4 SciDAC Software Infrastructure Project

4.1 Overview

The goal of the U.S. SciDAC software infrastructure project is to create a unified programming environment to achieve high efficiency on the multi-Terascale computer architectures targeted in this plan. By the creation of standards for communication interfaces, optimized low level algebraic kernels, optimized high level operators and other run-time functions, the valuable U.S. base of application codes can be easily ported and extended as these architectures evolve over time without wasteful or duplicative effort of the theoretical physics community. Further in light of the two architecture approach for a distributed topical center for QCD, it is essential to be able to rapidly move application codes between participating hardware platforms in order to balance the load and optimize performance system wide.

The focus of the first year of this project has been to define the overall structure of a QCD Application Programming Interface (API), and to flesh out the first 2 critical components of that API: a highly optimized communication (message passing) layer (QMP) and linear algebra (QLA) layers. The QCD API, illustrated in the block diagram below, provides the software matrix within which many of the other tasks are found. The full range of software development as define in the SciDAC proposal comprises 6 major tasks:

1. Design of QCD API and Code Libraries
2. Optimized Network Communication
3. Optimized QCD Kernels
4. Porting and Optimization of Application Codes
5. Data Management and Data Grid tools
6. Development of the Execution Environment

A more detailed breakdown into subtasks is given in Sec **??** below. Published standards and code releases are posted at the URL www.lqcd.org. Although the SciDAC software infrastructure will not be completed until the end the SciDAC grant, priorities have been set, so the critical components are ready in time to fully support the production QCDOC and cluster machines coming on line in FY03.

QCD-API Level Structure

Level 3
Dirac Operators, CG Routines, etc. (Optimized Plugins for critical sections)

QDP_xxx	Level 2
Data Parallel API: QCD Lattice Operations (overlapping Algebra and Messaging) e.g. A = SHIFT(B, mu) * C	
Lattice Wide Linear Algebra (No Communication) e.g. A = B * C	Lattice Wide Data Movement (Pure Communication) e.g. Atemp = SHIFT(A, mu-dir)

QLA_xxx	Level 1	QMP_xxx
Linear Algebra API: Single Site & Vector e.g. SU(3), Dirac algebra, ...		Message Passing API: (Maps Lattice Geometry into Network)

4.2 Integration of software and hardware development

Our plan for FY03 requires a careful co-ordination between software and hardware development. The central software components are:

1. Design, implementation and optimization of Message Passing Layer (QMP)
2. Optimization of core low level Linear Algebra routines (QLA).
3. High Level Dirac Inverters optimized for QCDOC and clusters.
4. Standardization of I/O routines and data file formats.
5. Performance milestones for core routines and full application codes.

An overview of the schedule for software development in FY03 is given in the Gantt chart in Fig **??**. The synchronization of software development tasks in FY03 with hardware deployment proceeds in two phases.

The first phase, begun Sept 01 using the QCDOC ASIC simulator and small clusters, is the optimization of level 1 code (QMP message passing and QLA linear algebra routines). These software components will be fully benchmarked, optimized and hardened first on the prototype clusters at JLab and FNAL, and then on the 128 node QCDOC prototype, with both platforms operating by June 1, 2003.

The second phase from June 1, 2003 to Dec 31, 2003 requires efficient full scale application codes to do production work on the 1.5 Tflops QCDOC development machine and the 0.25+ Tflops clusters. This necessitates that the major large scale applications designated by the Program Committee be ready to run by June 1, 2003 and that additional applications be developed as needed. In addition to doing very significant physics production in FY03 on the prototype machines, this

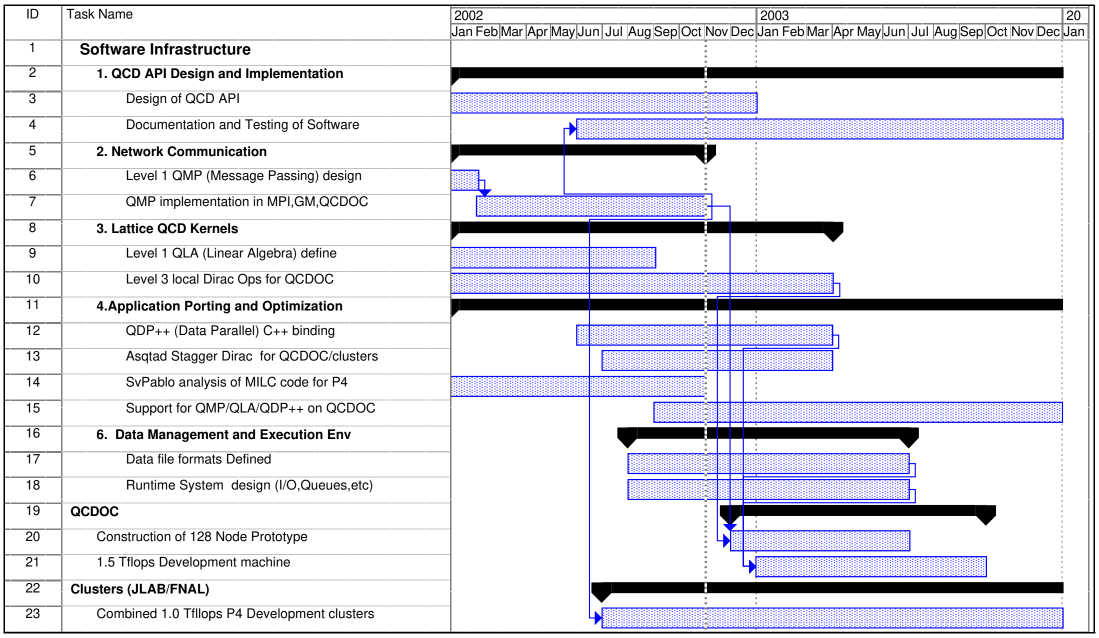

Figure 1: Overall schedule for SciDAC software in relationship to the deployment of QCDOC and cluster development machines in FY 03.

phase will allow for further optimization/testing/evaluation of SciDAC software and evaluation of the programming and execution environment needed prior to establishing in a coherent national topical center in lattice QCD with several multi-Terascale production machines.

4.2.1 Network Communication

It was clear from the beginning of the project that the message passing API (QMP) represented perhaps the single most time critical component. This layer defines a uniform subset of MPI-like functions equivalent to those used in existing QCD application code. In addition QMP extends this core set of MPI functions in two areas: (i) QMP partitions the QCD space-time lattice and maps it into the geometry of the hardware network providing a more convenient abstraction for the level 2 data parallel API (QDP); (ii) specialized routines are designed to access the full hardware capabilities of the QCDOC network and to aid optimization of low level protocols on networks in use and under development on the dedicated commodity clusters. The initial stage for the documentation and implementation of the message passing layer (QMP) has been largely the responsibility of Watson, Edwards and Chen at JLab in collaboration with Mawhinney and Chulwoo Jung at Columbia/BNL assuming the task for the native QCDOC implementation. The QMP definition, documentation and code are on the project website *http:://www.lqcd.org*. The released code includes a pure MPI implementation for portable code development, optimized implementation for Myrinet (in native GM code) and a native implementation for QCDOC tested on the ASIC simulator. Note that the design, documentation, implementation and testing procedure for QMP has set the framework and methodology for all the code modules that follow.

By itself, this message passing infrastructure (QMP) does permit one to quickly port C or C++ application codes to compile and run on the QCDOC and the clusters. Once these codes are linked to optimized level 3 Dirac inverters, reasonably efficient physics production becomes available. However, there are several additional tasks needed in this area during FY03. On the QCDOC, optimized routines for non-nearest neighbor communication are still needed for the non-local Dirac operator, Asqtad. In the future we anticipate a range of non-local operators which will employ these extended communication kernels, including the design of an efficient FFT. A test code suite for QMP is being designed by Mendes at Illinois and Simone at FNAL for all implementations, present and future. As described in the Commodity Cluster Sec 3.4 and 3.5, SciDAC software FTE's will also work on implementations of QMP over gigE and the FPGA NIC networks once they have been proven to be a more cost effective substitute for Myrinet.

4.2.2 Optimization of Linear Algebra

The linear algebra routines (QLA) and their optimization on the P4 are primarily the responsibility of DeTar, Pochinsky, Holmgren and Osborn. The "streaming SIMD" features of Pentium processors, known as the SSE and SSE2 instructions, are essential for high performance. By coding the fundamental QLA routines using SEE instructions on a 1.7 GHz Pentium 4, the single node performance in single precision for the Wilson Dirac operator in cache exceeds 2.5 Gigaflops. The full library for QLA routines is being generated in C using software tools in Scheme and Perl. Upon completion of this task by Jan 1, 2003, the optimized SSE library will be replace core routines in QLA to enhance the performance on the Pentium 4 clusters. Implementation of the QLA routines on QCDOC is under development by Mawhinney, Jung and Petrov with special emphasis on the optimizing of the core routines. Analysis of the impact on performance and practical pursuit of higher effeciency at this level is a major task leading up to the start of the prototype machine in June 1, 2003.

The Scientific Program Committee has identified three quark actions as being essential for early work on Terascale computers: Wilson clover, domain wall, and improved staggered (Asqtad). The first two of these have already been hand coded for QCDOC, as has the conventional staggered quark action. These Dirac inverters are also being optimized for clusters as part of the code development for production code at JLab and FNAL. The Asqtad action, which is a less local version of the conventional staggered action, is also required. Asqtad is being implemented by a joint effort of SciDAC developers in Columbia and Utah. Gregory at Utah has adapted the MILC Astad operator to use the QMP routines and Chulwoo at Columbia is running this code on the QCDOC ASIC simulator. At the same time the Asqtad fermion force term is being written by Gregory in the C implementation of QDP. Once this stage is completed, both the Dirac operator and the force term will be optimized as a level 3 QCDOC module callable from any C or C++ application code. The optimization of the Asqtad inverter is a priority which will be completed by the end of the first quarter of 2003 in time to be tested on the prototype QCDOC and to run production code on the QCDOC development machine by June 2003.

This Asqtad implementation is a typical example of the collaborative process between experts in low level code optimization and the general lattice QCD community, which the SciDAC software project seeks to establish as the norm. Clearly there will be an ongoing need to develop optimized plug-in (level 3) routines in order to adapt quickly to new Fermionic and Gauge actions for QCD and similar quantum field theories that are needed to go beyond the standard model. Thus a critical design objective of the software infrastructure project is to combine rapid prototyping in the convenient QDP (data parallel) interface with the subsequent coding of assembly language modules for the QCDOC, cluster architectures and their successors when the new algorithm is required for state of the art large scale production code and when it is shown to constitute a critical section of the overall application. However it should be noted that the overwhelming majority of application code (by line count) can remain in QDP (or even legacy C or C++) without substantial impact on through put of the simulation.

4.2.3 Data Parallel Framework and Porting Application Codes

QCD applications have the fortunate characteristic that they have a natural framework well suited to a data parallel interface, and a small number of linear algebra operations that must be optimized to obtain excellent performance, the inversion of the Dirac operator being the most notable. Consequently it is possible to write the vast majority of application code (by line count) in a uniform data parallel language. The SciDAC software project has defined a full data parallel QCD interface (QDP) to promote a community wide standard for code development. The C and C++ implementation for the level 2 QDP interface will be available in a serial implementation by Jan 1, 2003 so that software developers can use it to port legacy or write new application codes. A major task for software development starting in Jan 2003 is to produce a robust parallel implementation for QDP by June 2003 so that code written to this interface will run on all SciDAC prototype hardware.

The design objective for QDP is to enable new applications to be developed rapidly and to run with efficiency beyond the reach of generic C or C++ compiled code. Edwards, Mawhinney, Petrov and DiPierro are implementing QDP in C++, whereas DeTar and Osborn are implementing QDP in C. Both implementations conform to a single API and pains are taken to build them on a common set of lower level (level 1) fundamental C functions. Much of the underlying development and optimization can be shared between the C and C++ variants. It is expected that in most instances legacy code will be migrated to the QDP only on a case by case basis but new application code will adhere to the full API.

Obviously the principle responsibility for porting application code to the API interface largely falls to the authors of the particular code base: SZIN (Edwards, Richards, Pochinsky), MILC (DeTar, Osborn, Toussaint, Gregory) and CPS (Mawhanney and co-workers). However, the Software committee acts to review progress and to promote a uniform methodology and shared optimization strategies to minimize duplication of effort. Harmut Neff, who begins as a SciDAC postdoctoral fellow in Nov 1, 2002 at BU, will focus much of his effort on providing developers with example code and documentation to support this goal.

4.2.4 Data Management

Task 5 also has essential elements that must be defined before 2003. The data file formats and headers will be defined in the last quarter of 02 by adding I/O routine to the Level 2 API standards so that all application codes can share data files. This is a very active task at present for the Software Co-ordinating committee with separate teams assigned to address the programming interface in QDP (Edwards, Pochinsky, Brower), File Formats (Mawhinney, DeTar) and Web access and archiving (Watson, Holmgren). Simone and Edwards are representing the SciDAC project in order to co-ordinating with Richard Kenway in the UK data archiving project.

This project is also studying the use of XML, both as a tool to control data file I/0, and to locate relevant data using web based search methods employing XQL. Subsequently there will be a gradual evolution to a lattice portal with multi-site replicated data for security and rapid retrieval. These tools should allow a natural exploration of web based computing, including batch systems and uniform scripting tools. Chip Watson at JLAB is an active member of the SciDAC Particle Physics Data Grid project, working with others to develop and deploy advanced data grid software. In addition SciDAC is actively participating in the proposed ILDG (International Lattice Data Grid). These connections will enable us to deploy, rather than re-invent, grid tools as they become available. Obviously this ambitious program needs to be carefully considered and prioritized, so that practical extensions are brought on line without losing focus on the immediate needs of the physics program. This will become a higher priority in FY04 to further facilitate operating the combined computing resources at BNL, JLab and FNAL as a single topical center for QCD lattice research. Such a distributed meta-laboratory is clearly an ambitious agenda but the relatively narrow focus and maturity of the QCD computational program make it an ideal testbed for more general grid based computing facilities.

4.2.5 Execution Environment

The Execution Environment is a another important area of software development for this project. This category of software includes a number of application modules beyond those previous listed, as well as the batch and grid environments in which the applications run.

The application run time environment includes error detection and reporting, and standards for application (data) check pointing procedures. Because it is required that each platform gives bitwise reproducible results, global sums and random number generators will be rigorously deterministic on all networks.

Standards for the batch environment will allow not only applications, but also production batch scripts to be moved quickly between the different platforms. This implies a certain amount of standardization of file system naming conventions (or discovery), and standardization of batch submission commands and options. Declaration of input and output dataset names will eventually use global file names, names which are independent of the site at which they are located.

Uniform file format standards and compatible batch environments are precursors to the larger goal of putting in place a web based (or data grid) system to allow the three laboratories to be viewed as a single topical national facility. This has substantial advantages. It allows both load balancing and optimization by directing jobs to the architecture on which they will obtain the best performance per dollar expenditure.

4.2.6 Summary of Software Infrastructure subtasks through FY03

Below we give a summary of the progress to date on the software tasks and projected completion dates through FY03 — “on going” refers to subtasks active through out FY03 and continuing as part of software development during the life cycle of each platform. Each sub task is assign to a team and review by the entire Software Co-ordinating committee in weekly teleconference meetings and quarterly workshops. A chronology of the minutes of all meetings and working documents can be accessed through the web page *http://physics.bu.edu/ brower* whereas released standards and code are publish on *http://lqcd.org*.

Task/Subtasks	• Start Date	• Final Date
1. QCD API and Code Library:		
1.1 Overall design of QCD API	• July 01	• Dec 02
1.2 Compliance, Testing and Benchmarking in Application Codes	• Jan 03	• on going
1.3 Code Libraries and Documentation	• June 02	• on going
1.4 Data Analysis Tools defined and code archive	• Jan 03	• on going
2. Network Communication:		
2.1 Level 1 QMP (Message Passing) design (done)	• July 01	• Feb 02
2.2 QMP implemented on top of MPI, GM, QCDOC (done)	• Feb 02	• May 02
2.3 Fat SMP extensions using threads	• (when required)	
2.4 Alternate Networks (e.g. GigE/ FPGA-NIC/etc)	• May 02	• on going
3. Lattice QCD Kernels:		
3.1 Level 1 QLA (Linear Algebra) fully defined	• Sept 01	• Sept 02
3.2 QLA library issued in C with SSE opt for P4	• July 01	• Nov 02
3.3 Level 3 Local Dirac Operators for QCDOC/clusters (done)	• July 01	• June 02
3.4 Asqtad Staggered Dirac & force terms for QCDOC/clusters	• July 02	• April 03
4. Application Porting and Optimization		
4.1 C binding for QLA & QDP	• Sept 02	• Jan 03
4.2 QDP++ (Data Parallel binding to C++) for QCDOC/clusters	• June 02	• Jan 03
4.3 Parallel Support for QDP on QCDOC/clusters	• Jan 03	• June 03
4.4 Analysis of MILC code using SvPablo for P4	• July 01	• Jan 03
4.5 Optimization of Application Code on QCDOC/clusters	• Sept 02	• on going
5. Data Management and Data Grid		
5.1 Data File Formats Defined	• Aug 02	• Dec 02
5.2 Grid Work (w. International Lattice Data Grid (ILDG))	• July 01	• on going
6. Execution Environment		
6.1 Run-time environment: I/O, Error detection standards	• Aug 02	• June 03
6.2 Define standards for Batch Environment	• Aug 02	• June 03
6.3 BIOS , Chipsets, Network booting, Machine configuration,..	• April 02	• ongoing

The SciDAC grant in FY03 supports QCD software infrastructure at the level of 9 FTE. These resources are complemented by roughly an equal number of physicists and software engineers with no direct SciDAC support. Thus the size of the group actively participating in the SciDAC software infrastructure work is roughly 20 individuals. The seven member software co-ordinating committee meets weekly in conference calls to develop the overall design strategy. Subgroups take particular responsibility for separate tasks with intermittent participation of others to insure the unity of the overall design.

The software co-ordinating committee sets the deadline for deliverables to insure that critical components are available in time to keep the overall software and hardware project on track. On occasion this means reassigning effort in "burst mode" to avoid unanticipated delays on the critical path. This *modus operandi* has worked very well to date and we are confident that we have the means to deliver the software infrastructure as set out in the SciDAC grant.

4.3 Tests of Software Development and Final Evaluation

Our project follows an aggressive schedule of development and construction, which will take advantage of a unique window of opportunity to bring the U.S. lattice community back to the forefront of worldwide research. In order to reap its full benefits, it will be crucial to insure that the development of application specific software, as described in the previous section, meshes well with the development of hardware. Once the construction of the 1.5Tflops QCDOC machine and of the prototype clusters have been finished, it will also be important to run extensive tests of complete application packages, to validate the suitability of such special purpose computers for lattice QCD calculations and lay the ground for the construction of dedicated 10Tflops machines, as envisaged by our long range plan. For these purposes, we have devised a set of tests which will be conducted both during the development of our project and at its end, and which we will refer to as "software development tests" and "final milestones". The goals of such tests will be quite different. The software development tests will be conducted in parallel with the construction of the QCDOC prototype and development machine, indeed some are already being run on the simulators, and of the clusters. They will serve to make sure that the software development is on course, providing diagnostics for any aspect of it which may require remedies or extra efforts, so that full fledged applications may be run as soon as the development QCDOC and clusters are completed. The final milestones will consist in running complete QCD application codes and will serve to validate the efficiency and flexibility of the architectures, as well as their ability to sustain the efforts of our diversified lattice community.

In making these tests we will take advantage of the existence of three large, well established code packages, which are freely available to the entire U.S. lattice gauge theory community: the Columbia Physics System (CPS), the MILC code and the SZIN code. Each of these codes runs on single processor workstations. The MILC and SZIN codes have been validated and have achieved excellent performance on a wide variety of massively parallel computers, while the CPS has been validated and achieved excellent performance on the QCDSP, the predecessor of the QCDOC. Thus, we are in a position to test the correctness of new hardware at both the single and multiprocessor level, and to obtain a broad set of comparative benchmarks. The three codes include among them nearly all applications and algorithms in common use by lattice gauge theorists. We are thus convinced that their performance will be indicative of that which it will be possible to obtain on a wide range of lattice QCD applications with C or C++ codes conforming to the QCD API.

The ideal metric for judging hardware would be dollars per unit of science, but that cannot be measured directly. For example, several different actions are used to describe quarks on the lattice, and more are under development. The choice among them often depends upon the problem under consideration. Quark actions differ widely in the total number of floating point operations required to invert the Dirac operator, the most computationally intensive part of any lattice gauge theory calculation. They also differ in performance, as measured in Mflops per processor because of differences in the balance between floating point operations and data movement. Nevertheless, cost per sustained Mflops is a very important measure, once such differences are taken into account. The size of problems to which a platform can be applied, and the human resources required to create or modify code are also important considerations. The former is crucial for the large projects which are at the heart of our research, and the latter for the development of new algorithms and calculational methods, which is also a crucial component of our work. It is possible that no single architecture is optimal by itself for the entire range of jobs we envision. Thus, judgments regarding the suitability of hardware for our research must be based upon a spectrum of considerations. Final decisions will be made by the Executive Committee with advice from the Software, Scientific Program and Oversight Committees.

4.3.1 Software Development Tests

These tests will be done throughout the design and construction of the QCDOC prototype, the 1.5 Tflops QCDOC development machine, and the prototype clusters. As mentioned above, they will serve to insure that the software under development under the auspices of our SciDAC funded project is adequate. They will also serve to provide additional diagnostics for the development of machine specific software, such as the run time operating systems. The tests will consist in the following:

- Using C or C++ compilers and the QMP communications libraries, compile complete production codes for the three quark actions specified by the Scientific Program Committee: clover Wilson, improved staggered and domain wall quarks. The CPS, MILC and SZIN source codes will be used for these tests. The compiled codes should correctly evolve full QCD systems using hybrid Monte Carlo and molecular dynamics algorithms, including standard measurements such as hadron propagators, yielding answers in agreement with code on conventional hardware platforms. Tests should be done on lattice sizes per processor such that, when scaled up to the anticipated full machine or usable subsets of the full machine, they span the range of lattice sizes needed for both high temperature simulations (small lattices of size $18^3 \times 6$) and for studies of pion cloud phenomenology at zero temperature (large lattices of size $32^3 \times 96$). These tests will be done, to the extent possible, on the QCDOC simulator, to validate the correctness of the results, and to obtain estimates of the performance on the actual machine. They will be repeated on the machine components, as they become available (single node, motherboard, partial or full prototype machine). The target is to achieve at least 25% of the designed Megaflops/dollar for the updating part of the above codes on lattices with 8^4 or fewer lattice sites per processor, optimizing only at the C or C++ level. (Targeted performance figures are for Wilson based quark actions. Actions based on staggered quarks are expected to run approximately 30% slower). The ability to read and write lattices to disk should be demonstrated.

- The second test is to prepare and run (Level 3) optimized inverters for the three quark actions specified by the Scientific Program Committee. The inverters should be compatible with the corresponding suites of production codes, although it is permissible to reformat the data before and after the inversion. All of the inverters should achieve a performance of at least 80% of the designed Megaflops/dollar with the allowance for the difference between Wilson and staggered based quark actions noted above. The performance measurement must include communication and global sum overheads. These tests may be performed on lattices of limited size, but the scaling must be well enough understood to convince the Executive and Oversight Committees that scaling to a sustained Teraflops is achievable.

4.3.2 Final Milestones

The 1.5 Tflops QCDOC and 0.25 Tflops clusters must pass a rigorous set of milestones prior to the the deployment of large terascale platforms at the BNL, JLab or FNAL. These milestones will be used to test the ability of the hardware to advance our research, and to determine the appropriate mix of hardware for the national center for lattice gauge theory. The details of these milestones will be set out by the Executive Committee in consultation with the Program, Software and Oversight Committees. Their general form will be as follows:

- The milestones must demonstrate that both hardware and software are ready for multi-terascale deployment. Entire applications in the CPS, MILC, and SZIN codes, covering the full range of our research program, will be run and benchmarked. Tests will compare the performance of specially optimized (Level 3) inverters described in the previous subsection with that of inverters coded in pure C/C++ on top of optimized Level 1 routines, in the full data parallel API (Level 2). The maturity and stability of the runtime environment for data management, job queues, remote access, etc. will be evaluated. Statistics will be compiled for sustained production runs over a period of at least one quarter to measure overall throughput. The entire user community will be polled to ascertain the experience of users not intimately involved in either the hardware or software design.

- A measure will be made of the overall development environment for new and exploratory software that inevitably makes up a substantial fraction of lattice QCD applications in a research environment. The ongoing cost in terms of time and effort for the optimization of the critical components of production codes will be included in this assessment. The goal is to be sure that the computational environment allows for full access of the U.S. lattice gauge theory community of some 200 members, and that it fosters innovation in algorithms and new applications to particle physics phenomenology which are essential for progress.