Simulation of CESR-c Luminosity from Beam Functions

Abhijit C. Mehta
Trinity College, Duke University, Durham, North Carolina, 27708
(Dated: August 13, 2004)

It is desirable to have the ability to compute quickly and accurately the expected luminosity of a given CESR-c configuration. By replacing a number of numerical integrals with analytic integrals, we improve an algorithm that computes luminosity and attempt to develop a useful Fortran 90 subroutine that can calculate luminosity accurately and efficiently. Although our algorithm currently returns an answer that is incorrect by a modest factor, preliminary results indicate that it is not far from becoming a useful tool.

I. INTRODUCTION

In accelerator physics, luminosity is a measure of the rate of interactions per unit area when two beams collide. It is given by:

$$\mathcal{L} = \frac{N_1 N_2 f_{\mathrm{bb}}}{A} \tag{1}$$

where $\mathcal{L}$ is the luminosity, N_1 and N_2 are the number of particles in each bunch of the positron and electron beam, respectively, f_{bb} is the frequency at which bunches collide in the interaction region, and A is the cross-sectional overlap. It is desirable to configure CESR in a way that gives a high luminosity, since higher luminosities translate into more particle-particle interactions that can be studied.

Luminosity can be computed by geometrically calculating the overlap of two colliding bunches. If the distribution of a bunch centered at s is described by a function $f(x, x', y, y', z, \delta, s)$, (where x, x', y, y' are the horizontal and vertical phase-space coordinates, z and $\delta(= \Delta E/E)$ are the longitudinal phase-space coordinates, and $s(= ct)$ is the independent time-like variable), then, using Eqn. 1, one can write the luminosity as: [1]

$$\begin{aligned}\mathcal{L} \;=\; & N_1 N_2 f_{\mathrm{bb}} \int d^6V_1 \int ds_1 \int d^6V_2 \int ds_2 f_1(\mathbf{p_1}) f_2(\mathbf{p_2}) \cdot \\ & \delta(s_1 + z_1 - s_2 + z_2)\delta(s_1 + s_2)\delta(x_1 - x_2)\delta(y_1 - y_2)\end{aligned} \tag{2}$$

where the δ-functions represent the constraint that particles in the two beams be in the same physical location in order to interact.

Using Eqn. 2, it should be straightforward to create a computer program that calculates the luminosity expected from a particular CESR-c configuration. We assume that the bunch distribution functions are Gaussian, and perform the integration. Though this task seems straightforward, it is somewhat laborious; we have an integral over fourteen variables, and even though the presence of the four delta functions in Eqn. 2 means that we only effectively have to integrate over ten variables, it is still a CPU-intensive task if all of these integrals are computed numerically. Last year, an algorithm was developed to perform this integration numerically; the runtime, however, was too long to be practical. [2]. By performing some of the integration analytically, we should be able to drastically reduce runtime.

II. INTEGRATION

We can separate out the transverse integrals from Eqn. 2 and evaluate them analytically. First, we need to normalize the distribution functions to have unit values when integrated over all phase space. Following the steps taken in Appendix 2 of "Geometrical Calculation of Luminosity", we get, for each beam i: [1]

$$K_i = \frac{1}{(2\pi)^{5/2}\sigma_{z,i}\int_{-\infty}^{\infty} d\delta \epsilon_{u,i}\epsilon_{v,i}\exp(-\frac{\delta^2}{2\sigma_\delta^2})} \tag{3}$$

Taking advantage of the $\delta(x_1 - x_2)\delta(y_1 - y_2)$ term in Eqn. 2, we can write the transverse integrals in the form:

$$g(z_1, \delta_1, s_1 \ ; \ z_2, \delta_2, s_2) = K_1 K_2 \int_{-\infty}^{\infty} dx_1' dy_1' \int_{-\infty}^{\infty} dx_2 dx_2' dy_2 dy_2' \exp(E) \tag{4}$$

The argument, E, of the exponential in Eqn. 4 can be written in the form:

$$-2E = \mathbf{q^T Q q} + \mathbf{\Delta q^T \Omega q} + \mathbf{\Delta q^T} \omega \mathbf{\Delta q} \tag{5}$$

where $\mathbf{q}^T = (x_1', y_1', x_2, x_2', y_2, y_2')$, $\mathbf{\Delta q}$ is the difference between the closed orbit trajectories of the two beams, and $\mathbf{Q}$, $\mathbf{\Omega}$, and ω are 6x6 coefficient matrices. [1] Using a singular value decomposition, we diagonalize $\mathbf{Q}$ with a transformation $\mathbf{R}$ into new coordinates $\mathbf{q_r} = (r_1, r_2, r_3, r_4, r_5, r_6)^T$ such that $\mathbf{q} = \mathbf{R q_r}$. Writing Eqn. 5 in terms of these new coordinates, we have:

$$-2E = \mathbf{q_r^T R^T Q R q_r} + \mathbf{\Delta q^T \Omega R q} + \mathbf{\Delta q^T} \omega \mathbf{\Delta q} \tag{6}$$

Since $\mathbf{Q}$ is diagonalized, the eigenvalues λ_i of $\mathbf{Q}$ are given by $\lambda_i = [\mathbf{R^T Q R}]_{ii}$. It is now straightforward to integrate the right hand side of Eqn. 4:

$$\begin{aligned} g(z_1, \delta_1, s_1; z_2, \delta_2, s_2) &= K_1 K_2 \int_{-\infty}^{\infty} dr_1 ... dr_6 \exp(E) \\ &= \frac{\pi^3 K_1 K_2}{\sqrt{\lambda_1\lambda_2\lambda_3\lambda_4\lambda_5\lambda_6}} \exp(-q_0 + \sum_{j=1}^{6} \frac{q_j^2}{4\lambda_j}) \end{aligned} \tag{7}$$

Due to the $\delta(s_1 + z_1 - s_2 + z_2)\delta(s_1 + s_2)$ term in Eqn. 2, s_1, s_2, z_1, and z_2 are not all independent; we can set $s_2 = -s_1$ and $z_2 = -2s_1 - z_1$, and write g as a function g^* in terms of only 4 variables. Furthermore, since g only depends on the longitudinal position $(s_1 + z_1)$, and not on s_1 and z_1 individually, we can write g as a function G with only 3 arguments:

$$g(z_1, \delta_1, s_1; z_2, \delta_2, s_2) = g^*(z_1, \delta_1, s_1, \delta_2) = G(s_1 + z_1, \delta_1, \delta_2) \tag{8}$$

Now that we have evaluated the transverse integrals analytically, we can simplify Eqn. 2 to write an expression for luminosity that only requires us to integrate over four longitudinal variables. In the following expression for luminosity, z_{orbit} refers to the longitudinal spatial

[1] Appendix 2 of "Geometrical Calculation of Luminosity" by M. Billing describes in detail how to explicitly calculate the values of these matrices. [1]

displacement of the beam from the interaction point at the nominal collision time and δ_{orbit} refers to the fractional energy deviation of the beam with respect to the nominal energy.

$$\mathcal{L} = N_1 N_2 f_{\rm bb} \int_{-\infty}^{\infty} ds_1 dz_1 d\delta_1 d\delta_2 G(s_1 + z_1, \delta_1, \delta_2) \cdot \exp(-\frac{(z_1 - z_{orbit,1})^2}{2\sigma_z^2} - \frac{(\delta_1 - \delta_{orbit,1})^2}{2\sigma_\delta^2} - \frac{(-2s_1 - z_1 - z_{orbit,2})^2}{2\sigma_z^2} - \frac{(\delta_2 - \delta_{orbit,2})^2}{2\sigma_\delta^2}) \quad (9)$$

III. PROGRAM DESIGN

Last year, a Fortran 90 program called `test_lum_calc` was written to calculate the luminosity of a given storage ring configuration using Eqn. 2. [2] This summer, I modified the program so that it uses Eqn. 9 to calculate luminosity. Last year's version of the program needed to numerically integrate over 10 variables; now we only integrate numerically over 4 variables, though it is somewhat more complicated to construct the function G which we need to integrate. Below is an outline of our algorithm:

```
program test_lum_calc:

read lattice info
generate mesh
generate matrices from Eqn. 6
iterate over s_1 + z_1, δ_1, δ_2:
↪ calculate G(s_1 + z_1, δ_1, δ_2), store values
iterate over s_1, z_1, δ_1, δ_2:
↪ sum values to get the integral in Eqn. 9
normalize and use Eqn. 9 to output Luminosity.
```

The first part of our program reads in information about the bunch length, beam offsets, and CESR lattice that will be used for our luminosity calculation. In the next two steps, we construct a "mesh" in phase-space over which we shall integrate. The size of this mesh scales to the standard deviation of the bunch distribution function. First, we calculate and store Twiss parameters and other useful basic data at each mesh point. Then, we calculate and store the values of the matrices from Eqn. 6 – $\mathbf{Q}$, $\mathbf{\Omega}$, and ω – at each mesh point, and we diagonalize $\mathbf{Q}$ to get $\mathbf{R}$ and the λ's.

Once we have stored the necessary data at each mesh point, we begin to integrate. First, we calculate and store the values of $G(s_1 + z_1, \delta_1, \delta_2)$ at each $(s_1 + z_1,\ \delta_1,\ \delta_2)$ coordinate. When we use a fine mesh (i.e., a large number of mesh points per standard deviation (σ) of the bunch distribution), this is the most time-consuming step. [2] The next step iterates over s_1, z_1, δ_1 , and δ_2 to calculate the integral in Eqn. 9, using the stored values of G. Finally, we use Eqn. 9 to output the correctly scaled value of the chosen lattice's luminosity. Our program is approximately 1400 lines of Fortran 90 code.

[2] We do the calculation of G separately from the integration for two reasons. First, calculating G at each separate value of s_1 and z_1 would result in unnecessarily duplicated effort since G is a function of $s_1 + z_1$, not s_1 and z_1 individually. In addition, using three nested `do` loops to store values in an array without summing those values is a process that can be easily optimized for multi-CPU systems.

IV. TESTING

We tested our program on the DEC Alpha "CESR2F" using a bunch length of 10 mm and the "12WIG_CL_20040315_V1S" lattice. Table I contains some basic data which we obtained by running our program with different mesh sizes. The mesh size scales with σ, and we expect to get more accurate results from larger mesh sizes. Table II contains data that we collected using last summer's version of test_lum_calc, which performed all of the integrations numerically.

First, looking at Table I, we notice that there is very little variation in the luminosity values that we get when we run our program with different mesh sizes. After numerically integrating a simple one-dimensional Gaussian distribution, I found that this small amount of variation in luminosity is consistent with what one would expect to see when integrating a Gaussian distribution. Furthermore, these results are much more stable than the results we get from running last summer's program (Table II), so our results look promising.

Comparing run times, our new program is much faster than last summer's. The times for integration seem to increase like n^3 or n^4, as we would expect since we integrate over four variables. Unlike last year's program, our program is fast enough to calculate luminosity in a reasonable amount of time with a mesh size that will yield an accurate result.

TABLE I: Run times and variation in luminosity for different mesh sizes.
Note that mesh size scales with σ.

Number of mesh points per σ	Luminosity ($cm^{-2}s^{-1}$)	Mesh Generation Time (s)	Integration Time (s)	Total Run Time (s)
1	3.280×10^{29}	41	2	43
2	3.272×10^{29}	75	16	91
3	3.263×10^{29}	120	38	158
4	3.285×10^{29}	165	72	237
5	3.269×10^{29}	198	174	372
6	3.266×10^{29}	245	300	545
7	3.254×10^{29}	339	529	868

TABLE II: Run times and variation in luminosity for last year's version of test_lum_calc

Number of mesh points per σ	Luminosity ($cm^{-2}s^{-1}$)	Mesh Generation Time (s)	Integration Time (s)	Total Run Time (s)
1	2.67×10^{30}	33	231	264
2	1.04×10^{31}	76	22332	22408

To check for the possibility that the beams are missing each other in our simulation (thus giving us lower than expected luminosity values), we performed an offset scan in x, y, and z in which we adjusted the offset of the beams in each of the three spatial dimensions (one at a time) and measured the effect of the offset on calculated luminosity. Figure 1 contains the results of our scan. We expect that the curves for x and y will be Gaussian because of the following argument: Consider the case of the x offset. If we have an offset of Δx, then

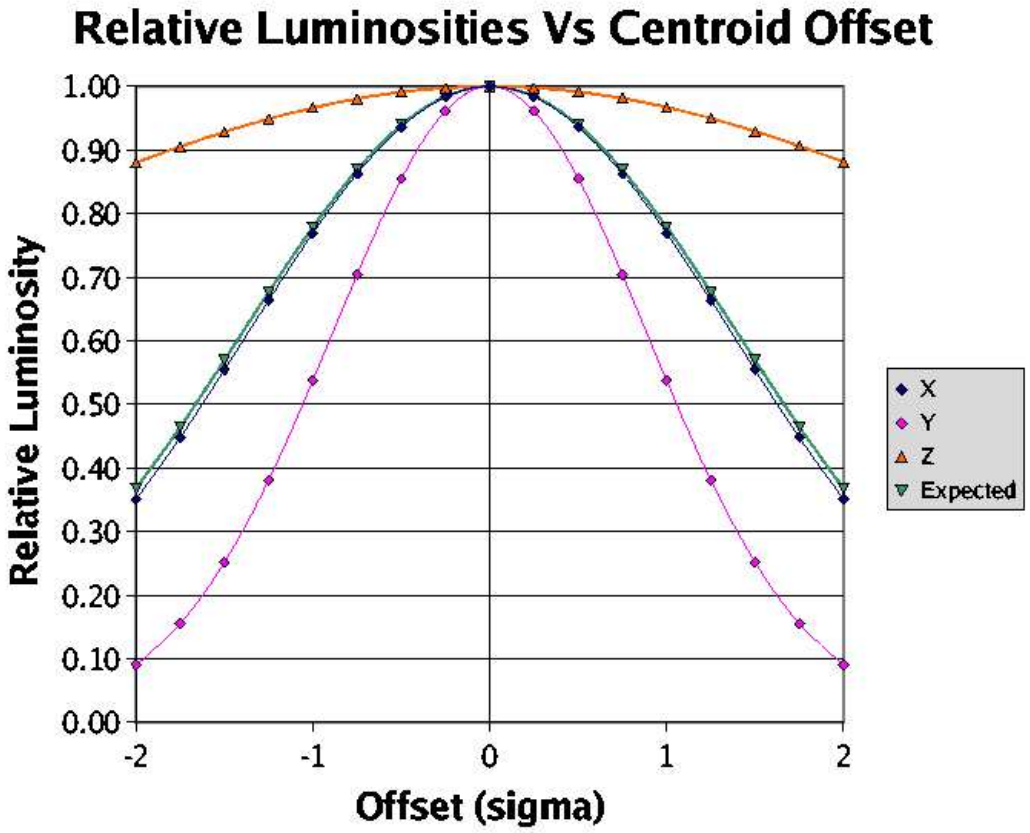

FIG. 1: Plot of relative luminosity ($\mathcal{L}/\mathcal{L}|_{offset=0}$) vs centroid displacement in σ's for x, y, and z, along with expected curve for x and y.

we are integrating an expression that looks like:

$$\int_{-\infty}^{\infty} \exp(-\frac{x^2}{2\sigma^2}) \exp(-\frac{(x-\Delta x)^2}{2\sigma^2}) \tag{10}$$

Simplifying the argument of the exponential in Eqn. 10, we have:

$$\int_{-\infty}^{\infty} \exp(-\frac{[2(x-\frac{\Delta x}{2})^2 + \Delta x^2 - \frac{\Delta x^2}{2}]}{2\sigma^2}) \tag{11}$$

Now, we can factor out an $\exp(-\frac{\Delta x^2}{4\sigma^2})$ from Eqn. 11 and integrate the rest of the expression to get a constant; thus, it is clear that as a function of offset, we expect the relative luminosity curve to have a Gaussian shape, specifically, $\exp(-\frac{\delta x^2}{4\sigma^2})$. The same argument holds for y. Our actual curve for x is very close to our expected curve; the variation from the expected curve in y is probably due to differences in the beam sizes and small crossing angles that our program takes into account.

We expect the longitudinal curve to be different, however. Near the interaction point, $\beta(s) = \beta^* + \frac{s^2}{\beta^*}$, (where $\sqrt{\beta}$ describes the envelope function of the beam around the ring, and β^* denotes the value of β at the interaction point). Luminosity is inversely proportional to $\sqrt{\beta}$ near the interaction point. Using $\sigma_z = 10mm$ and $\beta^* = 14mm$, we expect that at

an offset of $2\sigma_z$, the luminosity should be approximately 0.82 of its value at zero offset. Our actual result is close to this expected value. The correspondence between our predicted results and actual data for the offset scan leads us to believe that our algorithm is working properly.

Despite the promising nature of our results, they do not match up with the values of luminosity we would predict to get. We can estimate luminosity by using Eqn. 1 and substituting in values that are specific to the lattice we used. Doing this gives us a value of $5.7 \times 10^{30}\ cm^{-2}s^{-1}$, which is more than an order of magnitude greater than the result we get in Table I. The code is quite complex, and so it is quite possible that there is a factor missing from the computation. Some error testing leads us to believe that the problem may be related to the singular value decomposition which we perform.

V. CONCLUSIONS

Our results indicate that our algorithm, though still not correct, has great potential to become a useful tool for calculating luminosity. Compared to last summer's version of the code, our run time is quite short, which was the primary motivation for doing this project. Additionally, we demonstrated that the technique of collecting terms into a large exponential and manipulating the argument of that exponential with matrix operations is a good way to attack certain types of integration problems.

The order of magnitude difference between our result and the expected result still needs to be dealt with, however. The small amount of variation that we observe when we change the mesh size, combined with the results of our offset scan, suggests that there is not a fundamental error in our calculation. This leads us to believe that our method is basically correct, but that we are off by some nontrivial factor at some point in the calculation. Hopefully, this problem will be solved soon and our program will become a useful tool for CESR lattice design and testing.

VI. ACKNOWLEDGMENTS

I would like to thank Dr. Michael Billing of the Laboratory for Elementary-Particle Physics at Cornell University for being so generous with his time and for all of his patience in guiding me through this project, and Prof. Rich Galik for organizing the Cornell LEPP REU program. I learned a great deal this summer, and I had a lot of fun participating in this REU program.

This work was supported by the National Science Foundation REU grant PHY-0243687 and research co-operative agreement PHY-9809799.

---

[1] Billing, M. "Geometrical Calculation of Luminosity," 10 January, 2002.

[2] Marang, G. "Luminosity Calculation from Known Beam Functions." 16 August, 2003.