Taylor Approximation – a solution for round-off problems when modeling LINAC accelerating cavity

Serozhah Milashuk[a,*]

[a]*Cornell University, Laboratory for Elementary-Particle Physics, Ithaca, NY 14853*

* Corresponding author. Tel.: +1 574 2688153

E-mail address: sm757@cornell.edu

Address: 220 Hollister Hall, Cornell University, Ithaca, NY 14853

Abstract: LINAC accelerating cavities (lcavities) are heavily used in accelerators around the world. Bmad is a subroutine library for relativistic charged-particle simulations in accelerators and storage rings. It is currently used to model the proposed international linear collider. Bmad as well as some other particle simulation programs simulate a reference particle in addition to the actual particle. Reference particle assumes 0 errors while the actual particle takes into account in situ errors. Problem has been recognized and encountered where real computational software were unable to handle an exact formula for finding the displacement difference between the real and the reference particles upon their simulated exit from the lcavity. This paper introduces an approximation to the exact formula that allows real computer systems to handle this simulation case. The approximation is based on Taylor expansions of the exact formula. The results of the approximation are presented against results of the exact formula as solved in Fortran (bmad) and in Mathematica. A plot is presented showing the behavior of the typical exact equation in the region of its failure. The approximation is shown to be valid.

1. Introduciton

LINAC accelerating cavities (lcavity) are heavily used in synchrotrons around the world. At Cornell University lcavities are routinely utilized to accelerate electrons and protons. Bmad is a subroutine library for relativistic charged-particle simulations in accelerators and storage rings [1]. It is currently heavily used to model the proposed international linear collider. Bmad as well as some other particle simulation programs use a reference particle (with zero random and systematic errors) and actual particle (in situ errors taken into account). Problem has been recognized where real computational software were unable to handle an exact formula for finding the displacement difference (z) between the real and the reference particles in certain cases. The round-off error is assumed to cause the problem and has been recognized to be problematic in other programming projects [2]. Following is the exact equation for finding z:

$$z_2 = \frac{beta_2}{beta_1} * z_1 - beta_2 * \left(\frac{c * P_2^r - c * P_1^r}{G^r} - \frac{c * P_2^o - c * P_1^o}{G^o} \right) - \ Eq. 1$$

where *beta* is the standard relativistic factor, *c* is the speed of light, *P* and *G* are momentum and gradient, respectively. Superscript ^{r} stands for reference particle and superscript ^{o} stands for observed particle. Number 1 stands for lcavity entrance and 2 stands for lcavity exit. Observed gradient equals to reference gradient plus observed gradient error. Observed momentum can be calculated from observed gradient. Eq .1 contains three subtraction terms. At particularly high energies or low gradients, real software begin to lack the number of kept significant figures. The error during the subtraction can be highly magnified during the two consecutive division operations (Eq. 1). This results in problematic round-off errors [4] or, passed some point, inevitably results in program crashes. In this particular study it was discovered that equation 1 when solved with finite round-off errors in bmad and Mathematica failed to give reasonable results in two cases. As either one or both gradients G^{o} and G^{r} approached 0, one or both terms within the parenthesis of Eq. 1 approached 0/0. Round-off error in bmad and Mathematica increased significantly as G approached values in the 100eV range.

Furthermore, at high energies P_1 and P_2 become very close to each other and the round-off error in bmad and Mathematica also become very large. The objective of this study was to find a mathematically sound subroutine that would allow analytical solutions of Eq. 1 in programs with finite round-off errors and in previously described two cases. This would expand the range of applicability of Eq. 1 to high energy and low gradient simulations.

This paper introduces a program routine that takes advantage of Taylor expansions. Taylor approximation has been widely used in every scientific field (e.g. [5,6]) including solving programming shortcomings [7]. While the Eq. 1 cannot be directly incorporated into software, the introduced routine can be used in any one major programming languages (e.g. C++ and Fortran). The approximation is based on various Taylor expansions. The results of this routine are presented against results of the exact formula as solved in Fortran (bmad) and in Mathematica. The introduced error of the Taylor approximation can be easily estimated (e.g. [8]) if desired. While various improvements have been made in solving round-off related problems in different scientific fields, (e.g. [9]), Taylor expansion still maintains advantages in being "universal" and requiring the least CPU time [7].

2. Four case based Taylor approximation

Following is a subroutine written in Fortran 90 and implemented in Cornell's bmad software:

```
dE_true = gradient_true * length
 if (abs(dE/E_start) < 1e-4) then
  if (E_start > 100 * mc2) then
   f = (mc2 / pc_start)**2
   dp_dg = f/2 - f**2/8 + f**3/16
  else
   dp_dg = (E_start - pc_start) / pc_start
  endif
  f = (dE_true  / E_start)
  g = E_start / pc_start
  dp_dg = dp_dg + (mc2 / pc_start)**2 * (-f/2 + f**2 * g / 2 - f**3 * g**2 / 8)
 else
  dp_dg = (pc_end - pc_start) / dE_true - 1
 endif

 dE_ref = gradient_ref * length
```

```
if (abs(dE_ref/E_start_ref) < 1e-4) then
  if (E_start_ref > 100 * mc2) then
    f = (mc2 / pc_start_ref)**2
    dp_dg_ref = f/2 - f**2/8 + f**3/16
  else
    dp_dg_ref = (E_start_ref - pc_start_ref) / pc_start_ref
  endif
  f = (dE_ref  / E_start_ref)
  g = E_start_ref / pc_start_ref
  dp_dg_ref = dp_dg_ref + (mc2 / pc_start_ref)**2 * (-f/2 + f**2 * g / 2 - f**3 * g**2 / 8)
else
  dp_dg_ref = (pc_end_ref - pc_start_ref) / dE_ref - 1
endif

z2 = z1 * (beta_end / beta_start) - beta_end * length * (dp_dg - dp_dg_ref)
```

Where de_true is deo , de_ref is der , E_start_ref is e_1^r, E_start is e_1^o. The explanation of the subroutine is as follows:

First, calculate *de_true* where *de_true* =G^o * *length*, where *length* is the length of the lcavity. If de/e_1^o $<10^{-3}$ go to sequence 1. If $de/e_1^o>10^{-3}$ then solve $dpdg^o$ using the exact equation:

$$dpdg^o = \frac{P_2^o * c - P_1^o * c}{de} - 1$$

where –1 is added in the attempt to reduce the round-off error. Sequence 1 calculates $dpdg^o$ using a Taylor expansion to the first 6 terms of the exact equation:

$$dpdg^o = dpdg_{ft} + \left(\frac{m * c^2}{P_1^o * c}\right)^2 * \left(-\frac{f}{2} + f^2 * \frac{g}{2} - f^3 * \frac{g^2}{8}\right)$$

, where *f* is de/e_1^o, *g* is $e_1^o/(P_1^o * c)$ and $dpdg_{ft}$ is the first term of the Taylor expansion. Before this expansion is solved, another **if** statement is evaluated. If $e_1^o>100mc^2$ inequality is True, $dpdg_{ft}$ is solved for in exact form, i.e. $dpdg_{ft}=(e_1^o - (P_1^o * c))/ (P_1^o * c))$. If it is False, $dpdg_{ft}$ is solved for using another Taylor approximation: *f*/2-*f*^2/8+*f*^3/16, where *f* is $(mc^2/(P_1^o * c))^2$. This same basic sequence calculates both $((P_2^o * c) - (P_1^o * c))/\text{de}$ and $((P_2^r * c) - (P_1^r * c))/\text{de}$. These terms are then used to calculate *z2*.

4 Testing in Bmad and Mathematica

This approach for modeling the lcavity was then tested in many typical cases. These test cases included various energy gradients and particle energies ranging from 10^6 to 10^{110} eV (Appendix 1). The **If** statement program routines switch between equations based on dE_*/E_start_*. Figure 1 shows the dE/E_start values tested.

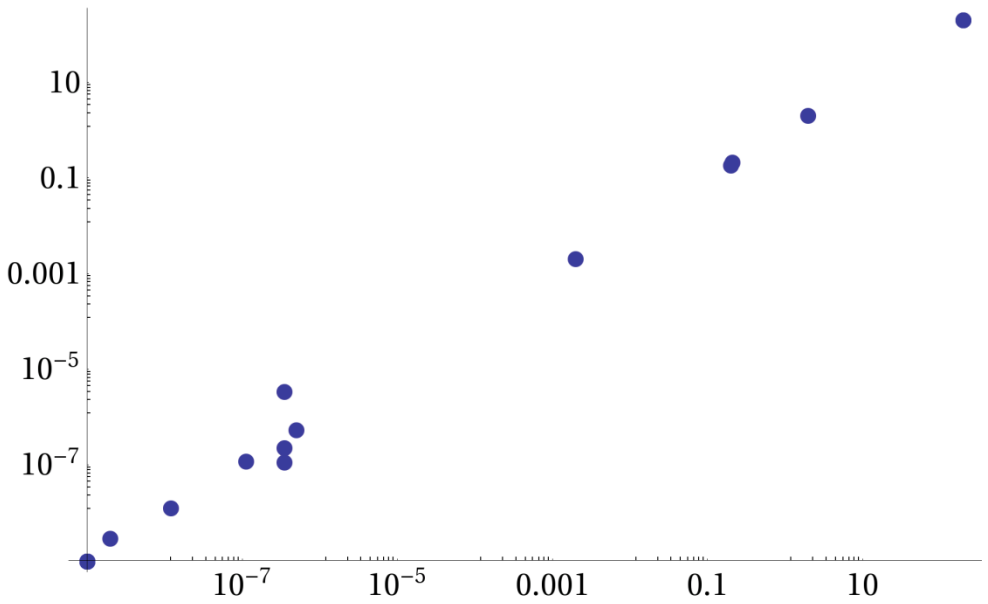

Fig. 1: tested values; $de/e_1{}^o$ vs. $de/e_1{}^r$; many data points are smaller than 10^{-10} (e.g. when energy is in the 10^{11} - 10^{130} eV range).

The usefulness of the "if (E_start …> 100 * mc2) then" statement was also tested. Figure 2 shows the resulting errors of 3 programs: exact calculation, section 2 routine and section 2 routine lacking the "E_start_... > 100 * mc2" **if** statements. Furthermore, both mathematica and Fortran codes were used for the implementation, each with its own internal precision settings. To show all results, in cases where error was zero or less than 10^-9, it was set equal to 10^-9.

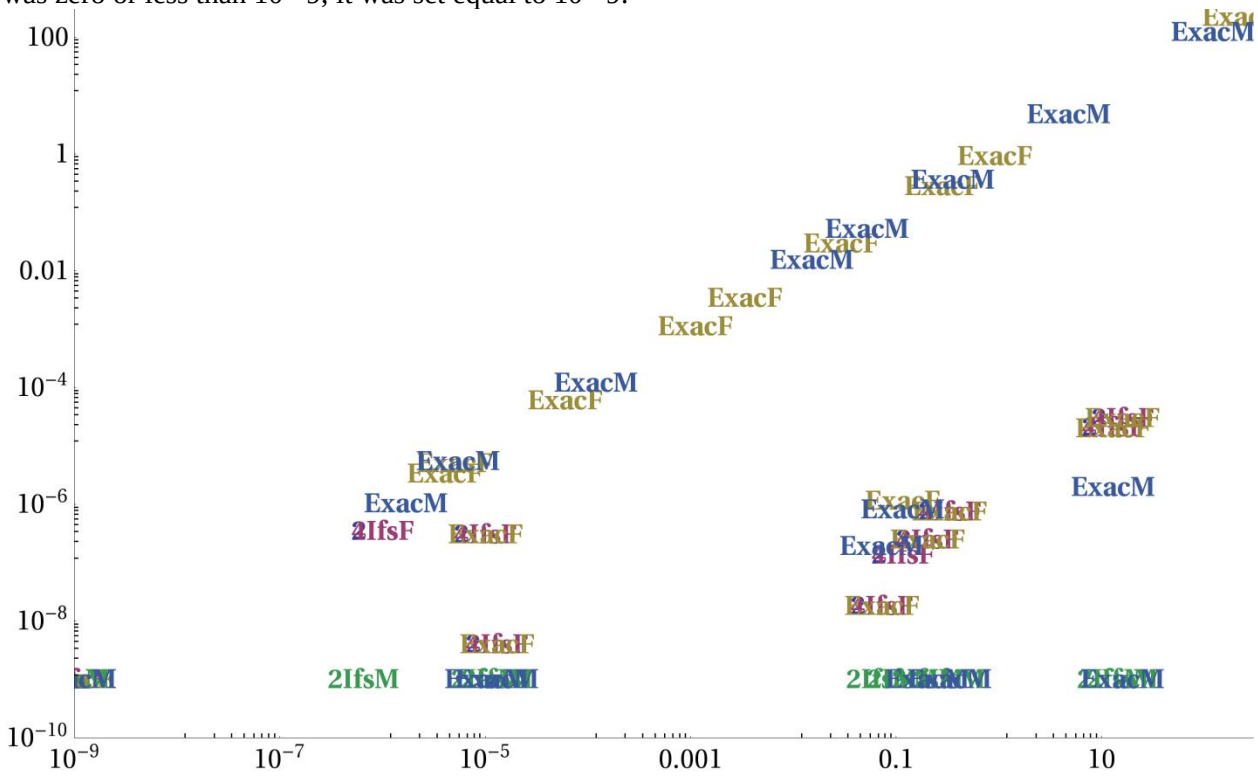

Fig. 2: *z2* resulting error (meters) assuming that 2 case **if** statement in Mathematica gives the correct answer (one without "E_start_... > 100 * mc2"); ***z2*** *vs.* **abs((*correct_z2* - *calculated_z2*))**; in the point markers: M stands for Mathematica, F stands for Fortran, 2 stands for a program without the "E_start_... > 100 * mc2" test, 4 stands for codes with all 4 **If** statements, Exac stands for programs that used exact Eq. 1; in cases where error was less than 10^-9, it was set to equal 10^-9.

While Fig. 2 showed semi-random isolated cases, Fig. 3 and Fig. 4 show continuous distributions in the high and low gradient regions for all Mathematica programs used for Figure 2 compilation.

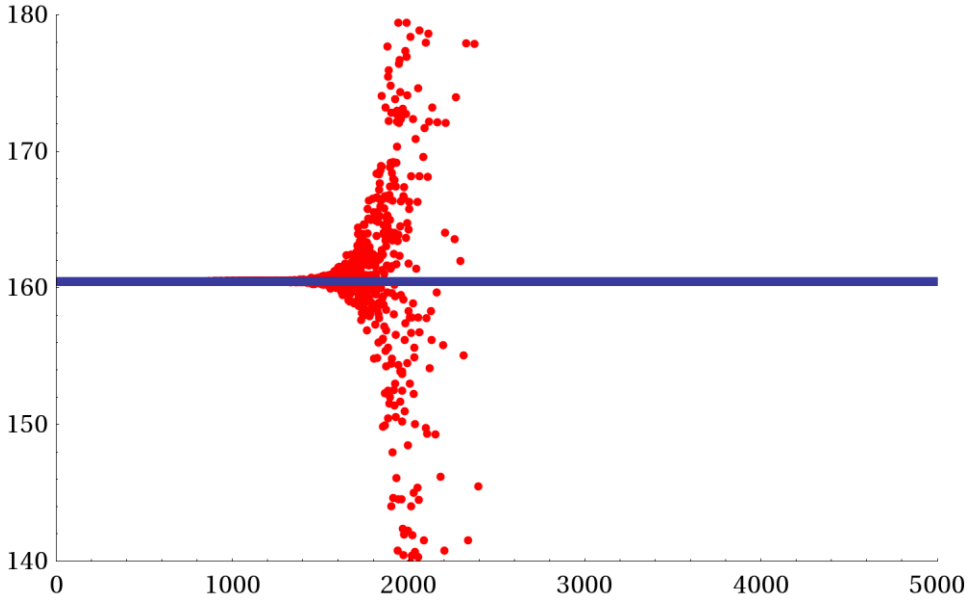

Fig. 3: *z2* (meters) in the low gradient region of input data; x axis follows the equation $gradient_{ref} = 1/e^{x/100}$, y axis is *z2*; both 2 and 4 **If** statement cases overlap into the thick blue horizontal line. The program that used raw eq. 1 for calculations (red) can be distinguished by its logarithmic growth deviation from the horizontal line.

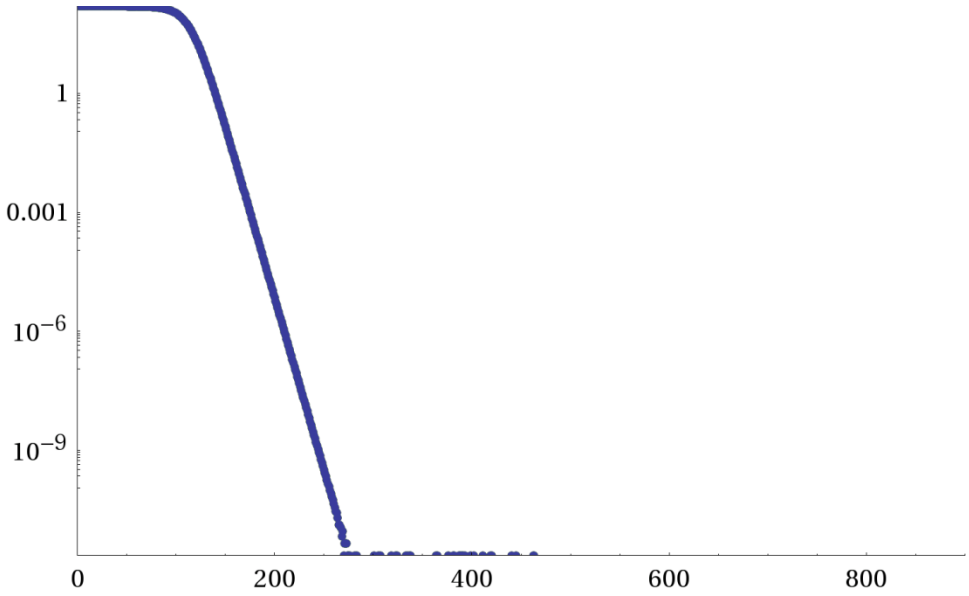

Fig. 4: *z2* (meters) in the high gradient region of input data; x axis follows the equation $G^r = e^{x/10}$, y axis is *z2* (meters); both 2 and 4 **If** statement cases overlap with exact equation case into the thick blue line.

The values used when generating data points for Fig. 3 and 4 are shown in Table 1, below.

Table 1: sample input values for z2 calculations

Lcavity length	G^r	gradient_error	(pc_strue-pc_sref)/pc_sref	e_1^r	particle
10 meters	varying	10^4 eV	0	10^6	electron

Discussion and Conclusion

As shown in the Figure 2, **if** statement subroutines give nearly equivalent results to the exact formulas in the region prior to failure of the exact equation (see overlapping points along the *y*=10^-9 line). This serves to show that in the de/e_1 range of about 10^-6 to about 1000 the **if** statement subroutines are valid. Starting at about 10^-6 and down, both Fortran exact formula and Mathematica exact formula "blew up." As de/e_1 decreased, *z2* remained approximately constant (due to dominant de/e_2 term). For the exact equation case, beyond some critical de/e_1 value, errors in *z2* began to increase with positive second order derivative (Fig. 2 and 3). The critical de/e_1 value seemed to not depend on the round-off error i.e., the number of significant figures the program used. Notice that the logarithmic growth of the error in Fig. 2 is equivalent for both Fortran and Mathematica codes. A sweep was made to show explicitly the critical de/e_1 for a sample case scenario and results were shown in Fig. 3. Fig. 3 shows explicitly the success of

Taylor approximations in expanding the range of computational viability of Equation 1 implementation. Fig 2-4 show that both two and four **If** statement cases programs performed equally well. In Fig. 1, z2 difference between all **If** based programs never exceeded 10 micrometers. However, this shows that 4 **If** statements case program does not need two of the **If** statements. In other words, 2 **If** statements case program should be utilized instead of the 4 **If** program, as it results in savings of CPU power requirements and code length as well as code lucidity improvements.

5. Conclusion

A single equation that reveals displacement error of observed particle passing through a linear accelerating cavity (lcavity) cannot be used in software directly. This equation has to be adapted for use in such simulation code packages as Cornell University's bmad library. One of the major difficulties that were encountered in practice was the finite round-off error of modern computers. The result was inaccurate calculations in regions of high traveling particle momentum and low simulated lcavity gradients. In fact, below certain lcavity energy gradient and above a certain particle momentum programs bombed failing to give any results. This paper presented a solution that allowed the integration of equation 1 into computational software. Furthermore, it has been shown that the presented computer calculation sequence provides reasonable results. A plot was presented of computer behavior when treating exact Equation 1 in the region of computational failure (Fig. 2 and 3). The calculation uncertainty follows an exponential function as the equation approaches low de/e values (Fig. 2 and 3). From the presented Figure 3, the envelope of the uncertainty growth apparently is symmetrical and follows an exponential function. It was also shown that the four **If** statements based program can be reduced to 2 **If** statements program with nearly equivalent results. Since this calculation routine is used as an integrated components to larger programs, the CPU savings from this reduction in various cases can be significant.

References:

[1] Sagan, D., 2010. Bmad manual. Available at: http://www.lepp.cornell.edu/~dcs/bmad/

[2] Fousse, L.; Hanrot, G.; Lefèvre, V.; Pélissier, P.; Zimmermann, P., 2007. MPFR: A multiple-precision binary floating-point library with correct rounding. ACM Transactions on Mathematical Software (TOMS)

[3] Kai, L.; Massoli, P., 1994. Scattering of electromagnetic-plane waves by radially inhomogeneous spheres: a finely stratified sphere model. Appl.Opt. 33, 3, 501-511, OSA

[4] Cucker, F.; Smale, S., 1998. Complexity estimates depending on condition and round-off error. Algorithms—ESA'98, 1-1, Springer

[5] Van Gunsteren, W. F.; Berendsen, H. J. C., 1977. Algorithms for macromolecular dynamics and constraint dynamics. Mol.Phys., 34, 5, 1311-1327, Taylor & Francis

[6] Brookner, E.,1998. Tracking and Kalman filtering made easy. 1998, John Wiley & Sons

[7] Agarwal, R.C.; Gustavson, F.G.; Schmookler, M.S., 1999. Series approximation methods for divide and square-root in the power3 processor

[8] Christianson, B., 1992. Reverse accumulation and accurate rounding error estimates for taylor series coefficient. Optim.Methods Software, 1992, 1, 1, 81-94, Taylor & Francis

[9] Wu, Z.S.; Guo, L.X.; Ren, K.F.; Gouesbet, G.; Gréhan, G., 1997. Improved algorithm for electromagnetic scattering of plane waves and shaped beams by multilayered spheres. Appl.Opt., 36, 21, 5188-5198, OSA

Appendix 1:

Figure A1: z2 values used for calculation of Figure 2 points (meters)

Row	Fortran, 2 if statements	Bmad, 4 if statements	Fortran, exact	Math old to_new	math new to_new	Math exact
1	15.94441	15.94441	15.94441	15.94438	15.94438	15.94438
2	0.000013	0.000013	0.000013	1.3E-05	1.3E-05	1.3E-05
3	0.00001	0.00001	0.00001	1.03E-05	1.03E-05	1.03E-05
4	0.331317	0.331317	0.331317	0.331316	0.331316	0.331316
5	0.331317	0.331317	0.331317	0.331316	0.331316	0.331316
6	0.072534	0.072534	0.072534	0.072534	0.072534	0.072534
7	0.116055	0.116055	0.116054	0.116055	0.116055	0.116054
8	-12.9004	-12.9004	-12.9004	-12.9004	-12.9004	-12.9004
9	0.20215	0.20215	0.20215	0.20215	0.20215	0.20215
10	0.000001	0.000001	0.000004	6.54E-07	6.54E-07	1.69E-06
11	0	0	-5E-06	1.79E-11	1.66E-11	-5.4E-06
12	0	0	-0.00006	1.79E-13	4.66E-14	-0.00012
13	0	0	-0.00338	1.79E-16	-1.3E-13	-0.01517
14	0	0	0.02913	1.79E-19	1.79E-19	0.052019
15	0	0	0.00111	1.79E-21	1.79E-21	0
16	0	0	NaN	0	1.79E-51	0
17	0	0	NaN	0	0	0
18	0	0	0	1.79E-25	1.79E-25	0
19	0	0	0.278109	1.8E-42	1.79E-42	0.357469
20	0	0	NaN	0	1.8E-102	0
21	0	0	NaN	0	1.8E-102	0
22	0	0	-0.91346	2.05E-45	1.79E-45	-4.76625
23	0	0	-220.161	0	1.79E-51	122.0161
24	0	0	-220.161	0	1.79E-51	122.0161

Figure A2: **If** statements values and logical results. Variable names and general notations is based on Section 2 routine (which is also a part of Cornell's bmad code).

Row	e_sture> 100 mc2	e_sture	e_sref> 100 mc2	e_sref	abs(dE/ E_start) < 1e-4	abs(dE/ E_start)	abs(dE_ref/ E_start_ref	abs(dE_ref/ E_start_ref)< 1e-4
1	FALSE	10^6	FALSE	10^6	FALSE	2.1	FALSE	2
2	TRUE	1E+08	TRUE	1E+08	FALSE	0.0021	FALSE	0.002

3	TRUE	9.5E+08	TRUE	9.5E+08	FALSE	0.221053	FALSE	0.210526
4	FALSE	1000000	FALSE	1000000	FALSE	210	FALSE	200
5	FALSE	1000000	FALSE	1000000	FALSE	210	FALSE	200
6	TRUE	1.5E+08	TRUE	1E+08	TRUE	2.33E-07	TRUE	3.5E-07
7	TRUE	3E+08	TRUE	1E+08	TRUE	1.17E-07	TRUE	3.5E-07
8	FALSE	10012917	TRUE	1E+08	TRUE	3.5E-06	TRUE	3.5E-07
9	FALSE	10000000	FALSE	10000000	FALSE	-0.19	FALSE	-0.2
10	FALSE	10000000	FALSE	10000000	TRUE	5.5E-07	TRUE	5E-07
11	TRUE	9E+08	TRUE	9E+08	TRUE	1.22E-07	TRUE	1.11E-07
12	TRUE	9E+08	TRUE	9E+08	TRUE	2.22E-09	TRUE	1.11E-09
13	TRUE	9E+08	TRUE	9E+08	TRUE	1.22E-11	TRUE	1.11E-11
14	TRUE	9E+09	TRUE	9E+09	TRUE	1.22E-12	TRUE	1.11E-12
15	TRUE	9E+11	TRUE	9E+11	TRUE	1.22E-10	TRUE	1.11E-10
16	TRUE	9E+21	TRUE	9E+21	TRUE	1.22E-20	TRUE	1.11E-20
17	TRUE	9E+110	TRUE	9E+110	TRUE	1.2E-109	TRUE	1.1E-109
18	TRUE	9E+14	TRUE	9E+14	TRUE	1.22E-08	TRUE	1.11E-08
19	TRUE	9E+20	TRUE	9E+20	TRUE	1.22E-13	TRUE	1.11E-13
20	TRUE	9E+40	TRUE	9E+40	TRUE	1.22E-33	TRUE	1.11E-33
21	TRUE	9E+40	TRUE	9E+40	TRUE	1.22E-33	TRUE	1.11E-33
22	TRUE	9E+21	TRUE	9E+21	TRUE	1.22E-14	TRUE	1.11E-14
23	TRUE	9E+23	TRUE	9E+23	TRUE	1.22E-16	TRUE	1.11E-16
24	TRUE	9E+23	TRUE	9E+23	TRUE	1.22E-16	TRUE	1.11E-16