ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPAK138

DEVELOPMENT OF EFFICIENT TREE-BASED COMPUTATION METHODS FOR THE SIMULATION OF BEAM DYNAMICS IN SPARSELY POPULATED PHASE SPACES

Ph. Amstutz*, Universität Hamburg, Hamburg, Germany
M. Vogt, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

Collective instabilities pose a major threat to the quality of the high brightness electron beams needed for the operation of a free electron laser. Multi-stage bunch compression schemes have been identified as a possible source of such an instability. The dispersive sections in these compressors translate energy inhomogeneities within the bunch into longitudinal charge density inhomogeneities. In conjunction with a collective force driving locally density-dependent energy modulations this leads to intricate longitudinal beam dynamics. As a consequence of the thin shape those bunches form in the longitudinal phase space, efficient simulation of such systems is not straight forward. At high resolutions, the numerical representation of the phase space density on a uniform grid is too wasteful, due to the large unpopulated phase space regions. In this contribution we present advances made in the development of a simulation code that addresses the problem of sparsely populated phase spaces by means of quadtree domain decomposition. A focus lies on the explanation of the underlying tree data structure.

INTRODUCTION

Recently, we have outlined a one-dimensional model, that is able to capture the dynamics of the formation of microbunching [1–4] driven by longitudinal space-charge (LSC) effects in bunch compressors of free-electron laser (FEL) injectors [5–7]. One remarkable feature of this model is its ability to describe the collective Coulomb self-interaction of the electron bunch, as well as the dynamics in all other relevant beam-line elements by strictly time-discrete maps. Following the Vlasov picture, this model allowed us to formulated the evolution of the phase-space density (PSD) of an electron bunch by employing Perron-Frobenius operators $\mathcal{M} \in \mathrm{lin}(\mathcal{L}^1(\mathbb{R}^2,\mathbb{R}), \mathcal{L}^1(\mathbb{R}^2,\mathbb{R}))$ associated to these maps $M : \mathbb{R}^2 \to \mathbb{R}^2$, which relate the current PSD $\Psi(t_n;\cdot)$ to the time-forward PSD $\Psi(t_{n+1};\cdot)$ via

$$\Psi(t_{n+1},\cdot) = \mathcal{M}_n\Psi(t_n,\cdot) \equiv \Psi(t_n,\cdot)\circ M^{-1}, \qquad (1)$$

where $\Psi : [t_0, t_{\max}]\times\mathbb{R}^2 \to \mathbb{R}^+$, which is to be interpreted in the sense that $\int_{A\subseteq\mathbb{R}^2} \Psi(t_n;\vec{z})\mathrm{d}\vec{z}$ gives the probability to find any of the electrons in the phase-space region A.

While this approach absolutely allows for interesting analytical investigations [6, 9], in the general case numerical simulation is still inevitable. Simulating such Vlasov systems is encumbered by the fact that phase-space densities of FEL-type electron bunches typically form a thin band that meanders through phase-space; a feature which we call *exotic*. This makes it impossible to capture the phase-space density in a tight-fitting rectangular simulation region. Consequently, the simulation region will contain vast areas that are actually void of any phase-space density, rendering the approach generally wasteful with respect to computational resources and unfeasible for high resolutions. To overcome this problem we have developed a simulation code, that relies on tree-based domain-decomposition to adapt to the phase-space density in order to minimize computational overhead. While the current working-area of the code is the aforementioned two-dimensional phase-space, we have designed the underlying data-structures and algorithms in a dimension-agnostic fashion to allow for an easier upgrade to higher (d) dimensions in the future. In this contribution we present some of the intricacies of the resulting 2^d-tree data-structure.

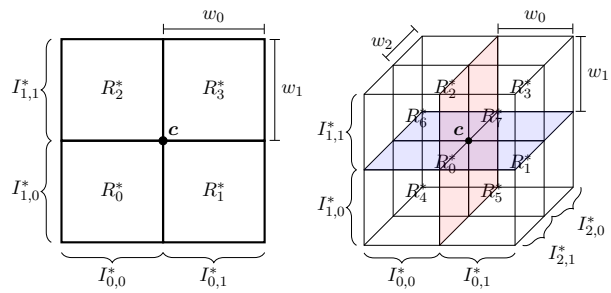

Figure 1: Refinement of two-dimensional (left) and three-dimensional (right) hyperrectangles.

2^d TREE DATA-STRUCTURE

The general idea of tree-based domain-decomposition is to recursively divide a hyperrectangular region into smaller *child*-regions, as illustrated in Figures 1; a process called *refinement*. By refining only those regions that are "interesting" for the problem at hand (in our case phase-space regions that contain significant density), this allows to efficiently describe even objects with an exotic shape. By keeping track of the parent-child relations a tree graph is formed. While this approach is well-known and used in many applications for two- and three-dimensional objects in form of quadtrees and octrees respectively [8], it can be generalized to arbitrary dimension d [9]. Assuming the d-dimensional parent-hyperrectangle covers the region given

* philipp.amstutz@desy.de

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPAK138

by the Cartesian product of d closed intervals

$$R = \bigotimes_{i=0}^{d-1} \underbrace{[u_i, v_i]}_{\equiv I_i} \tag{2}$$

where $\forall i\colon\ i \in \{0, \ldots, d-1\} \implies u_i, v_i \in \mathbb{R} \wedge u_i < v_i$, the first step is to find a suitable order for the 2^d child-hyperrectangles, covering the regions $R^*_0, \ldots, R^*_{2^d-1}$. Firstly we halve the original intervals into lower parts $I^*_{i,0}$ and upper parts $I^*_{i,1}$

$$I^*_{i,0} = \left[u_i, \frac{v_i - u_i}{2}\right], \quad I^*_{i,1} = \left(\frac{v_i - u_i}{2}, v_i\right], \tag{3}$$

so that $I_i = I^*_{i,0} \cup I^*_{i,1}$. The parent-hyperrectangle can then be reconstructed via

$$R = \bigotimes_{i=0}^{d-1} I^*_{i,0} \cup I^*_{i,1} = \bigcup_{\vec{b} \in \{0,1\}^d} \bigotimes_{i=0}^{d-1} I^*_{i,\vec{b}_i} = \bigcup_{n=0}^{2^d-1} \bigotimes_{i=0}^{d-1} I^*_{i,\vec{b}(n)_i}, \tag{4}$$

with an arbitrary, surjective ordering function

$$\vec{b}\colon \{0, \ldots, 2^d - 1\} \to \{0, 1\}^d, \tag{5}$$

which then defines the location of the nth child-hyperrectangle. A natural choice for this ordering function $\vec{b}(n)$ is the *binary decomposition* of n, implicitly defined by

$$\sum_{i=0}^{d-1} 2^i\, \vec{b}(n)_i = n, \tag{6}$$

which indeed is unique [10] and hence is surjective on this domain. It has the striking advantage that most contemporary programming languages represent integer types in this format so that the $\vec{b}(n)_i$ values are accessible by a simple memory look-up. Doing so, the center points $\vec{c}^*$ and widths $\vec{w}^*$ of the child-hyperrectangles are the related to those of their parent-hyperrectangle $\vec{c}, \vec{w}$ via

$$\vec{c}^*_{n,i} = c_i - (-1)^{\vec{b}(n)_i} \frac{\vec{w}_i}{2} \quad \text{and} \quad \vec{w}^*_{n,i} = \frac{\vec{w}_i}{2}, \tag{7}$$

compare also Figure 1. With this, we are now able to handle 2^d-trees in a well-defined manner. In two dimensions this ordering results in parent-child relations as depicted in Figure 2.

As can be seen, such a graph does not contain any direct information regarding neighborhood relations between the hyperrectangles. The length of a path connecting two neighboring hyperrectangles can be as large as $2r$, where r is the recursion depth, and one would have to resort to sophisticated algorithms in order to find the neighbor of a given hyperrectangle in the tree [11]. Because direct access to these neighborhood relations allows for many elegant methods in the context of Perron-Frobenius propagation of phase-space densities, we chose to add them in our implementation of the fundamental tree structure. Again, the first

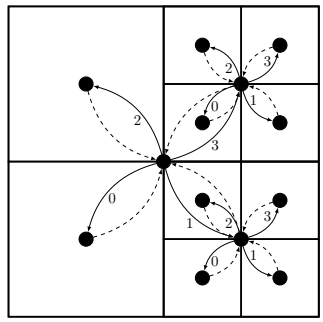

Figure 2: Parent-child relations in a rarely refined quadtree. Numbers at the solid arrows – pointing from a parent to a child – indicate the child-index n. Dashed arrows point from child to parent nodes.

step is to impose an ordering of the neighbors, i.e. define the notion of the "mth neighbor" of a hyperrectangle. Every hyperrectangle potentially has $2d$ neighbors, that is one in both, the positive and negative direction along each dimension, so that we have to relate the neighbor index m to the dimension/axis a and direction p of the neighbor. This can be achieved using the following interpretation m, based again on its binary decomposition

$$m = (\ldots b_3 b_2 b_1 \underbrace{b_0}_{p}\underbrace{\ldots b_3 b_2 b_1}_{a})_2, \tag{8}$$

so that the least significant bit gives the direction, while the others determine the direction. Using the C(-style) bit-wise operators [12] "`&`" and "`»`" these can be elegantly calculated via

$$a = \lfloor m/2 \rfloor = \texttt{m»1} \tag{9}$$

$$p = m \mod 2 = \texttt{m\&1}. \tag{10}$$

We make the observation that any neighbor B of a given hyperrectangle A is is either a *sibling* of A, i.e. they share the same parent hyperrectangle, or B is a child of a neighbor of the parent of A. The former is the case, if B lies in a direction in which A is already among the outermost of its siblings, which is determined by the child-index n of A. Using again bit-wise operators the statement "the mth neighbor of an nth is its sibling" can be checked by evaluating

```
(n»(m»1)) & 1 != m & 1 ,
```
(11)

of which the truth table is shown in Figure 3. If B is not a sibling of A, it can be seen that it is a child of the mth neighbor of A. Generally, the child-index of B in this parent hyperrectangle is determined by the fact A and B share their relative positions within their respective parent hyperrectangles (which might be the same) with respect to all dimension but the `m»1`th, in which it is the exact opposite. Hence, the mth neighbor of an nth child A is an oth child (either of the parent A, or of the mth neighbor of A), where o can be calculated by flipping the `m»1`th bit of n, which can be achieved by employing the bit-wise exclusive-or operation "`^`"

```
o = n ^ (1«(m»1)).
```
(12)

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPAK138

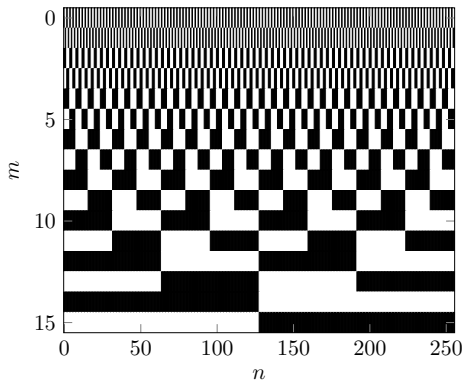

Figure 3: Truth table of the statement "The mth neighbor of an nth child is its sibling.". Cases in which the statement is true are marked in white, while black marks cases where it is false.

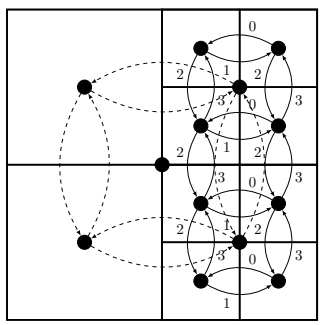

Figure 4: Neighborhood relations in a rarely refined quadtree. Numbers at the solid arrows indicate the neighbor-index m. First-level neighbors are indicated by dashed arrows and are not numbered.

With this, we have completely identified the location of all the neighbors of all hyperrectangles in the tree. Note, however, that this approach relies on the fact that the neighbors of the parent of any hyperrectangle is already known and therefore is not suitable to determine neighbors in a bare, unmodified 2^d-tree. The solution is to make sure that in the implementation, from the very start every time a hyperrectangle is generated while refining the tree, it determines its neighbors and stores references to them. As no hyperrectangle can be generated before its parent has been generated, it is hence ensured that the method above is applicable. The resulting neighborhood graph is exemplified for the two-dimensional case in Figure 4.

EXECUTION STRATEGY OF A PERRON-FROBENIUS STEP

In order to numerically evaluate Equation (1), the very first step is to determine a suitable time-forward simulation window to accommodate $\Psi(t_{n+1};\cdot)$, which would be a tricky task if no geometric information about the geometry of the initial phase-space density $\Psi(t_n;\cdot)$ was available. The presented tree structure, however, provides such information and offers an elegant method to find this minimum bounding-box. If we have a quadtree covering $\mathrm{supp}(\Psi(t_n,\cdot))$ – assuming $\Psi(t_n)$ was cut off at some threshold for the simulation – we can make use of the neighborhood relations to determine the boundary of the phase-space density $\delta\mathrm{supp}(\Psi(t_n,\cdot))$ by finding those hyperrectangles that lack at least one neighbor. Then, using the map M_n, points on this boundary can be tracked forward, resulting in a very good approximation of the boundary of the time-forward phase-space density

$$\delta\mathrm{supp}(\Psi(t_{n+1},\cdot)) = M(\delta\mathrm{supp}(\Psi(t_n,\cdot))), \qquad (13)$$

based on which a tight-fitting root hyperrectangle of the time-forward tree can be efficiently found, by determining the minimum bounding rectangle of these points.

Starting from this root rectangle, the time-forward tree can most efficiently be generated by a procedure similar to what is known as *flood-fill* in context of computer graphics. A *seed* point is determined by tracking forward some point from the initial support $\vec{z} \in \mathrm{supp}(\Psi(t_n,\cdot)) \implies M(\vec{z}) \in \mathrm{supp}(\Psi(t_{n+1},\cdot))$. Then, the initial root hyperrectangles is refined at this point until the required recursion depth is reached and the flood fill begins at the resulting seed hyperrectangle. In this flood-fill procedure, it is determined for each of the (potential) neighbors of an hyperrectangle whether they contain density above a given threshold. This can, for instance, be achieved in a heuristic manner by sampling the time-forward phase-space density $\Psi(t_{n+1},\cdot)$ on the interface between both hyperrectangle. If the test is positive, the tree is refined in a way so that the neighbor is actually generated and the process continues at the newly generated hyperrectangle. This approach minimizes unnecessary function evaluations, in the sense that $\Psi(t_{n+1},\cdot)$ is rarely evaluated outside of its support.

CONCLUSION

We have presented the fundamentals of a 2^d-tree data structure with neighborhood relations. Such a structure allows for the efficient Vlasov simulation of arbitrary-dimensional phase-space densities even it they exhibit an exotic structure. The structure provides additional geometric information about the shape of the phase-space density, which can be used to implement sophisticated simulation strategies, which may be used to overcome many difficulties arising in Vlasov simulation efforts.

REFERENCES

[1] E.L. Saldin, E.A. Schneidmiller and M.V. Yurkov, Nucl. Instrum. and Methods A **528**, 355 (2004).

[2] Zh. Huang, ICFA Beam Dynamics Newsletter **38**, 37 (2005).

[3] M. Vogt, T. Limberg, D.H. Kuk, *Proceedings of EPAC'08*, Genoa, Italy, THPC111 (2008).

[4] M. Clemens, M. Dohlus, S. Lange and G. Pöplau, *internal report* DESY TESLA-FEL Report 2009-02 (2009).

[5] Ph. Amstutz, M. Vogt, *Proceedings of IPAC'17*, Copenhagen, Denmark, THPAB014, (2017).

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPAK138

[6] M. Vogt, Ph. Amstutz, "Arbitrary Order Perturbation Theory for Time-Discrete Vlasov Systems with Drift Maps and Poisson Type Collective Kick Maps", *Proceedings of the NOCE 2017 Workshop* (submitted), Arcidosso, Italy, (2017).

[7] Ph. Amstutz, M. Vogt, "A Time-Discrete Vlasov Approach to LSC Driven Microbunching in FEL-like Beam Lines", *Proceedings of the NOCE 2017 Workshop* (submitted), Arcidosso, Italy, (2017).

[8] M. Bader, *Space-Filling Curves*, Springer, Heidelberg, (2013).

[9] Ph. Amstutz, "Vlasov Simulation of Exotic Phase-Space Densities via Tree-Based Domain-Decomposition", Master thesis, Universität Hamburg, (in preparation, 2018).

[10] G. H. Hardy, E. M. Wright, *An Introduction to the Theory of Numbers*, Fifth Edition, Oxford University Press, (1979).

[11] "Neighbor finding techniques for images represented by quadtrees", *Computer Graphics and Image Processing*, **18**, 1, (1982).

[12] B. W. Kernighan, D. M. Ritchie, *The C Programming Language*, Second Edition, Prentice Hall, (1988).