Change Notes for Version 14.4 of elegant

Advanced Photon Source
Michael Borland

June 7, 2001

Version 14.4 of elegant is now available. The most significant enhancement is to the CSRDRIFT element, which now provides more rigorous simulation of CSR in drift spaces. There is also support for body focusing in RFCA and RFCW elements, and two new elements for simulating RF cavities and solenoids from field maps.

1. The CSRDRIFT element has a number of new features. In order of increasing significance, these are:

   (a) Added the OL_MULTIPLIER parameter, which defaults to 1. If you give USE_OVERTAKING_LENGTH=1, then elegant uses the overtaking length as the attenuation length for the CSR wake. This parameter permits you to increase or decrease that attenuation length by the specified factor.

   (b) Added the BUNCHLENGTH_MODE parameter, so that the user can choose how the bunch length is computed for computation of the attenuation length. 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.

   (c) Added the USE_SALDIN54 parameter, which invokes use of a more rigorous method of computing the fall-off of the CSR wake in the drift. As the name suggests, it is based on equation 54 (and 53) from Saldin et al. (NIM A 398 (1997)). I no longer use any other mode for this element. Using USE_OVERTAKING_LENGTH tends to give similar results, however. I also added the SALDIN54_OUTPUT parameter, which allows one to specify a file into which the wake intensity vs z will be written. These equations are for a rectangular bunch; the length of this rectangular bunch is taken to be twice the bunch length computed according to the BUNCHLENGTH_MODE parameter. If your bunch is nearly rectangular, then you probably want BUNCHLENGTH_MODE of "90-percentile".

2. It's now possible to assign multiple errors to one item and get a "final" output file at the same time. In previous versions, this would cause the program to terminate.

3. Added the "string_value" parameter to alter_elements command, allowing alteration of string parameters of elements.

4. A bug was introduced in version 14.3 for the beam generation with bunched_beam. In particular, pair-wise randomization of the generated coordinates was performed by default. This

has been fixed. The randomize_order parameter of the bunched_beam namelist takes three possible values: 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, which retains correlations between x-x' etc but destroys those between x-y, x-p, etc. This is used with Halton sequences to remove banding. Be aware that if you provide values for the dispersion or its slope in the beam definition, that these will be nulled out by use of either type of randomization.

5. The bunched_beam command now allows specifying the longitudinal Twiss parameters, emit_z, beta_z, and alpha_z, instead of the energy spread, bunch length, and normalized correlation. You may also specify the energy spread, bunch length, and alpha_z together, avoiding the use of the troublesome dp_s_coupling parameter. Basically, which values `elegant` uses depends what you set to nonzero values. If you set emit_z, then sigma_dp, sigma_s, and dp_s_coupling are ignored. If you don'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. Paul Emma pointed out that using alpha_z is easier than dp_s_coupling some time ago and I just got around to adding it. 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 $R_{56} < 0$, you must have $\alpha_z < 0$ or $C > 0$.

6. The MAPSOLENOID element was added. It provides numerical integration through solenoid fields specified as Br(z,r) and Bz(z,r). Parameters are as follows:
   (a) L length in meters
   (b) DX, DY misalignment in meters
   (c) N_STEPS number of steps for nonadaptive integration
   (d) INPUTFILE name of SDDS file containing (Br, Bz) vs (r, z). Each page should have values for a fixed r.
   (e) RCOLUMN name of column containing r values
   (f) ZCOLUMN name of column containing z values
   (g) BRCOLUMN name of column containing Br values
   (h) BZCOLUMN name of column containing Bz values
   (i) FACTOR factor by which to multiply fields to get applied fields in Tesla
   (j) ACCURACY integration accuracy for adaptive integration (not recommended)
   (k) METHOD integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint). Recommend non-adaptive runge-kutta.

7. The RFTMEZ0 element was added. This element provides numerical integration through a TM-mode RF cavity, where the fields are specified by Ez(r=0) and an off-axis expansion. Parameters are as follows:
   (a) L length in meters
   (b) FREQUENCY frequency in Hz
   (c) PHASE in radians
   (d) EZ_PEAK Peak on-axis longitudinal electric field at crest (V/m)
   (e) TIME_OFFSET time offset (adds to phase)

(f) PHASE_REFERENCE phase reference number (to link to other time-dependent elements)

(g) DX, DY misalignment in meters

(h) N_STEPS number of steps for nonadaptive integration

(i) RADIAL_ORDER highest order in off-axis expansion

(j) CHANGE_P0 does element change central momentum?

(k) INPUTFILE name of SDDS file containing Ez vs z at r=0

(l) ZCOLUMN name of column containing z values

(m) EZCOLUMN name of column containing Ez values

(n) SOLENOID_FILE Name of SDDS file containing map of Bz and Br vs z and r. Each page contains values for a single r. This is the same organization as the file used by MAPSOLENOID.

(o) SOLENOID_ZCOLUMN, SOLENOID_RCOLUMN, SOLENOID_BZCOLUMN, SOLENOID_BRCOLUMN, SOLENOID_FACTOR These all correspond to the similarly-named parameters of MAPSOLENOID.

(p) ACCURACY integration accuracy for adaptive integration

(q) METHOD integration method (runge-kutta, bulirsch-stoer, non-adaptive runge-kutta, modified midpoint). Recommend non-adaptive runge-kutta.

(r) FIDUCIAL Specifies how to determine the fidicual particle. Syntax is `<coord>,<method>`, where coord is either 't' or 'p' and method is one of median, min, max, ave, first, or light. E.g., `t,median` means that the fiducial particle has the same arrival time as the median arrival time of the bunch.

The RFTMEZ0 and MAPSOLENOID elements were requested by Eliane Lessner, who helped in the testing.

8. Added the BEAMCENTERED parameter (flag) to the PFILTER element. If set to a nonzero value, then momentum filtering is done relative to the average beam momentum, rather than relative to the nominal central momentum of the beamline.

9. Added the CSR parameter (flag) to CSRCSBEND. If set to 0, then CSR is turned off in that element.

10. Fixed a bug in computation of the Twiss phase advance for elements that have negative drift length. This bug was discovered by Paul Emma.

11. The RFCA and RFCW elements now support rf focusing in the body of the element (i.e., not end focusing) using a simplification of Rosenzweig and Serafini's results (Phys. Rev. E 49 (2), 1599). The simplification is to assume that the field is due to a "pure pi-mode standing-wave". This was suggested by N. Towne (NSLS), and I used his results in checking the new feature. To activate this feature, set the BODY_FOCUS_MODEL parameter to "SRS" (Simplfied Rosenzweig Serafini). The default value of this parameter is "none." I'm willing to add other models using the Rosenzweig/Serafini formulation if people supply me with the parameters. Please note that this feature is independent of the existing end-focusing control through the END1_FOCUS and END2_FOCUS flags. This addition was suggested by Nathan Towne, and I used his note on the subject to help debug the new code.

12. A bug was fixed in the link_elements feature that resulted in double-counting of incremental changes implemented via links. As the manual states, the prior value of any linked-to quantity is on the stack when the equation is executed. Unforunately, links must be executed twice, so if you used that prior value you'd end up with unexpected results. This bug was discovered by Mark Woodley.

13. A bug was fixed in the load_parameters feature that resulted in failure to load the parameter values on the last step of a run with errors.

14. `elegant` now supports macro substitution in the input file. 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 rootname (in run_setup) so that you can get a new set of output files (assuming you use the suggested "%s" field in all the output file names).

15. A number of changes were made to the program sddsmatchtwiss: The -xPlane and -yPlane options now permit changing the emittances of the beam. I also added the -zPlane option, which permits matching the parameters of the longitudinal phase space. The new -oneTransform option allows computing the transform only for the first page, and using it for all pages; this is useful if you want to simulate correction of the beam properties with jitter about the correction.