



Fermi National Accelerator Laboratory

FERMILAB-Conf-95/213-T  
July 15, 1995

## MILXY WAY: HOW MUCH BETTER THAN VEGAS CAN ONE INTEGRATE IN MANY DIMENSIONS?<sup>a</sup>

G.I.Manankova<sup>i</sup>, A.F.Tatarchenko<sup>ii</sup> and F.V.Tkachov<sup>iii</sup>

<sup>i</sup> Center of Technical Lasers of Russ. Acad. Sci., Troitsk 142092, RUSSIA

<sup>ii</sup> Institute of High Pressure Physics of Russ. Acad. Sci., Troitsk 142092, RUSSIA

<sup>iii</sup> FNAL, P.O.Box 500, Batavia, IL 60510, USA and

Institute for Nuclear Research of Russ. Acad. Sci., Moscow 117312, RUSSIA<sup>b</sup>

**Abstract** MILX is an implementation of a universal adaptive algorithm of integration in many dimensions based on a systematic use of simplicial decompositions of integration domain and elements of AI to handle geometric structures and adaptation strategy. For a non-trivial class of integrands, it yields similar or better precision than Lepage's VEGAS. MILX surpasses VEGAS in flexibility, and smoothly handles diagonal structures that are a stumbling block for VEGAS. It offers a wide range of options for optimizations for concrete classes of integrands. We discuss implementation, directions of further development, fine-tuning options, and possible applications of the algorithms of the 'MILXy Way' family.

---

<sup>a</sup> A contribution to AINHEP-95 (Pisa, Italy, April 1995), extended version.

<sup>b</sup> Permanent address.

## 1. Introduction

The multidimensional integrals (Feynman diagrams) play the same role in high-energy physics as differential equations do in classical mechanics. The importance of the problem of numerical multidimensional integration (NMI) can therefore not be overstated. A good review of NMI can be found in Ref.1, and below we consider for comparison only Lepage's VEGAS 2 which seems to be the best program of its kind.

It is hard to imagine how VEGAS, which is just ~250 FORTRAN lines long, could possibly be beaten on the basis of quality-of-results-and-universality per line-of-code. However, VEGAS cannot satisfactorily handle diagonal structures (e.g. an integrand localized along the main diagonal of the hypercube,  $[0, \dots, 0]$ — $[1, \dots, 1]$ ). This is because it relies on a fixed coordinate system to slice the hypercubic integration domain by subdividing the coordinate axes. No satisfactory way to circumvent this difficulty within the framework of VEGAS has been found. VEGAS handles all the other (statistical and adaptive) aspects of Monte Carlo integration with a stunning economy, so that its geometric rigidity seems to be its only weakness.

We would like to report the results of the first, pilot stage of our MILX<sup>c</sup> project that aims to explore an adaptive NMI scheme based on simplicial (rather than hypercubic) decompositions of the integration domain.

The simplicial decompositions are the most universal and flexible tool to describe the structure of multi-dimensional geometric objects (manifolds etc.; see any textbook on algebraic topology). They are also well-known in numerical mathematics (e.g. the method of finite elements). However, their sufficiently flexible program implementation requires advanced programming techniques to handle complex data structures — the techniques that is better known in the context of AI research than numerical mathematics. In this respect it should be pointed out that writing a general integration routine based on simplicial decompositions was attempted in Ref.3 which used an exotic non-portable programming language. Apparently, the latter fact together with the success of VEGAS that was made public at about the same time, eclipsed the results of Ref.3.<sup>d</sup> However, the development of new programming methodologies in recent years (the object-oriented programming paradigm implemented in the highly portable C++) together with a proliferation of applied problems that involve NMI, allow one to reconsider the issue from an entirely different perspective and in a much more practical context.

## 2. Description

### 2.1. *The philosophy of MILX*

*Adaptability is an overriding consideration in higher dimensions.*

*G.P.Lepage, VEGAS<sup>2</sup>*

Experience shows that some form of adaptive Monte Carlo (MC) integration is an essential ingredient of any successful NMI program. This is because in many cases an appropriate subdivision of integration domain allows one to drastically reduce variance of the pure MC integration.<sup>1</sup> The crux of the matter is how well a program can perform such a subdivision. The latter depends, first, on how well the geometry of the integrand can be reflected in the resulting subdivision with a concrete subdivision technique, and second, on how well a particular subdivision strategy exploits available information about the integrand. The two

---

<sup>c</sup> The characters in MILX stand for Multidimensional Integration / List / simpleX. "MILXy way" seems an appropriate description of the underlying philosophy because MILX should best be thought of as a flexible environment rather than a single program — an environment that is capable of accomodating a vast range of adaptation strategies optimized for particular applied NMI problems.

<sup>d</sup> It is reviewed in Ref. 1 and we became aware of Ref. 3 only while preparing this report which may be as well because otherwise the decision to undertake the MILX project would have been more difficult. It is inevitable that there should be certain parallels between MILX and Ref. 3, and our results corroborate the optimistic conclusions of Ref. 1 concerning the potential of simplicial decompositions for NMI. In this respect we would like to emphasize that there are few new major technical elements in MILX. What is new though is an attempt to implement those elements in a practical tool.

aspects of adaptation strategy in NMI can be conveniently referred to as the “geometric” aspect and the “artificial intelligence” aspect.

On the other hand, there are many different applied problems with vastly different requirements, and because NMI is notoriously difficult one would like to make full use of any information on the integrand that may be available. This means that each problem has its own optimal subdivision strategy depending on whether the integrand is smooth or not; on the form of the integration domain; on the position and type of singularities/spikes etc. Then it is hardly possible to have one program to satisfy every need. A better approach is to have a program package that could be optimized for each particular problem — or to serve as a toolbox to be used when implementing a specific subdivision strategy.

### 2.1.1. *Geometric aspect*

VEGAS along with many other earlier programs uses hypercubic decompositions, so that the initial integration domain and all the subdomains are  $d$ -dimensional hypercubes. This requires fixing a coordinate system, and changing the latter proves to be unfeasible. On the other hand, algebraic topology (see any textbook) systematically uses simplices as simplest and extremely flexible building blocks from which to construct arbitrarily complex manifolds. An *oriented  $d$ -dimensional simplex*  $S = [v_0, \dots, v_d]$  in a  $D$ -dimensional space is a convex hull of  $d + 1$   $D$ -dimensional vectors  $v_i, i = 0, 1, \dots, d$ . The order of the vectors fixes the orientation of the simplex. In general,  $d \leq D$ . A 0-dimensional simplex is a point; a 1-dimensional simplex is a segment of straight line between two points; a 2-dimensional simplex is a triangle spanned by its three vertices; a 3-dimensional simplex is a tetrahedron spanned by its four vertices; etc. The numbers  $x_i, i = 0, \dots, d, \sum_{i=0}^d x_i = 1$ , are *local coordinates* of a point  $v$  in  $S$  if  $v = \sum_{i=0}^d x_i v_i$ . (The usual cartesian coordinates of  $v$  are its *global coordinates*.) If all  $x_i > 0$  then one deals with an internal point of the simplex. If for some  $i$   $x_i = 0$  then the point belongs to the boundary of  $S$ . For  $d = D$  the volume of the simplex is defined in a standard way. If the orientation of the vectors  $v_i - v_0$  coincides with that of the global coordinate axes then the volume is positive by definition; otherwise it is negative. Note that the boundary of a  $d$ -dimensional simplex consists of  $(d - 1)$ -dimensional simplices, etc. It is clear that one can approximate a domain of arbitrary shape using simplices, and that any simplicial decomposition can be refined as far as needed in many ways. This gives one ultimate geometrical flexibility but the program implementation may be extremely cumbersome if not done with greatest care.

In the context of MILX, we use the term “simplexon” to refer to a correct simplicial decomposition of an integration domain. 2.1.1

### 2.1.2. *Iterative construction of a simplexon*

In the simplest case the initial integration domain consists of one  $D$ -simplex. An adaptation program explores the integrand and decomposes the initial simplex into a sum of a larger number of simplices using a number of simple basic operations. Which operations to apply depends on the strategy adopted (see below). The number of possible basic operations is rather large:

One such basic operation consists in splitting a  $D$ -simplex into  $D + 1$   $D$ -simplices with a new vertex inside the simplex. A crucial point here is that one does not need to recompute the volumes of the new simplices because they are just fractions of the initial volume determined by the local coordinates of the new vertex. One can take any two vertices of a simplex and split the corresponding segment in two. This induces a split in two of all  $D$ -simplices that have the same segment as part of their boundary, and the volumes are split proportionately. One can similarly split any  $d$ -dimensional boundary simplex inducing splits of all adjacent  $D$ -simplices. The resulting simplices inherit the orientation of the initial one.

An important operation is a move of a vertex. Then the volumes of the adjacent simplices change accordingly. It may happen that the vertex moves too far so that there emerge simplices with negative volumes which means that some simplices overlap. This, however, proves to be of little import because the MC inte-

gration with infinite number of points would give a correct result if contributions of all simplices take into account their orientation. On the other hand, a multiple count of the same subdomain due to the overlap enhances the total variance and is, therefore, automatically avoided by any reasonable subdivision strategy.

An important class of operations ensures a possibility of nontrivial restructuring of the simplicial decomposition. For instance, merge any two adjacent vertices into one. Then all simplices that had both vertices in their boundary are eliminated. This can be generalized to more than two vertices. On the other hand, one can take any vertex, consider the body consisting of all its adjacent  $D$ -simplices, eliminate the vertex and rebuild the volume into a lesser number of simplices. Such rebuilds allow the adaptation algorithm explore, in principle, all possible simplicial decompositions of a given domain.

### 2.1.3. Analytical criteria

The purpose of any decomposition of integration domain is to reduce the variance of MC integration. This is the ultimate goal, and the *only* criterion of optimization. We emphasize this point because one may worry about ungainly geometry of the resulting simplexon (e.g. many "splinters" i.e. narrow long or almost flat simplices) and thus get ensnared in unnecessary complications. The optimal simplexons often look counter-intuitive especially when higher-order analytic quadratures are employed.

In general, a concrete class of problems determines the strategy of constructing an optimal simplexon. The minimal analytical information one uses is the current estimate of the variance for each simplex. We find analytic quadratures a powerful tool for NMI, so it is natural to use an interpolation formula as a zero-approximation and correct it by computing via MC the integral of the difference between the exact integrand and its interpolation. Then the variance measures the error of MC integration of that difference. A program can choose an interpolation method that yields a minimal variance (e.g. no interpolation; linear or quadratic interpolation etc.). The error estimates being a very difficult and important element of NMI, it is crucial that many ways to estimate errors were available e.g. based on Runge-Eitken process etc.<sup>6</sup>

Furthermore, it may be useful to have an information on how quickly the integrand or variance varies around some vertex (e.g. for deciding whether and where to move it). In some special cases one may wish to estimate variance of the integrand for all  $D + 1$  faces of a simplex and choose an operation accordingly.

### 2.1.4. Strategy and tactics

The specifics of the problem determine the global strategy of constructing an optimal simplexon. In general, the strategy consists of a few stages (that can be referred to as e.g. *debut*, *mittelspiel*, *endspiel* etc.) that involve different priorities and criteria for operations used to modify the simplexon. It is convenient to use the term *tactics* to refer to the concrete criteria for choosing and performing an operation. Thus, the *debut* stage may give priority to splits of simplices while neither moves of vertices nor rebuilds are attempted. Also, if there is a priori information on the position of a spike then the latter should be isolated within a simplex as early as possible using special techniques. The same principle applies if one knows that the integrand has (integrable) singularities near the boundaries of the integration domain.

On the other hand, the *endspiel* stage of the simplexon construction is characterized by lack of resources (e.g. insufficient memory to increase the number of simplices) and therefore should focus on fine-tuning positions of vertices and rebuilds.

An important strategic component of any complex construction problem is an option to backup if the construction process fails to result in much progress after several iterations. This can be achieved either as a simple backup (e.g. by restoring a previous version of simplexon saved earlier) or via more complex operations (such as rebuilds and eliminations of vertices described above) that could radically redirect the construction process.

---

<sup>6</sup> It should be emphasized that such tricks are allowed only to estimate errors of analytic quadratures for an individual simplex. The error for the entire integral is of a statistical nature and must be handled accordingly.

Quite obviously, the number of options is enormous, and an optimal choice (as our experience shows) can rarely be found by random experimentation. The challenge here is to formulate general “philosophic” principles of construction for each step depending on the a priori information about the integrand and a very clear understanding of the objectives, then to reduce those principles to an unambiguous heuristic, and finally to translate that heuristic into concrete formulas and prescriptions in a simple and direct way. Neither proves to be an easy task.

### 2.1.5. *Programming techniques*

It is clear that both to handle the complex geometric operations involved in the construction of an optimal simplex, and to allow a sophisticated and flexible investigation of the integrand to determine an optimal operation at each iteration, one needs rather complex dynamic data structures. One has to deal with several types of lists (to describe vertices with their global and local coordinates, simplices, cross-pointers to ensure fast search of, e.g., simplices adjacent to a given vertex, as well as all sorts of intermediate data, e.g. a temporary list of simplices that are to be affected by a rebuild, etc.). If one aims at a practical portable NMI system designed along the lines described above then the only viable choice seems to use the C/C++ programming language. The entire program package should then consist of several program layers:

- The lowest (technical) layer handles memory allocation and the list structures without regard to their meaning.
- Another (geometric) layer interpretes the geometrical information and performs various operations on the simplex.
- Yet another (“AI”) layer deals with strategy and tactics using analytical information and implements the “artificial intelligence” of the algorithm.
- Finally, the topmost layer ensures an interface with the user program (or with the user if the construction is interactive).

Depending on the user’s expertise the user may wish to work at a particular level. This emphasized the difficulty of the overall design of MILX-type algorithms: they are to provide more than one user interface of varying complexity in a coherent manner.

## 2.2. *Aims of the project*

We report here the results of the first, exploratory stage of the MILX project. A natural breakpoint occurred when — after much experimentation with a working model and its comparison with VEGAS — we became fully aware of the strength and weaknesses of both VEGAS and MILX, and a sufficient understanding was gained to allow us to begin a systematic work on a version oriented towards real-life applications. Our most important conclusion of a general character is that MILX-type algorithms are at their best when there is some a priori information that allows one to fine-tune the algorithms for a concrete application. Therefore, further progress can be best ensured in the context of a specific applied problem, and we hope to provide here some information to help one to see whether a MILX-type program may suit one’s needs.

Although a fairly general version of the algorithm is possible, both the spirit of MILX and our experience indicate that one should aim at developing a toolbox-type library of routines that would allow one to put together a customized algorithm for a specific application, together with a library of ready solutions (“strategies”) optimized for some typical problems.

The complexity of such a package requires a considerable effort to provide its complete self-contained description, and with the limited resources at our disposal that should not be expected to happen very soon. Even when such a description is available publicly, a successful customization for a non-standard problem would typically require a considerable experience with, and understanding the properties of, the MILX-type algorithms — especially because they sometimes tend to exhibit rather unpredictable reactions to seemingly minor changes in their control parameters. This, of course, makes them great fun to play with and points to hidden resources of optimization — but magic solutions should not be expected.

The working model of MILX described below was implemented using Turbo C version 2.0 on a 486/33 PC within the usual 640K PC memory limitation which proved sufficient for dealing with non-trivial examples in  $D = 5$ .

### 2.3. Data structures

The key elements of the data structure describing the simplexon are a list of all vertices and a list of all simplices. The concrete organization is chosen based on considerations of simplicity, flexibility of program maintenance and on how often a particular operation is performed (e.g. a search of all simplices adjacent to a given vertex or a search of the 10% of simplices with the largest variance, etc.).

A vertex is, essentially, a list of its global coordinates together with some analytical information (the value of the integrand at the vertex as well as some sort of variance-like parameter to describe the quality of simplexon construction around it; for simplicity we call this parameter *variance of vertex*). Each vertex is assigned a unique label when it is created. The multiply connected list of vertices is ordered lexicographically with respect to their labels, and there is a separate ordering with respect to their variances which is always maintained.

A simplex is an array of (labels of, and pointers to) vertices ordered lexicographically with respect to their labels (since the latter may not coincide with the orientation of the simplex a sign is also stored to correct for that) plus some analytical information (the current estimates of the integral and the variance for the simplex together with the number of integrand calls used for that, as well as the interpolation method that yields best results; the integral and variance are estimated relative to that interpolation as explained above). The multiply connected list of all simplices has several orderings: one is lexicographic with respect to their "names" (i.e. the arrays of their vertices); another is with respect to their variances.

The list of vertices is never large even in high dimensions because the number of simplices is much larger, and it is the resources taken up by the latter that determine the memory requirements.

The fact that various searches (e.g. during investigation prior to choosing an operation on simplexon) are performed rather often, explains that the necessary orderings of the two lists are continuously maintained. A relatively local nature of operations used to modify the simplexon ensures that only a few objects (vertices and simplices) are affected each time, so that full sorts are very seldom needed.

### 2.4. Adaptation strategies and tactics

At the pilot stage of our project we concentrated mostly on two generic forms of integrands. One is a characteristic function of a subregion within the standard simplex (an ellipsoid with arbitrary foci and radii allowing a partial overlap with the simplex, and a hypercube with arbitrary sides). This models integration over a domain of a complex shape embedded in a simplex (as required by the current version of MILX; note that this restriction can be eliminated). The second class of functions we considered is exponentials of the form  $\exp \sum_{i=1}^D y_i^2 / \sigma_i$  where  $y$  are global coordinates and  $\sigma$  are arbitrary parameters.

Note first of all that the simplicial algorithms prove to be supremely insensitive to orientation of the dominant integration subregion with respect to the global coordinate axes. Therefore, the problem of "diagonal" structures is unknown to MILX.

In both cases, we find that the debute and mittelspiel stages of the construction of simplexon are characterized by the predominant usefulness of the various splittings of simplices with a small admixture of rebuilds (however, we have not yet fully explored the effects of complex rebuild operations which may be non-negligible). The tactics that proves to be most useful consists in focusing on the simplices/vertices with largest variance and attempting to split them. In accordance with the philosophy of MILX, the actual split is performed with a maximal use of available information. First, a guess is made as to the optimal position of the new vertex (typically, a convex combination of vertices of the simplex being split with weights equal to their variances). Then an MC estimate of variances of the new simplices is made, and the position of the vertex is adjusted in the direction of the new simplices with maximal variances. One can perform as many adjustments as one wishes but a two-step procedure is satisfactory in a majority of cases.

The crucial problem with the integrands of the first class is the occurrence of tiny overlaps of the non-zero subregion with the seemingly empty surrounding simplices. Such overlaps are the main reason why the algorithm may show a "reluctance" to significantly increase precision after certain stage. VEGAS tackles this problem using a smearing between adjacent intervals and this trick is also possible in MILX. But MILX allows potentially more effective solutions. For instance, a simple and useful heuristic is to place vertices in suitable cases at positions where the integrand is zero. This significantly reduces the above overlaps but does not eliminate them altogether. One should regard this as a beginning of a new optimization stage that requires special tactics. Finding such overlaps is, in principle, straightforward (it is sufficient to examine in more detail the "internal" — with respect to the global "external" boundary of the integration domain — boundaries of the "empty" simplices) but we did not attempt that with the working model due to certain rigidity of the programming implementation of the latter. The new version is expected to allow a much better handling of this situation.

In general, the final stages of construction (prior to the straightforward MC integration at the very end) are more sensitive to the properties of the integrand and require more subtle optimization tactics. The latter require a flexible systematic organization of the instrumental layer of the program.

In the second class of integrands their smoothness suggests that emphasis should be shifted to the use of analytic quadratures. This indeed proves to be the case to such an extent that for wide smooth structures the details of how exactly the splits are done affect the precision of final results little.

On the whole it appears that with MILX, it is often more advantageous to spend resources on the optimal simplex construction rather than the direct MC integration at the end. Since the simplex construction is performed stochastically as a matter of principle, the error of the final result has a random nature even if analytic quadratures are used in all elementary simplices.

### 2.5. Instrumental means

The most important tool is a (pseudo-) graphical representation of the distribution of vertices and simplices with respect to variances. This allows one to monitor the progress of the algorithm and manually explore various operations. This is useful for accumulating experience with the algorithm, and is also a prototype of an interactive approach in which a human intervention may ensure a level of non-artificial intelligence unattainable by programming means.

A useful tool is a save-to-disk feature that can be used two-fold: First it provides a means of backing up of the algorithm if it fails to achieve a significant improvement after a certain number of attempts (or depending on a temperature-type stochastic criterion). Second, it can be useful in interactive optimization for inspection of the intermediate results by a human.

The working model also offers a graphical representation of the simplex that is specific for  $D=2$ . This proves to be great for demonstrations and adds much fun and fascination to playing with MILX. Unfortunately, the tricks that work in two dimensions will not necessarily be useful in higher-dimensional problems.

### 2.6. Results

The three tables below summarize the performance of MILX in comparison with VEGAS for three classes of integrands.

Note that all three examples are of a kind where the absence of diagonal structures gives an advantage to VEGAS. With distinct diagonal structures the performance of VEGAS — unlike that of MILX — quickly deteriorates.

The comparison of MILX and VEGAS is made difficult by the different geometry of the global integration domains of the two algorithms. Therefore the calculations with VEGAS were performed in two variants: the column VEGAS-1 shows the results for the integrand obtained by a natural continuous mapping of the standard  $D$ -dimensional simplex onto the unit hypercube taking into account the Jacobian of the trans-

formation. VEGAS-2 is for a natural embedding of the simplex into hypercube with the integrand extended to the entire hypercube using the same analytical formula.

All tables present the values of the variance  $\sigma$  in %, and the dimensionality of examples is  $D = 5$ .

The number of integrand evaluations is  $N = 200K$  unless indicated otherwise.

The cases when MILX matches or exceeds VEGAS precision-wise are shown with black frames.

### 2.6.1 Integration over a spherical subregion

In this example the function takes the values 1 and 0 within and outside a sphere of radius  $R$  placed at the center of the standard simplex. The sphere may extend beyond the simplex but the integration is restricted to the latter.

Table 1. Results for the characteristic function of a sphere.

$R$ (radius )	MILX	VEGAS-1	VEGAS-2
0.2	0.39%	0.34%	0.91% ( $N=400K$ )
0.3	0.17%	0.17%	N/A
0.4	0.08%	0.11%	N/A
0.5	0.04%	0.085%	N/A

### 2.6.2 Integration of a smooth function

In this example we integrate the spherically symmetric exponential

$$\exp\left[-(\mathbf{x} - \mathbf{x}_0)^2 / R^2\right], \quad (2.1)$$

where  $\mathbf{x}_0$  is the center of the standard simplex. Note that for  $R \rightarrow \infty$  the integrand becomes 1 so that the MC error is zero and adaptation plays no role.

Table 2. Results for the spherically symmetric exponential.

$R$ (radius )	MILX	VEGAS-1	VEGAS-2
0.1	0.390%	0.160%	0.140%
0.2	0.120%	0.096%	0.093%
0.3	0.057%	0.093%	0.088%
0.4	0.031%	0.076%	0.046%
0.5	0.020%	0.075%	0.027%

### 2.6.3 Singularity at the boundary

The integrand here has the form

$$\left[ \sum_{i=1}^M x_i + \varepsilon \right]^{-1}, \quad \varepsilon = 10^{-5}. \quad (2.2)$$

The results are shown in Table 3. Note that, again, the position of singularity is chosen such that it is very easy for VEGAS to isolate it especially for larger  $M$ .

Table 3. Example with a singularity at the boundary

$M$	MILX	VEGAS-1	VEGAS-2
5	0.017%	0.064%	0.017%
4	0.050%	0.07%	0.017%
3	0.061%	0.075%	0.027%
2	0.14%	0.09%	0.053%
1	0.2%	0.085%	0.054%

### 3. Discussion

The results presented above were obtained using a working model of MILX. We call this version a working model because it evolved largely in a trial-and-error fashion and it was impossible to predict which features would prove to be important eventually: the algorithm exhibited a rather complex behavior with respect to changes of its parameters etc. Also the aims of the project as we now understand it require a much stricter programming discipline than we had been prepared for. Nevertheless, even the imperfect working model allows one to make a definite conclusion about feasibility of this approach. Our target in a near perspective is to produce a functionally complete basic MILX package. In particular, we hope to implement optimized “tactics” for integrands containing delta-peaks, for highly smooth integrands, for integrands with integrable power singularities at the boundaries of the simplex (of the sort encountered in the theory of Feynman diagrams).

Since MILX is intended to be a customizable library rather than a universal routine, each applied problem should be approached concretely, and a useful program product may emerge only after several such problems are dealt with. Therefore we consider it essential to experiment with MILX in the context of concrete real-life projects.

Success of the project depends, as we realized, on the quality of design and programming implementation, in particular, a strict hierarchical implementation of data structures etc. A systematic use of an OOP discipline seems to be essential. There might be advantages having a LISP-like engine for handling the complex data structures but this may affect portability and cause problems of interface with routines for evaluation of integrand.

It turns out that the possibility to use analytic quadratures and interpolation formulas is a very powerful option in MILX. Only a limited number of mathematical results is available here because, apparently, of the hypercubic orientation of a great majority of existing integration routines. It should be fairly straightforward to develop, say, Gauss-type formulas for simplices (unlike hypercubes). On the other hand, the powerful Korobov-type formulas<sup>4</sup> are at their best with periodic integrands, which is not easily compatible with the simplicial geometry.

### 4. Conclusions

We emphasize that none of the major technical elements of the MILX project are new. The whole, however, appears to be non-trivial enough: the first results indicate that the technique of simplicial decompositions offers advantages over the hypercubic approach in a number of cases and opens a realistic prospect for a further progress with numerical multidimensional integration in a near future — after about 15 years of undisputed reign of VEGAS. We believe to have demonstrated both the overall feasibility of the approach, and a possibility of its practical C/C++ implementation.

The applications where the algorithms of the 'MILXy Way' family may have a clear advantage over conventional algorithms appear to be as follows:

- Applications requiring a high precision unattainable via a pure Monte Carlo. This becomes possible due to flexibility of simplicial decompositions plus a natural use of analytic quadratures. In fact, the straight-forward MC integration for a fixed decomposition of the integration domain at the very last stage is less important in MILX, and the stochastic nature of the error of the final result is due to the stochastic nature of the construction.
- Applications where an economical and precise description of the geometry of integrand is needed. (VEGAS may 'see' too many spurious regions with high variance if the integrand contains diagonal structures.)
- Applications involving many similar integration problems where one can design a customized optimization strategy to make full use of available a priori information and, perhaps, a precomputed simplicial decomposition that only has to be fine-tuned for each specific integrand.
- Applications where each evaluation of the integrand is extremely costly so that a maximal use of the information already available is necessary.
- Applications where a considerable a priori information about the integrand is available that can be used to optimize the strategy of constructing the decomposition of the integration volume.

### Acknowledgements

One of us (F.T.) is grateful to M.Fukugita and G.P.Lepage for providing copies of VEGAS and related materials which started our project and for stimulating discussions, and to the FERMILAB Theory Group where a part of this work was done, for hospitality. P.E.Zhidkov pointed out to us the Korobov formulas. We thank the members of the CompHEP project for a discussion of MILX. This work was made possible in parts by the Russian Fund for Fundamental Research (grant 95-02-05794) and the International Science Foundation (grants MP9000/9300).

### References

- 
1. F. James, *Rep. Prog. Phys.*, 43 (1980) 1143.
  2. G. P. Lepage, *J. Comput. Phys.* 27, 192 (1978).
  3. D.K. Kahaner and B. Wells, *ACM Trans. Math. Software* 5 (1979) 86.
  4. N.M. Korobov, *Trigonometric sums and their applications*, NAUKA, Moscow, 1989.