop_tracking_particle_limit = -1;
&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.
- `offset_by_orbit` — A flag indicating whether to offset the transverse beam coordinates by the closed orbit before tracking. Similar to `center_on_orbit`, but the initial centroids of the beam are preserved. The beam is simply displaced by the closed orbit rather than being centered on it.
- `offset_momentum_also` — A flag indicating whether to also offset the beam momentum to the momentum of the closed orbit. If the `start_from_centroid` or `start_from_dp_centroid` parameters are used on the `closed_orbit` command, this flag should be set to 0; otherwise, one will offset the beam central momentum by its own value.
- `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.
- `stop_tracking_particle_limit` — If a non-negative is given, then `elegant` will stop tracking when the number of particles falls below the given value. It will be as if all the particles were lost.

6.51 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 or by a matrix method. 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).

  The quantities computed are $\frac{\partial}{\partial A_x^n \partial A_y^m}\nu_p$, where $n \geq 0$ and $m \geq 0$ are integers and p is x or y. $A_q = (1+\alpha_q)q^2/\beta_q$, with $q = x$ or $q = y$.

```
&tune_shift_with_amplitude
    long turns = 2048;
    double x0 = 1e-6;
    double y0 = 1e-6;
    double x1 = 3e-4;
    double y1 = 3e-4;
    long grid_size = 6;
    long lines_only = 0;
    long sparse_grid = 0;
    long spread_only = 0;
    double nux_roi_width = 0.02;
    double nuy_roi_width = 0.02;
    double scale_down_factor = 2;
    double scale_up_factor = 1.05;
    double scale_down_limit = 0.01;
    double scale_up_limit = 1e-4;
    long scaling_iterations = 10;
    long use_concatenation = 0;
    long verbose = 0;
    long order = 2;
    STRING tune_output = NULL;
&end
```

- `turns` — The number of turns to track. If zero, then the concatenated matrix is used instead of tracking, and all other parameters of this command are irrelevant. The matrix method 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.

- `grid_size` — Size of the grid of points in x and y.
- `lines_only` — If nonzero, then instead of a full set of `grid_size`^{2} particles, only two lines of particles with $x = 0$ and/or $y = 0$ are tracked. In this case, no $A_x^i * A_y^j$ terms are computed (except for $i = 0$ or $j = 0$). However, in addition to being faster, the results may be more reliable, e.g., $\partial\nu_x/\partial A_y = \partial\nu_y/\partial A_x$ may be more closely satisfied.
- `sparse_grid` — If nonzero, then instead of a full set of `grid_size`^{2} particles, a sparse grid of particles is tracked. Will save time at some expense in accuracy.
- `spread_only` — Compute the tune spread only and don't bother with the tune shift coefficients. These tune spreads can be optimized and appear in the twiss output file under the names `nuxTswaLower`, `nuxTswaUpper`, and similarly for the y plane. This is the recommended way to reduce tune shift with amplitude, as the tune spread is more reliable than the coefficients of the expansion. (Particles that get lost are automatically ignored in both types of computations.)
- `nux_roi_width`, `nuy_roi_width` — Widths of the region of interest for x and y tunes. As the grid is filled in, `elegant` finds the tune for each tracked particle on the grid. Successive tune values are looked for in the region of the given width around the previous tune value. This prevents jumping from the main tune peak to another peak, which can happen when the tune spectrum has many lines.
- `scale_down_factor`, `scale_up_factor`, `scale_down_limit`, `scale_up_limit`, `scaling_iterations` — These control automatic scaling of the amplitudes. If `elegant` sees a tune shift larger than `scale_down_limit` it will decrease `x0` (or `y0`) by the factor `scale_down_factor`. If `elegant` sees a tune shift smaller than `scale_up_limit` it will increase `x0` (or `y0`) by the factor `scale_up_factor`. Suggestion: if you find yourself playing with these values and the initial amplitudes in order to get reliable TSWA coefficients, try just using the tune spread.
- `verbose` — If nonzero, information about the progress of the algorithm is printed to the screen.
- `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.

6.52 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.

7 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. The following is a brief description of each program. Full descriptions are available on subsequent pages.

- `csrImpedance` — Computes the shielded steady-state CSR impedance for a dipole magnet. The output can be used immediately with `elegant`'s `ZLONGIT` element. (Program by Y. Wang, H. Shang, and M. Borland.)

- `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. (Program by R. Soliday and M. Borland.)

- `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 and M. Borland.)

- `ibsEmittance` — Computes local intra-beam scattering rates for both storage ring and linac. Also computes the equlibrium 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, M. Borland, and A. Xiao)

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

- `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. (Program by M. Borland.)

- `sddsbrightness` — Uses twiss parameter output or data from `sddsanalyzebeam` to compute undulator brightness curves. (Program by H. Shang, R. Dejus, M. Borland, X. Jiao.)

- `sddsemitproc` — 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`. (Program by M. Borland.)

- `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. (Program by M. Borland.)

- `sddsurgent` — Uses algorithms from the programs US (by R. Dejus) and URGENT (by R. Walker) for computation of undulator radiation properties, including power density and intensity distributions. (Program by H. Shang, R. Dejus, M. Borland, X. Jiao.)

- `sddsrandmult` — Simulates the effect of random mechanical errors in a quadrupole or sextupole, generating multipole error data that can be used with `elegant`'s `KQUAD` and `KSEXT` elements. (Program by M. Borland.)
- `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. (Program by M. Borland and H. Shang.)
- `touschekLifetime` — This program calculates Touschek lifetime using A. Piwinski's formula. Input files are generated from "twiss_output" and "momentum_aperture". (Program by A. Xiao.)

7.1 csrImpedance

- **description:** Computes the steady-state CSR impedance with shielding by parallel plates. The computed impedance is for a dipole magnet that bends the beam in a complete circle.

- **synopsis:**

  `csrImpedance` *`outputFile`* `| -pipe[=out] -height=`*`valueInMeters`*
  `-radius=`*`valueInMeters`* `-frequencyLimit=maximum=`*`valueInHz`*`[,minimum=`*`valueInHz`*`]`
  `-n=`*`integer`* `[-filter=`*`cutoff1,cutoff2`*`]`

- **files:**

  – `outputFile` — SDDS file containing computed impedance. May be used directly with `elegant`'s `ZLONGIT` element. as columns, along with analysis and conditions as parameters.

- **switches:**

  – `-height` — The full height of the vacuum chamber, in meters.

  – `-radius` — The radius of the bending magnet, in meters.

  – `frequencyLimit` — Allows specifying the upper frequency limit (required), as well as the lower frequency limit, for the computed impedance. `elegant` will not accept the data if the lower limit is not 0. If the rms bunch length is σ_t, then it is suggested to have the maximum frequency much greater than $1/(2\pi\sigma_t)$.

  – `-n` — Allows specifying the number of data points to be computed. The number of points computed is $2^n + 1$, which is required by `elegant`. A reasonable value is $n = 10$ to $n = 14$.

  – `-filter` — Allows specifying the starting and ending frequency for a simple low-pass filter. The frequencies are given as fractions of the maximum frequency. The filter ramps linearly from 1 to 0 between the two cutoff values. If, for example, the cutoff is 0.2, then the highest frequency in the impedance corresponds to a wavelength of 10 bins (2/0.2) in `elegant`.

- **authors:** Y. Wang, H. Shang, ANL/APS. Based on a simplified form[26] of Warnock's [25] formula.

7.2 haissinski

- **description:** `haissinski` solves the Haissinski equation for the bunch steady-state longitudinal distribution in the presence of various impedances.

- **synopsis:**

```
haissinski twissFile resultsFile
-wakeFunction=file,tColumn=name,wColumn=name |
-model=[L=Henry|Zn=Ohms],R=Ohm -charge=C | -particles=value |
-bunchCurrent=A -steps=numberOfChargeSteps -outputLastStepOnly
-RF=Voltage=V,harmonic=value | -length=s
-harmonicCavity=Voltage=V,factor=harmonicFactor -superPeriods=number
-energy=GeV -integrationParameters=deltaTime=s,points=number,startTime=s,
iterations=number,fraction=value,tolerance=value
```

- **files:**
  - *twissFile* — Twiss output file from `elegant`, including radiation integral calculations.
  - `resultsFile` — SDDS file containing computed bunch longitudinal distributions as columns, along with analysis and conditions as parameters.

- **switches:**
  - `-wakeFunction=`*file*`,tColumn=`*name*`,wColumn=`*name* — Optionally specifies the impedance as a Greens function using values in an SDDS file. The time points must be equi-spaced.
  - `-model=[L=`*Henry*`|Zn=`*Ohms*`],R=`*Ohm* — Optionally specifies the impedance as an inductor `L` or broad-band value `Zn`, along with a resistance `R`.
  - `-charge=`*C* `| -particles=`*value* `| -bunchCurrent=`*A* — Various ways to specify the charge in each bunch.
  - `-steps=`*numberOfChargeSteps* — Number of values of bunch charge to compute up to the value specified with on the just-described options. Using more values can help convergence, as the result of each prior step is used as the starting point for the new step.
  - `-outputLastStepOnly` — Requests output for the last charge step (full charge) only.
  - `-RF=Voltage=`*V*`,harmonic=`*value* `| -length=`*s* — Two ways to specify the nominal bunch length.
  - `-harmonicCavity=Voltage=`*V*`,factor=`*harmonicFactor* — Specifies a harmonic cavity voltage and the ratio of the harmonic cavity frequency to the main frequency.
  - `-superPeriods=`*number* — Number of superiods of the lattice specified in *twissfile* to simulate. If one has an N cell ring but only gives 1 cell in the input, this value should be N. If one gives the whole ring, this value should be 1.
  - `-energy=`*GeV* — Beam energy. If not given, the value in the *twissfile* is used.

- `-integrationParameters=deltaTime=`*`s`*`,points=`*`number`*`,startTime=`*`s`*`,iterations=`*`number`*`,fraction=`*`value`*`,tolerance=`*`value` — Integration parameters, which must be set. `deltaTime` is the time interval for wake function and charge density evaluation. `points` is the number of time points, while `startTime` is the time (relative to synchronous phase) at which the time region starts. These values must be set by the user based on knowledge of the likely bunch length. For the others, we suggest 1000 iterations, a fraction of 0.01, and a tolerance of 10^{-4}.

- **authors:** L. Emery, M. Borland, ANL/APS.

7.3 ibsEmittance

- **description:** `ibsEmittance` computes growth rates and equilibrium emittances for electron rings due to intrabeam scattering (IBS). It will also integrate the growth rates to show the time evolution of the emittances. The IBS algorithm is based on the Bjorken and Mtingwa's [15] formula, and with an extension of including vertical dispersion. The program can also estimate IBS growth rates for and transport line or linac beam, provided special attention paid to the beam's energy change (splitting RF cavities as needed).

- **examples:** This example computes the IBS equilibrium parameters and the contributions to the growth rates (at equilibrium) vs position in the APS lattice.

```
ibsEmittance aps.twi aps.ibs -charge=5 -coupling=0.02
-rf=voltage=9,harmonic=1296
```

- **synopsis:**

```
ibsEmittance twissFile resultsFile -charge=<nC>|-particles=<value>
-coupling=<value> [-emitxInput=<value>] [-deltaInput=<value>]
[-superperiods=<value>] [-isRing=1|0]
{-RF=Voltage=<MV>,harmonic=<value>|-length=<mm>} [-energy=<MeV>]
[{-growthRatesOnly | -integrate=turns=<number>[,stepSize=<number>]}]
[-noWarning]
```

- **files:** *twissFile* is a twiss parameter file from the `twiss_output` command of `elegant`. You must use the `radiation_integrals` flag in `twiss_output`.

- **switches:**
  - **-charge**, **-particles** — Give the charge (in nanocoulombs) or the number of electrons.
  - **-coupling** — Give the emittance coupling ratio, ϵ_y/ϵ_x.
  - **-emitxInput** — Give the initial x emittance in meters. If not specified, the value from the parameter `ex0` in *twissFile* is used.
  - **-deltaInput** — Give the initial rms fractional momentum spread. If not specified, the value from the parameter `Sdelta0` in *twissFile* is used.
  - **-superperiods=*value*** — If given, the number of superperiods in the lattice. *twissFile* is taken to pertain to a single sector.
  - **-isRing** — Specify the calculation is done for stored beam (isRing=1, default) or transport line/linac beam (isRing=0). When isRing is set to 0, the energy scaling and integration calculation will be disabled.
  - **-RF=Voltage=*MV*,harmonic=*value*** — Specify rf voltage and harmonic number.
  - **-length=*mm*** — Specify the rms bunch length.
  - **-energy=*MeV*** — Specify the beam energy. By default, this is taken from the `pCentral` parameter in *twissFile*.

- **-growthRatesOnly** — If given, only the initial growth rates are computed. Equilibrium emittance values are not computed. *resultsFile* will contain columns of initial growth rate contributions from individual elements. Without this option, *resultsFile* would normally contain columns of growth rate contributions at equilibrium.
- **-integrate=turns=*number*[,stepSize=*number*]** — If given, then *resultsFile* contains the result of integrating the differential equations for the emittances for the given number of turns and not the contributions of individual elements of growth rates. The step size is the number of turns for each integration step, and can be adjusted to get faster results. The options -growthRatesOnly and -integrate are not compatible.
- **-noWarning** — Removes warning messages.

- **author:** A. Xiao, L. Emery, M. Borland, ANL/APS.

7.4 madto

- **description:** `madto` translates an `elegant`-style (or a MAD file, with some restrictions) into formats accepted by other programs, such as COSY, 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)*] |
-cosy=*quadAper(mm),sextAper(mm),p(MeV/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:**
  - **-cosy** — Provide data for the program COSY INFINITY. This can take a little while as the program must figure out the Enge coefficients that correspond to the `FINT` and `HGAP` values for all the dipoles. The user should test the output carefully.
  - **-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.

7.5 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] [-correctedOnly]
```

- **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.
  - `correctedOnly` — If given, only the corrected twiss parameters are computed and output. These parameters are the correct ones to match a beamline to, since they have the dispersive and mono-energetic terms properly separated.

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

7.6 sddsbrightness

- **description:** `sddsbrightness` computes undulator brightness curves using Twiss parameter data from `elegant` or `sddsanalyzebeam`. Several calculation methods are available.

- **examples:**

```
sddsbrightness run.twi run.bri -harmonics=3
-Krange=start=0.2,end=2.2,points=100
-current=0.1 -totalLength=2.4 -periodLength=0.027 -coupling=0.01
```

```
sddsanalyzebeam run.out -pipe=out -correctedOnly
| sddsbrightness -pipe=in run.bri -harmonics=3
-Krange=start=0.2,end=2.2,points=100
-current=0.1 -totalLength=2.4 -periodLength=0.027 -coupling=0.01
```

- **synopsis:**

```
sddsbrightness [-pipe=[input][,output]] [twissFile] [SDDSoutputfile]
-harmonics=integer -Krange=start=value,end=value,points=integer
-current=Amps -totalLength=meters -periodLength=meters
[-emittanceRatio=value | -coupling=value] [-noSpectralBroadening]
[-method=string,device=string,neks=value]]
```

- **files:**
  - *twissFile* — A Twiss output file from `elegant`, with radiation integral calculations included, or an output from `sddsanalyzebeam`. In the latter case, the `-correctedOnly` option should be used.
  - *SDDSoutputFile* — Contains the brightness data in column form. For each requested harmonic i, there are columns `photonEnergy`i, `wavelength`i, and `Brightness`i.

- **switches:**
  - `pipe` — The standard SDDS Toolkit pipe option.
  - `harmonics` — The number of harmonics to compute.
  - `Krange=start=`*value*`,end=`*value*`,points=`*integer* — The range of the K parameter for the undulator and the number of points to compute on that range.
  - `-current=`*Amps* — The current in amperes. If one gives the average current, one gets the average brightness.
  - `-totalLength=`*meters* — The total length of the undulator, in meters.
  - `-periodLength=`*meters* — The period length of the undulator, in meters.
  - `-emittanceRatio=`*value* `| -coupling=`*value* — In the case of a twiss output file from `elegant`, which does not contain the vertical emittance, one must supply one of these options. If `-emittanceRatio=`*R* is given, $\epsilon_y = \epsilon_0 * R$ and $\epsilon_x = \epsilon_0$. If `-coupling=`*k* is given, $\epsilon_x = \epsilon_0/(1+k)$ and $\epsilon_y = k * \epsilon_x$. ϵ_0 is the equilibrium emittance from the twiss output of `elegant`.

    In the case of twiss output from `sddsanalyzebeam`, both emittances are present and these options are ignored.

- `-method=`*`string`*`,device=`*`string`*`,neks=`*`value`*`]` — Choose which method to use for brightness calculations. Options are
  * `borland` — M. Borland's approximation method. Fast, but not as reliable as others.
  * `dejus` — R. Dejus' non-zero emittance, infinite-N+convolution method. This is the default.
  * `walkerinfinite` — R. Walker's method. Dejus' method is derived from this method.
  * `walkerfinite` — R. Walker's method using finite N without convolution. This is quite slow.

  The `device` qualifier may be `planar` or `helical`. `neks` is used to change the number of points used for finding the peak of the distribution.

- **authors:** M. Borland, H. Shang, R. Dejus (ANL).

7.7 sddsemitproc

- **description:**

  sddsemitproc analyzes quadrupole scan emittance measurement data. It accepts a file containing the transport matrix for each data point and measured beam sizes. Because sddsemitproc 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.

  sddsemitproc 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.

  The beam parameters computed by sddsemitproc pertain to the beginning of whatever system is simulated in elegant.

- **examples:**

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

- **synopsis:**

```
sddsemitproc [inputfile] [outputfile] [-pipe=[input][,output]]
[-sigmaData=xName,yName] [-errorData=xName,yName |
-errorLevel=valueInm,[{gaussian,nSigmas | uniform}]] [-nErrorSets=number]
[-seed=integer] [-limitMode=resolution | zero[,reject]
[-deviationLimit=xLevelm,yLevelm] [-resolution=xResolutionm,yResolutionm]
[-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.

  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; otherwise, they default to Sx and Sy. 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.

- **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`.
  - `-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.
  - `-errorData=`*`xName,yName`* — May be used to supply the names of columns in the input file that contain the error level for each measurement. This is an option instead of using `-errorLevel`, which allows varying the measurement error for each point.
  - `-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.
  - `-verbosity=`*`level`* — Higher values of *level* result in more informational printouts as the program runs.

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

7.8 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.

7.9 sddsrandmult

- **description:** `sddsrandmult` computes the multipole errors in a quadrupole or sextupole due to various construction errors. The program is based on the analysis of Halbach[16], with which I'll assume the reader is familiar. Instead of separately evaluating the effect of certain types of mechanical errors, it allows one to simulate several types of errors in order to get statistical distributions for the multipole perturbations.

- **examples:**

```
sddsrandmult quadpert.in
```

- **synopsis:**

  `sddsrandmult` *`inputFile`*

- **usage:**

  inputFile is a text file containing a series of namelist commands specifying the parameters of a quadrupole or sextupole, the type and amplitude of the errors to include, and the filenames for output. Each namelist command results in a complete computation and generation of output files.

  The namelist command is `perturbations`. It has the following fields:

  - `type` — A string value, either "quadrupole" (default) or "sextupole".
  - `name` — An optional string value giving the name of the element. This is used in preparing data for `elegant`.
  - `SDDS_output` — An required string value giving the name of an SDDS file to which data for each seed will be written. This file can be used to compute statistics or perform histograms.
  - `elegant_output` — An optional string value giving the name of a text file to which `elegant` commands and element definitions will be written. Note that this file is a mixture of commands and element definitions. As such, the user must manually edit the file and place the appropriate parts in the lattice file and the command file.
  - `kmult_output` — An optional string value giving the name of an SDDS file to which data will be written in the format accepted by the `RANDOM_MULTIPOLES` feature of the `KQUAD` and `KSEXT` elements. *This is the recommended data to use with* `elegant`.
  - `effective_length` — The effective length of the magnet, in meters.
  - `bore_radius` — The bore radius of the magnet, in meters.
  - `dx_pole` — The rms error, in meters, to be imparted to the horizontal position of each pole.
  - `dy_pole` — The rms error, in meters, to be imparted to the vertical position of each pole.
  - `dradius` — The rms error, in meters, in the bore radius.

- `dx_split` — The rms error, in meters, to be imparted to the horizontal distance between the left and right sides of the magnet.
- `dy_split` — The rms error, in meters, to be imparted to the vertical distance between the top and bottom halves of the magnet.
- `dphi_halves` — The rms error, in radians, to be imparted to the relative rotation of the top and bottom halves of the magnet.
- `n_cases` — The number of cases to simulate (default is 1000).
- `n_harm` — The number of harmonics to simulate. The default is 0, which results in computing all the harmonics for which Halbach indicates his treatment applies.
- `random_number_seed` — The initial seed for the random number generator. Should be a large integer.

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

7.10 sddsurgent

- **description:** `sddsurgent` uses algorithms from the program US (by R. Dejus) and URGENT (by R. Walker) for computation of undulator radiation properties, including power density and intensity distributions.

- **examples:** Take particle data from a tracking run and compute the power density using a 1 mm by 1 mm pinhole for a 72-period, 3.3-cm-period undulator set for a 5 keV first harmonic.

```
sddsanalyzebeam run.out -pipe=out -correctedOnly
| sddsurgent -pipe=in power.sdds -electronbeam=current=0.025
-calc=method=dejus,mode=powerDensity -us
-pinhole=dist=30,xsize=1,ysize=1,xnum=100,ynum=100
-undulator=period=0.033,number=72,energy=5e3
```

- **synopsis:**

sddsurgent *inputFile outputFile*
[-calculation=mode=*modeString*,method=*methodString*,harmonics=*integer*]
[-undulator=period=*meters*,numberOfPeriods=*integer*,
kx=*value*,ky=*value*,phase=*value*,energy=*eV*]
[-electronBeam=current=*Amp*,energy=*GeV*,spread=*fraction*,
xsigma=*mm*,ysigma=*mm*,xprime=*mrad*,yprime=*mrad*,nsigma=*number*]
[-pinhole=distance=*m*,xposition=*value*,yposition=*value*,
xsize=*value*,ysize=*value*,xnumber=*integer*,ynumber=*integer*]
[-alpha=steps=*integer*,delta=*value*] [-omega=steps=*integer*,delta=*value*]
[-nphi=*integer*] [-us] [-photonEnergy=maximum=*eV*,minimum=*eV*,points=*number*]
[-nowarnings] [-coupling=*value* | -emittanceRatio=*value*]

- **files:**
  - *inputFile* — A Twiss output file from `elegant`, with radiation integral calculations included, or an output from `sddsanalyzebeam`. In the latter case, the `-correctedOnly` option should be used with `sddsanalyzebeam`.
  - *outputFile* — Contains the output data, which varies depending on the calculation mode. Use `sddsquery` to view the file contents.

- **switches:**
  - `pipe` — The standard SDDS Toolkit pipe option.
  - `-calculation=mode=`*modeString*`,method=`*methodString*`,harmonics=`*integer*— Choose which calculation to perform and what method to us, as well as the number of undulator harmonics to compute. Values for *modeString* are
    - `1 | fluxDistribution`: Angular/spatial flux density distribution.
    - `2 | fluxSpectrum`: Angular/spatial flux density spectrum
    - `3 | brightness | brilliance`: On-axis brilliance spectrum

* 4 | `pinholeSpectrum`: Flux spectrum through a pinhole
* 5 | `integratedSpectrum`: Flux spectrum integrated over all angles
* 6 | `powerDensity`: Power density and integrated power

Values for *methodString* are

* 1: Non-zero emittance; finite-N.
* 2: Non-zero emittance; infinite-N.
* 3 | `WalkerFinite`: Zero emittance; finite-N.
* 4 | `Dejus`: Non-zero emittance; infinite-N + convolution (Dejus, with `-us` only).
* 14 | `WalkerInfinite`: Non-zero emittance; infinite-N + convolution (Walker, with -tt us only).

– `-emittanceRatio=`*`value`* | `-coupling=`*`value`* — In the case of a twiss output file from `elegant`, which does not contain the vertical emittance, one must supply one of these options. If `-emittanceRatio=`*`R`* is given, $\epsilon_y = \epsilon_0 * R$ and $\epsilon_x = \epsilon_0$. If `-coupling=`*`k`* is given, $\epsilon_x = \epsilon_0/(1+k)$ and $\epsilon_y = k * \epsilon_x$. ϵ_0 is the equilibrium emittance from the twiss output of `elegant`.

  In the case of twiss output from `sddsanalyzebeam`, both emittances are present and these options are ignored.

– `undulator=period=`*`meters`*`,numberOfPeriods=`*`integer`*`,`
  `kx=`*`value`*`,ky=`*`value`*`,phase=`*`value`*`,energy=`*`eV`*] — Specify undulator parameters. If energy (of first-harmonic photons) is given, `kx=0` and `ky` is computed, corresponding to a horizontally deflecting undulator. `phase` specifies the phase difference in degrees for a canted undulator.

– `-electronBeam=current=`*`Amps`*`,energy=`*`GeV`*`,spread=`*`fraction`*`,`
  `xsigma=`*`mm`*`,ysigma=`*`mm`*`,xprime=`*`mrad`*`,yprime=`*`mrad`*`,nsigma=`*`number`* specifies electron beam parameters. Only the current is needed, as other data will be drawn from the input file.

  * `current` — electron beam current in A. (default is 0.1A).
  * `energy` — electron energy in Gev. (default is 7.0Gev).
  * `spread` — electron energy spread.
  * `xsigma` — horizontal RMS beam size (mm)
  * `ysigma` — vertical RMS beam size (mm)
  * `xprime` — horizontal RMS divergence (mrad)
  * `yprime` — vertical RMS divergence (mrad)
  * `nsigma` — no. of standard deviations of electron beam dimensions (size and divergence) to be included.

– `-pinhole=distance=`*`m`*`,xposition=`*`value`*`,yposition=`*`value`*`,`
  `xsize=`*`value`*`,ysize=`*`value`*`,xnumber=`*`integer`*`,ynumber=`*`integer`* — Specifies pinhole parameters. Pinhole parameters are not needed for computing on-axis brilliance (i.e., mode=3).

  * `distance` — distance from the source (m) (distance=0.0 gives angular flux).
  * `xposition` — X-coordinate for center of pinhole (mm) or (mrad for distance=0)
  * `yposition` — Y-coordinate for center of pinhole (mm) or (mrad for distance=0)
  * `xsize` — X-size of pinhole (full width) (mm) or (mrad for distance=0)
  * `ysize` — y-size of pinhole (full width) (mm) or (mrad for distance=0)

    * `xnumber` — Number of subdivisions of pinhole in X (max 500)
    * `ynumber` — Number of subdivisions of pinhole in Y (max 500)
  - `nphi=`*`number`* — Specifies number of steps in phi between 0 and $\pi/2$. Must be less than 100. used in (calculation mode=1,2,3,4,5 calculation method=1,2).
  - `alpha=steps=`*`integer`*`,delta=`*`value`* — Specifies the number of steps in angle alpha (gamma*theta) (¡100). Delta specifies range of angles in alpha^2 to be used, in units of the angular equivalent to 1/N. Used in (mode=1, method=1) and method=3.
  - `omegasteps=`*`integer`*`,delta=`*`value`* — Specifies the number of steps in photon energy for the natural lineshape (¡5000). delta specifies range of photon energies to be included in the natural lineshape in units (energy of fundamental/N). The default value covers the range $\pm 2/N$ of the natural lineshape. Used in mode=2,3,4,5 method=1.
  - `photonEnergy=maximum=`*`eV`*`,minimum=`*`eV`*`,points=`*`number`* — Specifies the maximum and minimum photon energy in eV, and the number of energy points to be computed.

- **authors:** H. Shang, R. Dejus, M. Borland, X. Jiao (ANL).

7.11 touschekLifetime

- **description:** `touschekLifetime` computes Touschek lifetime using A. Piwinski's formula [23, 24].
- **examples:**

```
touschekLifetime aps.life -twiss=aps.twi -aper=aps.aper -part=2e10
-coupling=0.01 -length=6
```

- **synopsis:**

```
touschekLifetime outputFile -twiss=<twissFile>
-aperture=<momentumApertureFile> -charge=<nC>|-particles=<value>
-coupling=<value> -RF=Voltage=<MV>,harmonic=<value>|-length=<mm>
[-emitxInput=<value>] [-deltaInput=<value>] [-verbosity=<value>]
```

- **files:** *outputFile* — Contains resulting Touschek lifetime.
- **switches:**
  - `-twiss` — A twiss parameter file from `elegant`. You must use the `radiation_integrals` flag in `twiss_output`.
  - `-aperture` — A momentum aperture file from `elegant`. This file can contain a subset of elements of twissFile (for example: only Quadrupole elements). However, the Twiss and momentum aperture files *must* cover the same beamline. Having one file for a part of beamline (e.g., a few sectors) and one for the entire ring will yield incorrect results.
  - `-charge`, `-particles` — Give the charge (in nanocoulombs) or the number of electrons.
  - `-coupling` — Give the emittance coupling ratio, ϵ_y/ϵ_x. This is used to compute the horizontal and vertical emittance from the natural emittance.
  - `-RF=Voltage=`*MV*`,harmonic=`*value* — Specify rf voltage and harmonic number. N.B.: If the data files cover only a portion of the ring, using this option will give incorrect results!
  - `-length=`*mm* — Specify the rms bunch length. This is an alternative to giving rf parameters.
  - `-emitxInput` — Give the initial x emittance in meters. If not specified, the value from the parameter `ex0` in *twissFile* is used.
  - `-deltaInput` — Give the initial rms fractional momentum spread. If not specified, the value from the parameter `Sdelta0` in *twissFile* is used.
  - `-verbosity` — If nonzero, program execution information is printed to the standard output.
- **author:** A. Xiao, ANL/APS.

8 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_elements` 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.
- The maximum length of the name of an element or beamline is 100 characters.

`elegant`'s lattice parser translates all input into upper case, except where the input is protected by double quotes. However, various commands (such as `vary_element` or `link_elements`) that accept element names as input do not perform any translation. Hence, when referring to element names in commands, the user must enter the names in upper case unless they are protected by double quotes in the lattice file.

The rules for naming elements and beamlines are as follows:

- The name should start with an alphabetic character (i.e., a-z A-Z).
- The name may contain any of the following characters in addition to alphabetic characters and numbers: `~ @ $ % ^ & - _ + = { } [ ] \ | / ? < > . : |\verb`
- The name should not contain any of the following: `# * ! ‘‘ ’ ‘`
- The name should not contain spaces, tabs, or non-printing characters.

If using unusual characters in a name, it is a good idea to enclose the name in double quotes. This is required if `:` is in the name.

`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.

Comments may be embedded in the lattice file by starting a line with an exclamation point ("!"). Rpn expressions may be embedded separately from an element definition by starting a line with a percent sign ("%"). For example

```
! Define pi
% 1 atan 4 * sto Pi
% Pi 40 / sto myAngle
! Define a rectangular bend for a ring with 80 equal bends
B1: SBEN,L=1.0,ANGLE=''myAngle'',E1=''myAngle 2 /'',E2=''myAngle 2 /''
```

(Note that `pi` is already defined in the standard `defns.rpn` file.)

9 Element Dictionary

9.1 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.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
XMAX	M	double	0.0	size of alpha
XS1	M	double	0.0	inner scraper position relative to XMAX
XS2	M	double	0.0	outer scraper position relative to XMAX
DP1		double	-1	inner scraper fractional momentum deviation
DP2		double	1	outer scraper fractional 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]
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides a matrix-based implementation of an alpha magnet [5]. Matrices up to third order are available [4].

The parameter `XMAX` determines the size of the alpha, which is related to the gradient g in the magnet and the central momentum $\beta\gamma$ by

$$x_{max}[m] = 0.07504986\sqrt{\frac{\beta\gamma}{g[T/m]}}. \tag{1}$$

The path length of the central particle is $2.554x_{max}$.

Because an alpha magnet has large dispersion at the midplane, it is often used for momentum filtration in addition to bunch compression. The dispersion at the center is given by the simple relation

$$R_{16} = -\frac{1}{2}x_{max}. \tag{2}$$

To use an alpha magnet for momentum filtration in `elegant`, one must split the alpha magnet into two pieces. One may then either use the scraper features of the `ALPH` element or other elements such as `SCRAPER` or `RCOL`.

To split an alpha magnet, one uses the `PART` parameter. E.g.,

```
! First half, with momentum filter between -5% and +2.5%
AL1: ALPH,XMAX=0.11,PART=1,DP1=-0.05,DP2=0.025
! Second half
AL2: ALPH,XMAX=0.11,PART=2
AL:  LINE=(AL1,AL2)
```

As just illustrated, the parameters `DP1` and `DP2` may be used to filter the momentum by providing fractional momentum deviation limits. These are implemented in a physical fashion by computing the corresopnding horizontal position deviations and imposing these as limits on the particle coordinates. One may also do this directly using the `XS1` and `XS2` parameters, which specify maximum acceptable deviations from the nominal horizontal position. `XS1` is the allowed deviation on the low-energy side while `XS2` is the allowed deviation on the high-energy side.

9.2 BMAPXY

A map of Bx and By vs x and y.
Parallel capable? : yes

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)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates transport through a transverse magnetic field specified as a field map. It does this by simply integrating the Lorentz force equation in cartesian coordinates. It does not incorporate changes in the design trajectory resulting from the fields. I.e., if you input a dipole field, it is interpreted as a steering element.

The field map file is an SDDS file with the following columns:

- **x**, **y** — Transverse coordinates in meters (units should be "m").
- **Fx**, **Fy** — Normalized field values (no units). The field is multiplied by the value of the STRENGTH parameter to convert it to a local bending radius. For example, if Fx=y and Fy=x, then STRENGTH is the K1 quadrupole parameter.
- **Bx**, **By** — Field values in Tesla (units should be "T"). The field is still multiplied by the value of the STRENGTH parameter, which is dimensionless. Note: the default value of STRENGTH is 0, so if you don't set it to something, you'll get no effect!

The field map file must contain a rectangular grid of points, equispaced (separately) in x and y. There should be no missing values in the grid (this is not checked by `elegant`). In addition, the x values must vary fastest as the values are accessed in row order. To ensure that this is the case, use the following command on the field file:

sddssort *fieldFile* -column=y,incr -column=x,incr

9.3 BUMPER

A time-dependent uniform-field rectangular kicker magnet with no fringe effects. The waveform is in SDDS format, with time in seconds and amplitude normalized to 1.
Parallel capable? : yes

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
B2	$1/M^2$	double	0.0	Sextupole term: By=Bo*(1+b2*x$\hat{2}$)
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
N_KICKS		long	0	Number of kicks to use for simulation. 0 uses an exact result but ignores b2.
WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving kick factor vs time
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.4 CENTER

An element that centers the beam transversely on the ideal trajectory.
Parallel capable? : no

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?
ON_PASS		long	-1	If nonnegative, do centering on the nth pass only.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.5 CEPL

A numerically-integrated linearly-ramped electric field deflector.
Parallel capable? : no

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, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.6 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.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
TOTAL	C	double	0.0	total charge in beam
PER_PARTICLE	C	double	0.0	charge per macroparticle
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.7 CLEAN

Cleans the beam by removing outlier particles.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.8 CSBEND

A canonical kick sector dipole magnet.
Parallel capable? : yes

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	$1/M^4$	double	0.0	geometric octupole strength
K4	$1/M^5$	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	RAD	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

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)?
ISR1PART		long	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
SQRT_ORDER		long	0	Order of expansion of square-root in Hamiltonian. 0 means no expansion.
USE_RAD_DIST		long	0	If nonzero, overrides SYNCH_RAD and ISR, causing simulation of radiation from distributions, optionally including opening angle.
ADD_OPENING_ANGLE		long	1	If nonzero, radiation opening angle effects are add if USE_RAD_DIST is nonzero.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides a symplectic bending magnet with the exact Hamiltonian. For example, it retains all orders in the momentum offset and curvature. The field expansion is available to fourth order.

One pitfall of symplectic integration is the possibility of orbit and path-length errors for the reference orbit if too few kicks are used. This may be an issue for rings. Hence, one must verify that a sufficient number of kicks are being used by looking at the trajectory closure and length of an on-axis particle by tracking. Using `INTEGRATION_ORDER=4` is recommended to reduce the number

of required kicks.

Normally, one specifies the higher-order components of the field with the `K1`, `K2`, `K3`, and `K4` parameters. The field expansion in the midplane is $B_y(x) = B_o * (1 + \sum_{n=1}^{4} \frac{K_n \rho_o}{n!} x^n)$. By setting the `USE_bN` flag to a nonzero value, one may instead specify the `b1` through `b4` parameters, which are defined by the expansion $B_y(x) = B_o * (1 + \sum_{n=1}^{4} \frac{b_n}{n!} x^n)$. This is convenient if one is varying the dipole radius but wants to work in terms of constant field quality.

Setting `NONLINEAR=0` turns off all the terms above `K_1` (or `b_1`) and also turns off effects due to curvature that would normally result in a gradient producing terms of higher order.

Edge effects are included using a first- or second-order matrix. The order is controlled using the `EDGE_ORDER` parameter, which has a default value of 1. N.B.: if you choose the second-order matrix, it is not symplectic.

Incoherent synchrotron radiation, when requested with `ISR=1`, normally uses gaussian distributions for the excitation of the electrons. Setting `USE_RAD_DIST=1` invokes a more sophisticated algorithm that uses correct statistics for the photon energy and number distributions. In addition, if `USE_RAD_DIST=1` one may also set `ADD_OPENING_ANGLE=1`, which includes the photon angular distribution when computing the effect on the emitting electron.

Special note about splitting dipoles: when dipoles are long, it is common to want to split them into several pieces, to get a better look at the interior optics. When doing this, care must be exercised not to change the optics. `elegant` has some special features that are designed to reduce or manage potential problems. At issue is the need to turn off edge effects between the portions of the same dipole.

First, one can simply use the `divide_elements` command to set up the splitting. Using this command, `elegant` takes care of everything.

Second, one can use a series of dipoles *with the same name*. In this case, elegant automatically turns off interior edge effects. This is true when the dipole elements directly follow one another or are separated by a MARK element.

Third, one can use a series of dipoles with different names. In this case, you must also use the `EDGE1_EFFECTS` and `EDGE2_EFFECTS` parameters to turn off interior edge effects.

9.9 CSRCSBEND

Like CSBEND, but incorporates a simulation of Coherent Synchrotron radiation.
Parallel capable? : yes

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	$1/M^4$	double	0.0	geometric octupole strength
K4	$1/M^5$	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	RAD	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. If less than 1, no SG smoothing is performed.
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. Defaults to SG_HALFWIDTH (if positive) or else 1.
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

CSRCSBEND continued

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

Parameter Name	Units	Type	Default	Description
B2	$1/M^2$	double	0.0	b2 = B2/rho
B3	$1/M^3$	double	0.0	b3 = B3/rho
B4	$1/M^4$	double	0.0	b4 = B4/rho
ISR		long	0	include incoherent synchrotron radiation (scattering)?
ISR1PART		long	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
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.
WAKE_FILTER_FILE		STRING	NULL	Name of file supplying wakefield filtering data.
WFF_FREQ_COLUMN		STRING	NULL	Name of column supplying frequency values for wakefield filtering data.
WFF_REAL_COLUMN		STRING	NULL	Name of column supplying real values for wakefield filtering data.

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

Parameter Name	Units	Type	Default	Description
WFF_IMAG_COLUMN		STRING	NULL	Name of column supplying imaginary values for wakefield filtering data.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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) and G. Stupakov and P. Emma, SLAC LCLS-TN-01-12 (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. See equation 11 of [20].

In order respects, this element is just like the `CSBEND` element, which provides a symplectic bending magnet that is accurate to all orders in momentum offset. The field expansion is available to fourth order.

One pitfall of symplectic integration is the possibility of orbit and path-length errors for the reference orbit if too few kicks are used. This may be an issue for rings. Hence, one must verify that a sufficient number of kicks are being used by looking at the trajectory closure and length of an on-axis particle by tracking. Using `INTEGRATION_ORDER=4` is recommended to reduce the number of required kicks.

Normally, one specifies the higher-order components of the field with the `K1`, `K2`, `K3`, and `K4` parameters. The field expansion in the midplane is $B_y(x) = B_o * (1 + \sum_{n=1}^{4} \frac{K_n \rho_o}{n!} x^n)$. By setting the `USE_bN` flag to a nonzero value, one may instead specify the `b1` through `b4` parameters, which are defined by the expansion $B_y(x) = B_o * (1 + \sum_{n=1}^{4} \frac{b_n}{n!} x^n)$. This is convenient if one is varying the dipole radius but wants to work in terms of constant field quality.

Setting `NONLINEAR=0` turns off all the terms above `K_1` (or `b_1`) and also turns off effects due to curvature that would normally result in a gradient producing terms of higher order.

Edge effects are included using a first- or second-order matrix. The order is controlled using the `EDGE_ORDER` parameter, which has a default value of 1. N.B.: if you choose the second-order

matrix, it is not symplectic.

Special note about splitting dipoles: when dipoles are long, it is common to want to split them into several pieces, to get a better look at the interior optics. When doing this, care must be exercised not to change the optics. `elegant` has some special features that are designed to reduce or manage potential problems. At issue is the need to turn off edge effects between the portions of the same dipole.

First, one can simply use the `divide_elements` command to set up the splitting. Using this command, `elegant` takes care of everything.

Second, one can use a series of dipoles *with the same name*. In this case, elegant automatically turns off interior edge effects. This is true when the dipole elements directly follow one another or are separated by a MARK element.

Third, one can use a series of dipoles with different names. In this case, you must also use the `EDGE1_EFFECTS` and `EDGE2_EFFECTS` parameters to turn off interior edge effects.

N.B.: Splitting dipoles is not generally recommended for `CSRCSBEND` because the coherent synchrotron radiation computations start over at the beginning of each piece. This is only acceptable when using `STEADY_STATE=1`.

9.10 CSRDRIFT

A follow-on element for CSRCSBEND that applies the CSR wake over a drift.
Parallel capable? : yes

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)

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?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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.

9.11 CWIGGLER

Tracks through a wiggler using canonical integration routines of Y. Wu (Duke University).
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	Total length
B_MAX		double	0.0	Maximum on-axis magnetic field.
BX_MAX		double	0.0	Maximum on-axis magnetic field. Ignored if B_MAX is nonzero.
BY_MAX		double	0.0	Maximum on-axis magnetic field. Ignored if B_MAX is nonzero.
DX		double	0.0	Misaligment.
DY		double	0.0	Misaligment.
DZ		double	0.0	Misaligment.
TILT		double	0.0	Rotation about beam axis.
PERIODS		long	0	Number of wiggler periods.
STEPS_PER_PERIOD		long	10	Integration steps per period.
INTEGRATION_ORDER		long	4	Integration order (2 or 4).
BY_FILE		STRING	NULL	Name of SDDS file with By harmonic data.
BX_FILE		STRING	NULL	Name of SDDS file with Bx harmonic data.
BY_SPLIT_POLE		long	0	Use "split-pole" expansion for By?
BX_SPLIT_POLE		long	0	Use "split-pole" expansion for Bx?
SYNCH_RAD		long	0	Include classical synchrotron radiation?
ISR		long	0	Include incoherent synchrotron radiation (scattering)?
ISR1PART		long	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
SINUSOIDAL		long	0	Ideal sinusoidal wiggler? If non-zero, BX_FILE and BY_FILE are not used.
VERTICAL		long	0	If SINUSOIDAL is non-zero, then setting this to non-zero gives a vertical wiggler. Default is horizontal.

Tracks through a wiggler using canonical integration routines of Y. Wu (Duke University).

Parameter Name	Units	Type	Default	Description
HELICAL		long	0	Ideal helical wiggler? If non-zero and SINUSOIDAL is also non-zero, BX_FILE and BY_FILE are not used.
FORCE_MATCHED		long	1	Force matched dispersion for first harmonics? If non-zero, start and end of magnetic field will be inset from the ends of the device if phase is not 0 or π.
FIELD_OUTPUT		STRING	NULL	Name of file to which field samples will be written. Slow, so use only for debugging.
VERBOSITY		long	0	A higher value requires more detailed printouts related to computations.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

N.B.: at present this element is *not* included in computations of beam moments (`moments_output` command).

This element simulates a wiggler or undulator using a modified version of Ying Wu's canonical integration code for wigglers. To use the element, one must supply an SDDS file giving harmonic analysis of the wiggler field. The field expansion used by the code for a horizontally-deflecting wiggler is (Y. Wu, Duke University, private communication).

$$B_y = -\,|B_0| \sum_{m,n} C_{mn} \cos(k_{xl}x) \cosh(k_{ym}y) \cos(k_{zn}z + \theta_{zn}), \tag{3}$$

where $|B_0|$ is the peak value of the on-axis magnetic field, the C_{mn} give the relative amplitudes of the harmonics, the wavenumbers statisfy $k_{ym}^2 = k_{xl}^2 + k_{zn}^2$, and θ_{zn} is the phase.

The file must contain the following columns:

- The harmonic amplitude, C_{mn}, in column `Cmn`.
- The phase, in radians, in column `Phase`. The phase of the first harmonic should be 0 or π in order to have matched dispersion.
- The three wave numbers, normalized to $k_w = 2\pi/\lambda_w$, where λ_w is the wiggler period. These are given in columns `KxOverKw`, `KyOverKw`, and `KzOverKw`.

In Version 17.3 and later, for matrix computations `elegant` uses a first-order matrix derived from particle tracking when it encounterse a CWIGGLER. Tests show that this gives good agreement in the tunes from tracking and Twiss parameter calculations. For radiation integrals, an idealized sinusoidal wiggler model is used with bending radius equal to $B\rho/(B_0 \sum C_{mn})$ for each plane. Energy loss, energy spread, and horizontal emittance should be estimated accurately.

`elegant` allows specifying field expansions for on-axis B_y and B_x components, so one can model a helical wiggler. However, in this case one set of components should have $\theta_{zn} = 0$ or $\theta_{zn} = \pi$, while the other should have $\theta_{zn} = \pm\pi/2$. Using Wu's code, the latter set will not have matched dispersion. Our modified version solves this by delaying the beginning of the field components in question by $\lambda/4$ and ending the field prematurely by $3\lambda/4$. This causes all the fields to start and end at the crest, which ensures matched dispersion. The downside is that the (typically) vertical wiggler component is missing a full period of field. One can turn off this behavior by setting `FORCE_MATCHED=0`.

Additional field expansions

Y. Wu's code included field expansions for a vertically-deflecting wiggler as well as the horizontally-deflecting wiggler given above. In both cases, these expansions are suitable for a wiggler with two poles that are above/below or left/right of the beam axis. They are not always suitable for devices with more complex pole geometries.

Another geometry that is important is a "split pole" wiggler, in which each pole is made from two pieces. Such configurations are seen, for example, in devices used to produce variable polarization. In such cases, the expansion given above may not be appropriate. Here, we summarize the form of the various expansions that `elegant` supports. For brevity, we show the form of a single harmonic component.

Horizontal wiggler, normal poles, produces B_y only on-axis. Specified by setting `BY_SPLIT_POLE=0`, and giving `BY_FILE` or `SINUSOIDAL=1` with `VERTICAL=0`.

$$B_x = |B_0| \frac{k_x \cos(k_z z + \phi) \sin(k_x x) \sinh(k_y y)}{k_y} \tag{4}$$

$$B_y = -|B_0| \cos(k_x x) \cos(k_z z + \phi) \cosh(k_y y) \tag{5}$$

where $k_y^2 = k_x^2 + k_z^2$.

Horizontal wiggler, split poles, produces B_y only on-axis. Specified by setting `BY_SPLIT_POLE=1`, and giving `BY_FILE` or `SINUSOIDAL=1` with `VERTICAL=0`.

$$B_x = -|B_0| \frac{k_x \cos(k_z z + \phi) \sin(k_y y) \sinh(k_x x)}{k_y} \tag{6}$$

$$B_y = -|B_0| \cos(k_y y) \cos(k_z z + \phi) \cosh(k_x x) \tag{7}$$

where $k_y^2 = k_x^2 + k_z^2$.

Vertical wiggler, normal poles, produces B_x only on-axis. Specified by setting `BX_SPLIT_POLE=0`, and giving `BX_FILE` or `SINUSOIDAL=1` with either `VERTICAL=1` or `HELICAL=1`.

$$B_x = |B_0| \cos(k_y y) \cos(k_z z + \phi) \cosh(k_x x) \tag{8}$$

$$B_y = -|B_0| \frac{k_y \cos(k_z z + \phi) \sin(k_y y) \sinh(k_x x)}{k_x} \tag{9}$$

where $k_x^2 = k_y^2 + k_z^2$.

Vertical wiggler, split poles, produces B_x only on-axis. Specified by setting `BX_SPLIT_POLE=1`, and giving `BX_FILE` or `SINUSOIDAL=1` with either `VERTICAL=1` or `HELICAL=1`.

$$B_x = |B_0| \cos(k_x x) \cos(k_z z + \phi) \cosh(k_y y) \tag{10}$$

$$B_y = |B_0| \frac{k_y \cos(k_z z + \phi) \sin(k_x x) \sinh(k_y y)}{k_x} \tag{11}$$

where $k_x^2 = k_y^2 + k_z^2$.

9.12 DRIF

A drift space implemented as a matrix, up to 2nd order
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
ORDER		long	0	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.13 DSCATTER

A scattering element to add random changes to particle coordinates according to a user-supplied distribution function
Parallel capable? : no

Parameter Name	Units	Type	Default	Description
PLANE		STRING	NULL	Plane to scatter: xp, yp, dp (dp is deltaP/P)
FILENAME		STRING	NULL	Name of SDDS file containing distribution function.
VALUENAME		STRING	NULL	Name of column containing the independent variable for the distribution function data.
CDFNAME		STRING	NULL	Name of column containing the cumulative distribution function data.
PDFNAME		STRING	NULL	Name of column containing the probability distribution function data.
ONCEPERPARTICLE		long	0	If nonzero, each particle can only get scattered once by this element.
FACTOR		double	1	Factor by which to multiply the independent variable values.
PROBABILITY		double	1	Probability that any particle will be selected for scattering.
GROUPID		long	-1	Group ID number (nonnegative integer) for linking once-per-particle behavior of multiple elements.
RANDOMSIGN		long	0	If non-zero, then the scatter is given a random sign. Useful if distribution data is one-sided.
LIMITPERPASS		long	-1	Maximum number of particles that will be scattered on each pass.
LIMITTOTAL		long	-1	Maximum number of particles that will be scatter for each step.

DSCATTER continued

A scattering element to add random changes to particle coordinates according to a user-supplied distribution function

Parameter Name	Units	Type	Default	Description
STARTONPASS		long	0	Pass number to start on.
ENDONPASS		long	-1	Pass number to end on (inclusive). Ignored if negative.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.14 ECOL

An elliptical collimator.
Parallel capable? : yes

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
OPEN_SIDE		STRING	NULL	which side, if any, is open (+x, -x, +y, -y)
EXPONENT		long	2	Exponent for boundary equation. 2 is ellipse.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.15 EDRIFT

Tracks through a drift with no approximations (Exact DRIFT).
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.16 ELSE

Not implemented.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.17 EMATRIX

Explicit matrix input with data in the element definition, rather than in a file.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	Length (used only for position computation)
ANGLE	RAD	double	0.0	Angle (used only for position computation)
TILT	RAD	double	0.0	Tilt angle
ORDER		long	0	
C1	M	double	0.0	
C2		double	0.0	
C3	M	double	0.0	
C4		double	0.0	
C5	M	double	0.0	
C6		double	0.0	
R11		double	0.0	
R12	M	double	0.0	
R13		double	0.0	
R14	M	double	0.0	
R15		double	0.0	
R16	M	double	0.0	
R21	$1/M$	double	0.0	
R22		double	0.0	
R23	$1/M$	double	0.0	
R24		double	0.0	
R25	$1/M$	double	0.0	
R26		double	0.0	
R31		double	0.0	
R32	M	double	0.0	
R33		double	0.0	
R34	M	double	0.0	
R35		double	0.0	
R36	M	double	0.0	
R41	$1/M$	double	0.0	
R42		double	0.0	
R43	$1/M$	double	0.0	
R44		double	0.0	
R45	$1/M$	double	0.0	
R46		double	0.0	

EMATRIX continued

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
R51		double	0.0	
R52	M	double	0.0	
R53		double	0.0	
R54	M	double	0.0	
R55		double	0.0	
R56	M	double	0.0	
R61	$1/M$	double	0.0	
R62		double	0.0	
R63	$1/M$	double	0.0	
R64		double	0.0	
R65	$1/M$	double	0.0	
R66		double	0.0	
T111	$1/M$	double	0.0	
T121		double	0.0	
T122	M	double	0.0	
T131	$1/M$	double	0.0	
T132		double	0.0	
T133	$1/M$	double	0.0	
T141		double	0.0	
T142	M	double	0.0	
T143		double	0.0	
T144	M	double	0.0	
T151	$1/M$	double	0.0	
T152		double	0.0	
T153	$1/M$	double	0.0	
T154		double	0.0	
T155	$1/M$	double	0.0	
T161		double	0.0	
T162	M	double	0.0	
T163		double	0.0	
T164	M	double	0.0	
T165		double	0.0	
T166	M	double	0.0	
T211	$1/M^2$	double	0.0	
T221	$1/M$	double	0.0	
T222		double	0.0	

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
T231	$1/M^2$	double	0.0	
T232	$1/M$	double	0.0	
T233	$1/M^2$	double	0.0	
T241	$1/M$	double	0.0	
T242		double	0.0	
T243	$1/M$	double	0.0	
T244		double	0.0	
T251	$1/M^2$	double	0.0	
T252	$1/M$	double	0.0	
T253	$1/M^2$	double	0.0	
T254	$1/M$	double	0.0	
T255	$1/M^2$	double	0.0	
T261	$1/M$	double	0.0	
T262		double	0.0	
T263	$1/M$	double	0.0	
T264	1	double	0.0	
T265	$1/M$	double	0.0	
T266		double	0.0	
T311	$1/M$	double	0.0	
T321		double	0.0	
T322	M	double	0.0	
T331	$1/M$	double	0.0	
T332		double	0.0	
T333	$1/M$	double	0.0	
T341		double	0.0	
T342	M	double	0.0	
T343		double	0.0	
T344	M	double	0.0	
T351	$1/M$	double	0.0	
T352		double	0.0	
T353	$1/M$	double	0.0	
T354		double	0.0	
T355	$1/M$	double	0.0	
T361		double	0.0	
T362	M	double	0.0	
T363		double	0.0	

EMATRIX continued

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
T364	M	double	0.0	
T365		double	0.0	
T366	M	double	0.0	
T411	$1/M^2$	double	0.0	
T421	$1/M$	double	0.0	
T422		double	0.0	
T431	$1/M^2$	double	0.0	
T432	$1/M$	double	0.0	
T433	$1/M^2$	double	0.0	
T441	$1/M$	double	0.0	
T442		double	0.0	
T443	$1/M$	double	0.0	
T444		double	0.0	
T451	$1/M^2$	double	0.0	
T452	$1/M$	double	0.0	
T453	$1/M^2$	double	0.0	
T454	$1/M$	double	0.0	
T455	$1/M^2$	double	0.0	
T461	$1/M$	double	0.0	
T462		double	0.0	
T463	$1/M$	double	0.0	
T464	1	double	0.0	
T465	$1/M$	double	0.0	
T466		double	0.0	
T511	$1/M$	double	0.0	
T521		double	0.0	
T522	M	double	0.0	
T531	$1/M$	double	0.0	
T532		double	0.0	
T533	$1/M$	double	0.0	
T541		double	0.0	
T542	M	double	0.0	
T543		double	0.0	
T544	M	double	0.0	
T551	$1/M$	double	0.0	
T552		double	0.0	

Explicit matrix input with data in the element definition, rather than in a file.

Parameter Name	Units	Type	Default	Description
T553	$1/M$	double	0.0	
T554		double	0.0	
T555	$1/M$	double	0.0	
T561		double	0.0	
T562	M	double	0.0	
T563		double	0.0	
T564	M	double	0.0	
T565		double	0.0	
T566	M	double	0.0	
T611	$1/M^2$	double	0.0	
T621	$1/M$	double	0.0	
T622		double	0.0	
T631	$1/M^2$	double	0.0	
T632	$1/M$	double	0.0	
T633	$1/M^2$	double	0.0	
T641	$1/M$	double	0.0	
T642		double	0.0	
T643	$1/M$	double	0.0	
T644		double	0.0	
T651	$1/M^2$	double	0.0	
T652	$1/M$	double	0.0	
T653	$1/M^2$	double	0.0	
T654	$1/M$	double	0.0	
T655	$1/M^2$	double	0.0	
T661	$1/M$	double	0.0	
T662		double	0.0	
T663	$1/M$	double	0.0	
T664	1	double	0.0	
T665	$1/M$	double	0.0	
T666		double	0.0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.18 ENERGY

An element that matches the central momentum to the beam momentum, or changes the central momentum or energy to a specified value.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.19 FLOOR

Sets floor coordinates
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
X		double	0.0	X coordinate
Y		double	0.0	Y coordinate
Z		double	0.0	Z coordinate
THETA		double	0.0	theta value
PHI		double	0.0	phi value
PSI		double	0.0	psi value
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.20 FMULT

Multipole kick element with coefficient input from an SDDS file.
Parallel capable? : yes

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
SQRT_ORDER		long	0	Order of expansion of square-root in Hamiltonian. 0 means no expansion.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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 $1/m^n$.

3. A floating point column named `JnL` giving the integrated strength of the skew multipole, $J_n L$, where n is the order. The units are $1/m^n$.

The `MULT` element is also available, which allows the same functionality without an external file.

9.21 FRFMODE

One or more beam-driven TM monopole modes of an RF cavity, with data from a file.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
FILENAME		STRING	NULL	input file
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
RIGID_UNTIL_PASS		long	0	don't affect the beam until this pass
USE_SYMM_DATA		long	0	use "Symm" columns from URMEL output file?
FACTOR		double	1	factor by which to multiply shunt impedances
CUTOFF	HZ	double	0.0	If >0, cutoff frequency. Modes above this frequency are ignored.
OUTPUT_FILE		STRING	NULL	Output file for voltage in each mode.
FLUSH_INTERVAL		long	1	Interval in passes at which to flush output data.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is similar to `RFMODE`, but it allows faster simulation of more than one mode. Also, the mode data is specified in an SDDS file. This file can be generated using the APS version of URMEL, or by hand. It must have the following columns and units:

1. `Frequency` — The frequency of the mode in Hz. Floating point.
2. `Q` — The quality factor. Floating point.
3. `ShuntImpedance` or `ShuntImpedanceSymm` — The shunt impedance in Ohms, defined as $V^2/(2*P)$. Floating point. By default, `ShuntImpedance` is used. However, if the parameter `USE_SYMM_DATA` is non-zero, then `ShuntImpedanceSymm` is used. The latter is the full-cavity shunt impedance that URMEL computes by assuming that the input cavity used is one half of a symmetric cavity.

The file may also have the following column:

1. `beta` — Normalized load impedance (dimensionless). Floating point. If not given, the $\beta = 0$ is assumed for all modes.

In many simulations, a transient effect may occur when using this element because, in the context of the simulation, the impedance is switched on instantaneously. This can give a false indication of the threshold for instability. The `RAMP_PASSES` parameter should be used to prevent this by slowly ramping the impedance to full strength. This idea is from M. Blaskiewicz (BNL).

9.22 FTRFMODE

One or more beam-driven TM dipole modes of an RF cavity, with data from a file.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
FILENAME		STRING	NULL	input file
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
RIGID_UNTIL_PASS		long	0	don't affect the beam until this pass
USE_SYMM_DATA		long	0	use "Symm" columns from URMEL output file?
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
XFACTOR		double	1	factor by which to multiply shunt impedances
YFACTOR		double	1	factor by which to multiply shunt impedances
CUTOFF	HZ	double	0.0	If >0, cutoff frequency. Modes above this frequency are ignored.
OUTPUT_FILE		STRING	NULL	Output file for voltage in each mode.
FLUSH_INTERVAL		long	1	Interval in passes at which to flush output data.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is similar to `TRFMODE`, but it allows faster simulation of more than one mode. Also, the mode data is specified in an SDDS file. This file can be generated using the APS version of URMEL, or by hand. It must have the following columns and units:

1. `Frequency` — The frequency of the mode in Hz. Floating point.
2. `Q` — The quality factor. Floating point.

3. `ShuntImpedance` or `ShuntImpedanceSymm` — The shunt impedance in Ohms/m, defined as $V^2/(2*P)/x$ or $V^2/(2*P)/y$. Floating point. By default, `ShuntImpedance` is used. However, if the parameter `USE_SYMM_DATA` is non-zero, then `ShuntImpedanceSymm` is used. The latter is the full-cavity shunt impedance that URMEL computes by assuming that the input cavity used is one half of a symmetric cavity.

The file may also have the following columns:

1. `beta` — Normalized load impedance (dimensionless). Floating point. If not given, the $\beta = 0$ is assumed for all modes.

2. `xMode` — If given, then only modes for which the value is nonzero will produce an x-plane kick. Integer. If not given, all modes affect the x plane.

3. `yMode` — If given, then only modes for which the value is nonzero will produce an y-plane kick. Integer. If not given, all modes affect the y plane.

N.B.: Unlike `TRFMODE`, `FTRFMODE` does not include the longitudinal field that, strictly speaking, must also be excited. Generally this is a very small effect. It will be added in a future version.

In many simulations, a transient effect may occur when using this element because, in the context of the simulation, the impedance is switched on instantaneously. This can give a false indication of the threshold for instability. The `RAMP_PASSES` parameter should be used to prevent this by slowly ramping the impedance to full strength. This idea is from M. Blaskiewicz (BNL).

9.23 HISTOGRAM

Request for histograms of particle coordinates to be output to SDDS file.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
FILENAME		STRING		filename for histogram output, possibly incomplete (see below)
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 after the first histogram is made
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
NORMALIZE		long	1	normalize histogram with bin size and number of particles?
DISABLE		long	0	If nonzero, no output will be generated.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The output filename may be an incomplete filename. In the case of the `HISTOGRAM` point element, this means it may contain one instance of the string format specification "%s" and one occurence of an integer format specification (e.g., "%ld"). `elegant` will replace the format with the rootname (see `run_setup`) and the latter with the element's occurrence number. For example, suppose you had a repetitive lattice defined as follows:

```
H1: HISTOGRAM,FILENAME=''%s-%03ld.h1''
Q1: QUAD,L=0.1,K1=1
D: DRIFT,L=1
Q2: QUAD,L=0.1,K1=-1
CELL: LINE=(H1,Q1,D,2*Q2,D,Q1)
BL: LINE=(100*CELL)
```

The element `H1` appears 100 times. Each instance will result in a new file being produced. Successive

instances have names like "*rootname*-001.h1", "*rootname*-002.h1", "*rootname*-003.h1", and so on up to "*rootname*-100.h1". (If instead of "%03ld" you used "%ld", the names would be "*rootname*-1.h1", "*rootname*-2.h1", etc. up to "*rootname*-100.h1". This is generally not as convenient as the names don't sort into occurrence order.)

The files can easily be plotted together, as in

```
% sddsplot -column=dt,dtFrequency *-???.h1 -separate
```

They may also be combined into a single file, as in

```
% sddscombine *-???.h1 all.h1
```

In passing, note that if `H1` was defined as

```
H1: HISTOGRAM,FILENAME=''%s.h1''
```

or

```
H1: HISTOGRAM,FILENAME=''output.h1''
```

only a single file would be produced, containing output from the last instance only.

9.24 HKICK

A horizontal steering dipole implemented as a matrix, up to 2nd order.
Parallel capable? : yes

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$\hat{2}$) when y=0)
CALIBRATION		double	1	strength multiplier
EDGE_EFFECTS		long	0	include edge effects?
ORDER		long	0	matrix order
STEERING		long	1	use for steering?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.25 HMON

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

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)
CO_FITPOINT		long	0	If nonzero, then closed orbit value is placed in variable <name>#<occurence>.xco
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.26 IBSCATTER

A simulation of intra-beam scattering.
Parallel capable? : no

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?
VERBOSITY		long	0	Set verbosity level
FORCE_MATCHED_TWISS		long	0	Force computations to be done with twiss parameters of the beamline, not the beam.
ISRING		long	1	Is it storage ring?
INTERVAL		long	1	Interval in passes at which to update output file.
FILENAME		STRING	NULL	Output filename.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is used for simulation of intra-beam scattering (IBS). The IBS algorithm is based on the Bjorken and Mtingwa's [15] formula, and with an extension of including vertical dispersion. It can be used for both storage ring and Linac.

To initialize IBS calculation, one or more IBSCATTER elements must be inserted into the beamline. `elegant` calculates the integrated IBS growth rates between IBSCATTERs (or from beginning of the beamline to first IBSCATTER), then scats particles at each IBSCATTER element. Beam's parameters are updated for use in downstream elements.

Because the IBS growth rates are energy dependent, special caution is needed for calculations with accelerating beam. The user needs to split their accelerating cavity into several pieces, so that γ has no large changes between elements.

The user can examine the calculation through an optional SDDS output file - *filename*. The file has a multiple page structure. One IBSCATTER element at pass i occupies one page. Each page contains integrated IBS growth rates between IBSCATTERs (or from beginning of the beamline to first IBSCATTER) as parameters, and local rates for elements in between as tabular data.

9.27 ILMATRIX

An Individualized Linear Matrix for each particle for fast symplectic tracking with chromatic and amplitude-dependent effects
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	Length (used for position and time-of-flight computation)
NUX		double	0.0	Horizontal tune
NUY		double	0.0	Vertical tune
NUX1M		double	0.0	First chromatic derivative of the horizontal tune
NUY1M		double	0.0	First chromatic derivative of the vertical tune
NUX2M		double	0.0	Second chromatic derivative of the horizontal tune
NUY2M		double	0.0	Second chromatic derivative of the vertical tune
NUX3M		double	0.0	Third chromatic derivative of the horizontal tune
NUY3M		double	0.0	Third chromatic derivative of the vertical tune
NUX1AX		double	0.0	First amplitude derivative of the horizontal tune wrt Ax
NUY1AX		double	0.0	First amplitude derivative of the vertical tune wrt Ax
NUX1AY		double	0.0	First amplitude derivative of the horizontal tune wrt Ay
NUY1AY		double	0.0	First amplitude derivative of the vertical tune wrt Ay
BETAX	M	double	0.0	On-momentum horizontal beta function
BETAY	M	double	0.0	On-momentum vertical beta function
BETAX1M	M	double	0.0	First chromatic derivative of horizontal beta function
BETAY1M	M	double	0.0	First chromatic derivative of vertical beta function
ALPHAX		double	0.0	On-momentum horizontal alpha function

ILMATRIX continued

An Individualized Linear Matrix for each particle for fast symplectic tracking with chromatic and amplitude-dependent effects

Parameter Name	Units	Type	Default	Description
ALPHAY		double	0.0	On-momentum vertical alpha function
ALPHAX1M		double	0.0	First chromatic derivative of horizontal alpha function
ALPHAY1M		double	0.0	First chromatic derivative of vertical alpha function
ETAX	*M*	double	0.0	On-momentum horizontal eta function
ETAPX	*M*	double	0.0	On-momentum horizontal eta' function
ETAY	*M*	double	0.0	On-momentum vertical eta function
ETAPY	*M*	double	0.0	On-momentum vertical eta' function
ETAX1	*M*	double	0.0	First chromatic derivative of horizontal eta function
ETAPX1	*M*	double	0.0	First chromatic derivative of horizontal eta' function
ETAY1	*M*	double	0.0	First chromatic derivative of vertical eta function
ETAPY1	*M*	double	0.0	First chromatic derivative of vertical eta' function
ALPHAC		double	0.0	First-order momentum compaction factor
ALPHAC2		double	0.0	Second-order momentum compaction factor
TILT	*RAD*	double	0.0	Rotation angle about the longitudinal axis.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows fast, symplectic tracking of transport through a periodic cell with chromatic and amplitude-dependent tunes, beta functions, and dispersion. This is done by computing a linear matrix for every particle. The linear matrix is composed from Twiss parameters, tunes, dispersion, etc., supplied by the user. The user can also supply selected chromatic and amplitude derivatives of these quantities, which are used to compute the individual particle's beta functions, tune, dispersion, etc., which in turn allows computing the individual particle's linear matrix.

Using this element is very similar to using linear chromatic matrix tracking via the `setup_linear_chromatic_t` command. The advantage is that using ILMATRIX, one can split a ring into segments and place, for example, impedance elements between the segments.

This element was inspired by requests from Y. Chae (APS).

9.28 KICKER

A combined horizontal-vertical steering magnet implemented as a matrix, up to 2nd order.
Parallel capable? : yes

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?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.29 KPOLY

A thin kick element with polynomial dependence on the coordinates in one plane.
Parallel capable? : yes

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)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.30 KQUAD

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

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
HKICK	RAD	double	0.0	horizontal correction kick
VKICK	RAD	double	0.0	vertical correction kick
HCALIBRATION		double	1	calibration factor for horizontal correction kick
VCALIBRATION		double	1	calibration factor for vertical correction kick
HSTEERING		long	0	use for horizontal correction?
VSTEERING		long	0	use for vertical correction?
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
STEERING_MULTIPOLES		STRING	NULL	input file for multipole content of steering kicks
INTEGRATION_ORDER		long	4	integration order (2 or 4)
SQRT_ORDER		long	0	Order of expansion of square-root in Hamiltonian. 0 means no expansion.
ISR		long	0	include incoherent synchrotron radiation (scattering)?

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
ISR1PART		long	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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.

Specification of systematic higher multipoles due to steering fields is supported through the `STEERING_MULTIPOLES` field. This field gives the name of an SDDS file that supplies the multipole data. The file is 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 $(N_{poles} - 2)/2$. The order must be an even number because of the quadrupole symmetry.

3. Floating point column `an` giving the values for the normal multipole strengths, which are driven by the horizontal steering field. `an` is specifies the multipole strength as a fraction of the steering field strength measured at the reference radius, R: $a_n = \frac{K_n r^n/n!}{K_m r^m/m!}$, where $m = 0$ is the order of the steering field and n is the order of the error multipole. The `bn` values are deduced from the `an` values, specifically, $b_n = a_n * (-1)^{n/2}$.

The dominant systematic multipole term in the steering field is a sextupole. Note that `elegant` presently *does not* include such sextupole contributions in the computation of the chromaticity via the `twiss_output` command. However, these chromatic effects will be seen in tracking.

Apertures specified via an upstream `MAXAMP` element or an `aperture_input` command will be imposed inside this element, with the following rules/limitations:

- If apertures from both sources are present, the smallest is used.
- The apertures are assumed to be rectangular, even if the `ELLIPTICAL` qualifier is set for `MAXAMP`.

9.31 KQUSE

A canonical kick element combining quadrupole and sextupole fields.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
K1	$1/M^2$	double	0.0	geometric quadrupole strength
K2	$1/M^3$	double	0.0	geometric sextupole 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
FSE1	M	double	0.0	fractional strength error for K1
FSE2	M	double	0.0	fractional strength error for K2
N_KICKS		long	4	number of kicks
SYNCH_RAD		long	0	include classical synchrotron radiation?
INTEGRATION_ORDER		long	4	integration order (2 or 4)
ISR		long	0	include incoherent synchrotron radiation (scattering)?
ISR1PART		long	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.32 KSBEND

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

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	$1/M^4$	double	0.0	geometric octupole strength
K4	$1/M^5$	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	RAD	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

KSBEND continued

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.33 KSEXT

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

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)
SQRT_ORDER		long	0	Order of expansion of square-root in Hamiltonian. 0 means no expansion.
ISR		long	0	include incoherent synchrotron radiation (scattering)?
ISR1PART		long	1	Include ISR for single-particle beam only if ISR=1 and ISR1PART=1
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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.

Apertures specified via an upstream `MAXAMP` element or an `aperture_input` command will be imposed inside this element, with the following rules/limitations:

- If apertures from both sources are present, the smallest is used.

- The apertures are assumed to be rectangular, even if the `ELLIPTICAL` qualifier is set for `MAXAMP`.

9.34 LMIRROR

A mirror for light optics
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
RX	*M*	double	0.0	radius in horizontal plane
RY	*M*	double	0.0	radius in vertical plane
THETA	*RAD*	double	0.0	angle of incidence (in horizontal plane)
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
TILT	*RAD*	double	0.0	misalignment rotation about longitudinal axis
YAW	*RAD*	double	0.0	misalignment rotation about vertical axis
PITCH	*RAD*	double	0.0	misalignment rotation about transverse horizontal axis
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.35 LSCDRIFT

Longitudinal space charge impedance
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
LEFFECTIVE	M	double	0.0	effective length (used if L=0)
BINS		long	0	number of bins for current histogram
SMOOTHING		long	0	Use Savitzky-Golay filter to smooth current histogram?
SG_HALFWIDTH		long	1	Savitzky-Golay filter half-width for smoothing current histogram
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing current histogram
INTERPOLATE		long	1	Interpolate wake?
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.
RADIUS_FACTOR		double	1.7	LSC radius is (Sx+Sy)/2*RADIUS_FACTOR
LSC		long	1	Include longitudinal space-charge impedance? If zero, acts like ordinary drift.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates longitudinal space charge in a drift space using the method described in [22]. This is based on the longitudinal space charge impedance per unit length

$$Z_{lsc}(k) = \frac{iZ_0}{\pi k r_b^2}\left[1 - \frac{kr_b}{\gamma}K_1\left(\frac{kr_b}{\gamma}\right)\right] \tag{12}$$

If `L` is 0 and `LEFFECTIVE` is not, then the element provides a LSC kick with impedance given by $Z_{lsc}L_{effective}$. This can be used to insert an LSC kick that integrates the longitudinal space charge effect of a section of a lattice. This should be used only for cases where there is very little relative longitudinal motion of particles.

9.36 LSRMDLTR

A non-symplectic numerically integrated planar undulator including optional co-propagating laser beam for laser modulation of the electron beam.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
BU	*T*	double	0.0	Undulator peak field
PERIODS		long	0	Number of undulator periods.
METHOD	*NULL*	STRING	non-adaptive runge-kutta	integration method (runge-kutta, bulirsch-stoer, modified-midpoint, two-pass modified-midpoint, leap-frog, non-adaptive runge-kutta)
FIELD_EXPANSION	*NULL*	STRING	ideal	ideal, exact, or "leading terms"
ACCURACY	*NULL*	double	0.0	Integration accuracy for adaptive integration. (Not recommended)
N_STEPS		long	0	Number of integration steps for non-adaptive integration.
POLE_FACTOR1		double	0.155717533964439	Strength factor for the first and last pole.
POLE_FACTOR2		double	0.380687615192693	Strength factor for the second and second-to-last pole.
POLE_FACTOR3		double	0.80282999969846	Strength factor for the third and third-to-last pole.
LASER_WAVELENGTH	*M*	double	0.0	Laser wavelength. If zero, the wavelength is calculated from the resonance condition.
LASER_PEAK_POWER	*W*	double	0.0	laser peak power
LASER_W0	*M*	double	1	laser spot size at waist
LASER_PHASE	*RAD*	double	0.0	laser phase
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a planar undulator, together with an optional co-propagating laser beam that can be used as a beam heater or modulator. The simulation is done by numerical integration of the Lorentz equation. It is not symplectic, and hence this element is not recommended for long-term tracking simulation of undulators in storage rings.

The fields in the undulator can be expressed in one of three ways. The FIELD_EXPANSION parameter is used to control which method is used.

- The exact field, given by (see section 3.1.5 of the *Handbook of Accelerator Physics and Engineering*)

$$B_x = 0, \tag{13}$$

$$B_y = B_0 \cosh k_u y \cos k_u z, \tag{14}$$

and

$$B_z = B_0 \sinh k_u y \cos k_u z, \tag{15}$$

where $k_u = 2\pi/\lambda_u$ and λ_u is the undulator period. This is the most precise method, but also the slowest.

- The field expanded to leading order in y:

$$B_y = B_0(1 + \frac{1}{2}(k_u y)^2) \cos k_u z, \tag{16}$$

and

$$B_z = B_0 k_u y \cos k_u z. \tag{17}$$

In most cases, this gives results that are very close to the exact fields, at a savings of 10% in computation time.

- The "ideal" field:

$$B_y = B_0 \cos k_u z, \tag{18}$$

$$B_z = 0. \tag{19}$$

This is about 10% faster than the leading-order mode, but less precise. Small differences from results with the exact field may be seen. Generally, these are too small to be a concern. As a result, this is the default mode.

By default, if the laser wavelength is not given, it is computed from the resonance condition:

$$\lambda_l = \frac{\lambda_u}{2\gamma^2}\left(1 + \frac{1}{2}K^2\right), \tag{20}$$

where γ is the relativistic factor for the beam and K is the undulator parameter.

The adaptive integrator doesn't work well for this element, probably due to sudden changes in field derivatives in the first and last three poles (a result of the implementation of the undulator terminations). Hence, the default integrator is non-adaptive Runge-Kutta. The integration accuracy is controlled via the N_STEPS parameter. N_STEPS should be about 100 times the number of undulator periods.

The expressions for the laser field used by this element were provided by P. Emma (SLAC).

9.37 LTHINLENS

A thin lens for light optics
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
FX	*M*	double	0.0	focal length in horizontal plane
FY	*M*	double	0.0	focal length in vertical plane
DX	*M*	double	0.0	misalignment
DY	*M*	double	0.0	misalignment
DZ	*M*	double	0.0	misalignment
TILT	*RAD*	double	0.0	misalignment rotation about longitudinal axis
YAW	*RAD*	double	0.0	misalignment rotation about vertical axis
PITCH	*RAD*	double	0.0	misalignment rotation about transverse horizontal axis
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.38 MAGNIFY

An element that allows multiplication of phase-space coordinates of all particles by constants.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.39 MALIGN

A misalignment of the beam, implemented as a zero-order matrix.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.40 MAPSOLENOID

A numerically-integrated solenoid specified as a map of (Bz, Br) vs (z, r).
Parallel capable? : yes

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)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.41 MARK

A marker, equivalent to a zero-length drift space.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
DX	M	double	0.0	non-functional misalignment (e.g., for girder)
DY	M	double	0.0	non-functional misalignment (e.g., for girder)
FITPOINT		long	`0`	Supply local values of Twiss parameters, moments, floor coordinates, matrices, etc. for optimization?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

If `FITPOINT=0`, this element results only in generation of additional output rows in the various files that contain output vs s. For example, Twiss parameters, closed orbits, and matrices vs s will all contain a row for each occurrence of each marker element.

If `FITPOINT=1`, the element has additional functionality in the context of optimizations. In particular, for occurrence N of the defined element *Element*, a series of symbols are created of the form *Element#N.quantity*, where *quantity* has the following values:

- The quantity `pCentral` will be available, giving the reference value of $\beta\gamma$ at the marker location.
- The quantities `Cx`, `Cxp`, `Cy`, `Cyp`, `Cs`, and `Cdelta` will be available, giving coordinate centroid values from tracking to the marker location.
- The quantity `Particles` will be available, giving the number of particles tracked to the marker location.
- The quantities `s`ij will be available, giving $\langle (x_i - \langle x_i \rangle)(x_j - \langle x_j \rangle) \rangle$ at the marker location, where $1 \leq i \leq 6$ and $i < j \leq 6$.
- The quantities `R`ij will be available, for $1 \leq i \leq 6$ and $1 \leq j \leq 6$, giving the accumulated first-order transport matrix to the marker location.
- If the default matrix order (as set in `run_setup`) is 2 or greater, the quantities `T`ijk will be available, for $1 \leq i \leq 6$, $1 \leq j \leq 6$, and $1 \leq k \leq j$, giving the accumulated second-order transport matrix to the marker location.

- If Twiss parameter calculations are being performed (via `twiss_output` with `output_at_each_step=1`), then the quantities `alphax`, `betax`, `nux`, `etax`, `etapx`, and `etaxp`, along with similarly-named quantities for the vertical plane, will be available, giving twiss parameter values at the marker location. Note that `etapx` and `etaxp` are the same, being alternate names for η'_x.

- If coupled Twiss parameter calculations are being performed (via `coupled_twiss_output` with `output_at_each_step=1`), then the quantities `betax1`, `betax2`, `betay1`, `betay2`, `cetax`, `cetay`, and `tilt` will be available. (These are the two beta functions for x and y, the coupled dispersion values for x and y, and the beam tilt).

- If moments calculations are being performed (via `moments_output` with `output_at_each_step=1`), then the quantities `s`ij`m`, $1 \leq i \leq j \leq 6$, giving the 21 unique elements of the sigma matrix. The quantities `c`i`m`, $1 \leq i \leq 6$, are also created, giving the 6 centroids from the moments computation. The `m` on the end of the symbols is to distinguish them from the moments computed from tracking.

- If closed orbit calculations are being performed (via `correct` or `closed_orbit`), then the quantities `xco`, `yco`, `xpco`, and `ypco` will be available, giving the x and y closed orbits and their slopes, respectively, at the marker location.

- If floor coordinate calculations are begin performed (via `floor_coordinates`), then the quantities `X`, `Y`, `Z`, `theta`, `phi`, `psi`, and `s` will be available. These are, respectively, the three position coordinates, the three angle coordinates, and the total arc length at the marker location.

The misalignment controls for this element are non-functional, in the sense that they do not affect the beam. However, when combined with external scripts and the `GROUP` parameter, one can use this feature to implement girder misalignments using pairs of markers to indicate the ends of the girders. A future version of `elegant` will implement this internally.

9.42 MATR

Explicit matrix input from a text file, in the format written by the print_matrix command.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	*M*	double	0.0	length
FILENAME		STRING		input file
ORDER		long	1	matrix order
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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).

9.43 MATTER

A Coulomb-scattering and energy-absorbing element simulating material in the beam path.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
XO	M	double	0.0	radiation length
ELASTIC		long	0	elastic scattering? If zero, then particles will lose energy due to material.
ENERGY_STRAGGLE		long	0	Use simple-minded energy straggling model? Ignored for ELASTIC scattering.
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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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 $K_2 = \frac{\alpha^2 Z^{\frac{2}{3}}}{(\beta\gamma)^2}$, 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 $\frac{dE}{dx} = -E/X_o$. 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.

9.44 MAXAMP

A collimating element that sets the maximum transmitted particle amplitudes for all following elements, until the next MAXAMP.
Parallel capable? : yes

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?
EXPONENT		long	2	exponent for boundary equation in elliptical mode. 2 is a true ellipse.
OPEN_SIDE		STRING	NULL	which side, if any, is open (+x, -x, +y, -y)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element sets the aperture for itself and all subsequent elements. The settings are in force until another `MAXAMP` element is seen.

This can introduce unexpected behavior when beamlines are reflected. For example, consider the beamline

```
...
L1:  LINE=( ... )
L2:  LINE=( ... )
MA1: MAXAMP,X_MAX=0.01,Y_MAX=0.005
MA2: MAXAMP,X_MAX=0.01,Y_MAX=0.002
BL1: LINE=(MA1,L1,MA2,L2)
BL:  LINE=(BL1,-BL1)
```

This is equivalent to

```
BL:  LINE=(MA1,L1,MA2,L2,-L2,MA2,-L1,MA1)
```

Note that the aperture `MA1` is the aperture for all of the first instance of beamline `L1`, but that `MA2` is the aperture for the second instance, `-L1`. This is probably not what was intended. To prevent this, it is recommended to always use `MAXAMP` elements in pairs:

```
BL1: LINE=(MA2,MA1,L1,MA1,MA2,L2)
BL:  LINE=(BL1,-BL1)
```

which is equivalent to

```
BL:  LINE=(MA2,MA1,L1,MA1,MA2,L2,-L2,MA2,MA1,-L1,MA1,MA2)
```

Now, both instances of `L1` have the aperture defined by `MA1` and both instances of `L2` have the aperture defined by `MA2`.

9.45 MODRF

A first-order matrix RF cavity with exact phase dependence, plus optional amplitude and phase modulation.
Parallel capable? : yes

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
AMDECAY	$1/s$	double	0.0	exponetial decay rate 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
PMDECAY	$1/s$	double	0.0	exponetial decay rate of phase modulation
FIDUCIAL		STRING	NULL	mode for determining fiducial arrival time (light, tmean, first, pmaximum)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.46 MONI

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

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
CO_FITPOINT		long	0	If nonzero, then closed orbit values are placed in variables <name>#<occurence>.xco and <name>#<occurence>.yco
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.47 MULT

A canonical kick multipole.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KNL	M^{-ORDER}	double	0.0	integrated geometric strength
TILT	RAD	double	0.0	rotation about longitudinal axis
BORE	M	double	0.0	bore radius
BTIPL	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?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a multipole element using 4th-order sympletic integration. A single multipole order, n, is given. The multipole strength is specified by giving

$$K_n L = \left(\frac{\partial^n B_y}{\partial x^n}\right)_{x=y=0} \frac{L}{B\rho}, \tag{21}$$

where $B\rho$ is the beam rigidity. A quadrupole is $n = 1$, a sextupole is $n = 2$, and so on. The relationship between the pole tip field and $K_n L$ is

$$K_n L = \frac{n! B_{tip} L}{r^n (B\rho)}, \tag{22}$$

where r is the bore radius.

9.48 NIBEND

A numerically-integrated dipole magnet with various extended-fringe-field models.
Parallel capable? : yes

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	0.0	not used
FP3	*M*	double	0.0	not used
FP4	*M*	double	0.0	not used
FSE		double	0.0	fractional strength error
ETILT	*RAD*	double	0.0	error rotation about incoming longitudinal axis
ACCURACY		double	0.0001	integration accuracy (for non-adaptive integration, used as the step-size)
MODEL		STRING	linear	fringe model (hard-edge, linear, cubic-spline, tanh, quintic, enge1, enge3, enge5)
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?
ADJUST_BOUNDARY		long	1	adjust fringe boundary position to make symmetric trajectory? (Not done if ADJUST_FIELD is nonzero.)

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

Parameter Name	Units	Type	Default	Description
ADJUST_FIELD		long	0	adjust central field strength to make symmetric trajectory?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

For the NIBEND element, there are various fringe field models available. In the following descriptions, l_f is the extend of the fringe field, which starts at $z = 0$ for convenience in the expressions. Also, $K = \frac{1}{g}\int_{-}\infty^{\infty} F_y(z)(1 - F_y(z))dz$ is K. Brown's fringe field integral (commonly called FINT), where g is the full magnet gap and $\vec{F} = \vec{B}/B_0$, B_0 being the value of the magnetic field well inside the magnet.

- **Linear fringe field:**

$$F_y = zF_a \tag{23}$$
$$F_z = yF_a \tag{24}$$
$$F_a = 1/(6Kg) \tag{25}$$

For this model, the user specifies FINT and HGAP only.

- **Cubic-spline fringe field:**

$$F_y = F_a z^2 + F_b z^3 + y^2(-F_a - 3F_b z) \tag{26}$$
$$F_z = (2F_a z + 3F_b z^2)y \tag{27}$$
$$F_a = 3/l_f^2 \tag{28}$$
$$F_b = -2/l_f^3 \tag{29}$$
$$l_f = 70Kg/9 \tag{30}$$

For this model, the user specifies FINT and HGAP only.

- **Tanh-like fringe field:**

$$F_y = \frac{1}{2}(1 + \tanh F_a z) + \frac{1}{2}(yF_a \mathrm{sech} F_a z)^2 \tanh F_a z + \tag{31}$$
$$\frac{1}{24}(yF_a \mathrm{sech} F_a z)^4 \mathrm{sech} F_a z(11 \sinh F_a z - \sinh 3F_a z) \tag{32}$$
$$F_z = \frac{1}{2}yF_a \mathrm{sech}^2 F_a z + \frac{1}{6}(yF_a \mathrm{sech} F_a z)^3 \mathrm{sech} F_a z(2 - \cosh 2F_a z)) + \tag{33}$$
$$\frac{1}{120}(yF_a \mathrm{sech} F_a z)^5 \mathrm{sech} F_a z(33 - 26 \cosh 2F_a z + cosh 4F_z z) \tag{34}$$
$$F_a = 1/(2Kg) \tag{35}$$
$$l_f = P_1/F_a \tag{36}$$

For this model, the user specifies `FINT` and `HGAP`, along with the parameter `FP1`, which is the quantity P_1 in the last equation. It determines the length of the fringe field that is integrated.

- **Quintic-spline fringe field, to third order in y:**

$$
\begin{aligned}
F_y &= (F_a z^3 + F_b z^4 + F_c z^5) + y^2 z(3F_a + 6F_b z + 10F_c z^2) && (37)\\
F_z &= y(3F_a z^2 + 4F_b z^3 + 5F_c z^4) + y^3(-F_a - 4F_b z - 10F_c z^2) && (38)\\
F_a &= 10/l_f^3 && (39)\\
F_b &= -15/l_f^4 && (40)\\
F_c &= 6/l_f^5 && (41)\\
l_f &= 231Kg/25 && (42)
\end{aligned}
$$

For this model, the user specifies `FINT` and `HGAP` only.

- **Enge model with 3 coefficients:**

$$
\begin{aligned}
F_0 &= \frac{1}{1 + e^{a_1 + a_2 z/D + a_3 (z/D)^2}} && (43)\\
F_y &= F_0 - \frac{1}{2} y^2 F_0^{(2)} + \frac{1}{24} y^4 F_0^{(4)} && (44)\\
F_z &= y F_0^{(1)} - \frac{1}{6} y^3 F_0^{(3)} + \frac{1}{120} y^5 F_0^{(5)} && (45)
\end{aligned}
$$

where $F_0^{(n)} = \frac{\partial^n F_0}{\partial z^n}$.

The user may choose "enge1", "enge3", or "enge5", where the number indicates the order of the expansion of F_z with respect to y.

The need only specify `FINT` and `HGAP`. The Enge parameters are then automatically determined to give the correct linear focusing.

However, if user gives non-zero value for `FP2`, then `FINT` and `HGAP` are ignored. `FP2`, `FP3`, and `FP4` and taken as the Enge coefficients a_1, a_2, and a_3, respectively.

9.49 NISEPT

A numerically-integrated dipole magnet with a Cartesian gradient.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.50 OCTU

Not implemented–use the MULT element.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.51 PEPPOT

A pepper-pot plate.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.52 PFILTER

An element for energy and momentum filtration.
Parallel capable? : no

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.53 QUAD

A quadrupole implemented as a matrix, up to 3rd order.
Parallel capable? : yes

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
HKICK	RAD	double	0.0	horizontal correction kick
VKICK	RAD	double	0.0	vertical correction kick
HCALIBRATION		double	1	calibration factor for horizontal correction kick
VCALIBRATION		double	1	calibration factor for vertical correction kick
HSTEERING		long	0	use for horizontal steering?
VSTEERING		long	0	use for vertical steering?
ORDER		long	0	matrix order
FRINGE_TYPE		STRING	inset	type of fringe: "inset" or "fixed-strength"
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a quadrupole using a matrix of first, second, or third order.

By default, the element has hard edges and constant field within the defined length, `L`. However, soft-edge effects (up to second order) may be added using the `FFRINGE` and `FRINGE_TYPE` parameters. If `FFRINGE` is zero (the default), then the magnet is hard-edged. If `FFRINGE` is positive, then the magnet has linear fringe fields of length `FFRINGE*L/2` at each end. That is, the total length of fringe field from both ends combined is `FFRINGE*L`.

Depending on the value of `FRINGE_TYPE`, the fringe fields are modeled as contained within the length `L` ("inset" type) or extending symmetrically outside the length `L` ("fixed-strength" type).

For "inset" type fringe fields, the length of the "hard core" part of the quadrupole is `L*(1-FFRINGE)`. For "fixed-strength" type fringe fields, the length of the hard core is `L*(1-FFRINGE/2)`. In the latter case, the fringe gradient reaches 50% of the hard core value at the nominal boundaries of the

magnet. This means that the integrated strength of the magnet does not change as the `FFRINGE` parameter is varied. This is not the case with "inset" type fringe fields.

9.54 QUFRINGE

An element consisting of a linearly increasing or decreasing quadrupole field.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.55 RAMPP

A momentum-ramping element that changes the central momentum according to an SDDS- format file of the momentum factor vs time in seconds.
Parallel capable? : no

Parameter Name	Units	Type	Default	Description
WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving momentum factor vs time
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.56 RAMPRF

A voltage-ramped RF cavity, implemented like RFCA. The voltage ramp pattern is given by an SDDS-format file of the voltage factor vs time in seconds.
Parallel capable? : yes

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)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.57 RBEN

A rectangular dipole, implemented as a SBEND with edge angles, up to 2nd order.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.58 RCOL

A rectangular collimator.
Parallel capable? : yes

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
OPEN_SIDE		STRING	NULL	which side, if any, is open (+x, -x, +y, -y)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.59 RECIRC

An element that defines the point to which particles recirculate in multi-pass tracking
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
I_RECIRC_ELEMENT		long	0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.60 REFLECT

Reflects the beam back on itself, which is useful for multiple beamline matching.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
DUMMY		long	0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.61 REMCOR

An element to remove correlations from the tracked beam to simulate certain types of correction.
Parallel capable? : no

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?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.62 RFCA

A first-order matrix RF cavity with exact phase dependence.
Parallel capable? : yes

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 (for cavity that charges up to given voltage from 0)
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	set to 1 for long runs to avoid rounding error in phase
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?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The phase convention is as follows, assuming a positive rf voltage: `PHASE=90` is the crest for acceleration. `PHASE=180` is the stable phase for a storage ring above transition without energy losses.

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.

The `CHANGE_T` parameter may be needed for reasons that stem from `elegant`'s internal use of the total time-of-flight as the longitudinal coordinate. If the accelerator is very long or a large number of turns are being tracked, rounding error may affect the simulation, introducing spurious phase jumps. By setting `CHANGE_T=1`, you can force `elegant` to modify the time coordinates of the particles to subtract off NT_{rf}, where T_{tf} is the rf period and $N = \lfloor t/T_{tf} + 0.5 \rfloor$. If you are tracking a ring with rf at some harmonic h of the revolution frequency, this will result in the time coordinates being relative to the ideal revolution period, $T_{rf} * h$. If you have multiple rf cavities in a ring, you need only use this feature on one of them. Also, you can use `CHANGE_T=1` if you simply prefer to have the offset time coordinates in output files and analysis.

N.B.: *Do not use* `CHANGE_T=1` *if you have rf cavities that are not at harmonics of one another or if you have other time-dependent elements that are not resonant.*

9.63 RFCW

A combination of RFCA, WAKE, TRWAKE, and LSCDRIFT.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
CELL_LENGTH	M	double	0.0	cell length (used to scale wakes, which are assumed to be given for a cell)
VOLT	V	double	0.0	voltage
PHASE	DEG	double	0.0	phase
FREQ	Hz	double	500000000	frequency
Q		double	0.0	cavity Q (for cavity that charges up to voltage from 0)
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	see RFCA documentation
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

A combination of RFCA, WAKE, TRWAKE, and LSCDRIFT.

Parameter Name	Units	Type	Default	Description
WYCOLUMN		STRING	NULL	column containing y Green function
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	Use Savitzky-Golay filter to 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?
LSC		long	0	Include longitudinal space-charge impedance?
LSC_BINS		long	1025	Number of bins for LSC calculations
LSC_INTERPOLATE		long	1	Interpolate computed LSC wake?
LSC_HIGH_FREQUENCY_CUTOFF0		double	-1	Spatial frequency at which smoothing filter begins for LSC. If not positive, no frequency filter smoothing is done. Frequency is in units of Nyquist (0.5/binsize).
LSC_HIGH_FREQUENCY_CUTOFF1		double	-1	Spatial frequency at which smoothing filter is 0 for LSC. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
LSC_RADIUS_FACTOR		double	1.7	LSC radius is (Sx+Sy)/2*RADIUS_FACTOR
WAKES_AT_END		long	0	Do wake kicks at end of segment (for backward compatibility)?

A combination of RFCA, WAKE, TRWAKE, and LSCDRIFT.

Parameter Name	Units	Type	Default	Description
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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, as well as longitudinal space charge.

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.

This element simulates longitudinal space charge using the method described in [22]. This is based on the longitudinal space charge impedance per unit length

$$Z_{lsc}(k) = \frac{iZ_0}{\pi k r_b^2}\left[1 - \frac{kr_b}{\gamma}K_1\left(\frac{kr_b}{\gamma}\right)\right] \tag{46}$$

9.64 RFDF

A simple traveling or standing wave deflecting RF cavity.
Parallel capable? : yes

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 (odd integer)
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
STANDING_WAVE		long	0	If nonzero, then cavity is standing wave.
VOLTAGE_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving voltage waveform factor vs time
VOLTAGE_PERIODIC		long	0	If non-zero, voltage waveform is periodic with period given by time span.
ALIGN_WAVEFORMS		long	0	If non-zero, waveforms' t=0 is aligned with first bunch arrival time.
VOLTAGE_NOISE		double	0.0	Rms fractional noise level for voltage.
PHASE_NOISE	*DEG*	double	0.0	Rms noise level for phase.
GROUP_VOLTAGE_NOISE		double	0.0	Rms fractional noise level for voltage linked to group.
GROUP_PHASE_NOISE	*DEG*	double	0.0	Rms noise level for phase linked to group.
VOLTAGE_NOISE_GROUP		long	0	Group number for voltage noise.
PHASE_NOISE_GROUP		long	0	Group number for phase noise.
START_PASS		long	-1	If non-negative, pass on which to start modeling cavity.

RFDF continued

A simple traveling or standing wave deflecting RF cavity.

Parameter Name	Units	Type	Default	Description
END_PASS		long	-1	If non-negative, pass on which to end modeling cavity.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.65 RFMODE

A simulation of a beam-driven TM monopole mode of an RF cavity.
Parallel capable? : yes

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)
FREQ_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving frequency/f0 vs time, where f0 is the frequency given with the FREQ parameter

RFMODE continued

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

Parameter Name	Units	Type	Default	Description
Q_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving qualityFactor/Q0 vs time, where Q0 is the quality factor given the the Q parameter.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
BINLESS		long	0	If nonzero, use algorithm that doesn't requiring binning. Best for few particles, widely spaced.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.66 RFTM110

Tracks through a TM110-mode (deflecting) rf cavity with all magnetic and electric field components.
NOT RECOMMENDED—See below.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
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	peak deflecting voltage
PHASE_REFERENCE		long	0	phase reference number (to link with other time-dependent elements)
VOLTAGE_WAVEFORM		STRING	NULL	<filename>=<x>+<y> form specification of input file giving voltage waveform factor vs time
VOLTAGE_PERIODIC		long	0	If non-zero, voltage waveform is periodic with period given by time span.
ALIGN_WAVEFORMS		long	0	If non-zero, waveforms' t=0 is aligned with first bunch arrival time.
VOLTAGE_NOISE		double	0.0	Rms fractional noise level for voltage.
PHASE_NOISE	*DEG*	double	0.0	Rms noise level for phase.
GROUP_VOLTAGE_NOISE		double	0.0	Rms fractional noise level for voltage linked to group.
GROUP_PHASE_NOISE	*DEG*	double	0.0	Rms noise level for phase linked to group.
VOLTAGE_NOISE_GROUP		long	0	Group number for voltage noise.
PHASE_NOISE_GROUP		long	0	Group number for phase noise.
START_PASS		long	-1	If non-negative, pass on which to start modeling cavity.
END_PASS		long	-1	If non-negative, pass on which to end modeling cavity.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

NB: Although this element is correct insofar as it uses the fields for a pure TM110 mode, it is recommended that the `RFDF` element be used instead. In a real deflecting cavity with entrance and exit tubes, the deflecting mode is a hybrid TE/TM mode, in which the deflection has no dependence on the radial coordinate.

To derive the field expansion, we start with some results from Jackson[17], section 8.7. The

longitudinal electric field for a TM mode is just

$$E_z = -2iE_0\Psi(\rho,\phi)\cos\left(\frac{p\pi z}{d}\right)e^{-i\omega t}, \tag{47}$$

where p is an integer, d is the length of the cavity, and we use cylindrical coordinates (ρ,ϕ,z). The factor of $-2i$ represents a choice of sign and phase convention. We are interested in the TM110 mode, so we set $p=0$. In this case, we have

$$E_x = E_y = 0 \tag{48}$$

and (using CGS units)

$$\vec{H} = -2iE_0\frac{i\epsilon\omega}{ck^2}\hat{z}\times\nabla\Psi e^{-i\omega t}. \tag{49}$$

For a cylindrical cavity, the function Ψ for the $m=1$ aximuthal mode is

$$\Psi(\rho,\phi) = J_1(k\rho)\cos\phi, \tag{50}$$

where $k = x_{11}/R$, x_{11} is the first zero of $J_1(x)$, and R is the cavity radius. We don't need to know the cavity radius, since $k=\omega/c$, where ω is the resonant frequency. By choosing $\cos\phi$ for the aximuthal dependence, we'll get a magnetic field primarily in the vertical direction.

In MKS units, the magnetic field is

$$\vec{B} = \frac{2E_0}{kc}e^{-i\omega t}\left(\hat{\rho}\frac{J_1(k\rho)}{\rho}\sin\phi + \hat{\phi}\cos\phi\frac{\partial J_1(k\rho)}{\partial\rho}\right). \tag{51}$$

Using `mathematica`, we expanded these expressions to sixth order in $k*\rho$. Here, we present only the expressions to second order. Taking the real parts only, we now have

$$\begin{aligned}
E_z &\approx E_0 k\rho\cos\phi\sin\omega t && (52)\\
cB_\rho &\approx E_0\left(1-\frac{k^2\rho^2}{8}\right)\sin\phi\cos\omega t && (53)\\
cB_\phi &\approx E_0\left(1-\frac{3k^2\rho^2}{8}\right)\cos\phi\cos\omega t && (54)
\end{aligned}$$

The Cartesian components of $\vec{B}$ can be computed easily

$$\begin{aligned}
cB_x &= cB_\rho\cos\phi - cB_\phi\sin\phi && (55)\\
&= \frac{E_0}{4}\rho^2k^2\cos\phi\sin\phi\cos\omega t && (56)\\
cB_y &= cB_\rho\sin\phi + cB_\phi\cos\phi && (57)\\
&= E_0\left(1-\frac{k^2\rho^2(2\cos^2\phi+1)}{8}\right)\cos\omega t && (58)
\end{aligned}$$

The Lorentz force on an electron is $F = -eE_z\hat{z} - ec\vec{\beta}\times\vec{B}$, giving

$$\begin{aligned}
F_x/e &= \beta_z cB_y && (59)\\
F_y/e &= -\beta_z cB_x && (60)\\
F_z/e &= -E_z - \beta_x cB_y + \beta_y cB_x && (61)
\end{aligned}$$

We see that for $\rho \to 0$, we have $E_z = 0$, $B_x = 0$, and

$$cB_y = E_0 \cos \omega t. \tag{62}$$

Hence, for $\omega t = 0$ and $E_0 > 0$ we have $F_x > 0$. This explains our choice of sign and phase convention above. Indeed, owing to the factor of 2, we have a peak deflection of eE_0L/E, where L is the cavity length and E the beam energy. Thus, if $V = E_0L$ is specified in volts, and the beam energy expressed in electron volts, the deflection is simply the ratio of the two. As a result, we've chosen to parametrize the deflection strength simply by referring to the "deflecting voltage," V.

9.67 RFTMEZ0

A TM-mode RF cavity specified by the on-axis Ez field.
Parallel capable? : no

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
DZ	*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. If omitted, data is assumed to be for r=0 and an on-axis expansion is performed.

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

Parameter Name	Units	Type	Default	Description
SOLENOID_BZCOLUMN		STRING	NULL	column containing Bz values for solenoid map.
SOLENOID_BRCOLUMN		STRING	NULL	column containing Br values for solenoid map. If omitted, data is assumed to be for r=0 and an on-axis expansion is performed.
SOLENOID_FACTOR		double	1	factor by which to multiply solenoid fields.
SOLENOID_DX	*M*	double	0.0	misalignment
SOLENOID_DY	*M*	double	0.0	misalignment
SOLENOID_DZ	*M*	double	0.0	misalignment
SOLENOID_ETILT	*RAD*	double	0.0	misalignment
SOLENOID_EYAW	*RAD*	double	0.0	misalignment
SOLENOID_EPITCH	*RAD*	double	0.0	misalignment
BX_STRAY		double	0.0	Uniform stray horizontal field
BY_STRAY		double	0.0	Uniform stray vertical field
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")
FIELD_TEST_FILE		STRING	NULL	filename for output of test fields (r=0)
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.68 RMDF

A linearly-ramped electric field deflector, using an approximate analytical solution FOR LOW ENERGY PARTICLES.
Parallel capable? : no

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.69 ROTATE

An element that rotates the beam coordinates about the longitudinal axis.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
TILT	*RAD*	double	0.0	rotation about longitudinal axis
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.70 SAMPLE

An element that reduces the number of particles in the beam by interval-based or random sampling.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
FRACTION		double	1	fraction to keep
INTERVAL		long	1	interval between sampled particles
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.71 SBEN

A sector dipole implemented as a matrix, up to 2nd order.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

Special note about splitting dipoles: when dipoles are long, it is common to want to split them into several pieces, to get a better look at the interior optics. When doing this, care must be exercised not to change the optics. `elegant` has some special features that are designed to reduce

or manage potential problems. At issue is the need to turn off edge effects between the portions of the same dipole.

First, one can simply use the `divide_elements` command to set up the splitting. Using this command, `elegant` takes care of everything.

Second, one can use a series of dipoles *with the same name*. In this case, elegant automatically turns off interior edge effects. This is true when the dipole elements directly follow one another or are separated by a MARK element.

Third, one can use a series of dipoles with different names. In this case, you must also use the `EDGE1_EFFECTS` and `EDGE2_EFFECTS` parameters to turn off interior edge effects.

9.72 SCATTER

A scattering element to add gaussian random numbers to particle coordinates.
Parallel capable? : yes

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
PROBABILITY		double	1	Probability that any particle will be selected for scattering.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.73 SCMULT

Tracks through a zero length multipole to simulate space charge effects
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates transverse space charge (SC) kick using K.Y. Ng's formula (K.Y. Ng, FNAL, private communication). **N.B.: This element is not designed for space charge calculations in guns or linacs. It is only intended for simulating space charge in rings.**

The linear SC force is given by:

$$\Delta x' = \frac{K_{sc} L e^{-z^2/(2\sigma_z^2)}}{\sqrt{2\pi}\sigma_z} \frac{x}{\sigma_x(\sigma_x + \sigma_y)}$$

$$\Delta y' = \frac{K_{sc} L e^{-z^2/(2\sigma_z^2)}}{\sqrt{2\pi}\sigma_z} \frac{y}{\sigma_y(\sigma_x + \sigma_y)} \tag{63}$$

where $K_{sc} = \frac{2Nr_e}{\gamma^3\beta^2}$, L is the integrating length, $\sigma_{x,y,z}$ are rms beam size.

The non-linear SC force is given by:

$$\Delta x' = \frac{K_{sc} L e^{-z^2/(2\sigma_z^2)}}{2\sigma_z\sqrt{\sigma_x^2 - \sigma_y^2}} Im\left[w\left(\frac{x+iy}{\sqrt{2(\sigma_x^2-\sigma_y^2)}}\right) - e^{-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}} w\left(\frac{x\frac{\sigma_y}{\sigma_x} + iy\frac{\sigma_x}{\sigma_y}}{\sqrt{2(\sigma_x^2-\sigma_y^2)}}\right)\right]$$

$$\Delta y' = \frac{K_{sc} L e^{-z^2/(2\sigma_z^2)}}{2\sigma_z\sqrt{\sigma_x^2 - \sigma_y^2}} Re\left[w\left(\frac{x+iy}{\sqrt{2(\sigma_x^2-\sigma_y^2)}}\right) - e^{-\frac{x^2}{2\sigma_x^2}-\frac{y^2}{2\sigma_y^2}} w\left(\frac{x\frac{\sigma_y}{\sigma_x} + iy\frac{\sigma_x}{\sigma_y}}{\sqrt{2(\sigma_x^2-\sigma_y^2)}}\right)\right] \tag{64}$$

where $w(z)$ is the complex error function

$$w(z) = e^{-z^2}\left[1 + \frac{2i}{\sqrt{\pi}}\int_0^z e^{\zeta^2} d\zeta\right] \tag{65}$$

Equation 64 appear to diverge when $\sigma_x = \sigma_y$. In fact, this is not true, because the expressions inside the square brackets will provide zero too at $\sigma_x = \sigma_y$ to cancel the poles outside. In our code, we calculate this equation at $1.01\sigma_x$ and $0.99\sigma_x$, and average the total effects.

To invoke the calculation, one must use set up command "insert_sceffects" proceed "run_setup" and "Twiss_output" command proceed "track".

9.74 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.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
XO	M	double	0.0	radiation length
ELASTIC		long	0	elastic scattering? If zero, then particles will lose energy due to material.
ENERGY_STRAGGLE		long	0	Use simple-minded energy straggling model? Ignored for ELASTIC scattering.
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
POSITION	M	double	0.0	position of edge
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
INSERT_FROM		STRING	NULL	direction from which inserted (+x, -x, +y, -y)
DIRECTION		long	-1	obsolete, use insert_from instead
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.75 SCRIPT

An element that allows transforming the beam using an external script.
Parallel capable? : no

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.
USE_CSH		long	1	Use C-shell for execution (may be slower)?
VERBOSITY		long	0	Set the verbosity level.
START_PASS		long	-1	Start script action on this pass. Before that, behaves like a drift space.
ON_PASS		long	-1	Perform script action only on this pass. Other than that, behaveslike a drift space.
DIRECTORY		STRING	NULL	Directory in which to place input and output files. If blank, the present working directory is used.
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.
DRIFT_MATRIX		long	0	If nonzero, then for matrix calculations the element is treated as a drift space.

SCRIPT continued

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

Parameter Name	Units	Type	Default	Description
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
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

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

Parameter Name	Units	Type	Default	Description
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
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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows expanding `elegant` by using external scripts (or programs) as elements in a beamline. Here are requirements for the script:

- It must be executable from the commandline.
- It must read the initial particle distribution from an SDDS file. This file will have the usual columns that an `elegant` phase-space output file has, along with the parameter `Charge` giving the beam charge in Coulombs. The file will contain a single data page.
- It must write the final particle distribution to an SDDS file. This file should have all of the columns and parameters that appear in the initial distribution file. Additional columns and parameters will be ignored, as will all pages but the first.

The SCRIPT element works best if the script accepts commandline arguments. In this case, the COMMAND parameter is used to provide a template for creating a command to run the script. The COMMAND string may contain the following substitutable fields:

1. %i — Will be replaced by the name of the input file to the script. (elegant writes the initial particle distribution to this file.)

2. %o — Will be replaced by the name of the output file from the script. (elegant expects the script to write the final particle distribution to this file.)

3. %np0, %np1, ..., %np9 — Will be replaced by the value of Numerical Parameter 0, 1, ..., 9. This can be used to pass to the script values that are parameters of the element definition. For example, if one wanted to vary parameters or add errors to the parameter, one would use this facility.

4. %sp0, %sp1, ..., %sp9 — Will be replaced by the value of String Parameter 0, 1, ..., 9. This can be used to pass to the script values that are parameters of the element definition.

Here's an example of a SCRIPT COMMAND:

```
myScript -input %i -output %o -accuracy %np0 -type %sp0
```

In this example, the script myScript takes four commandline arguments, giving the names of the input and output files, an accuracy requirement, and a type specifier. By default, elegant will choose unique, temporary filenames to use in communicating with the script. The actual command when executed might be something like

```
myScript -input tmp391929.1 -output tmp391929.2 -accuracy 1.5e-6 -type scraper
```

where for this example I've assumed NP0=1.5e-6 and SP0=''scraper''.

If you have a program (e.g., a FORTRAN program) that does not accept commandline arguments, you can easily wrap it in a Tcl/Tk simple script to handle this. Alternatively, you can force elegant to use specified files for communicating with the script. This is done using the ROOTNAME, INPUT_EXTENSION, and OUTPUT_EXTENSION parameters. So if your program was crass and it expected its input (output) in files crass.in (crass.out), then you'd use

```
S1: script,command=''crass'',rootname=''crass'',input_extension=''in'',&
output_extension=''out''
```

For purposes of computing concatenated transport matrices, Twiss parameters, response matrices, etc., elegant will perform initial tracking through the SCRIPT element using an ensemble of 25 particles. If this is not desirable, then set the parameter DRIFT_MATRIX to a non-zero value. This will force elegant to treat the element as a drift space for any calculations that involve transport matrices. Examples of where one might want to use this feature would be a SCRIPT that involves randomization (e.g., scattering), particle loss, or particle creation.

9.76 SEXT

A sextupole implemented as a matrix, up to 3rd order
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.77 SOLE

A solenoid implemented as a matrix, up to 2nd order.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
KS	RAD/M	double	0.0	geometric strength, -Bs/(B*Rho)
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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.78 SREFFECTS

Simulation of synchrotron radiation effects (damping and quantum excitation).
Parallel capable? : yes

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		double	0.0	reference equilibrium fractional momentum spread
DDELTAREF		double	0.0	reference fractional momentum change per turn due to SR (negative value)
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, less rf effects?
QEXCITATION		long	1	include quantum excitation?
LOSSES		long	1	include average losses?
CUTOFF		double	100	cutoff (in sigmas) for gaussian random numbers
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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`, `DDELTAREF`, 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. This will occur only if `PREF=0`. The `COUPLING` parameter can be used to change the partitioning of quantum excitation between the horizontal and vertical planes.

In versions 19.0 and later, `elegant` includes the effect of `SREFFECTS` on the closed orbit. This presents a dilemna when automatic initialization is used, because in order to perform automatic initialization, `elegant` has to compute the optics functions. However, it must determine the closed orbit to compute the optics functions. The solution to this is for the user to pre-compute the twiss parameters and radiation integrals using `twiss_output` with `output_at_each_step=0`. The user is free to subsequently give `twiss_output` with `output_at_each_step=1` to obtain the results on the closed orbit.

N.B.: Computation of Twiss parameters does not fully include the effects of synchrotron radiation losses when these are imposed using `SREFFECTS` elements. If `PREF=0` (automatic initialization), these effects are completely missing. If `PREF` is non-zero, then `elegant` will use the `DDELTAREF` parameter to compute the energy offset from the element, and thus its effect on the beam trajectory.

9.79 STRAY

A stray field element with local and global components. Global components are defined relative to the initial beamline direction.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates stray fields. These fields are considered perturbations, in that they change the trajectory (or orbit), but not the floor coordinates. Local stray fields (`LBX` and `LBY`) are referenced to the local coordinate system. Global stray fields (`GBX`, `GBY`, `GBZ`) are referenced to the global coordinate system, which coincides with the local coordinate system only at the start of the beamline (unless there is no bending, in which case the two systems are identical).

9.80 TFBDRIVER

Driver for a transverse feedback loop
Parallel capable? : no

Parameter Name	Units	Type	Default	Description
ID		STRING	NULL	System identifier
STRENGTH		double	0.0	Strength factor
KICK_LIMIT	*RAD*	double	0.0	Limit on applied kick
DELAY		long	0	Delay (in turns)
OUTPUT_FILE		STRING	NULL	File for logging filter output and driver output
A0		double	1	Filter coefficient
A1		double	0.0	Filter coefficient
A2		double	0.0	Filter coefficient
A3		double	0.0	Filter coefficient
A4		double	0.0	Filter coefficient
A5		double	0.0	Filter coefficient
A6		double	0.0	Filter coefficient
A7		double	0.0	Filter coefficient
A8		double	0.0	Filter coefficient
A9		double	0.0	Filter coefficient
A10		double	0.0	Filter coefficient
A11		double	0.0	Filter coefficient
A12		double	0.0	Filter coefficient
A13		double	0.0	Filter coefficient
A14		double	0.0	Filter coefficient
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is used together with the `TFBPICKUP` element to simulate a digital transverse feedback system. Each `TFBDRIVER` element must have a unique identification string assigned to it using the `ID` parameter. The same identifier must be used on a `TFBPICKUP` element. This is the pickup from which the driver gets its signal. Each pickup may feed more than one driver, but a driver can use only one pickup.

A 15-term FIR filter can be defined using the `A0` through `A14` parameters. The output of the filter is simply $\sum_{i=0}^{14} a_i P_i$, where P_i is the pickup filter output from i turns ago. The output of the filter is optionally delayed by the number of turns given by the `DELAY` parameter.

To some extent, the `DELAY` is redundant. For example, the filter $a_0 = 0, a_1 = 1$ with a delay of 0 is equivalent to $a_0 = 1, a_1 = 0$ with a delay of 1. However, for long delays or delays combined with many-term filters, the `DELAY` feature must be used.

The output of the filter is multiplied by the `STRENGTH` parameter to get the kick to apply to the beam. The `KICK_LIMIT` parameter provides a very basic way to simulate saturation of the kicker output.

See Section 7.2.14 of *Handbook of Accelerator Physics and Engineering* (Chao and Tigner, eds.) for a discussion of feedback systems.

9.81 TFBPICKUP

Pickup for a transverse feedback loop
Parallel capable? : no

Parameter Name	Units	Type	Default	Description
ID		STRING	NULL	System identifier
PLANE		STRING	x	"x" or "y"
RMS_NOISE	M	double	0.0	RMS noise to add to position readings.
A0		double	0.0	Filter coefficient
A1		double	0.0	Filter coefficient
A2		double	0.0	Filter coefficient
A3		double	0.0	Filter coefficient
A4		double	0.0	Filter coefficient
A5		double	0.0	Filter coefficient
A6		double	0.0	Filter coefficient
A7		double	0.0	Filter coefficient
A8		double	0.0	Filter coefficient
A9		double	0.0	Filter coefficient
A10		double	0.0	Filter coefficient
A11		double	0.0	Filter coefficient
A12		double	0.0	Filter coefficient
A13		double	0.0	Filter coefficient
A14		double	0.0	Filter coefficient
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element is used together with the `TFBDRIVER` element to simulate a digital transverse feedback system. Each `TFBPICKUP` element must have a unique identification string assigned to it using the `ID` parameter. This is used to identify which drivers get signals from the pickup.

A 15-term FIR filter can be defined using the `A0` through `A14` parameters. The input to the filter is the turn-by-turn beam centroid at the pickup location. The output of the filter is simply $\sum_{i=0}^{14} a_i C_i$, where C_i is the position from i turns ago. Note that $\sum_{i=0}^{14} a_i$ must be zero. Otherwise, the system will attempt to correct the DC orbit. The output of the filter is the input to the driver element(s).

See Section 7.2.14 of *Handbook of Accelerator Physics and Engineering* (Chao and Tigner, eds.) for a discussion of feedback systems.

9.82 TMCF

A numerically-integrated accelerating TM RF cavity with spatially-constant fields.
Parallel capable? : no

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, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.83 TRCOUNT

An element that defines the point from which transmission calculations are made.
Parallel capable? : no

Parameter Name	Units	Type	Default	Description
DUMMY		long	0	
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.84 TRFMODE

A simulation of a beam-driven TM dipole mode of an RF cavity.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
RA	Ohm/m	double	0.0	shunt impedance
RS	Ohm/m	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
SAMPLE_INTERVAL		long	1	passes between output to RECORD file
PER_PARTICLE_OUTPUT		long	0	If non-zero, then in BINLESS mode, provides per-particle output of RECORD data.
RECORD		STRING	NULL	output file for cavity data
SINGLE_PASS		long	0	if nonzero, don't accumulate field from pass to pass
RIGID_UNTIL_PASS		long	0	don't affect the beam until this pass
DX	M	double	0.0	misalignment
DY	M	double	0.0	misalignment
XFACTOR		double	1	factor by which to multiply shunt impedances
YFACTOR		double	1	factor by which to multiply shunt impedances
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
BINLESS		long	0	If nonzero, use algorithm that doesn't requiring binning. Best for few particles, widely spaced.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.85 TRWAKE

Transverse wake specified as a function of time lag behind the particle.
Parallel capable? : yes

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		double	1	factor by which to multiply both wakes
XFACTOR		double	1	factor by which to multiply x wake
YFACTOR		double	1	factor by which to multiply y wake
N_BINS		long	128	number of bins for current histogram
INTERPOLATE		long	0	interpolate wake?
SMOOTHING		long	0	Use Savitzky-Golay filter to 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
TILT	*RAD*	double	0.0	rotation about longitudinal axis
XPOWER		long	1	Power of x that x kick depends on.
YPOWER		long	1	Power of y that y kick depends on.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the wake to full strength.

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

Parameter Name	Units	Type	Default	Description
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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. Time values must be equally spaced.

The sign convention for W_q (q being x or y) is as follows: a particle with $q > 0$ will impart a positive kick ($\Delta q' > 0$) to a trailing particle following t seconds behind if $W_q(t) > 0$. A physical wake function should be zero at $t = 0$ and also be initially positive as t increases from 0.

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`).

The `XFACTOR` and `YFACTOR` parameters can be used to adjust the strength of the wakes if the location at which you place the `TRWAKE` element has different beta functions than the location at which the object that causes the wake actually resides.

The `XPOWER` and `YPOWER` parameters can be used to change the dependence of the wake on the x and y coordinates, respectively, of the particles. Normally, `XPOWER=1` and `YPOWER=1`. This is an ordinary dipole wake in a (supposedly) symmetric chamber.

If you have an asymmetric chamber, then you will have a transverse wake kick even if the beam is centered. (Of course, you'll need a 3-D wake code like GdfidL or MAFIA to compute this wake.) This part of the transverse wake is described with `XPOWER=0` and `YPOWER=0`. It will result in an orbit distortion, but conceivably could have other effects, such as emittance dilution. If `XPOWER=0` or `YPOWER=0`, the units for the x or y wake (respectively) must be V/C. A negative value of the wake corresponds to a kick toward negative x (or y).

9.86 TUBEND

A special rectangular bend element for top-up backtracking.
Parallel capable? : yes

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
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.87 TWISS

Sets Twiss parameter values.
Parallel capable? : no

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
ETAX	M	double	0.0	horizontal eta function
ETAXP		double	0.0	slope of horizontal eta function
ETAY	M	double	0.0	vertical eta function
ETAYP		double	0.0	slope of vertical eta function
FROM_BEAM		long	0	compute transformation from tracked beam properties instead of Twiss parameters?
COMPUTE_ONCE		long	0	compute transformation only for first beam or lattice functions?
APPLY_ONCE		long	1	apply correction only on first pass through for each beam?
VERBOSE		long	0	if non-zero, print extra information about transformations
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This elements allows transformation of the twiss parameters of a beam with a first-order matrix. The matrix is computed in various ways based on initial and final twiss parameters. Depending on how you set it up, the final twiss parameters for your beam may not be the twiss parameters you specify.

The twiss parameter values `BETAX`, `BETAY`, etc. specified in the element definition specify the target values of the transformation. To completely specify the transformation, one must know the initial values as well.

Lattice-Function-Based Transformation

If `FROM_BEAM` is zero, which is the default, then the initial values are taken from the incoming lattice functions computed by `twiss_output`. This provides a way to transform the lattice functions between two parts of a transport line without designing intervening optics. A beam that is matched at the beginning of the transport line will remain matched. A beam that is mismatched at the beginning of the transport line *will not* be matched after the `TWISS` element.

By default, each time the twiss parameters are recomputed, the transformation is updated to

maintain the desired lattice functions at the exit of the `TWISS` element. Setting `COMPUTE_ONCE` to a non-zero value specifies that `elegant` should compute the transformation matrix only once, i.e., for the first set of computed lattice functions.

By default, the transformation is applied to the beam only the first time it passes the element. Setting `APPLY_ONCE` to a zero will result in application of the transformation at each pass.

Beam-Ellipse-Based Transformation

If `FROM_BEAM` is non-zero, the the initial values for the transformation are computed from a beam. This provides a way to transform the beam ellipse to the desired twiss parameters irrespective of the lattice. The results from `twiss_output` will not necessarily be matched downstream of this element. Only if the beam ellipse and lattice ellipse are the same will this occur.

By default, each time a new beam is generated, the transformation will be updated to maintain the desired beam ellipse at the exit of the `TWISS` element. Setting `COMPUTE_ONCE` to a non-zero value specifies that `elegant` should compute the transformation matrix only once, i.e., for the first beam it sees.

By default, the transformation is applied to the beam only the first time it passes the element. Setting `APPLY_ONCE` to a zero will result in application of the transformation at each pass. This would make sense, for example, if the `TWISS` element was filling in for a section of a ring. It wouldn't make sense if the `TWISS` element was being used to match the beam from a transport line to a ring.

9.88 TWLA

A numerically-integrated first-space-harmonic traveling-wave linear accelerator.
Parallel capable? : no

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, non-adaptive 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?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.89 TWMTA

A numerically-integrated traveling-wave muffin-tin accelerator.
Parallel capable? : no

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")
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.90 TWPL

A numerically-integrated traveling-wave stripline deflector.
Parallel capable? : no

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, non-adaptive runge-kutta, modified midpoint)
FIDUCIAL		STRING	t,median	{t|p},{median|min|max|ave|first|light} (e.g., "t,median")
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.91 UKICKMAP

An undulator kick map (e.g., using data from RADIA).
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
FIELD_FACTOR		double	1	Factor by which to multiply the magnetic fields.
INPUT_FILE		STRING	NULL	Name of SDDS file with undulator kickmap data.
N_KICKS		long	1	Number of kicks into which to split the element.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element provides simulation of undulators using kick maps [27]. A script (`km2sdds`) is provided with the `elegant` distribution to translate RADIA [28] output into SDDS for use by `elegant`.

The input file has the following columns:

- `x` — Horizontal position in meters.
- `y` — Vertical position in meters.
- `xpFactor` — Horizontal kick factor in T^2m^2.
- `ypFactor` — Vertical kick factor in T^2m^2.

The values of `x` and `y` must be laid out on a grid of equispaced points. It is assumed that the data is ordered such that `x` varies fastest. This can be accomplished with the command

```
% sddssort -column=y,increasing -column=x,increasing input1.sdds input2.sdds
```

where `input1.sdds` is the original (unordered) file and `input2.sdds` is the new file, which would be used with `UKICKMAP`.

This was requested by W. Guo (BNL), who also assisted with the implementation and debugging.

9.92 VKICK

A vertical steering dipole implemented as a matrix, up to 2nd order.
Parallel capable? : yes

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?
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.93 VMON

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

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)
CO_FITPOINT		long	0	If nonzero, then closed orbit value is placed in variable <name>#<occurence>.yco
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

9.94 WAKE

Longitudinal wake specified as a function of time lag behind the particle.
Parallel capable? : yes

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	Use Savitzky-Golay filter to 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?
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the wake to full strength.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

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.

A positive value of $W(t)$ results in energy *loss*. A physical wake function should be positive at $t = 0$.

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`).

The algorithm for the wake element is as follows:

1. Compute the arrival time of each particle at the wake element. This is necessary because `elegant` uses the longitudinal coordinate $s = \beta ct$.

2. Find the mean, minimum, and maximum arrival times (t_{mean}, t_{min}, and t_{max}, respectively). If $t_{max} - t_{min}$ is greater than the duration of the wakefield data, then `elegant` either exits (default) or issues a warning (if `ALLOW_LONG_BEAM` is nonzero). In the latter case, that part of the beam that is furthest from t_{mean} is ignored for computation of the wake.

3. If the user has specified a fixed number of bins (not recommended), then `elegant` centers those bins on t_{mean}. Otherwise, the binning range encompasses $t_{min} - \Delta t$ to $t_{max} + \Delta t$, where Δt is the spacing of data in the wake file.

4. Create the arrival time histogram. If any particles are outside the histogram range, issue a warning.

5. If `SMOOTHING` is nonzero, smooth the arrival time histogram.

6. Convolve the arrival time histogram with the wake function.

7. Multiply the resultant wake by the charge and any user-defined factor.

8. Apply the energy changes for each particle. This is done in such a way that the transverse momentum are conserved.

9. If `CHANGE_P0` is nonzero, change the reference momentum of the beamline to match the average momentum of the beam.

9.95 WATCH

A beam property/motion monitor–allowed modes are centroid, parameter, coordinate, and fft.
Parallel capable? : no

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
END_PASS		long	-1	pass on which to end (inclusive). Ignored if negative.
FILENAME		STRING		output filename, possibly incomplete (see below)
LABEL		STRING		output label
MODE		STRING	coordinates	coordinate, parameter, centroid, or fft. For fft mode, you may add a space and a qualifer giving the window type: hanning (default), parzen, welch, or uniform.
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)
DISABLE		long	0	If nonzero, no output will be generated.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

The output filename may be an incomplete filename. In the case of the `WATCH` point element, this means it may contain one instance of the string format specification "%s" and one occurence of an integer format specification (e.g., "%ld"). `elegant` will replace the format with the rootname (see `run_setup`) and the latter with the element's occurrence number. For example, suppose you

had a repetitive lattice defined as follows:

```
W1: WATCH,FILENAME=''%s-%03ld.w1''
Q1: QUAD,L=0.1,K1=1
D: DRIFT,L=1
Q2: QUAD,L=0.1,K1=-1
CELL: LINE=(W1,Q1,D,2*Q2,D,Q1)
BL: LINE=(100*CELL)
```

The element `W1` appears 100 times. Each instance will result in a new file being produced. Successive instances have names like "*rootname*-001.w1", "*rootname*-002.w1", "*rootname*-003.w1", and so on up to "*rootname*-100.w1". (If instead of "%03ld" you used "%ld", the names would be "*rootname*-1.w1", "*rootname*-2.w1", etc. up to "*rootname*-100.w1". This is generally not as convenient as the names don't sort into occurrence order.)

The files can easily be plotted together, as in

```
% sddsplot -column=t,p *-???.w1 -graph=dot -separate
```

They may also be combined into a single file, as in

```
% sddscombine *-???.w1 all.w1
```

In passing, note that if `W1` was defined as

```
W1: WATCH,FILENAME=''%s.w1''
```

or

```
W1: WATCH,FILENAME=''output.w1''
```

only a single file would be produced, containing output from the last instance only.

N.B.: confusion sometimes occurs about some of the quantities related to the `s` coordinate in this file when in parameter mode. Please see Section 4 above.

9.96 WIGGLER

A wiggler or undulator for damping or excitation of the beam.
Parallel capable? : yes

Parameter Name	Units	Type	Default	Description
L	M	double	0.0	length
RADIUS	M	double	0.0	peak bending radius
K		double	0.0	Dimensionless strength parameter. Ignored if radius is nonzero.
DX		double	0.0	Misaligment.
DY		double	0.0	Misaligment.
DZ		double	0.0	Misaligment.
TILT		double	0.0	Rotation about beam axis.
POLES		long	`0`	number of wiggler poles
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element simulates a wiggler or undulator. There are two aspects to the simulation: the effect on radiation integrals and the vertical focusing. Both are included as of release 15.2 of elegant.

N.B.: at present this element is *not* included in computations of beam moments (`moments_output` command).

If the number of poles should be an odd integer, we include half-strength end poles to match the dispersion, but only for the radiation integral calculation. For the focusing, we assume all the poles are full strength (i.e., a pure sinusoidal variation). If the number of poles is an even integer, no special end poles are required, but we make the unphysical assumption that the field at the entrance (exit) of the device jumps instantaneously from 0 (full field) to full field (0).

The radiation integrals were computed analytically using Mathematica, including the variation of the horizontal beta function and dispersion. For an odd number of poles, half-strength end-poles are assumed in order to match the dispersion of the wiggler. For an even number of poles, half-length end poles are assumed (i.e., we start and end in the middle of a pole), for the same reason.

The vertical focusing is implemented as a distributed quadrupole-like term (affecting ony the vertical, unlike a true quadrupole). The strength of the quadrupole is (see Wiedemann, *Particle Accelerator Physics II*, section 2.3.2)

$$K_1 = \frac{1}{2\rho^2}, \tag{66}$$

where ρ is the bending radius at the center of a pole. The undulator is focusing in the vertical plane.

The wiggler field strength may be specified either as a peak bending radius ρ (RADIUS parameter) or using the dimensionless strength parameter K (K parameter). These are related by

$$K = \frac{\gamma \lambda_u}{2\pi\rho}, \tag{67}$$

where γ is the relativistic factor for the beam and λ_u is the period length.

9.97 ZLONGIT

A simulation of a single-pass broad-band or functionally specified longitudinal impedance.
Parallel capable? : yes

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
MAX_N_BINS		long	0	Maximum 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	Use Savitzky-Golay filter to smooth current histogram?
SG_ORDER		long	1	Savitzky-Golay filter order for smoothing
SG_HALFWIDTH		long	4	Savitzky-Golay filter halfwidth for smoothing
REVERSE_TIME_ORDER		long	0	Reverse time-order of particles for wake computation?
FACTOR		double	1	Factor by which to multiply impedance.

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

Parameter Name	Units	Type	Default	Description
START_ON_PASS		long	0	The pass on which the impedance effects start.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
HIGH_FREQUENCY_CUTOFF0		double	-1	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	Frequency at which smoothing filter is 0. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows simulation of a longitudinal impedance using a "broad-band" resonator or an impedance function specified in a file. The impedance is defined as the Fourier transform of the wake function

$$Z(\omega) = \int_{-\infty}^{+\infty} e^{-i\omega t} W(t) dt \tag{68}$$

where $i = \sqrt{-1}$, $W(t) = 0 for t < 0$, and $W(t)$ has units of V/C.

For a resonator impedance, the functional form is

$$Z(\omega) = \frac{R_s}{1 + iQ(\frac{\omega}{\omega_r} - \frac{\omega_r}{\omega})}, \tag{69}$$

where R_s is the shunt impedance in $Ohms$, Q is the quality factor, and ω_r is the resonant frequency.

When providing an impedance in a file, the user must be careful to conform to these conventions.

Other notes:

1. The frequency data required from the input file is *not* ω, but rather $f = \omega/(2\pi)$.
2. The default smoothing setting (`SG_HALFWIDTH=4`), may apply too much smoothing. It is recommended that the user vary this parameter if smoothing is employed.

9.98 ZTRANSVERSE

A simulation of a single-pass broad-band or functionally-specified transverse dipole impedance.
Parallel capable? : yes

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
MAX_N_BINS		long	0	Maximum number of bins for current histogram
SMOOTHING		long	0	Use Savitzky-Golay filter to 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

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

Parameter Name	Units	Type	Default	Description
DY	M	double	0.0	misalignment
FACTOR		double	1	Factor by which to multiply x and y impedances.
XFACTOR		double	1	Factor by which to multiply x impedance.
YFACTOR		double	1	Factor by which to multiply y impedance.
WAKES		STRING	NULL	filename for output of wake
WAKE_INTERVAL		long	1	interval in passes at which to output wake
START_ON_PASS		long	0	The pass on which the impedance effects start.
RAMP_PASSES		long	0	Number of passes over which to linearly ramp up the impedance to full strength.
HIGH_FREQUENCY_CUTOFF0		double	-1	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	Frequency at which smoothing filter is 0. If not given, defaults to HIGH_FREQUENCY_CUTOFF0.
GROUP		string	NULL	Optionally used to assign an element to a group, with a user-defined name. Group names will appear in the parameter output file in the column ElementGroup

This element allows simulation of a transverse impedance using a "broad-band" resonator or an impedance function specified in a file. The impedance is defined as the Fourier transform of the wake function

$$Z(\omega) = \int_{-\infty}^{+\infty} e^{-i\omega t} W(t) dt \tag{70}$$

where $i = \sqrt{-1}$, $W(t) = 0 for t < 0$, and $W(t)$ has units of $V/C/m$. Note that there is no factor of i in front of the integral. Thus, in `elegant` the transverse impedance is simply the Fourier transform of the wake. This makes it easy to convert data from a program like ABCI into the wake formalism using `sddsfft`.

For a resonator impedance, the functional form is

$$Z(\omega) = \frac{-i\omega_r}{\omega} \frac{R_s}{1 + iQ(\frac{\omega}{\omega_r} - \frac{\omega_r}{\omega})}, \tag{71}$$

where R_s is the shunt impedance in $Ohms/m$, Q is the quality factor, and ω_r is the resonant frequency.

When providing an impedance in a file, the user must be careful to conform to these conventions. Other notes:

1. The frequency data required from the input file is *not* ω, but rather $f = \omega/(2\pi)$.

2. The default smoothing setting (`SG_HALFWIDTH=4`), may apply too much smoothing. It is recommended that the user vary this parameter if smoothing is employed.

10 Examples

Example runs and post-processing files are available in a separate tar file. The examples are intended to demonstrate program capabilities with minimal work on the user's part. However, they don't pretend to cover all the capabilities.

Each demo is (typically) invoked using a command (usually a C-shell script) that can both run `elegant` and post-process the output. The post-processing is often 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.

The examples are organized into a number of directories and subdirectories. In each area, the user will find a "Notebook" file (a simple ASCII file) that describes the example and how to run it.

Many examples for storage ring simulations reside in the `PAR` subdirectory. The PAR (Particle Accumulator Ring) is a small storage ring in the APS injector that is good for quick examples because of its size.

Here's a helpful tip in searching the examples on UNIX/LINUX systems: suppose one wants to find an example of the `frequency_map` command. One can search all the elegant command files very quickly with this command:

```
find . -name '*.ele' | xargs fgrep frequency_map
```

Similarly, to find all examples that use `CSBEND` elements, one could use

```
find . -name '*.lte' | xargs fgrep -i csbend
```

11 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 Electron Laser Driven by SLAC LINAC".

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

[15] J.D. Bjorken, S.K. Mtingwa, "Intrabeam Scattering," Part. Acc. Vol. 13, 1983, 115-143.

[16] K. Halbach, "First Order Perturbation Effects in Iron-Dominated Two-Dimensional Symmetrial Multipoles", NIM **74-1**, 1969, 147-164.

[17] J. D. Jackson, *Clasical Electrodynamics*, second edition.

[18] G. Ripken, DESY Report No. R1-70/04, 1970 (unpublished).

[19] *Handbook of Accelerator Physics and Engineering*, A. Chao and M. Tigner eds., 1998.

[20] Ya. S. Derbenev, J. Rossbach, E. L. Saldin, V. D. Shiltsev, "Microbunch Radiative Tail-Head Interaction," September 1995, TESLA-FEL 95-05.

[21] A. Xiao *et al.*, "Direct Space-Charge Calculation in `elegant` and its Application to the ILC Damping Ring," Proc. PAC2007, 3456-3458.

[22] Z. Huang *et al.*, Phys. Rev. ST Accel. Beams **7** 074401 (2004).

[23] A. Piwinski, " The Touschek effect in strong focusing storage rings," DESY-98-179, Nov 1998.

[24] A. Xiao *et al.*, "Touschek Effect Calculation and its Application to a Transport Line," Proc. PAC07, 3453-3455 (2007).

[25] W. Warnock, "Shielded Coherent Synchrotron Radiation and Its Effect on Very Short Bunches," SLAC-PUB-5375, 1990.

[26] T. Agoh and K. Yokoya, "Calculation of coherent synchrotron radiation using mesh," Phys. Rev. ST Accel. Beams 7, 054403 (2004).

[27] P. Elleaume, "A New Approach to Electron Beam Dynamics in Undulators and Wigglers," Proc. EPAC 1992, 661-663.

[28] http://www.esrf.eu/Accelerators/Groups/InsertionDevices/Software/Radia