Step-by-step guide to using `volterra_mat` and some notes on the code

Cheng-Ying Tsai[1]

School of Electrical and Electronic Engineering, Huazhong University of Science and Technology

July 9, 2020

To quantify microbunching instability in a single-pass transport or multi-pass recirculation systems, we have developed a code, named `volterra_mat`, to estimate microbunching amplification gain factor along an arbitrary linear beamline. The formulation behind this code is based on a linearized Vlasov equation [1, 2]. Solving by the method of characteristics and together with linear single-particle equations of motion, the governing equations of the phase space density and energy modulations can be simply expressed as a linear integral equations of Volterra type. This code solves the semi-analytical integral equations, with extension of previous work [1, 2] to include both horizontal and vertical bends, as well as more relevant impedance models. The input information is split into two parts: one from `elegant` [3] (the initial beam and lattice parameters) and the other provided by the users (numerical parameters, and choice of collective effects). During the development of this code, more relevant beam physics effects have been added and the use of it is improved by graphical user interface (GUI). It can serve as an efficient tool for microbunching analysis, or, later, be integrated into an optimization tool for beamline design when microbunching instability is an issue of interest. In this note, we introduce the physics models and numerical algorithms behind the code, and then step-by-step illustrate the usage of this code. Finally, we summarize possible improvements in future versions and some notes on this code.

Contents

[1]Please send any questions or comments about the code to jcytsai@hust.edu.cn

1 Very short history of this code

The numerical code `volterra_mat` was mainly developed along with my Ph.D. thesis [4], when I was a graduate student at Virginia Tech/Jefferson Lab, during 2013 and 2017.

During v1, many efforts were spent in benchmarking this code with the analytical results (for the three-dipole model chicane [2]) and with particle tracking simulations `elegant` for several lattice designs (most of them provided by David Douglas of JLab).

The early versions of this code were not intended to provide other people's use, so there was no graphical user interface (GUI) until the birth of v2.2. At that time, this code was mainly used by myself, Chris Tennant (JLab), and Irwin Setija (ASML). During v2, the main improvements were inclusion of transient coherent synchrotron radiation (CSR) impedances, including entrance and exit transients, and the longitudinal space charge (LSC) impedances.

Starting from v3, there were further improvements in both impedance models and beam dynamics. For the former, a new CSR drift impedance model was included based on Ref. [5], which allows for non-ultrarelativistic beams. More aspects of LSC models were added based on Ref. [6]. For the latter, since the theoretical formulation for both density and energy modulations was developed, the new features were soon incorporated into the code. This code becomes a useful tool to study how both the density and energy modulations evolve along a beamline and how they respond to different initial modulation wavelengths. It was found that the direct scalar multiplication of microbunching gain factors $G_1G_2...G_N$ may *underestimate* the overall microbunching effects [7]. Therefore, a comprehensive study should not only include the density gain. When including both density and energy modulations, the concept of scalar gains has been extended to gain matrix $\mathcal{G}_1\mathcal{G}_2...\mathcal{G}_N$, associated with state vector $[\mathbf{b}_k, \mathbf{p}_k]^T$. See Ref. [7] for more details.

During development of v3, a decision was made to split the code into two: one for constant-energy beam transport and the other for beam transport with acceleration or deceleration. Another extension, which was also separately implemented, incorporates the transport of a transversely coupled (or, magnetized) beam [8]. Note that these two options, i.e., beam transport involving energy change and magnetized beam transport, are not included in this released version (v4.2.1_r).

There are several options under development. If you have any questions/comments regarding this code, or find any potential bug, please kindly let me know and I will try my best to improve it in the next update.

2 Introduction

Using computer simulation to study accelerator physics topics, such as machine beamline design or beam dynamics, has become a very popular and indispensable way. For the latter, i.e., beam dynamics studies, in general, an intuitively straightforward way is resorted to the use of general purpose (macro-)particle tracking, such as `elegant` [3]. However, sometimes a specific code for a particular problem can be developed in order to perform more systematic studies. In this note, we are interested in the microbunching instability in a single-pass or multi-pass system. Such issue was observed in particle tracking simulation. However, it was then understood that such microbunching instability is relevant/sensitive to noise information seeded in the beam

from the very beginning. Therefore, although particle tracking simulation can be feasible, it encounters several challenging problems, such as how to generate a beam distribution with a noise level of physical sense, how to distinguish physical noise from numerical noise in a beam distribution, and how to suppress the numerical noise in the preparation of initial beam phase space distribution and during beam transport. A straightforward solution is to increase the number of (macro-)particles until the results become converged or, alternatively, to use a realistic number of particles in a realistic beam distribution. On a personal computer, the latter solution would require incredibly large amount of computation time and resources.

To systematically study the microbunching instability, particularly the underlying physics of microbunching gain development, using particle tracking simulation becomes uneconomical, although it can still serve as an excellent benchmarking tool. Motivated by the above mentioned issues, we developed a code, based on linearized Vlasov equation (a kinetic collisionless model), in order to study microbunching instability problem. Since 2014, we continue to improve this code by adding more physics models relevant to microbunching instability, creating more flexibility to inputs/outputs, and developing a GUI, so that this code can be easier to use for interested users.

In this note we first introduce and summarize the essentials of this code, including physics models (Sec. 3,4) and numerical algorithms (Sec. 5) implemented in the code. Then, we step-by-step illustrate the use of this code by a simple example (Sec. 6). We expect that this note can serve as a user manual for the code. For those who are not interested in the details of the code but would like to have a first try, they can simply jump to Sec. 6 by skipping Sec. 3,4,5.

This note was prepared partly based on JLab tech notes and my Ph.D. thesis [4], where more details may be found, including references.

3 Electron dynamics: Volterra integral equation

This section will sketch the governing equations that this code solves. To characterize the microbunched electron beam dynamics, let us define the density modulation (or, bunching factor) and energy modulation as

$$b(k_z;s) = \frac{1}{N}\int f_1(\mathbf{X};s)e^{-ik_z z_s}\mathrm{d}\mathbf{X} \tag{3.1}$$

$$p(k_z;s) = \frac{1}{N}\int (\delta_s - h_s z_s) f_1(\mathbf{X};s)e^{-ik_z z_s}\mathrm{d}\mathbf{X} \tag{3.2}$$

where $\mathbf{X}$ is the six-dimensional phase space coordinate, $\mathbf{X} = (x, x', y, y', z, \delta)$ with $\delta = \frac{\Delta E}{E_0}$, E_0 being the reference/nominal beam energy. h describes the correlation between z and δ (i.e., the chirp) of the bunch. The subscript s indicates the location of the (global) transport line. $k_z = \frac{2\pi}{\lambda_z}$ is the modulation wavenumber corresponding to a single modulation wavelength λ_z, assuming λ_z is much shorter than the full bunch duration, i.e., the coasting beam approximation. Here we use f_1 to indicate the *perturbed* phase space distribution by the collective effects. The complete description of the phase space distribution is $f = f_0 + f_1$, where f_0 evolves due to pure-optics beam transport. In what follows, we restrict ourselves to the linear regime, where $|f_1| \ll f_0$, i.e., the onset of microbunching instability.

The evolution of the beam phase space distribution $f(\mathbf{X};s)$ is governed by the collisionless Boltzmann equation or Vlasov equation. Direct solution in six-dimensional phase space is

extremely difficult. Here we are interested in the dynamics of small perturbed f_1. Therefore, Vlasov equation can be linearized. Moreover, the relevant quantities are not just f_1 itself, but the Fourier transform of it, i.e., Eq. (3.1), and one with weighting factor, i.e., Eq. (3.2). The linearized Vlasov equation in terms of f_1 can then be transformed from z to the spectral k_z domain. Solving the linearized Vlasov equations for both $b(k_z;s)$ and $p(k_z;s)$ will give us the following two integral equations [4]

$$b(k_z;s) = b_0(k_z;s) + i\int_0^s \frac{I_b(\tau)}{\gamma I_A} k_z(s) R_{56}(\tau\to s) Z_\parallel(k_z;\tau)\left\{\text{L.D.};\tau,s\right\} b(k_z;\tau)\mathrm{d}\tau \tag{3.3}$$

$$= b_0(k_z;s) + \int_0^s K(\tau,s) b(k_z;\tau)\mathrm{d}\tau \tag{3.4}$$

and

$$p(k_z;s) = p_0(k_z;s) - \int_0^s \frac{I_b(\tau)}{\gamma I_A}\left[1 - k_z^2(s) R_{56}^2(\tau\to s)\sigma_{\delta 0}^2\right] Z_\parallel(k_z;\tau)\left\{\text{L.D.};\tau,s\right\} b(k_z;\tau)\mathrm{d}\tau \tag{3.5}$$

$$= p_0(k_z;s) + \int_0^s \left[M(\tau,s) - L(\tau,s)\right] b(k_z;\tau)\mathrm{d}\tau \tag{3.6}$$

where I_b is the bunch current, γ is the relativistic factor of the beam reference energy, I_A is Alfven current (≈ 17045 A), $Z_\parallel(k_z;\tau)$ is the longitudinal impedance per unit length evaluated at $k = k_z$ at $s = \tau$, and $\sigma_{\delta 0}$ is the initial slice/uncorrelated energy spread of the beam. The phase space smearing factor $\{\text{L.D.};\tau,s\}$ will be defined below. The overall integrands, except $b(k_z;\tau)$ and $p(k_z;\tau)$, are written as the kernel functions $K(\tau,s)$, $L(\tau,s)$, and $M(\tau,s)$, or in the script `kernel_mod.m`, `kernel_mod_L.m`, and `kernel_mod_M.m`, respectively. In the absence of collective effects, the kernel functions vanish and $b(k_z;s) = b_0(k_z;s), p(k_z;s) = p_0(k_z;s)$. Note that the explicit expressions of $b_0(k_z;s)$ and $p_0(k_z;s)$ are available.

In Eqs. (3.3) and (3.5), the *relative* transport function $R_{56}(\tau\to s)$ can be determined by the relation $\mathbf{R}(\tau\to s) = \mathbf{R}^{-1}(\tau)\mathbf{R}(s)$, or explicitly written as

$$R_{56}(\tau\to s) = R_{56}(s) - R_{56}(\tau) + R_{51}(\tau)R_{52}(s) - R_{51}(s)R_{52}(\tau) + R_{53}(\tau)R_{54}(s) - R_{53}(s)R_{54}(\tau) \tag{3.7}$$

and the smearing factor

$$\{\text{L.D.};\tau,s\} = \exp\left\{-\frac{k_z^2(s)}{2}\left[\begin{array}{l}\epsilon_{x0}\beta_{x0}\left(R_{51}(\tau,s) - \dfrac{\alpha_{x0}}{\beta_{x0}}R_{52}(\tau,s)\right)^2 + \dfrac{\epsilon_{x0}}{\beta_{x0}}R_{52}^2(\tau,s) \\ +\epsilon_{y0}\beta_{y0}\left(R_{53}(\tau,s) - \dfrac{\alpha_{y0}}{\beta_{y0}}R_{54}(\tau,s)\right)^2 + \dfrac{\epsilon_{y0}}{\beta_{y0}}R_{54}^2(\tau,s) + \sigma_{\delta 0}^2 R_{56}^2(\tau,s)\end{array}\right]\right\} \tag{3.8}$$

with the *difference* transport function defined as $R_{5j}(\tau,s) = R_{5j}(s) - R_{5j}(\tau)$ with $j = 1,2,3,4,6$. For the trivial case, $R_{5j}(0,s) = R_{5j}(s)$. Those Courant-Snyder or Twiss functions, geometric emittances ϵ_{x0}, and $\sigma_{\delta 0}$ shown in Eq. (3.8) are read from `elegant` outputs and fed into `volterra_mat`. Note that in `elegant`, `*.lte` file, users must not turn on any collective effects, i.e., they should avoid using `CSRCSBEND`, `CSRDRIFT`, or `LSCDRIFT`.

In the `voltera_mat` code, it solves Eqs. (3.3) and (3.5) in vector-matrix form $\mathbf{AV} = \mathbf{V_0}$ (see Sec. 5 for more details). Depending on the impedance models specified by users, it will fill out the (strictly lower triangular) kernel matrices and solve for $\mathbf{V} = \mathbf{A^{-1}V_0}$.

4 Impedance models

As has been known, impedance plays an important role in microbunching phenomena. In this section, we briefly introduce the impedance models used in the code. We have applied some of these impedance models in some lattice designs, such as isochronous transport arcs [9],[10],[11],[13],[14], compressor arcs [12],[15], magnetic bunch compressor chicanes, low-energy beam merger lattices [16] and so on. Most of the results were compared with `elegant` tracking simulation and excellent agreement has been reached. For more details of these impedance models, we refer the interested readers to Ref. [4] and references therein.

4.1 Coherent synchrotron radiation (CSR)

4.1.1 Steady-state ultrarelativistic CSR impedance (s.s. UR-CSR)

For an ultrarelativistic beam ($\beta \to 1, \gamma \to \infty$) traversing an individual dipole, the steady-state CSR impedance in free space can be expressed as

$$Z_{\mathrm{CSR}}^{\mathrm{s.s.UR}}\left(k(s);s\right) = -\frac{ik^{1/3}(s)A}{\left|\rho(s)\right|^{2/3}} \tag{4.1}$$

where $k(s) = k_z(s)$ is the modulation wavenumber, $\rho(s)$ is the (signed) bending radius, and the constant $A = 3^{-1/3}\Gamma\left(\frac{2}{3}\right)\left(\sqrt{3}i - 1\right) = -2\pi\left[\frac{\mathrm{Bi}'(0)}{3} + i\mathrm{Ai}'(0)\right]$.

This simplest model is the mostly used impedance model when considering CSR effect. Note that the impedance is independent of beam energy. At low energy case, such as in a low-energy merger lattice (e.g., $\leq$50 MeV), it is suggested to use the non-ultrarelativistic CSR impedance expression in the next subsection.

4.1.2 Steady-state non-ultrarelativistic CSR impedance (s.s. NUR-CSR)

The above commonly used ultrarelativistic analysis can be extended to the lower energy regime. Without derivation, here we only quote the results. For a non-ultrarelativistic (NUR) electron beam ($\beta \to 1$, but $\gamma < \infty$) traversing a bending dipole, the free-space steady-state CSR impedance per unit length can be expressed as:

$$\mathrm{Re}\left[Z_{\mathrm{CSR}}^{\mathrm{s.s.NUR}}(k(s);s)\right] = -\frac{2\pi k(s)^{1/3}}{\left|\rho(s)\right|^{2/3}}\mathrm{Ai}'\left(x\right) + \frac{k(s)\pi}{\gamma^2}\left(\int_0^x \mathrm{Ai}(\varsigma)\mathrm{d}\varsigma - \frac{1}{3}\right) \tag{4.2}$$

$$\mathrm{Im}[Z_{\mathrm{CSR}}^{\mathrm{s.s.NUR}}(k(s);s)] \simeq \frac{2\pi k(s)^{1/3}}{\left|\rho(s)\right|^{2/3}}\left\{\frac{1}{3}\mathrm{Bi}'(x) + \int_0^x \left[\mathrm{Ai}'(x)\mathrm{Bi}(t) - \mathrm{Ai}(t)\mathrm{Bi}'(x)\right]\, dt\right\} \tag{4.3}$$

where $x = \left(k(s)\left|\rho(s)\right|\right)^{2/3}\Big/\gamma^2$, $k = 2\pi/\lambda$ is the modulation wave number, $\rho(s)$ is the bending radius, and Ai and Bi are Airy functions.

Equations (4.2) and (4.3) reduces back to Eq. (4.1) when $\gamma \to \infty$. Note that when evaluating these Airy functions (we use the special function `airy()` built in MATLAB), we find the poor convergence of Airy functions with large arguments, i.e., large x, which correspond to the case with lower energy or smaller wavelength [or larger $k(s)$]. Similar situations occur at the evaluation of the shielded CSR impedance; see below.

Figure 1 compares the steady-state ultrarelativistic CSR (s.s. UR-CSR) impedance with non-ultrarelativistic one (s.s. NUR-CSR), in the low (50 MeV) and high (750 MeV) energy situations. It can be seen, at higher energy, the two expressions give almost the same results.

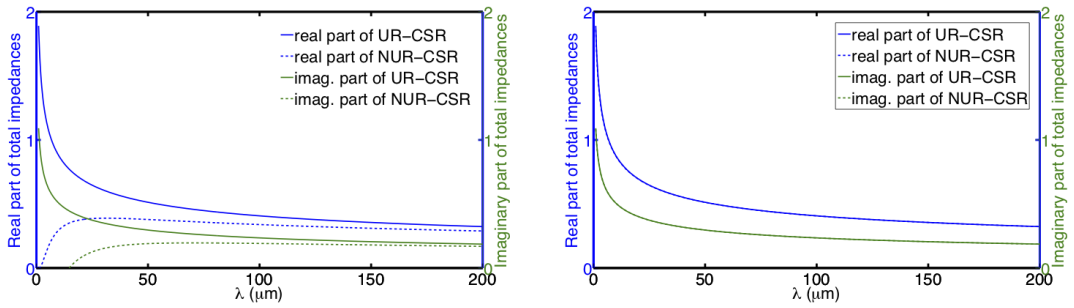

Figure 1: Comparison of steady-state UR-CSR and NUR-CSR at two different energies. (Left) 50 MeV; (right) 750 MeV. Here $\rho = 2$ m.

Figure 2 further illustrates how the s.s. NUR-CSR depends on the beam energy, approaches to the s.s. UR-CSR (left panel) and how its corresponding wakefield[2] behaves and compare with the following well-known CSR wakefield expression (right panel).

$$\left.\frac{\mathrm{d}\mathcal{E}(z)}{c\mathrm{d}t}\right|_{\mathrm{CSR,ss}} = -\frac{2Ne^2}{(3\rho^2)^{1/3}}\int_{-\infty}^{z}\frac{1}{(z-\zeta)^{1/3}}\frac{\partial\lambda(\zeta)}{\partial\zeta}\mathrm{d}\zeta = -\frac{2Ne^2}{\sqrt{2\pi}3^{1/3}\rho^{2/3}\sigma_z^{4/3}}F_0\left(\frac{z}{\sigma_z}\right) \tag{4.4}$$

with $F_0(x) = \int_{-\infty}^{x}\mathrm{d}\chi\frac{1}{(x-\chi)^{1/3}}\frac{\partial}{\partial\chi}e^{-\frac{\chi^2}{2}}$.

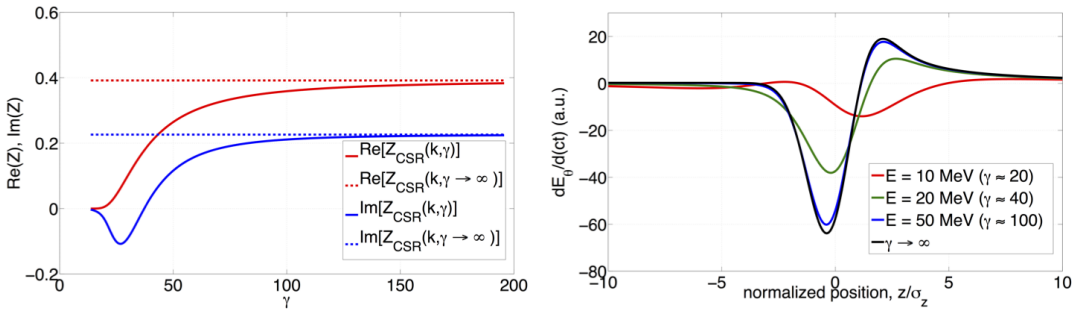

Figure 2: (Left) Energy dependence of the NUR-CSR impedance. Here we assume $k \approx 314$ cm^{-1} (or, $\lambda \approx 200$ μm) and $\rho = 1.5$ m. (Right) Comparison of CSR wakefields for $\gamma \to \infty$ using Eq. (4.4), with Eqs. (4.2) and (4.3) for finite γ via inverse Laplace transform (i.e., conversion from impedance back to wakefield).

4.1.3 Entrance transient CSR impedance

Prior to reaching steady-state interaction, the beam entering a bend from a straight section would experience the so-called entrance transient state, where the impedance can be obtained by Laplace transformation of the corresponding wakefield expression

$$Z_{\mathrm{CSR}}^{\mathrm{ent}}\left(k(s);s\right) = -\frac{4}{s^*}e^{-4i\mu(s)} + \frac{4}{3s^*}[i\mu(s)]^{1/3}\Gamma\left(-\frac{1}{3}, i\mu(s)\right) \tag{4.5}$$

[2]In time domain we assume the bunch z-distribution is Gaussian.

where $\mu(s) = k(s)z_L(s)$, $z_L(s) = \frac{(s^*)^3}{24|\rho(s)|^2}$, s^* the longitudinal coordinate measured from dipole entrance, and Γ the upper incomplete Gamma function. Note that the argument of the incomplete Gamma function is complex, instead of real. The built-in function `gammainc` in MATLAB only takes real argument. In the code, we call `mfun` for this particular feature of the special function.

Figure 3 compares the entrance transient impedance with steady-state UR-CSR for three different locations at the dipole entrance. Here one can see that, at the location very near dipole entrance, both the real and imaginary parts of the entrance transient impedances are smaller than the steady-state UR-CSR over all spectral ranges.

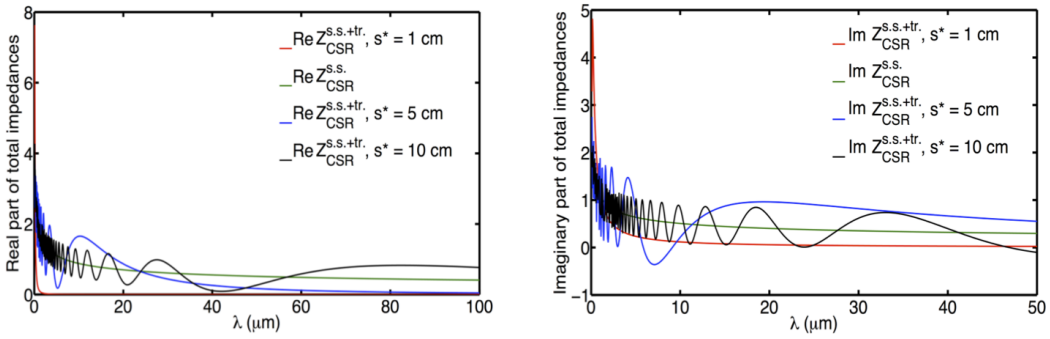

Figure 3: Illustration of entrance transient CSR impedance at several different entrance locations and their comparison with steady-state case. Here $\rho = 1.5$ m.

Note that if your MATLAB does not support `mfun`, one can open/edit the source file `csr1d_tr.m` in the file folder, and change the value of the variable `method` from 2 to 1. Then, this subroutine `gammaincc.m` would evaluate incomplete Gamma function in an alternative way (however, the numerical accuracy might not be guaranteed).

4.1.4 Exit transient CSR impedance

In addition to steady-state and entrance transient CSR impedances, there are exit transient effects as the beam exits from a dipole. For the case with fields generated from an upstream electron (at a retarded time) propagating *across* the dipole to downstream straight section, the corresponding impedance can be obtained

$$Z_{\rm CSR}^{\rm exit}(k(s);s) = -\frac{4}{L_b+2s^*}e^{-i\frac{k(s)L_b^2}{6|\rho(s)|^2}(L_b+3s^*)} \tag{4.6}$$

where s^* is the longitudinal coordinate measured from dipole exit (do not get confused with that defined in entrance transient case) and L_b is the dipole length.

For the other (and usually more dominant) part of exit CSR impedances, we use Bosch's expression (with proper modification)

$$Z_{\rm CSR}^{\rm drift}(k(s);s) \approx \begin{cases} \frac{2}{s^*} & \text{if } |\rho|^{2/3}\lambda^{1/3} \le s^* \le \frac{\gamma^2\lambda}{2\pi}, \\ \frac{2k(s)}{\gamma^2} & \text{if } s^* \ge \frac{\gamma^2\lambda}{2\pi} \\ 0 & \text{if } s^* \le |\rho|^{2/3}\lambda^{1/3} \end{cases} \tag{4.7}$$

where s^* is the longitudinal coordinate measured from dipole exit. This expression assumes that the exit impedance comes primarily from coherent edge radiation in the near-field region, and in our simulation we only include such transient effects right after a nearest upstream bend.

Note that this impedance model has only real/resistive part, and decays with s^*. Note also that this model may not be valid at extremely small wavelength regime. Figure 4 illustrates how this impedance model corresponds to the different modulation wavelengths and how it decays with s^*.

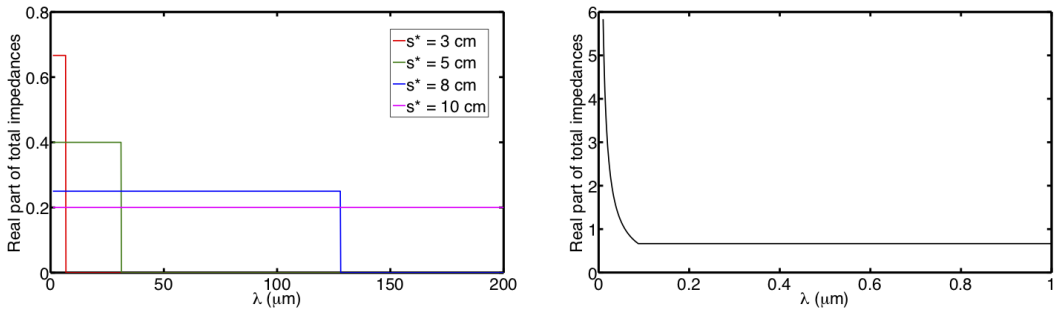

Figure 4: Illustrattion of exit CSR impedance (left) at different locations at downstream a bending dipole. Here the beam energy is assumed 750 MeV and $\rho = 2$ m. (Right) a special case when the wavelength is extremely small.

In addition to the above Bosch's expression implemented as our exit impedance model, we have another more dedicated exit CSR impedance model, direct quote without detailed derivation

$$Z_{\mathrm{CSR}}^{\mathrm{drif}}\left[k_z(s);s\right] = \frac{4}{\rho}\int_0^{\theta_m} \mathrm{d}\vartheta \frac{\mathrm{d}f(s^*,\vartheta)}{\mathrm{d}\vartheta} e^{-ik_z(s)\Delta z(s^*,\vartheta)} \tag{4.8}$$

where

$$f(s^*,\vartheta) = \frac{\frac{2}{\gamma^2}\left(\frac{s^*}{\rho}+\vartheta\right)+\vartheta^2\left(\frac{2s^*}{\rho}+\vartheta\right)}{\frac{4}{\gamma^2}\left(\frac{s^*}{\rho}+\vartheta\right)^2+\vartheta^2\left(\frac{2s^*}{\rho}+\vartheta\right)^2} \tag{4.9}$$

and

$$\Delta z(s^*,\vartheta) = \frac{s^*+\rho\vartheta}{2\gamma^2} + \beta\frac{\rho\vartheta^3}{24}\frac{4s^*+\rho\vartheta}{s^*+\rho\vartheta} \tag{4.10}$$

where s^* is again the longitudinal coordinate measured from the exit of the dipole and θ_m is the angle of a bending dipole with radius ρ. β and γ are relativistic Lorentz factors. Here we remind that in this code we only include the exit transient effects at a nearby upstream bend. Figure 5 illustrates the impedance spectrum from such a CSR drift expression, which depends on both the wavenumber and the distance away from the dipole. From the figure we can see that the most contributing regime occurs near dipole exit (or, at shorter distances) and longer wavelengths (or small k)

4.1.5 Steady-state ultrarelativistic CSR impedance with shielding effect by parallel plates

Close proximity of metallic walls to the beam orbit can possibly suppress the CSR effect. The wall shielding effect becomes important when the condition $h \leqslant \left(|\rho(s)|\,\lambda^2\right)^{1/3}$ is met, where h

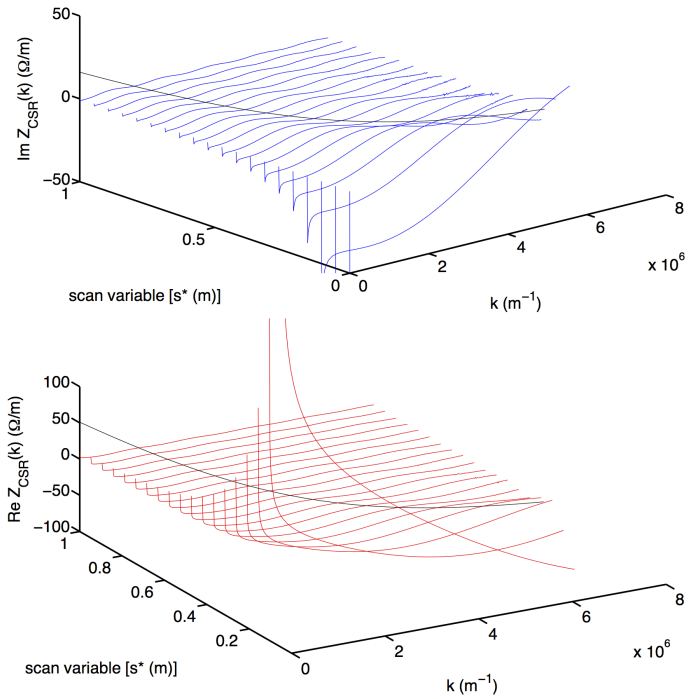

Figure 5: Real (bottom) and imaginary (top) parts of CSR drift impedance behavior evaluated by Eq. (4.8). Here $E_0 = 161$ MeV, $L_b = 40$ cm, and $\rho = 1.5$ m. The solid black curve is used to distinguish the slow and fast (larger k, longer s^*) oscillation.

is the *full* height of the beam pipe wall.

In this situation[3], one should consider the shielded CSR impedance in evaluating the CSR-induced microbunching gain. Currently we adopt the steady-state impedance in parallel-plate model as

$$Z_{\rm CSR}^{\rm pp}\left(k(s);s\right)=\frac{8\pi^2}{h}\left(\frac{2}{k(s)\left|\rho(s)\right|}\right)^{1/3}\sum_{p=0}^{\infty}F_0(\beta_p) \tag{4.11}$$

with $F_0(\beta)=\mathrm{Ai}'(\beta^2)\left[\mathrm{Ai}'(\beta^2)-i\mathrm{Bi}'(\beta^2)\right]+\beta^2\mathrm{Ai}(\beta^2)\left[\mathrm{Ai}(\beta^2)-i\mathrm{Bi}(\beta^2)\right]$ and $\beta_p=(2p+1)\frac{\pi}{h}\left(\frac{|\rho(s)|}{2k^2(s)}\right)^{1/3}$.

For a more realistic beam pipe geometry, such as the rectangular cross section, we consider to load impedance data externally calculated by other dedicated program. This part, albeit in the working plan, is however not implemented yet.

Note here that, similar to the case with entrance transient CSR impedance expression, due to poor convergence in MATLAB when evaluating Airy functions with large argument, here we adopt an option for the shielded CSR impedances from a specialized `elegant/SDDS` tool, called `csrImpedance`, for our code. The analytical expression used in `csrImpedance` is exactly the same as Eq. (4.11).

Figure 6 compares the shielded parallel-plate CSR impedances for different beam pipe heights with the free-space UR-CSR. It can be seen that both results become to merge when the beam pipe wall becomes larger and larger.

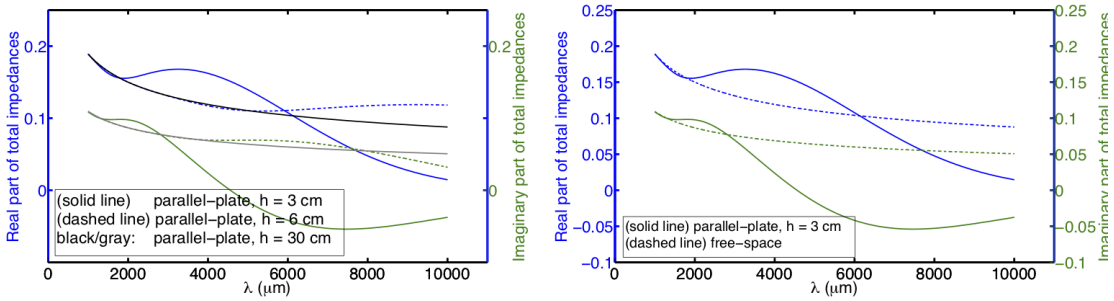

Figure 6: Comparison of shielded parallel-plate CSR impedance (left) for different beam pipe height and (right) with free-space UR-CSR case. Here $\rho = 2$ m.

4.2 Longitudinal space charge (LSC)

We care about space-charge-induced microbunching instability in situations such as along a long transport line or at low beam energy. A long beamline (e.g., a long drift section) can accumulate an amount of energy modulation due to longitudinal space charge (LSC) interaction. The energy modulation can be converted (via R_{56}) to density modulation downstream a dispersive region (or, nonvanishing momentum compaction region) and thus possibly amplify the microbunching gain. Here we implement three different LSC impedance models in the code.

[3]When the shielding CSR effect is turned on, the code automatically determines the condition $h \leqslant \left(|\rho(s)|\,\lambda^2\right)^{1/3}$ and, if the condition satisfies, replaces $Z_{\rm CSR}^{\rm s.s.UR}$ or $Z_{\rm CSR}^{\rm s.s.NUR}$ by the parallel-plate impedance $Z_{\rm CSR}^{\rm pp}$; otherwise, the code keeps using the original free-space impedance values.

4.2.1 LSC impedance with transverse circular uniform density

The first and the most commonly used one is the LSC impedance with transverse circular uniform beam distribution (on-axis model),

$$Z_{\rm LSC}^{\text{on-axis}}\left(k(s);s\right)=\frac{4i}{\gamma(s)r_b(s)}\frac{1-\xi K_1(\xi)}{\xi} \tag{4.12}$$

where $\xi=\frac{k(s)r_b(s)}{\gamma(s)}$ and the transverse beam size is determined to be $r_b(s)=\frac{1.747}{2}\left(\sigma_x(s)+\sigma_y(s)\right)$. Note here that $\sigma_x(s)$ and $\sigma_y(s)$ [adopted from `elegant *.sig` output] are transverse root-mean-squared (rms) beam sizes in the absence of space charge effect[4].

4.2.2 LSC impedance with transverse uniform averaging over r

The second model considers the average of the transverse beam distribution (called *average* model), where it relates the first model by the following relation,

$$Z_{\rm LSC}^{\rm ave}\equiv\frac{2}{r_b^2(s)}\int_0^{r_b} rZ_{\rm LSC}^{\text{on-axis}}(k,r)\mathrm{d}r \tag{4.13}$$

and the explicit impedance expression has the form,

$$Z_{\rm LSC}^{\rm ave}\left(k(s);s\right)=\frac{4i}{\gamma(s)r_b(s)}\frac{1-2I_1(\xi)K_1(\xi)}{\xi} \tag{4.14}$$

where $\xi=\frac{k(s)r_b(s)}{\gamma(s)}$ and the transverse beam size is determined to be $r_b(s)=\frac{1.747}{2}\left(\sigma_x(s)+\sigma_y(s)\right)$.

Figure 7 illustrates the slight difference between the first (on-axis) and the second (average) models.

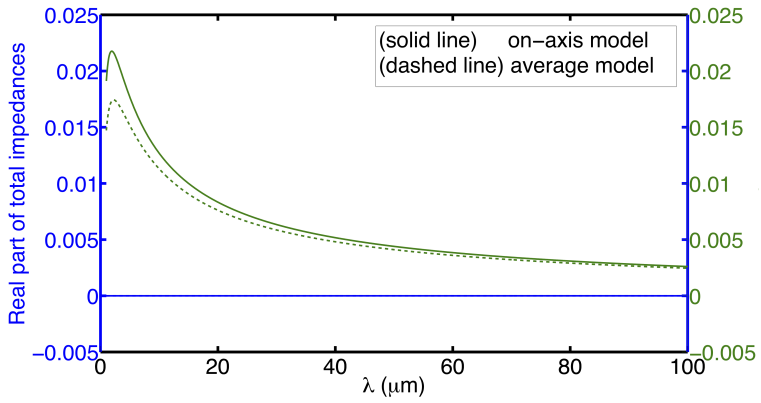

Figure 7: Comparison of two LSC impedances (in unit of cm^{-1}). Here the beam energy is assumed 750 MeV and $\sigma_x=\sigma_y=0.5$ mm. The blue lines are the real part and the green lines are imaginary part of the impedance.

[4]This means that users should avoid using `LSCDRIFT` or `CSRDRIFT` in `*.lte`. Instead, change all `LSCDRIFT` and `CSRDRIFT` to `DRIF`.

4.2.3 LSC impedance with transverse axis-symmetric Gaussian distribution averaging over r

The third model, taking into account the axisymmetric transverse Gaussian distribution, has the following form,

$$Z_{\rm LSC}^{\rm ave,Gaussian}\left(k(s);s\right)=i\frac{\xi_\sigma e^{\xi_\sigma^2/2}}{\sigma_{x,y}(s)\gamma(s)}{\rm Ei}\left(-\frac{\xi_\sigma^2}{2}\right) \tag{4.15}$$

where $\xi_a=\frac{k(s)\sigma}{\gamma}$, ${\rm Ei}(x)$ is the exponential integral function, defined as ${\rm Ei}(x)=-\int_{-x}^{\infty}\frac{e^{-t}}{t}dt$ and $\sigma(s)=\sigma_{x,y}(s)$.

In this code, we also employ the following expression for LSC impedance with round pipe and uniform round beam,

$$Z_{\rm LSC}^{\substack{\text{uniform-round}\\ \text{round-pipe}}}(k)=\frac{4i}{\gamma r_b}\frac{1}{\xi_b}\left[1-\xi_b\left(K_1(\xi_b)-K_0\left(\frac{\xi_b r_p}{r_b}\right)\frac{I_1(\xi_b)}{I_0\left(\frac{\xi_b r_p}{r_b}\right)}\right)\right] \tag{4.16}$$

where the radius of the beam pipe is r_p and assumed with perfect conduction. At high-frequency limit, for $r_p/r_b>1$, the second term in the round bracket decreases exponentially with $\xi_b=kr_b/\gamma$, so the expression is reduced to the free-space on-axis case, as expected.

Note that all the above LSC impedance models, except for the free-space on-axis one, are not benchmarked. For the above four LSC impedance models, the transverse beam sizes (in the absence of space charge effect) are read from `Sx` and `Sy` of `elegant` output (`*.sig`).

4.3 High-frequency linac geometric impedance

If a beam experiences acceleration, deceleration, or chirping along a section of a linac with RF cavities, the periodic structure in general features a linac geometric impedance. Here we consider this effect by adopting the following expression,

$$Z_{\rm Linac}^{\rm UR}\left(k(s);s\right)\approx\frac{4i}{k(s)a^2}\frac{1}{1+(1+i)\frac{\alpha L}{a}\sqrt{\frac{\pi}{gk(s)}}} \tag{4.17}$$

with $\alpha\approx 1-0.4648\sqrt{\frac{g}{L}}-0.07\left(\frac{g}{L}\right)$. In Eq. (4.17), a is average iris radius, g is the gap distance between irises, and L is the cell/period length. Figure 8 shows the geometric cell of this simplified model.

Figure 9 demonstrates the geometric impedance spectrum, based on a typical SLAC S-band traveling-wave linac. The real part of the impedance scales as $\lambda^{3/2}$, while the imaginary part increases with λ.

5 Numerical algorithms

5.1 Solution to Volterra integral equation

In the previous two sections we have prepared required information as our main building block of the solver. In this subsection, we briefly summarize the numerical algorithm to solve the

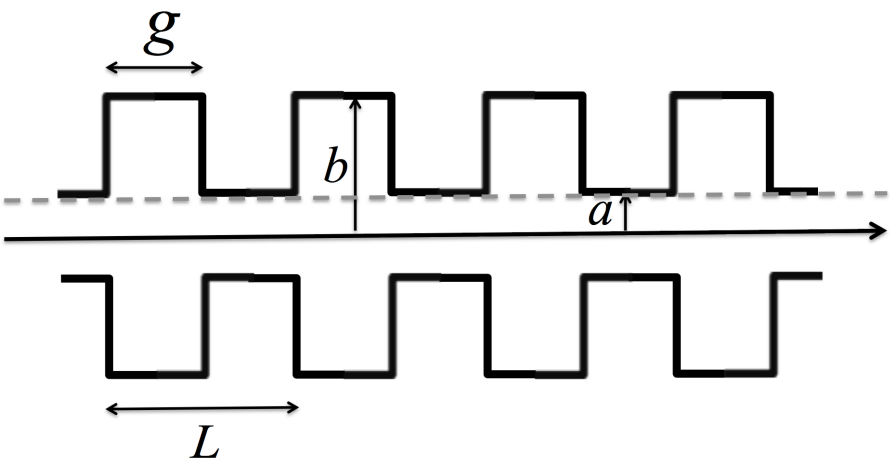

Figure 8: Geometric illustration of a cell/periodic structure.

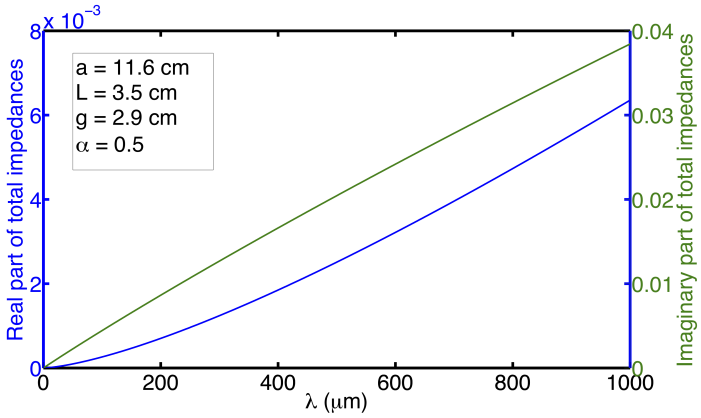

Figure 9: Illustration of the linac geometric impedance with parameters based on SLAC S-band traveling-wave linac.

Volterra integral equations Eqs. (3.3) and (3.5). The numerical algorithm is based on the trapezoidal quadrature rule, which splits in the discrete sum as,

$$b(s_i) = b_0(s_i) + \Delta s \left(\frac{1}{2} K(s_i, 0) b_0(0) + \sum_{j=1}^{i-1} K(s_i, u_j) b(u_j) \right) \tag{5.1}$$

for $s_i = s_0 + i\Delta s$ and $u_j = u_0 + j\Delta s$ being the grid points along the path length, with i and j the mesh/grid indices. Here Δs is the mesh spacing, s_0 and u_0 are the initial positions. In our case, we assume $s_0 = u_0 = 0$. We further express Eq. (5.1) in the matrix form to be,

$$\begin{bmatrix} b(s_1) \\ b(s_2) \\ b(s_3) \\ b(s_4) \\ \vdots \\ b(s_{N-1}) \\ b(s_N) \end{bmatrix} = \begin{bmatrix} b_0(s_1) \\ b_0(s_2) \\ b_0(s_3) \\ b_0(s_4) \\ \vdots \\ b_0(s_{N-1}) \\ b_0(s_N) \end{bmatrix}$$

$$+ \Delta s \begin{pmatrix} 0 & & & & & 0 \\ \frac{1}{2}K(s_2, s_1) & 0 & & & & \\ \frac{1}{2}K(s_3, s_1) & K(s_3, s_2) & 0 & & & \\ \frac{1}{2}K(s_4, s_1) & K(s_4, s_2) & K(s_4, s_3) & 0 & & \\ \vdots & \vdots & \vdots & \vdots & & \\ \frac{1}{2}K(s_{N-1}, s_1) & K(s_{N-1}, s_2) & K(s_{N-1}, s_3) & \cdots & 0 & \\ \frac{1}{2}K(s_N, s_1) & K(s_N, s_2) & K(s_N, s_3) & \cdots & K(s_N, s_{N-1}) & 0 \end{pmatrix} \begin{bmatrix} b(s_1) \\ b(s_2) \\ b(s_3) \\ b(s_4) \\ \vdots \\ b(s_{N-1}) \\ b(s_N) \end{bmatrix} \tag{5.2}$$

or, in a shorthand notation,

$$\mathbf{b} = \mathbf{b}_0 + \mathbf{K}\mathbf{b} \tag{5.3}$$

It can be seen that the upper-right elements of the kernel matrix $\mathbf{K}$ vanish. For the remaining (bottom-left) elements of $\mathbf{K}$, the corresponding $\{\text{L.D.};\tau, s\}$ are evaluated by linear interpolating the associated transport matrix functions extracted from `elegant` with proper transformations of dynamic variables, including sign change for R_{5j}. Depending upon the validity of criteria for the impedance models mentioned above, the overall resultant impedance at a certain location s' is evaluated by taking the *sum* over individual effects[5].

Numerical integration of the above integral equation becomes equivalent to finding the inverse of the matrix $(\mathbf{I} - \mathbf{K})$ and multiplied by $\mathbf{b}_0$, i.e., $\mathbf{b} = (\mathbf{1} - \mathbf{K})^{-1}\mathbf{b}_0$. Numerical convergence requires that the step size of the numerical integration be small enough to resolve the fastest variation of the relevant impedance along a beamline. The convergence tests should be done before producing the microbunching gain spectral curves. An example is shown in Fig. 10. General sawtooth behavior may be expected due to, for example, finite number of meshes on individual dipoles along s-integration. As the mesh number increases, such sawtooth amplitude is reduced and the results would converge.

[5]This indeed greatly enhances the power of employing frequency-domain treatments in the analysis of collective instability studies.

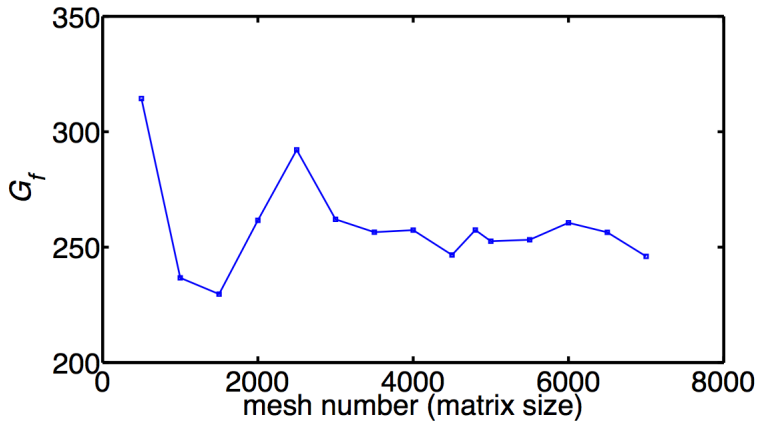

Figure 10: Convergence test of mesh number for the 1.3 GeV high-energy transport arc lattice. In this convergence test, the simulation parameters are the same as those shown in Table 6-1 (see Ref. [4]) and the initial modulation wavelength $\lambda = 40$ μm.

When constructing the kernel matrix elements, $K(s_i, s_j)$ as well as $\mathbf{b}_0$, the code reads the relevant initial beam and Twiss parameters and extracts transport functions from `elegant` output files. It takes an advantage of treating a general linear lattice[6] and also utilizes most of the capabilities born in `elegant`.

5.2 Evaluation of various impedances

- Entrance transient CSR (ent. tr. CSR)
  In this code, we first determine if the global coordinate s locates within a dipole. If so, we determine the local coordinate s^* with respect to the dipole entrance and then evaluate the entrance transient CSR impedance.

  In evaluation, we need to compute a value associated with incomplete Gamma function Γ with complex argument. Unfortunately, MATLAB of current version (2017a) does not support such implementation with complex argument. As an alternative way, we call a special function by using `mfun`. Because of this reason, this code would take longer time (compared to that including only steady-state CSR impedance) if the entrance transient effect is considered.

- Steady-state non-ultrarelativistic CSR (s.s. NUR-CSR)
  As summarized in Eqs. (4.2) and (4.3), the evaluation of this impedance model sometimes involves the computation of Airy functions with large argument (e.g., for the cases with lower energy or smaller wavelength). In this situation, MALTAB `airy()` can have poor convergence; sometimes the function may return `NaN`.

- Steady-state parallel-plate shielded CSR
  The above situation also occurs in the evaluation of shielded CSR impedance [see Eq. (4.11)]. Fortunately, we can apply a specialized subroutine developed in `elegant/SDDS`, `csrImpedance`, which takes care of this situation and evaluates such effect by using the same analytical expression as that we intend to use.

[6] In `elegant`, there is no such restriction. This limit stems from our theoretical formulation.

- Longitudinal space charge impedances (on-axis, average, axisymmetric Gaussian)
  As already mentioned earlier, evaluation of LSC impedances requires the information of the transverse beam sizes (`Sx` and `Sy` in `elegant`). Here we adopt these two parameters from `elegant` output file (`*.sig`) and take a weighted average of them.

- Drift element along the beamline
  To simulate LSC-induced microbunching gain, we require the non-ultrarelativistic effect along a straight/drift section. When $\gamma = \infty$ or $R_{56}(\tau \to s) = 0$, there would be no LSC. After reading the transport functions (e.g., R_{56}) along a beamline, we make slight modification to the momentum compaction function as

$$R_{56}(s) \to R_{56}(s) + \frac{s}{\gamma^2} \tag{5.4}$$

  There are two advantages with such slight modification:

  1. the space-charge-induced plasma oscillation can be properly accounted for in the microbunching gain calculation;
  2. Landau damping (or phase space smearing) effect on microbunching gain for CSR interaction on a drift following a dipole can be also considered more accurately.

- RF cavity element
  In this released version, we assume no RF element in the beamline. Note that we have developed a method to derive the 6×6 transport matrix by directing tracking a small (but, sufficient) number of independent (macro-)particles in `elegant` at various locations and extracting the dependences between dynamical variables or coordinates. It is found that in this way we can treat an even more general linear lattice including beam acceleration (or, deceleration).

6 Step-by-step illustration: LCLS BC2 as an example

If one has `elegant` and `SDDS` Toolkit successfully installed and cannot wait reading the step-by-step guide before running the code, one can simply unzip the downloaded file `volterra_mat_v4.2.1r.zip`. The first thing is to find out where your MATLAB is[7], and the version of it. In the script `run_script` it is assumed that the location is in `/Applications/MATLAB_R2017a.app/bin/matlab`. If your version is not 2017a or your location of MATLAB is not as addressed, you will need use your address where your MATLAB is installed.

Then, open a terminal screen (e.g., `iTerm.app`), go to this folder, and type `bash run_script`. `volterra_mat` will run `elegant` and soon MATLAB and a GUI will pop up. Then you can enjoy using this code. If you use Windows, you may need to modify the commands in the bash script `run_script` to make sure you have the corresponding functionalities, e.g., the stream editor command `sed` and the command `rm` to remove directory entries.

6.1 Before running `volterra_mat`

`volterra_mat` is the Vlasov solver for microbunching gain calculation. It uses several outputs from `elegant` fed as `volterra_mat` input files. There are several steps that users need to setup in `elegant` for subsequent successful `volterra_mat` calculation

[7]The following instruction assumes that you run this code on MacOS.

6.1.1 Install `elegant` and `SDDS` Toolkit

Before running this code, we need `elegant` and its accompanying `SDDS` Toolkit to be successfully installed because we need many `elegant` input and output results and utilize several `SDDS` subroutines.

Relevant information, source codes and executables, can be found and downloaded at: https://www.aps.anl.gov/Accelerator-Operations-Physics/Software

6.1.2 Input format restriction for `elegant` files

To run this code, we place further restrictions on the input format of `elegant` files (`*.ele` and `*.lte`) in order for our code to read relevant information

1. Suggest to always use lower-case characters for variable names.
2. Do not set more than one of the following parameters in a single command line in the input file (`*.ele`) ▶ one parameter in one command line
   (a) initial Twiss parameters (`betax, betay, alphax, alphay`), and
   (b) initial beam parameters (`emit_nx, emit_ny, sigma_dp, sigma_s or sigma_t, and momentum_chirp`)[8].

   See Fig. 11 for the allowed and disallowed examples.

```
&bunched_beam
        n_particles_per_bunch = 500,
        emit_nx = 1.0e-6, emit_ny = 1.0e-6,
        beta_x = 105, beta_y = 22,
        eta_x = 0.0, eta_y=0.0,
        alpha_x = 5, alpha_y = 0,
        sigma_dp = 3.0e-6,
        sigma_s = 195.0e-6,
        momentum_chirp = 39.83,
        distribution_type[0] = 3*"gaussian",
        distribution_cutoff[0] = 3*10,
        halton_sequence[0] = 3*1,
        halton_radix[0] =5,7,5,7,2,3,
        randomize_order[0] = 3*2,
        optimized_halton = 1,
&end
```

```
&bunched_beam
        n_particles_per_bunch = 500,
        emit_nx = 1.0e-6,
        emit_ny = 1.0e-6,
        beta_x = 105,
        beta_y = 22,
        eta_x = 0.0,
        alpha_x = 5,
        alpha_y = 0,
        sigma_dp = 3.0e-6,
        sigma_s = 195.0e-6,
        momentum_chirp = 39.83,
        distribution_type[0] = 3*"gaussian",
        distribution_cutoff[0] = 3*10,
        halton_sequence[0] = 3*1,
        halton_radix[0] =5,7,5,7,2,3,
        randomize_order[0] = 3*2,
        optimized_halton = 1,
&end
```

Figure 11: (left) disallowed format; (right) allowed format.

3. It is known that `elegant` can use an exclamation mark `(!)` as a starting symbol over a line comment. Note that, in `*ele`, if, for some reason, a parameter is assigned its value more than once, remember to put the desired one (i.e., those to be read by `volterra_mat`) at the top of those commented lines. See Fig. 12 for the allowed and disallowed examples.

[8]The parameter `momentum_chirp` is optional to assign, while the remaining initial Twiss and beam parameters are required.

```
&bunched_beam
        n_particles_per_bunch = 500,
        emit_nx = 1.0e-6,
        emit_ny = 1.0e-6,
!       beta_x = 105,
!       beta_y = 22,
        beta_x = 100,
        beta_y = 20,
        eta_x = 0.0,
        alpha_x = 5,
        alpha_y = 0,
        sigma_dp = 3.0e-6,
        sigma_s = 195.0e-6,
        momentum_chirp = 39.83,
        distribution_type[0] = 3*"gaussian",
        distribution_cutoff[0] = 3*10,
        halton_sequence[0] = 3*1,
        halton_radix[0] =5,7,5,7,2,3,
        randomize_order[0] = 3*2,
        optimized_halton = 1,
&end
```

```
&bunched_beam
        n_particles_per_bunch = 500,
        emit_nx = 1.0e-6,
        emit_ny = 1.0e-6,
        beta_x = 100,
        beta_y = 20,
!       beta_x = 105,
!       beta_y = 22,
        eta_x = 0.0,
        alpha_x = 5,
        alpha_y = 0,
        sigma_dp = 3.0e-6,
        sigma_s = 195.0e-6,
        momentum_chirp = 39.83,
        distribution_type[0] = 3*"gaussian",
        distribution_cutoff[0] = 3*10,
        halton_sequence[0] = 3*1,
        halton_radix[0] =5,7,5,7,2,3,
        randomize_order[0] = 3*2,
        optimized_halton = 1,
&end
```

Figure 12: (left) disallowed format; (right) allowed format.

4. In the lattice file (`*.lte`), the only information we directly need is the amount of bunch charge. We use the value of bunch charge and the rms bunch length (`sigma_s` or `sigma_t`) from `*.ele` file to calculate the (peak) bunch current[9]. If either bunch charge or rms bunch length is not given, users can manually assign the (peak) beam current in main program (GUI) later. The setting of bunch charge is restricted to the following format:
   `q:  charge, total=`
   In other words, please use `q` (the lower case) as the symbol, `charge` and `total` as the variables.

5. Thus far the existing formulation of microbunching theory does not allow any transverse-coupling element in the beamline, e.g., a solenoid element should be avoided. Reducing the length of solenoid section to zero can avoid giving incorrect results. Keeping the solenoid section as the drift section might be okay if the overall 4×4 transport matrix is presumed the identity matrix. The formulation becomes invalid if there is *bend solenoid* (i.e., combined-function dipole with superimposed solenoidal field).

6. Suggest to add the following section of action command into the top of input file (`*.ele`):

   ```
   &divide_elements
   name = *,
   type = SBEN
   maximum_length = 0.01,
   &end
   ```

   When simulating CSR-induced microbunching gain, it may be important to resolve the gain variation within an individual dipole. Without using the above action command, the code can only perform a simple linear interpolation to obtain the transport functions (e.g., R_{56}) inside a dipole, and this may lead to inaccurate results.

7. For `elegant` output files, the following are required for running `volterra_mat`,

[9] In this code, the bunch current (in Ampere) is defined as I_b = (bunch charge in Coulomb)/(2.35 × `sigma_t` in sec). The factor $2.35 \approx 2\sqrt{2 \ln 2}$.

Name of file extension	Purpose of use
`ele`	read basic beam parameters
`lte`	read bunch charge to calculate peak current I_b
`sig`	read rms transverse beam sizes $\sigma_x(s), \sigma_y(s)$
`cen`	read beam reference energy/momentum
`mag`	read lattice profile
`param`	read lattice element parameters
`flr`	read dipole geometric information to calculate (signed) bending radius $\rho(s)$
`mat`	read 6×6 transport functions $R_{ij}(s)$

8. Those `elegant` input and output files must share the same name (i.e., rootname) because the code is set to search the relevant information assuming that a common rootname is used ▶ always set the filename and rootname the same in `*.ele` and `*.lte` files; it is set in `&run_setup`, see the example file (`LCLSBC2.ele`) attached in the package file.

6.1.3 Run the script

After correctly setting up the `elegant` input files (`*.ele` and `*.lte`), we provide a simple bash script (`run_script`) to control the whole simulation flow, including running `elegant`, `volterra_mat`. To start the calculation, one should specify in the file `run_script` the rootname of the lattice, which users would like to simulate. In the default, it was put 'LCLSBC2' (without quote). Then, in the terminal screen, type

```
bash run_script
```

6.1.4 Some notes

Users are reminded that there is no need to impose too many (macro-)particles when running `elegant` input/output preparation. In general, users can use moderate number of (macro-)particles to ensure the convergence of `Sx` and `Sy` in `elegant`. If LSC is not considered, one can even use a single particle to run `elegant` preparation.

6.2 Main program: `volterra_mat_for_GUI.m`

When `elegant` run is done (it should be done in a few seconds, if the beamline is not too complicated), the main program `volterra_mat` is launched by `run_script` and MATLAB Command Window should show up. In MATLAB Command Window, one would see descriptive text messages, where one can input additionally required physics and numerical parameters. Afterwards, a GUI will pop up, where one can change or (re-)adjust those parameters at one's desire. See Fig. 13 for a snapshot of the main program GUI.

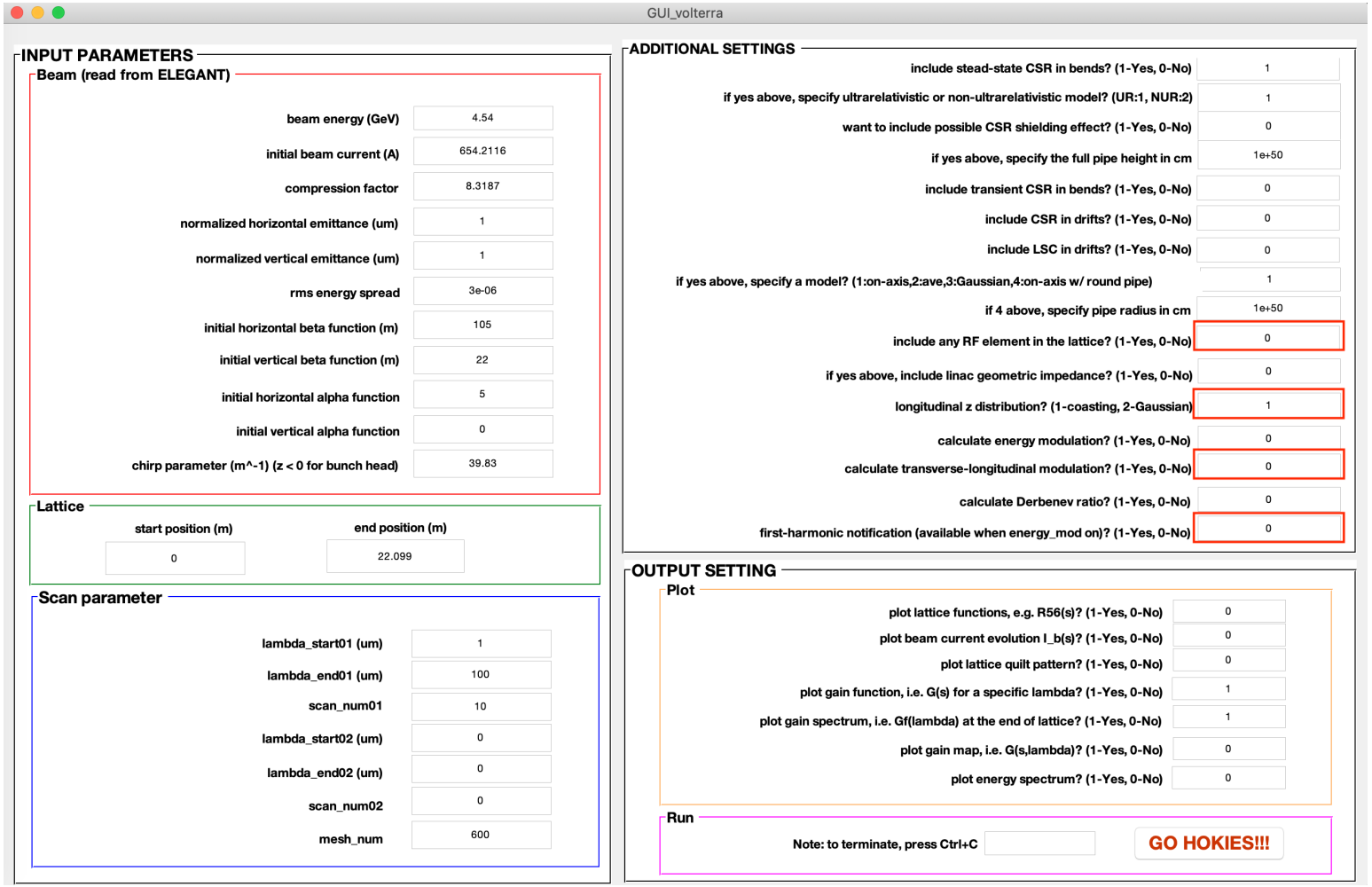

Figure 13: GUI of the main program. The four options marked with red boxes in the right panel are not available in this version.

The upper left area displays the (initial) beam parameters taken from `elegant`, where the definition of the initial beam current is described before, assuming a bunch charge is given. If either bunch charge or rms bunch length is not given, one can manually assign the initial beam current here. The initial compression factor is defined as $C = [R_{55}(s_0) - h_0 R_{56}(s_0)]^{-1}$.

The bottom left area is for numerical setting to calculate gain functions and/or gain spectra. The `mesh_num` refers to the number of integration steps used to evaluate the governing integral equations, where `mesh_num = (s_end-s_start)/ds`. A convergence test is always suggested to perform to ensure the results stay converged with respect to `mesh_num` (thus ensure the smallness of `ds`). In additional to `mesh_num`, there are additional numerical parameters that must be assigned to produce gain spectra $G_f(\lambda)$: $\lambda_{\rm start}$, $\lambda_{\rm end}$ and `scan_num` $= (\lambda_{\rm start} - \lambda_{\rm end})/\Delta\lambda$. In some situations, the gain spectra may feature multi-bumps; if one would only like to resolve the peak gain, this program can give two sets of separated scanning intervals.

The output information produced from `volterra_mat` consists:

1. gain function: $G(s)$ as a function of global path-length coordinate s
2. gain spectrum: $G_f(\lambda)$ at the exit of beamline as a function of (initial) modulation wavelength λ
3. gain 'map': $G(s, \lambda)$ as functions of both s and λ

When shielded CSR impedance and/or linac geometric impedance become of interest, one needs to set up relevant structure parameters manually in the file `kernel_mod.m`. As for the shielded CSR impedance, the required parameter is the full height (h) of the beam pipe in cm. As for linac geometric impedance, it requires three additional parameters: the average iris radius (a) in cm, the gap distance between irises (g) in cm, and the cell/period length (L) in cm.

Below are some solutions if special situations are encountered:

1. If one closes GUI during the program execution, one can go back to Command Window and type the following command to resume GUI:

   ```
   GUI_volterra;
   ```

2. If one changes the integration numbers for solving the integral equation, i.e., `mesh_num` (this mostly happens when doing convergence tests at the beginning of studying a new problem), one needs to clear all the relevant variables (scalars, vectors, and arrays). Please type in Command Window

   ```
   clear all;
   ```

3. In some situation, one may be interested in the gain spectrum over a certain location other than the exit of a lattice. Please always keep the starting position zero (`start_pos = 0`) and change or (re-)adjust the end position (`end_pos`).

6.3 Benchmarking with particle tracking simulation and analytical formulas

Here we provide a set of example files (`*.ele` and `*.lte`) for the second bunch compressor lattice of LCLS (parameters based on Ref. [1, 2]).

Figures 14 and 15 show the results obtained from `volterra_mat` and `elegant` tracking. They show excellent agreement between the two distinct numerical algorithms (kinetic model vs. particle tracking). More detailed description and convergence procedures for `elegant` tracking were discussed in Ref. [17, 18] or [19].

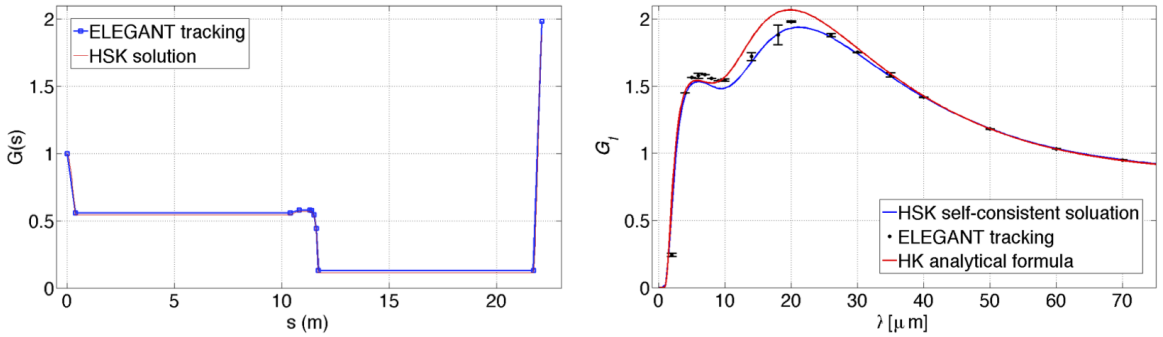

Figure 14: (Left) CSR gain function $G(s)$. The modulation wavelength is 20 μm and the initial modulation amplitude is assume 2.0 % in `elegant` tracking. (Right) CSR gain spectrum $G_f(\lambda)$. The dots are from the average of gains extracted from the scanned modulation amplitudes (ranging from 0.05 % to 0.2 %). The error bar length is quantified as two times the standard deviations of those numerical gains obtained for each different modulation amplitudes.

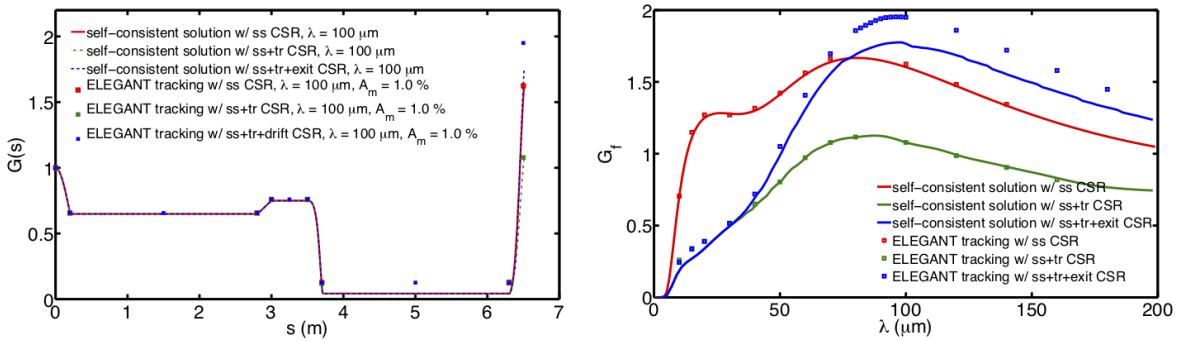

Figure 15: CSR gain functions (left) and spectra (right) with inclusion of entrance and/or exit transient effects for LCLS BC1.

To benchmark LSC-induced microbunching gain evolution, we set up a simple 1.75-m drift with a relatively low beam energy 5.5 MeV for benchmarking. Figure 16 compares the `volterra_mat` solutions with `elegant` and `TStep`.

We note that there are still two impedance models that we have not yet benchmarked, in particular:

1. linac geometric impedance;
2. parallel-plate shielding CSR impedance.

Benchmarking the two model cases with `elegant` for a single-pass system may not be trivial, as they are treated in the lumped models.

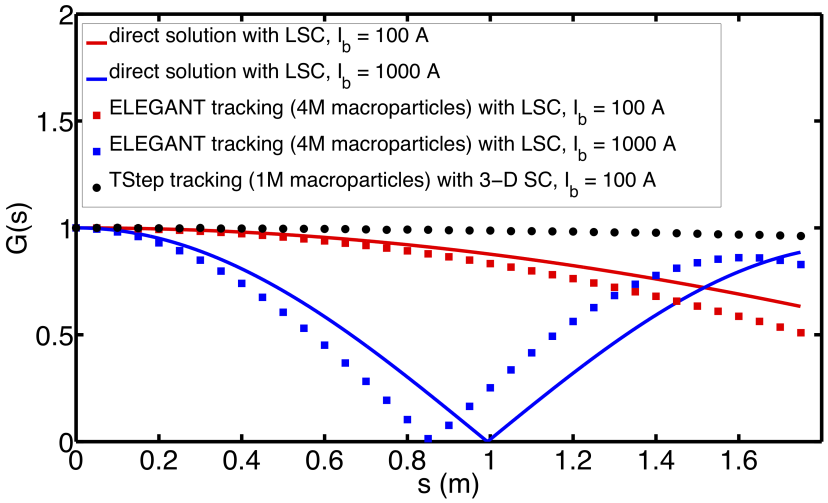

Figure 16: LSC gain functions for a 1.75-m drift. Note that we have only 1-D LSC model implemented in our code (as well as in `elegant`), `TStep` results with 3-D SC are for further comparison of the $G(s)$. A slight difference between `volterra_mat` solutions (red or blue dots) and `elegant` results (red or blue curves) comes from the fact of the transverse beam distortion (thus the change of rms beam sizes $\sigma_x(s)$ (`Sx`) and $\sigma_y(s)$ (`Sy`) would not be properly accounted for the transverse beam distributions in `voltera_mat`).

7 Future possible improvements

Since this code is still under development, many issues listed below are to be improved or added later in future versions:

7.1 GUI

More friendly main program or postprocessing GUI should be developed to facilitate users for microbunching gain calculation.

7.2 Physics issues

7.2.1 Adopt numerical impedance models:

Thus far, this code only uses analytical impedance models, which are, in most situations, over-simplified. Possible high-frequency impedances of various configurations should be taken into account by numerically adopting external (thus more general) impedance tabular data.

7.2.2 Element-based modeling of microbunching gain calculation:

Unlike s-based modeling as implemented in this code, one can develop the so-called *element-based* modeling for microbunching gain estimation. Microbunching gain is calculated for a certain beamline in a lumped manner, and can serve as a quick estimation for a more complicated system.

7.2.3 Bunched-beam model:

The existing coasting-beam model can be improved to a more general one, the bunched-beam model, for microbunching analysis. The modulation wavelengths are no longer independent; the bunched-beam model becomes necessary in simulation of some ERLs or recirculating accelerators.

7.2.4 Parasitic compression:

In some situations, a bunch can experience the so-called 'parasitic' compression, where the coasting-beam model is no longer valid. This can be treated by the extension to bunched-beam model.

7.3 Numerical issues

7.3.1 Issues on computation time, efficiency

As for using this code to optimize/minimize microbunching instability for a lattice design, performance (in particular, the computation time and memory usage) would become an issue. Currently, MATLAB takes much time in:

1. fill elements into the kernel matrix;
2. find the inverse of the kernel matrix.

For the former, the process involves a nested *for* loop and the performance may be improved by mixing programing with `C/C++, Python`, or by even translating `volterra_mat` from MATLAB to `C/C++,Python`, if necessary.

7.3.2 Issues of Ai, Bi, and $\Gamma(a+ib)$

As mentioned, we encounter the numerical difficulty in evaluating Airy function (of large argument) and the incomplete Gamma function (of complex input argument) because of poor convergence in MATLAB. This issue should be resolved in some future version.

7.4 Output format

In `elegant`, several input/output data manipulations have been done by using `SDDS` (Self-Describing Data Sets) file protocol. With this powerful tool, we may integrate this code more to `elegant` through `SDDS` Toolkit once this code gets matured.

8 Summary

In this note, we introduced this developed code, `volterra_mat` for microbunching gain calculation along a general linear transport line, summarized the relevant impedance models, and outlined numerical algorithm behind the code.

Then, the step-by-step guide to running this program is illustrated. We use the example lattice, LCLS BC2, to benchmark the results from `volterra_mat` against particle tracking simulations `elegant`. The results match well with each other.

Finally, we summarize the possible future improvements of this code from the graphical interface, physics models, numerical computation, and input/output perspectives. Users are always welcome and appreciated to give feedbacks and suggestions to this code.

9 Acknowledgements

Over the years, I would like to express my sincere gratitude to Rui Li (JLab) and am grateful to David Douglas (JLab, now retired) for his kind sharing with me many perfectly designed lattices for my graduate study and during the development of this code. I would also like to thank Chris Tennant (JLab) for many helpful suggestions on this code and format translation from `DIMAD` to `elegant`, and Steve Benson (JLab) for many stimulating suggestion, encouragement, and discussion on possible improvement of impedance models.

This work is supported by the Fundamental Research Funds for the Central Universities under Project No. 5003131049, National Natural Science Foundation of China under project No. 11905073., and was by Jefferson Science Associates, LLC, under U.S. DOE Contract No. DE-AC05-06OR23177.

References

[1] S. Heifets *et al.*, Coherent synchrotron radiation instability in a bunch compressor, Phys. Rev. ST Accel. Beams 5, 064401 (2002) and its erratum paper.

[2] Z. Huang and K.-J. Kim, Formulas for coherent synchrotron radiation microbunching in a bunch compressor chicane, Phys. Rev. ST Accel. Beams 5, 074401 (2002) and its erratum paper.

[3] M. Borland, `elegant`: A Flexible SDDS-Compliant Code for Accelerator Simulation, Advanced Photon Source LS-287, September 2000.

[4] C.-Y. Tsai, *Investigation of Microbunching Instabilities in Modern Recirculating Accelerators*, Ph.D. dissertation, Virginia Polytechnic Institute and State University, 2017, https://vtechworks.lib.vt.edu/handle/10919/77429.

[5] R. Li and C.-Y. Tsai, Entrance and exit CSR impedance for non-ultrarelativistic beams, IPAC 2017 (WEPIK113).

[6] Y. Li, Studies of the microwave instability in the small isochronous ring, Ph.D. thesis, Michigan State University (2015)

[7] C.-Y. Tsai, Concatenated analyses of phase space microbunching in high brightness electron beam transport, NIMA 940, 464-474 (2019).

[8] C.-Y. Tsai *et al.*, Vlasov analysis of microbunching instability for magnetized beams, PRAB 20, 054401 (2017).

[9] C. -Y. Tsai, *et al.*, Theoretical Investigation of coherent synchrotron radiation induced microbunching instability in transport and recirculation arcs, FEL Conference 2014 (THP022)

[10] C. -Y. Tsai *et al.*, CSR induced microbunching gain estimation including transient effects in transport and recirculation arcs, IPAC'15 (MOPMA025)

[11] C. -Y. Tsai *et al.*, Linear microbunching gain estimation including CSR and LSC impedances in recirculation machines, ERL Workshop 2015 (TUICLH2034)

[12] C. -Y. Tsai *et al.*, Linear Vlasov solver for microbunching gain estimation with inclusion of CSR, LSC and linac geometric impedances, FEL Conference 2015 (MOP052)

[13] D. R. Douglas *et al.*, Control of coherent synchrotron radiation and microbunching effects during transport of high brightness electron beams, arXiv:1403.2318v1 [physics.acc-ph]

[14] D. Douglas *et al.*, Control of synchrotron radiation effects during recirculation, IPAC'15 (TUPMA038)

[15] D. Douglas *et al.*, Control of Coherent Synchrotron Radiation Effects During Recirculation with Bunch Compression, IPAC'15 (TUPMA037)

[16] R. Li and C. -Y. Tsai, Studies of CSR-induced Microbunching Gain for Beam Transport in Project E (unpublished)

[17] M. Borland, Controlling Noise and Choosing Binning Parameters for Reliable CSR and LSC Simulation in elegant, OAG-TN-2005-027.

[18] M. Borland, Modeling of microbunching instability, Phys. Rev. ST Accel Beams 11, 030701 (2008).

[19] C.-Y. Tsai and R. Li, Simulation of coherent synchrotron radiation induced microbunching gain using ELEGANT, JLAB-TN-14-016