Lattice software, algadd algorithms

Wolfgang Orthuber

Klinik für Kieferorthopädie der Universität Kiel
Arnold-Heller-Str. 16, D-24105 Kiel, Germany
further info: http://www.orthuber.com

Abstract

We have shown that only an a priori finite mathematical model can have an exact equivalent in physical reality [11] and that discrete considerations are adequate to get a deeper understanding of basic physical principles [8]. Because this leads to difficult combinatorics we have developed open-ended software [10] which is designed as general help to handle discrete numerical spaces in the form of multidimensional numerical lattices and to test algorithms on them. It is possible to define many algorithms ("*algadd*" algorithms) by using only a table. The aim is to find algorithms whose results correspond to the results of physical experiments better and better. This paper gives an overview of the software and shows examples of its usage.

PACS: 03.65.Ta, 02.10.Ox, 02.70.Bf
Keywords: Discrete physics, random walk, combinatorics

Contents

List of Figures

1 Introduction

We have shown [11] that only an a priori finite mathematical model can have an exact equivalent in physical reality. This means that it implies an only finite number of basic arithmetic operations on an a priori finite numerical space[1] which can be represented without using irrational numbers.

Up to now there is not much experience in this area: Important physical equations are defined on continuous (a priori infinite) sets and often written as partial differential equations. If we want to find the natural finite basis of them, first we have to replace differential calculus by finite-difference calculus. This can soon lead to difficult combinatorics, especially in case of interactions across several dimensions. But increasing performance of computers offers new possibilities. The mentioned numerical space can be represented by finite dimensional numerical lattices (finite dimensional point lattices with numbers resp. *quantities* assigned to every point) which can be handled adequately by a computer. So we developed as help software [10] for handling numerical lattices and for studying the results of numerical algorithms on them.

2 Software architecture

The software is written in C++ and open-ended. The file interface is clear and all data generated by the user (lattices, definitions, comments, also individual configuration) are by default stored in the program directory, without spread and in readable ASCII format. Possibilities to generate statistics like sums of the (squared) quantities in selectable lattice subspaces or subsets are available, with graphical representation. We will not deepen this here but concentrate on the basics, especially on the lattice characteristics and on the algorithm interface.

2.1 Numerical lattices

We use the term *numerical lattice* or shortly *lattice* for a finite dimensional point lattice in which a complex number resp. *quantity* is assigned to every

[1] Nevertheless both can increase without boundary *when* time increases without boundary (infinite potential).

point. A *lattice-family* is an indexed set of these lattices. It is the central object which is handled by the software. All points resp. quantities in it are addressed by integer coordinates: The lattice-index l, the time-like coordinate n and 30 free coordinates $k1, ..., k30$.[2]

The lattice-index. The coordinate l is the lattice-index, i.e. the index of the lattice within the lattice-family which is under consideration. This allows to combine an only by computer hardware limited number of multidimensional lattices, each having its individual name in form of its lattice-index l. The reservation of l for this purpose can e.g. facilitate the discrete implementation of physical equations which connect different kinds of physical quantities (quantities with different names). For implementation of complex algorithms it can be also helpful to assign different l to different components of a vector as shown in 3.4 and [9]. By combining many different l resp. lattices it is possible to study results of algorithms with total complexity far beyond reach of human brain. (Initially, however, the situation may be not too complicated and it is recommendable to work with as few lattices as possible.)

The time-like coordinate. The coordinate n is reserved for time-like behavior. This means that every algorithm which uses lattice quantities whose n is smaller or equal to $n0$ ($n <= n0$) should only influence quantities whose n is greater than $n0$ ($n > n0$).

The free coordinates. Additionally there are 30 free coordinates which are simply accessible by their number $1, ..., 30$ and we will call them $k1, ..., k30$ subsequently. In most cases only a part of them is used. Unused coordinates are by default 0 and it is not necessary to take care about them.

Denote the lattice quantity at the point with coordinates $(l, n, k1, k2, ..., k30)$ by $q(l, n, k1, k2, ..., k30)$. It can be a complex rational or floating point number. Initially all quantities are by default 0. This convention (that the quantity at every unused coordinate is 0) allows the handling of the lattice(s) by a computer and is also adequate because lattices (with distinguishable points) exist not a priori but have to be created by execution of branching algorithms

[2]The number of free coordinates resp. dimensions could be greater and is only restricted by computer hardware.

(cf. 2.2.1). Internally the lattices are organized as lattice family in a map (an associative container of the C++ standard template library) which stores all non-zero quantities. It is ordered by a structure which contains the 32 integer coordinates and n has the highest priority.

As shown in Fig. 1 all lattice quantities can be displayed and edited in a table which contains a selectable two-dimensional subset. Consequent working with rational numbers can preserve exactness. Of course clearness may be lost in case of rational numbers with many digits. If there are more than 10 necessary digits for numerator or denominator of a rational number, the implemented arithmetic automatically converts it to a floating point number. The possibility of using complex numbers (with non-zero imaginary part) is offered as bridge to current concepts and should make usage more convenient. Later complex numbers can be replaced as shown in Fig. 9, e.g. for detailed combinatorial and graph theoretical studies.

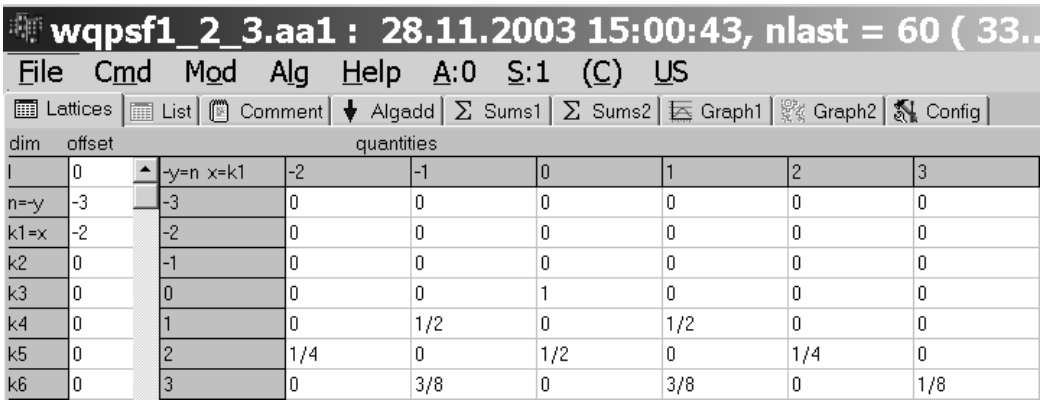

Figure 1: Editing lattices. The left table determines the lattice and its part which can be edited in the right table. Here lattice $l = 0$ is displayed, the negative y-coordinate (the vertically downwards increasing row number) corresponds to the time-like coordinate n, the horizontal x-coordinate (the column number) corresponds to the coordinate $k1$. The lattice contains the sequence of symmetric binomial distributions resulting e.g. from iterations of *algadd-random* which is defined by the table in Fig. 2.

2.2 Algorithms

2.2.1 General guidelines

It is important to realize that in physical nature the number of distinguishable possibilities of experimental results is finite at given time [11] and is increasing only *together* with time. Furthermore physical laws are by definition stable, i.e. they are valid iteratively at every time. Therefore nature conform software algorithms must be iterative in time and must operate on discrete spaces resp. numerical lattices, starting with a finite number of different (lattice) quantities and producing an increasing number of new quantities during their iteration. Furthermore the results of [8] suggest that these algorithms should generate ***branching loops***, i.e. iteratively in the course of time (iteratively after a finite number of iterations of the algorithm) the quantities should have effect on quantities at other new coordinates[3] which in turn later have effect backwards.

We now focus our interest on the algorithm interface.

2.2.2 User algorithms

Of course the user can develop algorithms by writing code according to own ideas. At this it is not necessary to understand all details of the software architecture, it is also possible to use the readable ASCII file format of saved data as interface.

2.2.3 Algadd algorithms

The software is already prepared for a large class of algorithms which we call *algadd*-algorithms. It is possible to define these algorithms without writing code. This is done by usage of a table with several entries as shown in Fig. 2. The in C++ written open-ended software [10] is designed for editing of numerical lattices and for generation of statistics like sums over the absolute or squared quantities of subspaces, with graphical representation. Using the ASCII file format of saved data as interface or writing own code the user can check the results of own algorithms. Additionally it is possible to define

[3]More exactly: Effect on quantities at coordinates which are different resp. separated not only in time coordinate from up to now used (resp. created) coordinates. In physical nature this separation can be a measurable potential barrier.

algorithms without writing code. This is done by usage of a table with several entries. Every entry defines a propagation between two lattices in form of an addition from the first (the source lattice with index l) to the second (the destination lattice with index $ldest$). The strength of the propagation is determined by the *propagator*[4] p: During every iteration of the algorithm all non-zero quantities of the source lattice with $n = nlast$, i.e. quantities at points with maximal coordinate n before iteration, are added to shifted locations with $n = nlast + 1$ within the destination lattice after multiplication by p.

Source and destination lattices and the multidimensional relative shift resp. offset $(dk1, ..., dk30)$ are selectable using integer numbers, the propagator p can be a complex number, if desired. Per iteration of algadd only quantities with $n = nlast$ are used and influence only new quantities with $n = nlast+1$. Because of this convention every existing quantity remains untouched and can be later used for analysis of development. If initially $nlast = 0$ like in chapter 3, then n simply is the number of the iteration which has generated the associated quantity. For the purpose of clarity we will summarize this and give a compact definition:

Definition (algadd). An algadd algorithm, shortly *algadd*, is an iterative algorithm which operates on the in 2.1 defined lattice-family. Let $nlast_0$ resp. $nlast_1$ denote the maximal time-like coordinate n of all lattice points with non-zero quantities before resp. after the iteration and let $M := \{0, 1, 2, ..., j_{max}\}$ denote the set of all entry numbers in the algadd table as shown partially in Fig. 2 (in the current version $j_{max} = 255$). For $j \in M$ let l_j and $ldest_j$ denote indices of source and destination lattices of entry j, p_j its propagator and $dk1_j, dk2_j, ..., dk30_j$ its coordinate offsets. Then per iteration of algadd the following additions are done:

$$\begin{aligned} & & q(ldest_j,\ nlast_0 + 1,\ k1 + dk1_j,\ k2 + dk2_j, ...,\ k30 + dk30_j) \\ = & & q(ldest_j,\ nlast_0 + 1,\ k1 + dk1_j,\ k2 + dk2_j, ...,\ k30 + dk30_j) \\ + & p_j & q(l_j,\ nlast_0,\ k1,\ k2\ , ...,\ k30) \qquad \text{for all } j \in M. \end{aligned}$$

[4]There is a relationship to the Feynman propagator, but p is elementary because it propagates along minimal dn (resp. dt) and every p represents with its entry exactly one component of the total propagation (which can have many components). Furthermore it does not propagate to $n < nlast$ resp. *past* - according to the definition of the word *past*.

Result after the iteration. If the iteration has not created new non-zero quantities, then $nlast_1 = nlast_0$ and this and also further iterations will have no effect. If, however, new non-zero quantities have been created, then $nlast_1 = nlast_0 + 1$ and after the iteration holds:

$$q(l,\ nlast_1,\ k1,\ k2, ...,\ k30)$$
$$= \sum_{j \in M \wedge ldest_j = l} p_j \quad q(l_j,\ nlast_1 - 1,\ k1 - dk1_j,\ k2 - dk2_j, ...,\ k30 - dk30_j).$$

Lattices | List | Comment | Algadd | Σ Sums1 | Σ

0=aanr	0	1	2	3	4	5	6
p	1/2	1/2	0	0	0	0	0
ldest	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0
dn	1	1	1	1	1	1	1
dk1	-1	1	0	0	0	0	0
dk2	0	0	0	0	0	0	0

Figure 2: The algadd table. Only the entries resp. columns with non-zero p (here entries 0 and 1) cause some addition, the other are empty and not used. Not shown entries are empty, not shown rows contain zero offsets. This also holds for the other examples.

In this table the algorithm *algadd-random* is defined, n iterations of it generate the symmetric binomial distribution resp. probability distribution which results from a symmetric Bernoulli random walk with n steps. The only non-zero component of the offsets $(dk1, ..., dk30)$ is $dk1$. It determines the direction of the steps so that a binomial distribution of $k1$ results (Fig. 1 and Fig. 3). During every iteration all quantities of lattice $l = 0$ with coordinates $(0, nlast, k1, k2, ..., k30)$ are added to those of lattice $ldest = 0$ with coordinates $(0, nlast + 1, k1 - 1, k2, ...k30)$ and $(0, nlast + 1, k1 + 1, k2, ..., k30)$ after multiplication by $p = 1/2$. At this $nlast$ denotes the maximal time-like coordinate n of all lattice points which have non-zero quantities before iteration.

3 Some exemplary applications

Now we will describe some examples of algadd algorithms and their results. Before their iteration all lattice quantities are zero except in the origin where the quantity is 1. So the initial conditions are:

$$q(0,0,0,0,0,...,0) = 1 \quad \text{else} \quad q(l,n,k1,k2,k3,...,k30) = 0.$$

3.1 Simple random walk

The remarkable characteristics of the binomial distribution and its modifications was one of the reasons which motivated us to introduce the more general algadd concept. The definition of *algadd-random* in Fig. 2 represents the standard example, n iterations of it generate the symmetric binomial distribution of $k1$ which is identical to the probability distribution resulting from a "simple" symmetric Bernoulli random walk with equal probabilities $p = 1/2$ for steps in directions $-k1$ or $k1$. The result after 60 iterations is shown in Fig. 3 and has the well known bell-shape.

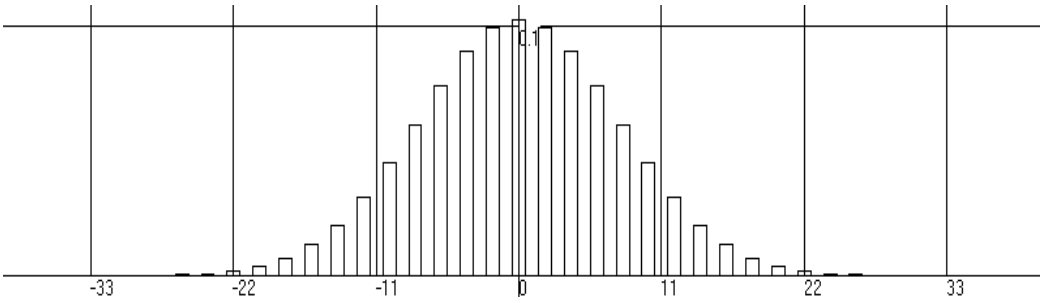

Figure 3: The binomial distribution resulting after 60 iterations of *algadd-random*. Abscissa: $k1$, Ordinate: $q(0,60,k1,0,0,0,...,0)$.

3.2 Finite differences

It is easy to generate finite differences along $k1$ by *algadd-fdiff*.

1=aanr	0	1	2	3	4	5	6
p	1/2	-1/2	0	0	0	0	0
ldest	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0
dn	1	1	1	1	1	1	1
dk1	-1	1	0	0	0	0	0
dk2	0	0	0	0	0	0	0

Figure 4: Definition of *algadd-fdiff* which generates finite differences along coordinate k1. In comparison to the definition of *algadd-random* in Fig. 2 only the sign of the propagator p in entry 1 is exchanged.

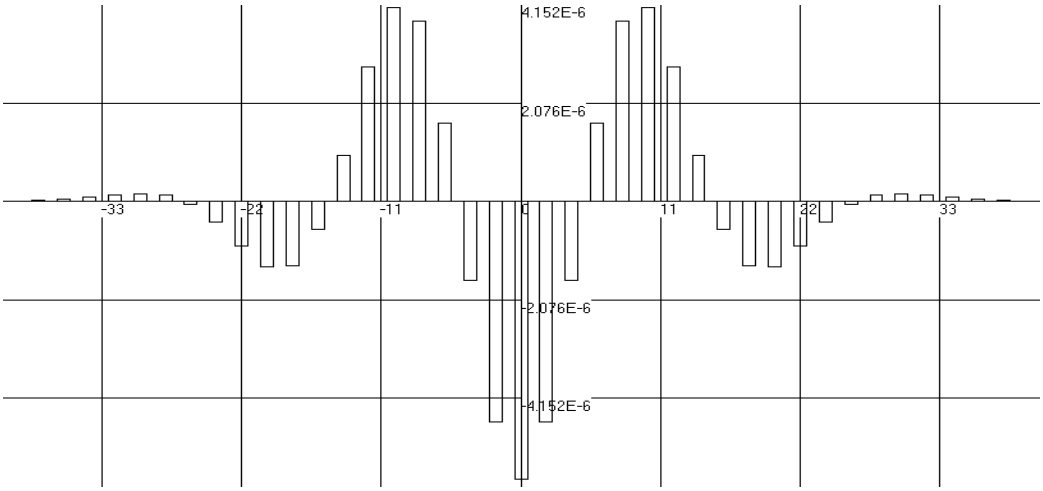

Figure 5: 6-th order finite difference of the binomial distribution shown in Fig. 3. Result after 6 iterations of *algadd-fdiff* and 60 iterations of *algadd-random*. Abscissa: $k1$, Ordinate: $q(0, 66, k1, 0, 0, 0, ..., 0)$.

Fig. 5 shows the 6-th order finite difference of the binomial distribution[5] in Fig. 3, generated by 6 iterations of *algadd-fdiff* and 60 iterations of *algadd-random*. At this the in Fig. 5 shown result in $n = 66$ is only dependent on the number of iterations of *algadd-fdiff* and *algadd-random*, not on their order as shown in Fig. 6 and Fig. 7.

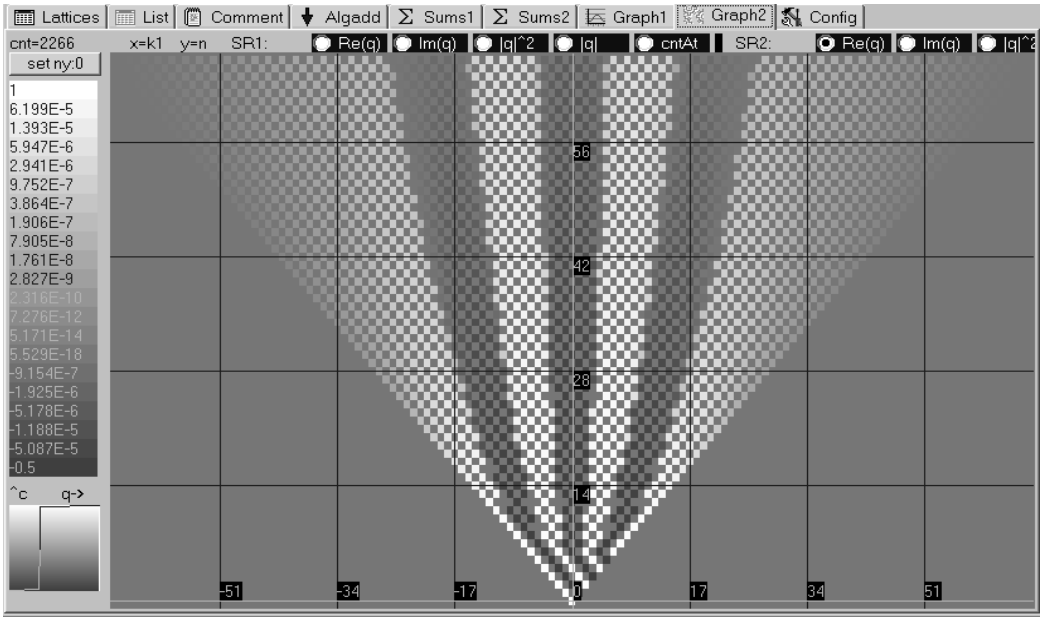

Figure 6: Initially generated finite differences (grayscale graph). The grayscale graph illustrates the lattice quantities in dependence of *two* independent coordinates. Here it shows the development of lattice $l = 0$ during 6 initial iterations of *algadd-fdiff* followed by 60 iterations of *algadd-random*. The vertical coordinate is n (the number of iterations) and the horizontal coordinate is $k1$. The grayscale represents the quantity $q(0, n, k1, 0, 0, 0, ..., 0)$. The brighter the color, the more positive is the corresponding quantity. Here the most negative quantity is -0.5 and represented by dark gray, the most positive quantity is 1.0 and represented by white. The scaling is automatically adapted that frequency of used different shades is approximately equal to maximize the information of the picture.

[5]Recall that the differentiation of the binomial distribution (resp. of its analytic borderline case which is proportional to the function $e^{-(x/a)^2}$, where a is a constant) is important e.g. for generation of the eigenfunctions of the harmonic oscillator [1] [12].

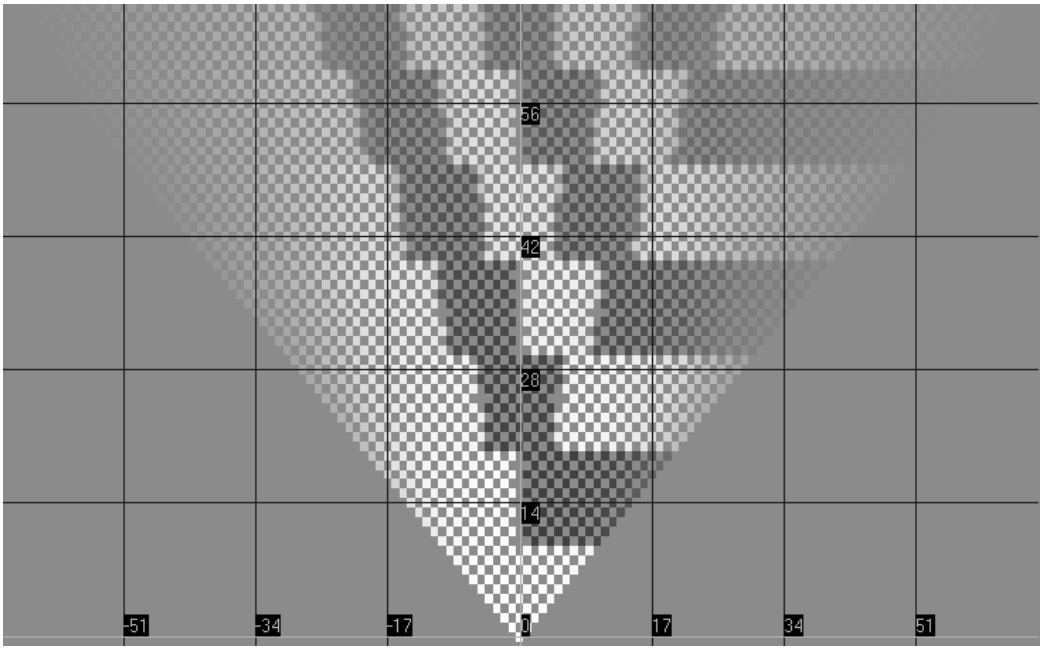

Figure 7: In-between generated finite differences. The result in $n = 66$ is the same as in Fig. 6 and in detail shown in Fig. 5. It is only dependent on the number of iterations of *algadd-fdiff* and *algadd-random*, not on their order. Here the 6 iterations of *algadd-fdiff* are not done initially but in between after every 9 iterations of *algadd-random*. Abscissa: $k1$, Ordinate: n, grayscale: $q(0, n, k1, 0, 0, 0, ..., 0)$.

3.3 Discrete representations of analytic functions

First we must not forget that analytic functions with infinite power series cannot have an (exact) equivalent in physical reality [11]. But they can be helpful in many cases, particularly for approximative considerations. Therefore they are commonly used and it is appropriate to show bridges. For example it is possible to generate discrete representations[6] of standard wave functions like sin and cos by very concise algadd definitions. The most evident possibility for this is the usage of a complex propagator p with $|p| = 1$ which causes a rotation, see Fig. 8 and Fig. 9. Though the rotation angle could be arbitrarily small we chose it big enough to make gradation clearly visible, especially in the right graph in Fig. 9. The second algadd definition in Fig. 9 also demonstrates the replacement of complex numbers by 2×2 matrices. This can be helpful for advanced graph theoretical considerations.

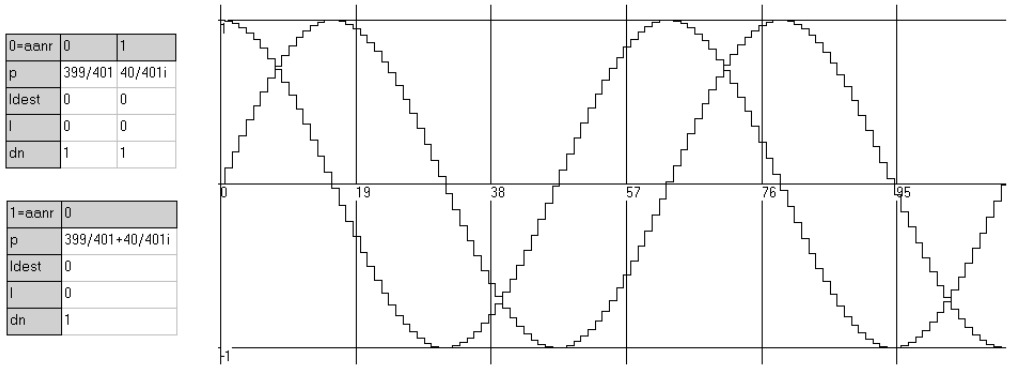

Figure 8: Discrete representation of sin, cos. Both tables contain equivalent definitions of *algadd-sincos*. Every iteration of it results in the multiplication by a complex number which causes a rotation. $40/401i$ means $(40/401)i$. The graph shows the development in dependence of the iteration number n. Abscissa: n, Ordinate: real resp. imaginary part of $q(0, n, 0, 0, 0, 0, ..., 0)$.

[6] representations which have a discrete but arbitrarily fine subdivided range of values

5=aanr	0	1	2	3
p	1680/3361	0	2911/3361i	0
ldest	0	0	0	0
l	0	0	0	0
dn	1	1	1	1

6=aanr	0	1	2	3
p	1680/3361	1680/3361	2911/3361	-2911/3361
ldest	1	0	1	0
l	1	0	0	1
dn	1	1	1	1

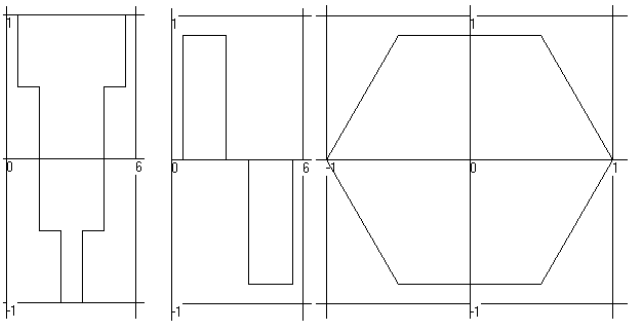

Figure 9: Complex numbers as 2×2 matrices. The algadd defined by the upper left table causes a rotation like in Fig. 8, but with greater angle (approx. $\pi/3$). The graphs show the development during 6 iterations, the left and middle graphs show the real part and imaginary part in dependence of the iteration number, the right graph shows the imaginary part in dependence of the real part. The algadd defined by the lower left table causes analogous graphs, but the imaginary part is replaced by an additional lattice with index $l = 1$ and the multiplication by a complex number is replaced by the multiplication by a rational 2×2 rotation matrix.

3.4 Discretization of differential equations

Above we have chosen simple examples to introduce the algadd concept. Now follows a nontrivial example. In 3.2 we have seen that the generation of finite differences is a standard application of algadd algorithms. So it is reasonable to use this for discretization of approved (partial) differential equations in mathematical physics. Because of their elementary importance and their combinatorial potential we chose the vacuum Maxwell equations and converted them into finite-difference equations. Fig. 10 shows a corresponding algadd definition and Fig. 11 shows one result, [9] contains a more detailed treatment of the subject.

2=ax	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
p	0.08	-0.08	-0.08	0.08	-0.08	0.08	0.08	-0.08	0.08	-0.08	-0.08	0.08	-0.08	0.08	0.08	-0.08	0.08	-0.08	-0.08	0.08	-0.08	0.08	0.08	-0.08	1	1	1	1	1	1
ldes	4	4	5	5	3	3	5	5	3	3	4	4	1	1	2	2	0	0	2	2	0	0	1	1	0	1	2	3	4	5
l	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5	0	1	2	3	4	5
dn	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
dk1	0	0	0	0	0	0	1	-1	0	0	1	-1	0	0	0	0	0	0	1	-1	0	0	1	-1	0	0	0	0	0	0
dk2	0	0	1	-1	0	0	0	0	1	-1	0	0	0	0	1	-1	0	0	0	0	1	-1	0	0	0	0	0	0	0	0
dk3	1	-1	0	0	1	-1	0	0	0	0	0	0	1	-1	0	0	1	-1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10: Maxwell equations discretized. This algadd definition shows a discrete version of the vacuum Maxwell equations. It uses 6 four-dimensional lattices with indices $l \in \{0, 1, 2, 3, 4, 5\}$ for representation of the in [9] defined field components $\hat{E}_x, \hat{E}_y, \hat{E}_z, \hat{B}_x, \hat{B}_y, \hat{B}_z$. Apart from the right 6 copying entries the absolute value of the propagator p is $\sqrt{\alpha} = 0.085424542921$ which is the square root of the fine structure constant.

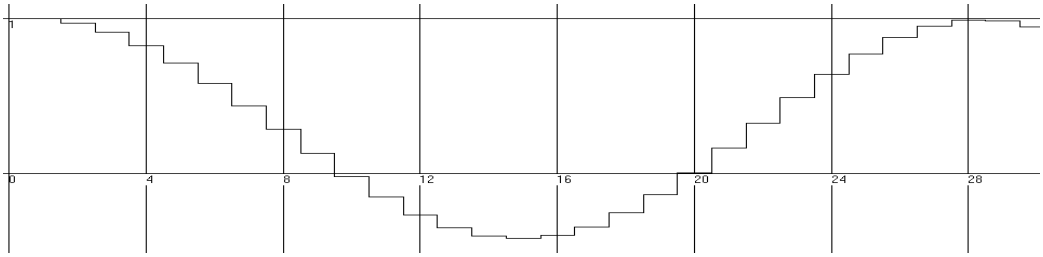

Figure 11: The first wave. Short term development of lattice $l = 0$ during the first iterations of the in Fig. 10 defined algadd algorithm. Abscissa: n, Ordinate: $q(0, n, 0, 0, 0, 0, ..., 0)$. It is not trivial that the second maximum at $n = 28$ has nearly the initial quantity 1.

4 Discussion

The motivation for the algadd algorithm scheme arises from the fact that it is simple and nevertheless very general. We see in the exemplary definition of *algadd-random* in Fig. 2 that the propagator p can have the meaning of the probability of a step in one direction. This can be used to study (superpositions of) random walks which play a fundamental role in (quantum) physics. For example the second order finite difference along the location coordinate k of a symmetric Bernoulli random walk is equal to the first order finite difference along the time-like coordinate n like in the Schroedinger equation as mentioned in [8]. This is also valid for linear combinations of these random walks. Negative combinations can lead to finite differences as shown in 3.2. For quantum theoretical considerations it is also useful that p can represent a complex probability amplitude because it is not restricted to the interval $[0, 1]$. Furthermore the examples in chapter 3.3 show that it is easy to generate discrete representations of periodic functions which are traditionally used in mathematical physics. But it should be annotated, that these examples don't satisfy the criteria of 2.2.1. They are not time conform because their defining tables in Fig. 8 and Fig. 9 contain only zero offsets (i.e. $(dk1, dk2, ..., dk30) = (0, 0, ..., 0)$) and with that no branching entries. A restriction of the algadd scheme is the constancy of the propagator p. In physical reality p doesn't need to be constant. Even if it is interpreted as an mean value (like a probability), it can change (systematically) during a sequence of iterations. It would be possible to expand the scheme and to take this into consideration, e.g. to define p as a function of previous lattice quantities. Without concrete information, however, this seems premature. Already now the algadd scheme allows it to make p dependent on last individual history of a random walk: For example in case of two possible directions v_{-1} and v_1 per step we can use different lattices l_{-1} and l_1 as destination for steps in these directions and define different propagators p_{-1} and p_1 in those entries of the algadd table which use l_{-1} and l_1 as source lattices. The result is simply that the next p is dependent on the direction of the previous step. It is possible to expand this method to more steps backwards, but we will not deepen this here. The above examples demonstrate the adaptability of the algadd concept and that it can be used to introduce consequent discrete considerations in mathematical physics.

5 Conclusions

We have seen that many means for the study of algorithms on numerical lattices are provided and that it is possible to define a large class of algorithms without writing code, already in the current stage. The architecture of the software is open-ended. It is possible that we gain new surprising information [9] when we use it to check our ideas. Of course we have to ask the right questions.

References

[1] M. Aunola, *The discretized harmonic oscillator: Mathieu functions and a new class of generalized Hermite polynomials*, J. Math. Phys. 44, (2003), 1913-1936; http://arxiv.org/abs/math-ph/0207038

[2] L. H. Kauffman, *Non-commutative Calculus and Discrete Physics*, http://arxiv.org/abs/quant-ph/0303058

[3] A. Khrennikov, Y. Volovich, *Discrete Time Leads to Quantum-Like Interference of Deterministic Particles*, Proc. Int. Conf. Quantum Theory: Reconsideration of Foundations. Ser. Math. Modelling in Phys., Engin., and Cogn. Sc., Växjö Univ. Press (2002), 441-454; http://arxiv.org/abs/quant-ph/0203009

[4] A. Khrennikov, Y. Volovich, *Interference effect for probability distributions of deterministic particles*, Proc. Int. Conf. Quantum Theory: R. of Foundations. Ser. Math. Modelling in Phys., Engin., and Cogn. Sc., Växjö Univ. Press (2002), 455-462; http://arxiv.org/abs/quant-ph/0111159

[5] D. Levi, P. Tempesta, P. Winternitz, *Umbral Calculus, Difference Equations and the Discrete Schroedinger Equation*, http://arxiv.org/abs/nlin.SI/0305047

[6] M. Lorente, *Quantum Mechanics on discrete space and time*, Proceedings: M. Ferrero, A. van der Merwe, eds. New Developments on Fundamental Problems in Quantum Physics (1997), 213-224; http://arxiv.org/abs/quant-ph/0401004

[7] W. Orthuber, *A discrete and finite approach to past physical reality*, International Journal of Mathematics and Mathematical Sciences 2004:19 (2004) 1003-1023.

[8] W. Orthuber, *A discrete and finite approach to past proper time*, http://arxiv.org/abs/quant-ph/0207045

[9] W. Orthuber, *A discrete approach to the vacuum Maxwell equations and the fine structure constant*, http://www.orthuber.com/wqps1.pdf

[10] W. Orthuber, *The Recombination Principle: Mathematics of decision and perception* (init. 2000), http://www.orthuber.com

[11] W. Orthuber, *To the finite information content of the physically existing reality*, http://arxiv.org/abs/quant-ph/0108121

[12] A. Turbiner, *Canonical Discretization. I. Discrete faces of (an)harmonic oscillator*, Int. J. Mod. Phys. A16, (2001), 1579-1604; http://arxiv.org/abs/hep-th/0004175