PARAXIAL APPROXIMATION IN CSR MODELING USING THE DISCONTINUOUS GALERKIN METHOD*

D. A. Bizzozero†, J. A. Ellison, K. A. Heinemann, S. R. Lau
Department of Mathematics and Statistics, University of New Mexico,
Albuquerque, New Mexico 87131, USA

Abstract

We continue our study [1, 2] of CSR from a bunch moving on an arbitrary curved trajectory. In that study we developed an accurate 2D CSR Vlasov-Maxwell code (VM3@A) and applied it to a four dipole chicane bunch compressor. Our starting point now is the well-established paraxial approximation [3–7] with boundary conditions for a perfectly conducting vacuum chamber with uniform cross-section. This is considerably different from our previous approach [1, 2] where we calculated the fields from an integral over history, using parallel plate boundary conditions. In this study, we present a Discontinuous Galerkin (DG) method for the paraxial approximation equations. Our basic tool is a MATLAB DG code on a GPU using MATLAB's `gpuArray`; the code was developed by one of us (DB). We discuss our results in the context of previous work and outline future applications for DG, including a Vlasov-Maxwell study.

STATEMENT OF THE PROBLEM

Statement of the Mathematical Problem

We study the initial boundary value problems for two nonhomogeneous Schrödinger type equations which arise in a paraxial approximation to Maxwell's equations. The PDEs are

$$\partial_s E_x^r = \frac{i}{2k}\nabla_\perp^2 E_x^r + \frac{ikx}{\rho}E_x^r + \frac{ikx}{\rho}E_x^b(x,y) \qquad (1a)$$

$$\partial_s E_y^r = \frac{i}{2k}\nabla_\perp^2 E_y^r + \frac{ikx}{\rho}E_y^r + \frac{ikx}{\rho}E_y^b(x,y), \qquad (1b)$$

on the domain $0 \le s \le L, -a \le x \le a, -b \le y \le b$. Here the 2D Laplacian is $\nabla_\perp^2 := \partial_x^2 + \partial_y^2$, the real parameters $k \in \mathbb{R}$ and $\rho > 0$, and the nonhomogeneous terms are determined by

$$E_x^b = C\frac{x}{x^2+y^2}, \quad E_y^b = C\frac{y}{x^2+y^2}, \qquad (2)$$

where C is defined in the next subsection. The boundary conditions for $E_x^r = E_x^r(x,y,s;k)$ are

$$\begin{aligned} \partial_x E_x^r &= \partial_y E_y^b, \quad \text{on} \quad x = \pm a \\ E_x^r &= -E_x^b, \quad \text{on} \quad y = \pm b, \end{aligned} \qquad (3)$$

and the boundary conditions on $E_y^r = E_y^r(x,y,s;k)$ are

$$\begin{aligned} E_y^r &= -E_y^b, \quad \text{on} \quad x = \pm a \\ \partial_y E_y^r &= \partial_x E_x^b, \quad \text{on} \quad y = \pm b. \end{aligned} \qquad (4)$$

The initial conditions are given uniquely by

$$\nabla_\perp^2 E_x^r = 0, \quad \nabla_\perp^2 E_y^r = 0, \quad \text{at} \quad s = 0, \qquad (5)$$

with the same boundary conditions, i.e. (3), (4). We note that the above two initial boundary value problems are uncoupled and that the boundary and initial conditions are independent of k.

In addition, the field quantity $E_s^r = E_s^r(x,y,s;k)$, defined by

$$E_s^r = \frac{i}{k}(\partial_x E_x^r + \partial_y E_y^r), \qquad (6)$$

is needed for $0 \le s \le L$ in order to compare with the impedance calculation in [5]. The impedance in our notation is given by

$$Z = -\frac{Z_0}{2\pi C}\int_0^\infty ds E_s(0,0,s;k), \qquad (7)$$

where Z_0 is the free space impedance, C is the parameter in (2), and the calculation of $E_s(0,0,s;k)$ for $s \ge L$ is discussed in the numerical implementation section. Note that C simply scales the fields.

Statement of the Physical Problem

Derivations of the paraxial approximation can be found in [3], [6] and [7]. The starting point is Maxwell's equations with a source given by a line charge moving at near the speed of light, on a circular arc of radius ρ and length L, and in a perfectly conducting rectangular vacuum chamber. As in [3], [6] and [7], we take the special case where the line charge is reduced to a single point. Maxwell equations are written in beam frame coordinates (x,y,s) where the arc is in the (x,s) plane, s is the distance along the arc, and (x,y) are perpendicular to the arc. Thus the electric field can be written

$$\boldsymbol{\mathcal{E}}(x,y,s,t) = (\mathcal{E}_x, \mathcal{E}_y, \mathcal{E}_s). \qquad (8)$$

where $(\mathcal{E}_x, \mathcal{E}_y, \mathcal{E}_s)$ are the components of the field along the unit vectors $(\mathbf{e}_x(s), \mathbf{e}_y, \mathbf{e}_s(s))$ along the reference curve

* Work supported by DOE under DE-FG-99ER41104
† dbizzoze@math.unm.edu

ISBN 978-3-95450-126-7

at $x = y = 0$. The field $\boldsymbol{\mathcal{E}}$ is transformed to $\mathbf{E} = \mathbf{E}(x, y, s; k)$, where they are related by a Fourier type transform

$$\boldsymbol{\mathcal{E}}(x, y, s, t) \propto \int_{-\infty}^{\infty} dk \mathbf{E}(x, y, s; k) e^{ik(s-ct)}. \quad (9)$$

Since the particle is moving near the speed of light, $\mathbf{E}$ is expected to be slowly varying in s. Ignoring second derivatives in s and assuming a/ρ is small, (1) and (6) are obtained where

$$\mathbf{E} = (E_x, E_y, E_s), \quad (10)$$

and where

$$E_x = E_x^r + E_x^b, \quad E_y = E_y^r + E_y^b, \quad E_s = E_s^r + E_s^b. \quad (11)$$

$\mathbf{E}_b$ was introduced in [3] to reduce the effect of the singularity in $\mathbf{E}$, and E_s^b is taken as 0.

Note that the electric field $\boldsymbol{\mathcal{E}}$ satisfies –via (3),(4),(6)– the boundary conditions of a perfect conductor at the boundary $\partial\Omega$ of the vacuum chamber $\Omega := [-a, a] \times [-b, b]$, where $\partial\Omega$ consists of those points of Ω for which either $x = \pm a$ or $y = \pm b$. The initial conditions are defined assuming that the fields have reached a steady-state from an infinite straight prior to entering the bend. The C in (2) is a physical parameter containing the bunch charge, see [3, 6, 7].

NUMERICAL IMPLEMENTATION

We use the discontinuous Galerkin (DG) method, a high-order method which shares features with both finite element and finite volume methods. It has seen rapid development with a myriad of applications over the past 15 years, however, we are not aware of its use in the beam physics community. We therefore give a short overview of our approach, closely following the treatment of the heat equation in [8]. DG formulations typically involve decomposition of the computational domain Ω into triangular elements (in 2D), local representations of the relevant operators and solution on each element, and coupling of adjacent elements through flux terms along common boundaries. The element-wise local solutions are represented by polynomials, but these local solutions may be discontinuous across elements. See Refs. [8–11] for more thorough and general treatments of DG formulations.

Overview of our DG Approach

To make the discussion of the DG approach simpler, we introduce dimensionless variables through the rescalings

$$x \to a\xi, \quad y \to a\eta, \quad s \to 2ka^2\zeta, \quad E_x^r \to Cu/a,$$

but in this subsection we will cavalierly write (s, x, y) for the dimensionless variables (ζ, ξ, η). In terms of these variables (1a) and (3) respectively become

$$-i\partial_s u = \widetilde{\nabla}_\perp^2 u + \frac{2k^2a^3}{\rho}\Big(xu + \frac{x^2}{x^2+y^2}\Big), \quad (12)$$

and

$$\partial_x u = \partial_y \frac{y}{x^2+y^2} \quad \text{on} \quad x = \pm 1$$
$$u = -\frac{x}{x^2+y^2} \quad \text{on} \quad y = \pm\frac{b}{a}. \quad (13)$$

Likewise, the initial condition (5) becomes

$$\widetilde{\nabla}_\perp^2 u|_{s=0} = 0, \quad (14)$$

with the same boundary conditions.

Similarly, with the rescaling $E_y^r \to Cu/a$, (1b) and (4) become

$$-i\partial_s u = \widetilde{\nabla}_\perp^2 u + \frac{2k^2a^3}{\rho}\Big(xu + \frac{xy}{x^2+y^2}\Big), \quad (15)$$

and

$$u = -\frac{y}{x^2+y^2} \quad \text{on} \quad x = \pm 1$$
$$\partial_y u = \partial_x \frac{x}{x^2+y^2} \quad \text{on} \quad y = \pm\frac{b}{a}. \quad (16)$$

The initial condition does not change, and we continue to write (s, x, y) in place of (ζ, ξ, η). We note that u only depends parametrically on $2k^2a^3/\rho$ and b/a, and that the integration domain $0 \le s \le L$ becomes $0 \le s \le L/2ka^2$. The parameter C only enters into the magnitude of the fields $\mathbf{E}$.

Both (12) and (15) can be written as a system,

$$-i\partial_s u = \partial_x q_x + \partial_y q_y + F, \; q_x = \partial_x u, \; q_y = \partial_y u, \quad (17)$$

where F represents either of the last terms in (12,15). We partition Ω into triangular elements, focus on a single element $D \subset \Omega$, and assume that on D the local solution $u \in P^N(D)$ is polynomial of degree N. Multiplication of each equation (17) by its own test polynomial $v \in P^N(D)$ and subsequent integration over D yields

$$-i\int_D dA(v\partial_s u) = \int_D dA(v\partial_x q_x + v\partial_y q_y + vF)$$
$$\int_D dA(vq_{x,y}) = \int_D dA(v\partial_{x,y} u), \quad (18)$$

where a subscript x, y indicates two equations, one for x and one for y. The integration formulas in (18) are exact, but involve only the local polynomials u, q_x, q_y, and v on D. To couple adjacent elements, we now replace the above equations by

$$-i\int_D dA(v\partial_s u) = \int_D dA(v\partial_x q_x + v\partial_y q_y + vF) - \int_{\partial D} dLv\big[n_x(q_x - q_x^*) + n_y(q_y - q_y^*)\big]$$
$$\int_D dA(vq_{x,y}) = \int_D dA(v\partial_{x,y} u) - \int_{\partial D} dLn_{x,y} v(u - u^*). \quad (19)$$

ISBN 978-3-95450-126-7

In these formulae dL and (n_x, n_y) respectively specify the arc-length measure and normal vector for ∂D. Moreover, the terms u^*, q_x^*, q_y^* are numerical expressions which depend not only on the local solutions u, q_x, q_y on D but also on the solutions belonging to adjacent elements. We define these expressions below. One way to "derive" (19) from (18) is to first integrate by parts, shifting all derivatives onto each test polynomial v. This process generates boundary integrals. Next, in these boundary integrals one makes the intermediate replacements $u, q_x, q_y \to u^*, q_x^*, q_y^*$, and then invokes a second round of integration by parts to arrive at (19). Of course, without the intermediate replacements, this process would arrive back at (18), as is easily seen by taking $(u^*, q_x^*, q_y^*) = (u, q_x, q_y)$ in (19).

To obtain matrix formulas from (19), we first express the local solution and each test polynomial as follows:

$$u(x,y) = \sum_{j=1}^{N_p} u_j \ell_j(x,y), \quad v(x,y) = \ell_i(x,y), \tag{20}$$

with the expansions for q_x and q_y similar to the one for u. Here, N_p denotes the number of nodes on the element D, which is related to the polynomial order N through $N_p = (N+1)(N+2)/2$. Moreover, $\ell_i(x,y)$ is the Lagrange basis polynomial which is 1 at the ith node $(x_i, y_i) \in D$, but zero at all other nodes (x_j, y_j), $j \neq i$. Since $\ell_i \in P^N(D)$, our choice for the test polynomial v is permissible, and by taking $i \in \{1, \ldots, N_p\}$ as arbitrary we sample the whole test space. With expansion coefficients as in (20), we define a corresponding vector, for example,

$$\mathbf{u} = (u_1, u_2, \ldots, u_{N_p})^T. \tag{21}$$

Substitution of (20) and the similar expansions for q_x and q_y into (19) yields

$$\begin{aligned} -i\frac{d\mathbf{u}}{ds} &= \mathcal{M}^{-1}\mathcal{S}_x\mathbf{q}_x + \mathcal{M}^{-1}\mathcal{S}_y\mathbf{q}_y + \mathbf{F} \\ &\quad - \mathcal{M}^{-1}\int_{\partial D} dL n_x(\mathbf{q}_x - \mathbf{q}_x^*)\boldsymbol{\ell} \\ &\quad - \mathcal{M}^{-1}\int_{\partial D} dL n_y(\mathbf{q}_y - \mathbf{q}_y^*)\boldsymbol{\ell} \end{aligned} \tag{22a}$$

$$\mathbf{q}_x = \mathcal{M}^{-1}\mathcal{S}_x\mathbf{u} - \mathcal{M}^{-1}\int_{\partial D} dL n_x(\mathbf{u} - \mathbf{u}^*)\boldsymbol{\ell} \tag{22b}$$

$$\mathbf{q}_y = \mathcal{M}^{-1}\mathcal{S}_y\mathbf{u} - \mathcal{M}^{-1}\int_{\partial D} dL n_x(\mathbf{u} - \mathbf{u}^*)\boldsymbol{\ell}. \tag{22c}$$

In these expressions the mass $\mathcal{M}$ and stiffness $\mathcal{S}_x$, $\mathcal{S}_y$ matrices are defined as

$$\mathcal{M}_{ij} = \int_D dA \ell_i \ell_j, \quad (\mathcal{S}_{x,y})_{ij} = \int_D dA(\partial_{x,y}\ell_i)\ell_j. \tag{23}$$

Moreover, $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_{N_p})^T$. To reach the given form (22a,b,c) of the local semi-discrete equations, we have inverted the local mass matrix, and to put these equation into their final form we now define u^*, q_x^*, q_y^* as follows:

$$q_{x,y}^* = \{\{q_{x,y}\}\} - \tau[\![u]\!]_{x,y}, \quad u^* = \{\{u\}\}.$$

The $\{\{\cdot\}\}$ and $[\![\cdot]\!]$ operations respectively denote the average value and jump in a value across a boundary. For example, if two elements D^+ and D^- share a common boundary segment $\partial D^{\pm}$, then along the segment $q_x^* = \frac{1}{2}(q_x^+ + q_x^-) - \tau(n_x^- u^- + n_x^+ u^+)$. For the penalty parameter τ we use the expression described on p263 of [8]. Every element follows the same construction yielding a total $N_p \times K$ nodes for Ω, where K is the total number of elements.

Numerical Computation of Fields and Impedances

This section presents the steps used to obtain the electric fields and longitudinal impedances for the curved and straight portions of the vacuum chamber. This 4 step process is repeated for every wave number k. We now return to the notation in the section on the statement of the problem, since our actual code employs the physical variables.

Construction of the Elements and Matrices This step begins by dividing the rectangular cross-section Ω of the vacuum chamber into rectangles and then subdividing each rectangle diagonally into two triangles. The number of elements is $K = 2N_x^{\text{res}}N_y^{\text{res}}$, with N_x^{res} and N_y^{res} denoting the resolution of elements in the x and y directions. Figure 1 shows an example configuration of nodes and elements with $N_x^{\text{res}} = 6$, $N_y^{\text{res}} = 2$, and $N = 4$. The mass and

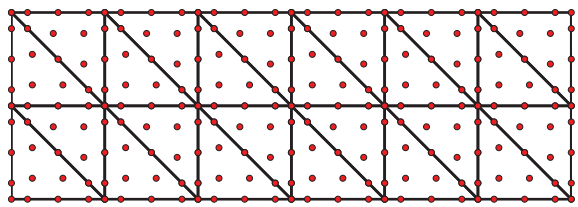

Figure 1: Example of a 24 element domain with 15 nodes per element. The nodes (red dots) are not equally spaced.

stiffness matrices are computed via (23). These matrices are then used to obtain the collocation derivative matrices $\mathcal{D}_x = \mathcal{M}^{-1}\mathcal{S}_x$, $\mathcal{D}_y = \mathcal{M}^{-1}\mathcal{S}_y$ appearing in (22).

Construction of the Initial Data The initial conditions for the transverse fields $E_{x,y}^r$ are solved numerically with the DG Poisson solver described on p275-280 of [8]. Next, E_s^r is computed with the derivative matrices as [cf. Eq. (6)]

$$E_s^r = \frac{i}{k}(\mathcal{D}_x E_x^r + \mathcal{D}_y E_y^r). \tag{24}$$

Evolution of the Fields The transverse fields $E_{x,y}^r$ are evolved using the classical 4th order explicit Runge-Kutta scheme. The time step for the evolution is first determined by

$$\Delta s = C_{\text{CFL}} \cdot k \cdot r_{\min}^2, \tag{25}$$

ISBN 978-3-95450-126-7

where $r_{\min}$ is minimal distance between the nodes. Note that $r_{\min}$ decreases quadratically with the order N; therefore, for large N this time step restriction is severe. We have typically taken the CFL constant C_{CFL} as 0.25 for our computations.

The evaluations of $d\mathbf{u}/ds$ required by the Runge-Kutta scheme are implemented as follows: $\mathbf{q}_x$ and $\mathbf{q}_y$ are obtained from (22b,c), with the results then substituted into (22a). Finally, E_s^r is computed from $E_{x,y}^r$ at each timestep with (24).

Computation of the Impedance We separate the impedance integral in (7) into two parts, $Z = Z_b + Z_s$, where

$$\begin{aligned} Z_b &= -\frac{Z_0}{2\pi C}\int_0^L ds E_s(0,0,s;k) \\ Z_s &= -\frac{Z_0}{2\pi C}\int_L^\infty ds E_s(0,0,s;k). \end{aligned} \tag{26}$$

Throughout the evolution we record E_s^r at the origin. Z_b is then approximated by the trapezoidal rule

$$\begin{aligned} Z_b \approx -\frac{Z_0}{2\pi C}\Delta s\Big[&\frac{1}{2}E_s(0,0,0;k) + \frac{1}{2}E_s(0,0,L;k) \\ &+ \sum_{n=1}^{N_{\mathrm{steps}}-1} E_s(0,0,n\Delta s;k)\Big]. \end{aligned} \tag{27}$$

Following [7], we evaluate the remaining integral Z_s as a mode expansion

$$Z_s \approx \frac{Z_0}{2\pi C}\sum_{m=1}^{M}\sum_{p=1}^{P} D_{mp}\sin\left(\frac{m\pi}{2}\right)\sin\left(\frac{p\pi}{2}\right), \tag{28}$$

where the coefficients D_{mp} are determined by the transverse fields $E_{x,y}^r$ at the end of the bend through the following expressions:

$$\begin{aligned} D_{mp} &= \frac{8(A_{mp}k_x + B_{mp}k_y)}{k_x^2 + k_y^2} \\ A_{mp} &= \frac{1}{ab}\iint_\Omega dA E_x^r(x,y,L)\cos(k_x x)\sin(k_y y) \\ B_{mp} &= \frac{1}{ab}\iint_\Omega dA E_y^r(x,y,L)\sin(k_x x)\cos(k_y y) \\ k_x &= \frac{m\pi}{2a}, \quad k_y = \frac{p\pi}{2b}. \end{aligned} \tag{29}$$

The values of M and P should ideally be infinite; however, in practice, $M \gtrsim (N+1)N_x^{\mathrm{res}}$ and $P \gtrsim (N+1)N_y^{\mathrm{res}}$ are sufficient for a good approximation.

NUMERICAL RESULTS

DG Results

Although comprehensive strategies [12, 13] exist for optimized DG simulations on GPUs, we have adopted a simple approach based on MATLAB's `gpuArray`. Our simulations have been performed on an NVIDIA GTX Titan with the following parameters: $a = 60$ mm, $b = 20$ mm, $L = 200$ mm, $\rho = 1$ m, and $k = 8$ mm^{-1}. We have used K elements with $N_p = (N+1)(N+2)/2$ nodes per element, and the internal penalty parameter τ mentioned earlier. We consider results for both (i) impedance calculations and (ii) performance and accuracy.

The initial condition is shown in Fig. 2 and is independent of k. Figure 3 shows an example of the real and imaginary parts of E_x^r, for $k = 8$ mm^{-1}, at $s = 200$ mm, i.e. at the end of the bend.

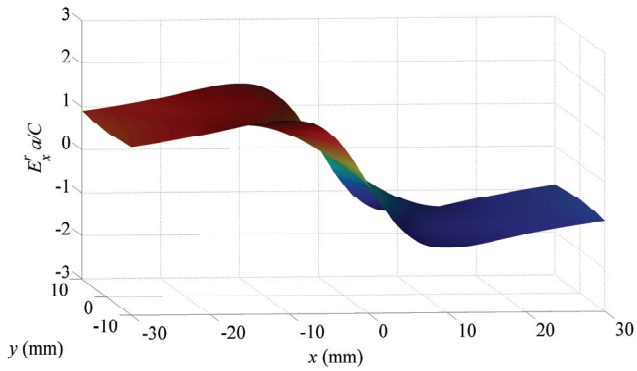

Figure 2: Initial condition for E_x^r.

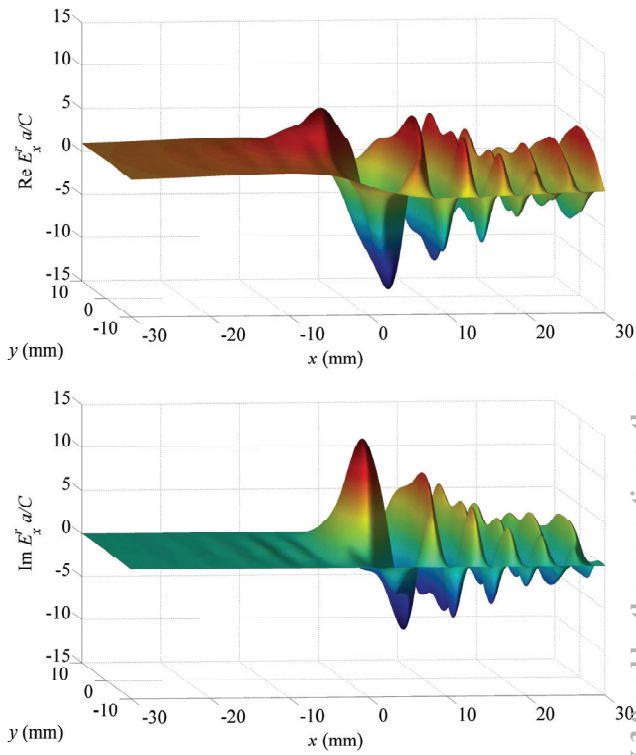

Figure 3: Real (top) and Imaginary (bottom) parts of E_x^r for $\rho = 1$ m, $L = 200$ mm, $a = 30$ mm, $b = 10$ mm, $k = 8$ mm^{-1}.

We have performed a high resolution simulation with $(N, K) = (8, 2400)$, and taken the resulting numerical so-

ISBN 978-3-95450-126-7

lution as the "exact" solution. This simulation took about one hour. Errors for lower-resolution solutions have been computed against this "exact" solution, with the results for E_x^r at $s = 200$ mm and $k = 8$ mm^{-1} shown in Table 1.

Table 1: DG E_x^r Error and Computation Time

N\K	150	600	1350	2400
2	8.107e-1	3.915e-1	7.471e-2	2.453e-2
	8.032e-1	2.528e-1	4.291e-2	1.484e-2
	9s	28s	49s	81s
	122	486	1093	1943
4	1.265e-1	4.897e-3	9.344e-4	3.590e-4
	8.017e-2	3.427e-3	9.462e-4	4.342e-4
	29s	88s	202s	392s
	539	2156	4850	8622
6	1.122e-2	5.177e-4	1.407e-4	5.483e-5
	6.974e-3	6.187e-4	1.932e-4	8.045e-5
	83s	283s	677s	1319s
	1691	6764	15218	27054
8	1.569e-3	1.672e-4	5.612e-5	N\A*
	1.493e-3	2.098e-4	7.473e-5	N\A*
	208s	723s	1867s	3630s
	4174	16693	37559	66771

*:Used for comparison to other tests.

The table shows relative L^∞ (top) and relative L^2 (upper middle) errors corresponding to E_x^r evaluated on a 31×11 grid. This evaluation grid was the largest set of nodes common to all DG grids. Computation times (lower middle) and time-step counts (bottom) are also listed.

The large errors for $(N, K) = (2, 150)$, $(2, 600)$, and $(4, 150)$ are likely due to the oscillating fields not being spatially resolved. We are checking this. The errors decrease and the time-step counts increase with increasing K and N. The table shows that the stepsize $\Delta s \propto 1/(KN^2)$ as expected from (25).

Our MATLAB DG code was first written for a CPU; the GPU version required little additional work. Our GPU calculations become more efficient for larger matrix systems, and when less communication between the CPU and GPU is required. For the lower-order tests, the GPU functioned at around 20% of its maximum capacity. However, for the higher-order tests, the GPU efficiency increased to over 60%. In our GPU simulations we have observed speed-ups of up to ~ 10 over our CPU simulations.

FD Results and Impedance Comparison

We have also written a MATLAB finite difference (FD) code modeled after the FD method discussed in [3, 5–7]. This FD method uses leap-frog as the time-stepper. Our DG code allows for any spatial order N, whereas the FD code employs a second-order stencil. For future work, a comparison between a high-order FD method and the current DG approach might be of interest, since the time-step restriction would be less severe for the high-order FD method.

The FD grids were of size $(N_x^{\text{res}} + 1) \times (N_y^{\text{res}} + 1)$. As a test of the two codes, we calculated the impedance from (7) and (26)-(29). For both the FD and DG approaches, Fig. 4 depicts the resulting real and imaginary parts. To

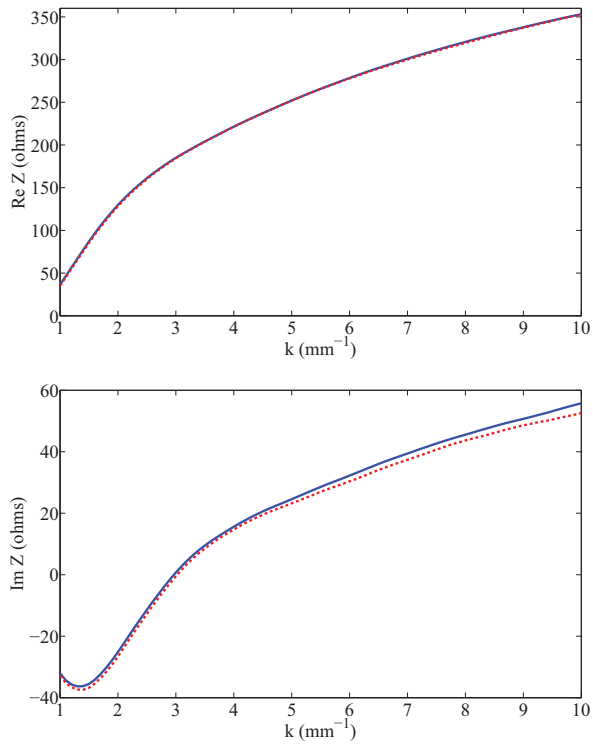

Figure 4: Real (top) and imaginary (bottom) parts of the longitudinal impedance for $\rho = 1$ m, $L = 200$ mm, $a = 30$ mm, $b = 10$ mm. DG (blue solid), FD (red dashed)

the eye these impedances are in agreement with Fig. 3 of [5]. The discrepancy between our DG and FD results is being investigated. The difference likely stems from how the spatial derivatives in (6) are computed. The FD method uses lower order difference stencils instead of collocation derivative matrices.

We have run the FD code for $(N_x^{\text{res}}, N_y^{\text{res}}) = (60, 20), (120, 40), (180, 60), (240, 80)$, and the results are shown in Table 2. The errors were calculated with respect to the high resolution DG calculation, using the 61×21 grid which is common to all the cases. The organization of the table is as in Table 1, with listings for the relative L^∞ and L^2 error, run time, and time-step count. Clearly, our DG GPU code outperforms our FD CPU code.

Table 2: FD E_x^r Error

Grid	61×21	121×41	181×61	241×81
	3.845e-1	1.126e-1	4.016e-2	3.440e-2
	4.539e-1	1.223e-1	4.551e-2	2.840e-2
	8s	125s	642s	2032s
	200	800	1800	3200

ISBN 978-3-95450-126-7

SUMMARY AND FUTURE WORK

We have developed a DG algorithm and GPU code for a CSR study using the paraxial approximation. As the DG method may be new to many readers we have given an overview of the method. Our DG GPU code was written in MATLAB (as was the described FD code) by one of us (DB), and it can be made available to interested readers subject to the copyright conditions in [8]. We have done an error analysis on both codes by comparing results with a high resolution DG calculation. Our results for the impedance compare well with Fig. 3 in [5].

We consider this work as a first step step toward our more ambitious goal of implementing a DG approach to Vlasov-Maxwell systems, e.g. as a possible alternative to our work in [1, 2].

Our future plans are as follows.

1. Examine more fully all sources of numerical error in our simulations. In addition, find optimal (N, K) for a given error requirement. Experience suggests that high order works best.

2. Extend the results to arbitrary arcs and straights, e.g. a 4 dipole chicane.

3. Design a high-order FD code, and compare it with our DG code. The FD approach would seem better suited to the simple geometry of the vacuum chamber considered here. However, the DG approach can handle more complicated geometries, and therefore might prove particularly useful in studies of tapered or corrugated vacuum chambers.

4. Perform a singular perturbation analysis. Since the coefficient $2k^2a^3/\rho$ in (12) is large, such an analysis might offer insights into the paraxial solutions. For example, there does appear to be a boundary layer effect in Fig. 3 which might be amenable to such an analysis.

5. Use a DG algorithm for the 3D Maxwell equations to explore the validity of the paraxial approximation in the context of the work reported here.

6. Finally, as a long-term goal, develop a DG approach to the Vlasov-Maxwell equations and compare with our work in [1, 2].

ACKNOWLEDGMENTS

Our MATLAB DG code was built upon the generic 2D DG codes written by Hesthaven and Warburton (see nudg.org). We thank T. Agoh and D. Zhou for sharing their work, D. Brewer for help with our GPU system, and D. Appelo for comments and suggestions. This work was supported by DOE under DE-FG-99ER41104. DB's work with the DG method in general has also been partially supported by NSF grant No. PHY 0855678.

REFERENCES

[1] G. Bassi, J. A. Ellison, K. Heinemann, R. Warnock, "Microbunching Instability in a Chicane: Two-Dimensional Mean Field Treatment", Phys. Rev. ST Accel. Beams **12**, 080704 (2009).

[2] K. Heinemann, D. Bizzozero, J. A. Ellison, S. R. Lau, G. Bassi, "Rapid integration over history in self-consistent 2D CSR modeling", Proceedings of ICAP2012, Rostock-Warnemunde, Germany, August 2012. See jacow.org for the paper and slides at TUSDC2, i.e., `http://accelconf.web.cern.ch/AccelConf/ICAP2012/papers/tusdc2.pdf`

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

[4] G.V. Stupakov and I.A. Kotelnikov, "Calculation of coherent synchrotron radiation impedance using the mode expansion method", Phys. Rev. ST Accel. Beams **12**, 104401 (2009).

[5] D. Zhou, K. Ohmi, K. Oide, L. Zang, and G. Stupakov, "Calculation of Coherent Synchrotron Radiation Impedance for a Beam Moving in a Curved Trajectory", Jpn. J. Appl. Phys. 51, 016401 (2012).

[6] T. Agoh, *Dynamics of Coherent Synchrotron Radiation by Paraxial Approximation*, Ph.D. Dissertation, University of Tokyo, December (2004).

[7] D. Zhou, *Coherent Synchrotron Radiation and Microwave Instability in Electron Storage Rings*, Ph.D. Dissertation, The Graduate University for Advanced Studies, September (2011).

[8] J. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods* (New York: Springer, 2008).

[9] B. Cockburn, "Discontinuous Galerkin Methods", ZAMM, Journal of Applied Mathematics and Mechanics, Volume 83, Issue 11, p. 731-754, November (2003)

[10] B. Rivière, *Discontinuous Galerkin Methods for Solving Elliptic and Parabolc Equations* (Philadelphia: SIAM, 2008).

[11] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method* (Mineola N.Y.: Dover, 2009).

[12] A. Klöckner, T. Warburton, J. Bridge, J. S. Hesthaven, "Nodal discontinuous Galerkin methods on graphics processors", J. Comput. Phys. 228, issue 21, 7863-7882 (2009).

[13] A. Klöckner, T. Warburton, J. S. Hesthaven, "High-Order Discontinuous Galerkin Methods by GPU Metaprogramming" in *GPU Solutions to Multi-scale Problems in Science and Engineering*, edited by D. A. Yuen, L. Wang, X. Chi, L. Johnsson, W. Ge, and Y. Shi (Springer, 2013). Also available as `arXiv:1211.0582 [cs.MS]`.

ISBN 978-3-95450-126-7