I. INTRODUCTION

The problem of evaluating the radiation spectrum from accelerating charges is relevant in various contexts, and has in many cases been solved analytically [1]. However valuating radiation from arbitrary motion is not always possible by analytical solutions. In this paper we describe a numerical code designed to evaluate both classical and semi-classical radiation spectra from classical particle trajectories. This problem is numerically very difficult since evaluating the radiation spectra involve integration of rapidly oscillating functions. Our code utilizes the enormous parallelism available on GPUs. Thus evaluating radiation from ultra-relativistic particles is now possible with the enhanced double precision capabilities in modern GPUs such as the Nvidia GP100 [2], GV100 [3] and the Titan V (which is equipped with the same chip as the GV100 [4]). The code is written in C++/CUDA [5], where the serial task of solving the equation of motion of single particles is done on a CPU, while the radiation integrals are solved on the GPU.

Other codes for the purpose of evaluating radiation from relativistic particles have been developed, see e.g. [6], an implementation in the existing Meso Bio Nano explorer software (MBN Explorer). This software is very capable, but for larger calculations it requires large CPU clusters. For many research-groups, gaining access to large CPU clusters is not possible, and we show in this paper that by using GPUs, these very demanding numerical tasks can be performed on personal workstations.

This paper is organized as follows: First we describe the theory which is implemented in the code, then the implementation of the theory described and then we describe several tests against theory and experiment to validate the different implementations. Last we compare computation times of the radiation integrals using different CPUs and GPUs.

II. RADIATION INTEGRALS

From the Lienard-Wiechert fields one obtains the classical radiation spectrum, which is evaluated by the integral [1]

$$\frac{d^2 I}{d\omega d\Omega} = \frac{e^2}{4\pi^2} \left| \int_{-\infty}^{\infty} f(t, \mathbf{n}) e^{i\omega(t-\mathbf{n} \cdot \mathbf{x}(t))} dt \right|^2,$$

$$f(t, \mathbf{n}) = \frac{\mathbf{n} \times (\mathbf{n} - \mathbf{\beta}) \times \dot{\mathbf{\beta}}}{(1 - \mathbf{\beta} \cdot \mathbf{n})^2},$$

where $e$ is the electric charge, $c$ is the speed of light, $\omega$ is the frequency of the emitted radiation, $\mathbf{n} = (\sin \vartheta \cos \varphi, \sin \vartheta \sin \varphi, \cos \vartheta)$ is the direction of emission with polar and azimuthal angles $\vartheta$ and $\varphi$ defined relative to an observer, $d\Omega = \sin \vartheta d\vartheta d\varphi$ and $\mathbf{\beta} = \mathbf{v}/c$ and $\mathbf{\beta} = \dot{\mathbf{v}}/c$ are the position, velocity and acceleration of the charge at time $t$. Since radiation emission from relativistic particles often is of interest, we are dealing with small differences between large numbers. To increase the numerical precision, we do a series expansion of these large quantities, as is done in [7], keeping only the leading order terms of the changes to these large numbers. Assuming the particle travels in the $z$-direction, one can solve the equation of motion for the values $\delta z(t) = z(t) - \beta_0 t \delta v_z(t) = v_z(t) - \beta_0$ where $\beta_0 \approx 1 - \frac{\gamma^2}{2}$. Since almost all radiation is emitted in the forward direction we also have that $n_z = \sqrt{1 - n_x^2 - n_y^2} \approx 1 - \frac{n_x^2}{2}$, where $\gamma^2 = n_x^2 + n_y^2$. Substituting the above variables in eq. (1) and eq. (2) we
get [7]

\[ f_x = \left\{ (\dot{v}_x + n_y \dot{v}_y)(n_x - v_x) \right\}, \]

\[ f_y = \left\{ (\dot{v}_y + n_x \dot{v}_x)(n_y - v_y) \right\}, \]

\[ f_z = \left\{ \left[ -\theta^2 - \frac{1}{2} \right] (n_x \dot{v}_x + n_y \dot{v}_y) \right\}, \]

where

\[ g = \left( \frac{1}{2 \gamma^2} + \frac{\theta^2}{2} - \delta v_z - n_x v_x - n_y v_y \right)^{-2}, \]

and the exponential phase becomes

\[ \omega(t - n \cdot x) = \omega \left( \left( \frac{1}{2 \gamma^2} + \frac{\theta^2}{2} \right) t - \delta z - n_x x - n_y y \right), \]  

This expansion is used in both the classical radiation integral and the following two semi-classical integrals.

In many cases the largest quantum effect to the radiation process is the photon recoil. It is a kinematical effect that can be taken into account by a simple substitution of the frequency variable in the classical photon number spectrum regardless of the details of the motion [8]. By substituting the frequency \( \omega \rightarrow \omega^* = \omega/(1 - \hbar \omega/E) \) in the classical number spectrum one gets exactly the quantum number spectrum

\[ \frac{dN_c}{d\omega^*} = \frac{dN_i}{d\omega}, \]  

which means that the quantum intensity spectrum becomes

\[ \frac{dI_i}{d\omega} = \frac{\omega}{\omega^*} \frac{dI_c}{d\omega^*}. \]  

When quantum effects related to the spin of the particles can be neglected, which is generally the case for electron/positron energies below approximately 2 TeV/Z, where Z is the charge number of the crystal, employing the substitution of the frequency variable in eq. (1) according to eq. (5) will reproduce the full quantum spectrum. We call this model the substitution model.

To include effects of both the quantum recoil and of the particle spin in the radiation process we apply a result obtained on the basis of the semi-classical method by Baier and Katkov [9], in which the particle motion is still treated classically whereas the interaction with the radiation field is quantal. Belkacem, Cuc, and Kimball give the semi-classical spectrum for a general trajectory as (first seen in [10] and derivations shown in [7, 11]):

\[ \frac{dI}{d\omega} = \epsilon^2 \frac{E^*}{4\pi^2} \frac{E^2}{2E^2} \left| I^2 + \frac{1}{2E^2 \gamma^2} |J|^2 \right|, \]

where \( E^* = E - \hbar \omega \) and I and J are given by

\[ I = \int_{-\infty}^{\infty} \frac{n \times [(n - \beta) \times \beta]}{(1 - n \cdot \beta)^2} e^{i\omega^*(t-n \cdot r)} dt, \]  

\[ J = \int_{-\infty}^{\infty} \frac{n \cdot \beta}{(1 - n \cdot \beta)^2} e^{i\omega^*(t-n \cdot r)} dt. \]  

We call this the BCK model.

The substitution model and the BCK model are both derived in a first order approximation in the interaction with the radiation field, which means that they apply for single photon emission and only for pure Lorentz force trajectories. Due to the energy dependent phase factor, the radiation spectrum from a particle following a long trajectory, in which it loses energy, will be incorrect. If the particle loses energy along the trajectory, the substituted frequency variable becomes time dependent and the result depends on the initial phase, which is non-physical.

To account for this we divide the trajectory into smaller sections in which the energy loss of the particle has no effect on the spectrum. We then fix the energy of the particle in the radiation integral to the initial value it has when entering the given section. The initial conditions of the particle when entering a new section is then the final conditions from the previous section. The full spectrum for an entire trajectory is the sum of the spectra from each section. The section length is chosen by checking that the spectrum from a pure Lorentz force trajectory is the same as a trajectory where the particle loses energy due to radiation emission. For 50 GeV positrons channeled in the (110) plane of Si this length will be 0.1 mm. One also has to ensure that the section length is longer than the formation length [12, 13] \( l_f = 2\gamma^2(E - \omega)/E\omega \), which in the case of a 50 GeV positron emitting a 5 GeV photon is \( l_f = 71 \) nm.

### III. CRYSTAL CHANNELING

Channeling of charged particles in crystals has been studied in great detail [9], and has been shown to significantly affect radiation emission from charged particles [14]. Here we describe the physics of the channeling process which is to be implemented in the C++ code.

The particles move in the electric field of the crystal lattice, described in our model by the thermally averaged Doyle-Turner continuum potential [6, 10]. The potential
pertaining to a single plane and a single axial atomic string, respectively, for a particle with charge $e$ is given by [17]

$$U(x) = 2\sqrt{\pi}e^2 a_0 N d \pi \sum_{i=1}^{4} \frac{a_i}{\sqrt{B_i + \rho^2}} \exp \left( \frac{-x^2}{B_i + \rho^2} \right),$$  \hspace{1cm} (13)

$$U(r) = \frac{2e^2 a_0}{d} \sum_{i=1}^{4} \frac{a_i}{\sqrt{B_i + \rho^2}} \exp \left( \frac{-r^2}{B_i + \rho^2} \right).$$  \hspace{1cm} (14)

Here $Z_1$ is the charge number of the projectile, $\rho$ is the two dimensional thermal vibration amplitude at room temperature, $a_0$ is the Bohr radius, $d_p$ is the distance between two planes, $d$ is the mean distance between atoms in a string, $x$ and $r$ is the distance to a plane or a string respectively and $B_i = b_i/4\pi^2$. The values of $b_i$ and $a_i$ are fitting parameters determined by electron atomic scattering factors. These have first been measured by Doyle and Turner [15] for the potentials in eq. (13) and eq. (14) and later for more elements by Peng [18].

Multiple Coulomb scattering on individual nuclei and electrons in the crystal are accounted for by Monte Carlo based successive elastic changes in direction of motion in each integration step. The mean squared scattering angle per length traveled is the sum of the mean squared scattering angles on electrons and nuclei [19]

$$\frac{\Delta \theta^2}{\Delta z} = \frac{\Delta \theta^2_n}{\Delta z} + \frac{\Delta \theta^2_e}{\Delta z}. \hspace{1cm} (15)$$

Here the mean squared scattering angle due to collision with nuclei is given by [20]

$$\frac{\Delta \theta^2_n}{\Delta z} = \frac{c^2}{E^2 X_0} n(r),$$ \hspace{1cm} (16)

where $X_0$ is the radiation length, $E$ is the energy of the particle, $\epsilon_s = 10.6$ MeV and

$$n_p(r) = N \frac{d_p}{\sqrt{\pi} \rho^2} \exp \left( -\frac{r^2}{\rho^2} \right),$$ \hspace{1cm} (17)

$$n_e(r) = N \frac{1}{\pi \rho^2 N d} \exp \left( -\frac{r^2}{\rho^2} \right),$$ \hspace{1cm} (18)

is the local density of nuclei pertaining to a single plane or string respectively [21]. One sums the density from all nearby planes or strings to get the total local density. Here $N$ is the mean nuclei density in the material and $r$ is the distance to the plane or the string. The mean squared scattering angle due to collision with valence electrons is given by [19]

$$\frac{\Delta \theta^2_{el}}{\Delta z} = \frac{\pi r_0^2}{\gamma^2 \beta^4} \left[ \log \left( \frac{2m_e \epsilon_s^2 \beta^2}{I} \right) - \beta^2 \right] n_{el}(r),$$ \hspace{1cm} (19)

where $r_0$ is the classical electron radius, $m_e$ is the electron rest mass, $I$ is the average ionization energy of the target atoms and

$$n_{el,p}(r) = \frac{d_p}{\pi} \sum_{i=1}^{4} \frac{a_i (X)}{B_i^{(X)} + \rho^2} \exp \left( -\frac{\gamma^2}{B_i^{(X)} + \rho^2} \right) + c^X n_p(r),$$ \hspace{1cm} (20)

$$n_{el,s}(r) = \frac{1}{\pi d} \sum_{i=1}^{4} \frac{a_i (X)}{B_i^{(X)} + \rho^2} \exp \left( -\frac{r^2}{B_i^{(X)} + \rho^2} \right) + c^X n_s(r),$$ \hspace{1cm} (21)

is the local electron density [22] for a plane and a string respectively. Here the values of $a_i^{(X)}$, $b_i^{(X)}$ and $c^{(X)}$ are the X-ray scattering factors also measured by Doyle and Turner [15]. In eq. (19) we have removed a factor of 2 to ensure that the radiation yield from scattering on nuclei and electrons differs by a factor of $1/Z$ [20].

To account for the loss of energy due to radiation emission we include the Landau-Lifshitz (LL) force in the equation of motion of the particles, which gives [23]

$$m \frac{du^\mu}{ds} = eF^{\mu \nu} u_\nu + \frac{2}{3} e^2 \left[ \frac{1}{m} (\partial_\alpha F^{\mu \nu}) u^\alpha u_\nu + \frac{c^2}{m^2} (F^{\mu \nu} u_\nu)(F_{\alpha \lambda} u^\lambda) u^\mu \right].$$ \hspace{1cm} (22)

Here, $m$ and $e$ denote the particle mass and charge respectively, $F^{\mu \nu}$ is the external electromagnetic field tensor, $u^\alpha$ is the particle four-velocity, and $s$ its proper time, in units where $c = 1$. Because the LL equation depends on the electric field, it does not react to the instantaneous "kick" the particle get when it undergoes multiple Coulomb scattering, as it is put in by hand. For a uniform constant density of nuclei and electrons the radiation spectrum evaluated should reproduce the Bethe-Heitler bremsstrahlung spectrum, which means that the energy loss due to bremsstrahlung is not accounted for. We implement this by the scheme developed in [24], where the energy loss of the particle due to one deflection is given by

$$dE_{MS} = \frac{m_e \alpha}{2 \pi} \gamma^3 \theta^2.$$ \hspace{1cm} (23)

Here $\theta$ is the Monte-Carlo sampled deflection angle in every step from multiple Coulomb scattering.

**IV. CODE IMPLEMENTATION**

The entire program can be divided into two parts; calculating the trajectory and calculating the radiation integral, given the trajectory. The two parts work independently and the radiation integrals are evaluated on the GPU while the trajectories are evaluated on the CPU. The reason why the trajectories are not evaluated on the GPU is because the task of solving the equation of motion is a serial task which only has to be done once for
Electric fields with the ODE solver, and afterwards change the trajectory included, like multiple scattering, we first do one step. Since Monte Carlo based processes of the theoretical models used to calculate the fields and particles, we estimate is much lower than the inaccuracy reached in only a few steps during a simulation of 1000 error imposed by setting a fixed lower step size, which is taking a lower bound on the modulated step size in order to avoid step sizes converging to zero due to stiffness of the differential equation only affects the part of the trajectory for each particle and the number of particles usually does not exceed 2000 per simulation. This means that we are not even close to saturating high end GPUs.

On fig. [1] a diagram of the code structure is shown. This outlines the basic steps that are taken in the process of evaluating a radiation spectrum. The trajectories are evaluated with a Runge–Kutta (RK45) Ordinary Differential Equation (ODE) solver which features step size modulation. We have implemented this method including a lower bound on the modulated step size in order to avoid step sizes converging to zero due to stiffness of the differential equation. The stiffness of the differential equation only affects the part of the trajectory for electrons very close to the string in the axial case. The error imposed by setting a fixed lower step size, which is reached in only a few steps during a simulation of 1000 particles, we estimate is much lower than the inaccuracy of the theoretical models used to calculate the fields and multiple scattering. Since Monte Carlo based processes are included, like multiple scattering, we first do one step with the ODE solver, and afterwards change the trajectory by hand, based on the Monte Carlo processes. This step can be modulated in size within the solver but it returns only the trajectory at the point in time specified by the step. In this way we can control the resolution of the trajectory, this is vital when calculating the radiation spectrum, as for high-frequency photon emission the radiation integral oscillates much more rapidly and more points in time are required for the integral to converge.

We have written a library that evaluates the effective electric field and gradient of the field as a function of position within the crystal and which crystal type/orientation we are using. This library features the placement of strings and planes for FCC and BCC crystals in the ⟨111⟩, ⟨110⟩, ⟨100⟩, ⟨111⟩, ⟨110⟩ and ⟨100⟩ orientations. The elements which have been implemented are Silicon, Diamond, Germanium and Tungsten, with easy implementation of new elements which have the same crystal structure.

When the trajectory is evaluated, it is copied onto the GPU memory on which the spectrum pertaining to the trajectory is evaluated and copied back to the CPU. By only copying the trajectory and the spectrum once we limit the amount of data copied between GPU and CPU and therefore reach more than 99% workload on the GPU during the entire simulation.

To produce a spectrum of the type \( dI/d\omega \) we need to evaluate the spatial integral over \( d\Omega \) for every frequency \( \omega \) in the spectrum. We therefore have to evaluate the integral over the same trajectory \( N_x \times N_y \times N_z \) times, where \( N_x \) and \( N_y \) is the number of points in the spatial integral in the \( x \) and \( y \) direction respectively and \( N_z \) is the number of points in the spectrum. As an example, it is not unreasonable to have \( N_x = N_y = 50 \) and \( N_z = 100 \), when multiple scattering is included, which amounts to \( 0.25 \times 10^6 \) integrals over the same trajectory. When multiple scattering is included the spatial features of the spectrum are spread out and only 50 points in both spatial dimensions are needed, but if multiple scattering is not included, and the particle follows a periodic motion, the spatial features become very narrow, and depending on how many periods the integral contains, we could need more than 3000 points in both spatial dimensions. This would amount to \( 900 \times 10^6 \) individual integrals over the same trajectory, which at this point begin to approach the limit of how much memory is accessible on the GPU. The average spectrum of a beam of particles would then be the average spectrum of many particles with different initial conditions entering the crystal.

The above point is also one of the reasons why we evaluate the trajectories on the CPU individually, and only evaluate the spectrum pertaining to one trajectory on the GPU at a time. The GPU is saturated with work by evaluating a spectrum from just one particle, which means that we do not lose much performance by evaluating trajectories on the CPU. To always keep the GPU saturated with work we use openMP to evaluate trajectories in parallel on the CPU, each CPU thread then instructs the GPU to evaluate the spectrum pertaining to
its trajectory. This means that the GPU can be working on the spectrum from one CPU thread, while another CPU thread is evaluating another trajectory. Depending on which energy, crystal orientation and number of integrals per trajectory we need, the time it takes to evaluate a trajectory can either be faster or slower than evaluating a spectrum. So it can in some cases affect the performance a lot if you are balancing between which process is the fastest. Evaluating a trajectory for a planar oriented crystal is generally faster by up to an order of magnitude than evaluating a trajectory in the axial case.

The workload on the GPU is distributed such that each thread on the GPU evaluates one integral and saves the result in a cube (3 dimensional matrix corresponding to the dimensions \( N_x, N_y \) and \( N_z \)). This cube is copied back onto the CPU where we do the final sum over the spatial integrals evaluated by the GPU and produce the final spectrum.

As already mentioned the radiation spectra rely on small differences between large numbers, this means that we need high precision on the data type used, and we are required to use double precision numbers when calculating the trajectory and the radiation spectrum. This is required even though we do a series expansion as explained in section 11.

A version of the code can be found in the repository on [26].

Test of integral implementation

To test the numerical implementation of eq. (1), we evaluated the radiation emitted from an electron traversing a plane strong undulator in the \( z \)-direction, with a magnetic field in the \( y \)-direction:

\[
H_y = B_y \sin \left( \frac{2\pi z(t)}{\lambda} \right), \quad (24)
\]

In the test we used a magnetic field strength \( B_0 = 1 \, \text{T} \), an undulator period length \( \lambda = 53 \, \text{mm} \), an electron at 2 GeV and \( N = 20 \) periods. The resulting radiation is compared to the analytical expression found in [26]. On fig. 3 the radiation evaluated with the CUDA program in the forward direction is shown together with an analytical solution to the problem and their relative error. We clearly see good agreement between the two solutions. In vicinity of the peaks the error drops to around 1-2% and in the areas where there is no peak, we see rather large error oscillations up to 20%, and even a few larger than 100% which are not shown in the figure. These large errors occur in areas where the functions are very small. While the analytical solution might be zero by construct in the valleys, small numerical errors in the program result in large deviations from the analytical solution. These error patterns are therefore expected.

We also evaluated the energy radiated per solid angle \( dI/d\Omega \) in an area with a maximal opening angle of \( 5/\gamma \), numerically integrating eq. (1) with respect to \( \hbar \omega \) from 0-500 eV. This is shown on fig. 3 together with the relative error in percent between the solution found by the CUDA program and the analytical solution. We again see good agreement between the CUDA program and the analytical solution, with relative errors within a few percent, except in a small area. In fig. 3 we only show relative errors up to 14% in order to highlight the smaller errors, but in the center of the area with the large errors lies a few points with up to 200% error which is not shown. These large errors, we believe, are again results of small numerical errors in the CUDA program. The function values we compare in the area are very small, and a small numerical error will result in large differences between the functions.

From this test we conclude that the CUDA program written to evaluate radiation emitted from accelerating charged particles given the particle trajectory works, and could be used to evaluate the radiation emitted from arbitrary motion, e.g. positrons and electrons channeled in crystals. We therefore proceed to testing this as described in the next section.

Test of trajectory implementation

We test the implementation of the continuum potential by evaluating the radiation emitted from a 50 GeV positron traversing a (110) oriented Si crystal, initially moving at an angle of 300 µrad with respect to the crystal planes with an initial position between two planes. These simulations are compared to quantum mechanical calculations using the formulation of coherent bremsstrahlung found in [27, 28]. Here the particle is treated as a plane wave traversing the crystal planes with an angle. This model breaks down for small angles, while the continuum model breaks down for large angles. Even though the models break down in different regimes, the position of the harmonic peaks is correct, as this is a kinematic effect related to the frequency of which the particles traverse a plane. On fig. 4 the spectra using the classical radiation integral and the BCK model is shown together with the coherent bremsstrahlung calculation which we call Maratea, produced by Allan H. Sørensen from Aarhus University. It is clear that the classical model no longer agrees with the quantum calculations, since effects like the photon recoil is neglected. The discrepancy between the classical model and the BCK model starts to show even at 5 GeV photon energies (10% of the incoming particle energy). We see good agreement of the location of the harmonic peaks between the BCK and the Maratea model. It is clear that including multiple Coulomb scattering is necessary to describe the level difference at high photon energy, which relates to incoherent bremsstrahlung.

A final test of the implementation of radiation emission when multiple Coulomb scattering in oriented crystals is included, involve evaluating the radiation spectrum from 6.7 GeV positrons and electrons traversing...
the (110) plane of a 0.105 mm thick Si crystal. On fig. 5 we show a comparison of the theoretical spectrum with experimental data measured by the authors from [29]. By considering the beam divergence and angular resolution in the experimental cuts, the entry angle of the simulated particles is chosen to follow a uniform distribution of angles between ±110 µrad. We see good agreement between simulation and experiment, even when looking at the spectral features of the radiation from the positrons.

V. GPU VS CPU

In this section we compare computation times of the radiation integrals between different GPUs and CPUs. We will be using a variety of GPUs to also illustrate the performance increase between GPUs. The two CPUs used in the test are the Intel 7820x processor, which features 16 parallel cores and the second CPU is the Xeon Gold 6140 processor, featuring 36 parallel cores. The GPUs used in the test are the GeForce 1080ti, Titan V and the Quadro GP100. The code used for the CPU’s are nearly identical to the GPU version, as the data structure and function evaluations are the same. Since the data structure used are only c-style arrays and the fact that we only use low-level math functions like sines and square roots, the GPU code is also optimized well for CPU’s. The only difference is that no data is transferred between GPU and CPU, and that now only one thread is working on the spectrum from a single trajectory if many particles are simulated. In the following tests we only simulate one particle, which means that all threads available on the CPU will be working on the spectrum just like in the GPU case, evaluating one integral each.

The test involves measuring the computation time of a certain amount of radiation integrals of the form in eq. (1). Each integral consist of 3000 points in time, which corresponds to ≈ 0.01 mm thick planar oriented crystal. To do the spatial integration of \(d\Omega\), one has to calculate the integral eq. (1) for a certain number of points in space, which in the case of the spectra in fig. 4 amounts to \(5 \cdot 10^6\) points in space, since the spatial spectral features become very narrow when propagating a particle along the same oscillatory motion for an extended time. Then to produce a spectrum one has to also calculate each integral for each photon frequency in the spectrum, corresponding to \(5 \cdot 10^8\) integrals in total. It is therefore necessary to see how each chip performs as a function of the number of integrals.

On fig. 6 the computation time for each chip is shown. The processors are run in single core mode, doing every integral sequential, and in parallel mode, meaning that their 16 and 36 cores respectively are working in parallel. As expected the CPUs have faster computation time for very low number of threads, we also see that the single core processors are faster than the parallel processor, which is due to the overhead of starting a parallel process. We see a flat trend for each GPU for \(\approx 10^4\) threads, this
7

Figure 3. $\frac{dI}{d\Omega}$ from a 2 GeV electron in an undulator with an undulator period length $\lambda = 53$ mm, $N = 20$ periods and a magnetic field strength $B_0 = 1$ T. The integrated photon energy range is 0-500 eV. Left figure is the solution found by the CUDA program. Right figure is the relative error in percent between the solution found by the CUDA program and the analytical solution found in [26].

Figure 4. Radiation spectrum from a 50 GeV positron traversing the (110) plane of Si at an angle of 300 $\mu$rad. The label MS tells whether multiple Coulomb scattering is included or not. The classical and BCK model are both the result from the CUDA/C++ program, and the curve labeled Maratea is the result of a quantum mechanical calculation done by Allan H. Sørensen at Aarhus University, using the model described in [27, 28].

Figure 5. Radiation spectra from 6.7 GeV positrons and electrons traversing the (110) plane of a 0.105 mm thick Si crystal. Entry angles of the particles in the simulation are uniformly distributed between $\pm 110$ $\mu$rad. The squares and triangles are experimental data points digitized from [29].

is due to the amount of cores available on the GPUs not saturated with working threads. The number of threads needed before each GPU is saturated is different for each GPU and we see that the first GPU to reach saturation in threads is the 1080ti which is expected. What is perhaps not expected is that the most expensive GPU, the Quadro GP100, reaches saturation before the Titan V. After saturation the computation time for the GPUs become linear in number of threads, as the CPUs. From fig. 6 we also see that the number of threads needed for the Xeon CPU and the Titan V GPU to reach identical performance is only 350, after which the Titan V is faster.
Figure 6. Computation time as a function of the number of evaluated radiation integrals (eq. (1)), each integral has 3000 points in time. The 7820x CPU has 16 parallel cores, and the Xeon processor has 36 parallel cores.

Figure 7. Speed up with respect to the 7820x processor running in parallel mode. The speed up value for each chip is listed above their respective bar.

On fig. 7, the speed-up of each chip is shown compared to the 7820x processor in parallel mode. Here each chip is fully saturated by threads. As expected we see a factor of $\approx 2$ between the 36 cores from the Xeon processor and the 16 cores on the 7820x processor, while we reach a speed up of 205.1 for the Titan V.

This shows that if one had a parallel job capable of running effectively on GPUs, a single Titan V could replace a computer cluster running almost 100 nodes, each equipped with a 36 core Xeon processor.

One thing also worth noting is that these integrals require double precision operations. The architecture on the Titan V and the GP100 allows for much faster double precision calculations than earlier models like the 1080ti [30] and even the new Titan RTX which is not built to support effective double precision calculations [4]. Choosing a GPU for a specific job therefore depends on the datatype used in the calculation, which becomes more important when calculations require double precision.

VI. CONCLUDING REMARKS AND FUTURE IMPLEMENTATIONS

The current version of the CUDA code is a program capable of evaluating both classical and quantum mechanical radiation spectra from charged relativistic spin 0 and 1/2 particles following arbitrary classical trajectories. These calculations are done on a single GPU and an even higher speedup could be achieved using a cluster of GPUs. The bottleneck in computation time is calculating the actual trajectory of particles when simulating particles traversing oriented single crystals. The C++ program feature straight axial and planar oriented crystal trajectory simulations for Tungsten, Diamond, Germanium and Silicon. Every particle trajectory is calculated in parallel on the GPU and an implementation where many particle trajectories are evaluated in parallel on the GPU is of interest. Also of interest is simulation of bent crystals, which can be used as crystalline undulators or as a beam extraction device for accelerators like the LHC.

It is clear from our analysis that GPUs can be a very powerful tool when extensive calculations become impossible due to computation time. The evolution of computation performance of GPUs the past decade has pushed the boundaries of what is numerically possible and we believe that GPUs are the future of computation in many areas of science, not only physics.

The numerical results presented in this work were partly obtained at the Centre for Scientific Computing, Aarhus [http://phys.au.dk/forskning/cscac] and by support from Nvidias GPU grant program.

[1] J. D. Jackson, *Classical Electrodynamics*, 3rd ed. (John Wiley and Sons, Inc, 1998).
[2] NVIDIA TESLA P100 GPU ACCELERATOR, Nvidia (2016), data sheet.
[3] NVIDIA TESLA V100 GPU ACCELERATOR, Nvidia (2018), data sheet.
[4] NVIDIA TURING GPU ARCHITECTURE - Graphics Reinvented, Nvidia, wP-09183-001_v01.
[5] CUDA is a C/C++ parallel computing platform designed to implement the use of GPUs for
general purpose programming. This allows the user to launch thousands of parallel processes. https://www.geforce.com/hardware/technology/cuda.

[6] G. B. Sushko, V. G. Bezchastnov, I. A. Solovyov, A. V. Korol, W. Greiner, and A. V. Solovyov, Journal of Computational Physics 252, 404 (2013)

[7] T. N. Wistisen, Phys. Rev. D 90, 125008 (2014)

[8] J. Lindhard, Phys. Rev. A 43, 6032 (1991)

[9] V. N. Baier, V. M. Katkov, and V. M. Strakhovenko, Electromagnetic processes at high energies in oriented single crystals (1998).

[10] A. Belkacem, N. Cue, and J. Kimball, Physics Letters A 111, 86 (1985)

[11] J. Kimball, N. Cue, and A. Belkacem, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 13, 1 (1986)

[12] K. Andersen, S. Andersen, H. Knudsen, R. Mikkelsen, H. Thomsen, U. Uggerhøj, T. Wistisen, J. Esberg, P. Sona, A. Mangiarotti, and T. Ketel, Physics Letters B 732, 309 (2014)

[13] K. K. Andersen, S. L. Andersen, J. Esberg, H. Knudsen, R. Mikkelsen, U. I. Uggerhøj, P. Sona, A. Mangiarotti, T. J. Ketel, and T. Ballestrero (CERN NA63), Phys. Rev. Lett. 108, 071802 (2012)

[14] U. I. Uggerhøj, Rev. Mod. Phys. 77, 1131 (2005)

[15] P. A. Doyle and P. S. Turner, Acta Crystallographica Section A 24, 390 (1968)

[16] J. Andersen, E. Bonderup, E. Laegsgaard, B. Marsh, and A. Sørensen, Nuclear Instruments and Methods in Physics Research 194, 209 (1982)

[17] S. P. Møller, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 361, 403 (1995)

[18] L.-M. Peng, Micron 30, 625 (1999)

[19] A. Babaev and S. Dabagov, The European Physical Journal Plus 127 (2012), 10.1140/epjp/i2012-12062-6

[20] M. Tanabashi et al. (Particle Data Group), Phys. Rev. D 98, 030001 (2018)

[21] M. Kitagawa and Y. Ohtsuki, Phys. Rev. B 8 (1973), 10.1103/PhysRevB.8.3117

[22] T. V. Jensen and A. H. Sørensen, Phys. Rev. A 87, 022902 (2013)

[23] L. Landau and E. Lifshitz, The Classical Theory of Fields, 4th ed. (Pergamon press, 1979).

[24] T. N. Wistisen, Phys. Rev. D 92, 045045 (2015)

[25] Https://gitlab.au.dk/au483748/channelingradiation.git.

[26] A. Hofmann, The Physics of Synchrotron Radiation, 1st ed. (Cambridge university press, 2004) Chap. 8.

[27] G. D. Palazzi, Rev. Mod. Phys. 40, 611 (1968)

[28] M. L. Ter-Mikaelian, High-energy electromagnetic processes in condensed media (Wiley-Interscience, 1972).

[29] J. Bak, J. Ellison, B. Marsh, F. Meyer, O. Pedersen, J. Petersen, E. Uggerhøj, K. Østergaard, S. Møller, A. Sørensen, and M. Suffert, Nuclear Physics B 254, 491 (1985)

[30] White Paper, NVIDIA GeForce GTX 1080, Gaming Perfected, Nvidia.