SCREMS: The Frontiers of Representation Theory, Number Theory, and Mathematical Physics

Major Highlights

- **Representation Theory:** We will compute the *Kazhdan-Lusztig-Vogan polynomials* for all simple Lie groups up to rank 9 and make the results of these computations readily available to researchers worldwide. We will also continue to explore the combinatorial infrastructure of W-graphs and relate it to representation theoretical invariants.
- **Number Theory:** We will carry out major computations of modular forms and L-functions, and greatly enhance our understanding of the *Birch and Swinnerton-Dyer conjecture and the Riemann Hypothesis*, two of the central problems in number theory.
- **Mathematical Physics:** We will complete the first major step in the *classification of off-shell representations of Supersymmetry*, a problem which has remained open for three decades.
- **Broader Impact:** We will produce new algorithms, data, web pages, and free open source tools (such as Sage) for collaboration and data sharing in mathematical research.
- **Unique Hardware:** The two 128GB 16-core servers that we are requesting fill a unique need that can't be addressed any other way. They would update hardware purchased in 2005 using NSF Grant DMS-0555776 that has had a broad impact on mathematics.
- This grant would fund a talented undergraduate, who would administer the system and learn substantial mathematics through their involvement with the project.

1 Introduction

In 2005 Stein used \$34K of funds from his personal ANTC NSF grant DMS-0555776 to purchase a computer with sixteen 1.8 GHz processing cores and 64 GB of RAM. This computer has opened up a whole world of possibilities—the E_8 computation mentioned below alone has spawned an immense amount of interest, in addition to all the other exciting work that has already been done on this computer.

- This computer is the so-called "Sage supercomputer" that was used by the Atlas for Lie Groups FRG (DMS 0554278) to compute the character table of E_8, which made a huge splash in the international media in the Spring of 2007. In fact, Representative Jerry McNerney (D. CA) addressed Congress to describe this collaborative work as a shining example of NSF spending money wisely.
- This computer is the home for Sage (see Section 3), which is an open source mathematical software development project. About 100 Sage developers have accounts on this machine. Having this computer as a collaborative meeting place for work, sharing of partial results,

data, testing, etc., has had a tremendous positive influence on the project. Moreover, the large memory and multiprocessor nature of the machine has been critical in pushing algorithms to the limit and ensuring that Sage is capable of using parallel computing resources.

- This computer has been the main computation server for prestigous NSF-funded workshops and conferences, e.g., the Arizona Winter School (which Stein co-organizes).
- This computer has played a central role in Jim Morrow's successful summer REU program at University of Washington, which Stein now helps organize.

We are requesting two identical servers, each having 16 very fast 3 GHz processing cores and 128 GB RAM, and 1.5 terabytes of fast hard disks. We will use these machines to carry out major computational research projects involving Lie groups, supersymmetry, and number theory. Nothing comparable to this advanced hardware is currently available to us, even at supercomputing centers. This hardware addresses major needs we have that are not addressed by available university-wide or national supercomputing equipment:

1. The requested servers have much more RAM than individual nodes on supercomputers we use. Large RAM is critical for the computations described in this proposal.
2. These computers will have a huge amount of fast disk space, which will be made immediately available over the Internet; this makes collaboration and sharing our results much easier. Moreover, output of computations of data about L-functions, modular forms, representations, etc., is sometimes huge. For example, David Vogan describes one of our past calculations as follows:

   > He managed to gain access to a machine [at Cornell] with 128GB of RAM and 128GB of swap space. He used it to run the E_8 computation to the end. [...] Unfortunately he had no reasonable way to write the results to disk, so they disappeared.
   >
   > — David Vogan [Vo3]

3. Each machine will have 16 very fast processing cores, which will make them highly suitable for interactive work. Such interactive exploration is of immense importance when formulating mathematical conjectures, e.g., Stein's forthcoming joint work with Mazur to refine the Sato–Tate conjecture. Also, multiple fast processors may be the difference between success and failure in the Supersymmetry classification (see Section 2.3).
4. Because each machine is fast, this hardware will also make it possible to run extremely powerful web services, built on the unique new technology that has come out of the Sage project (see Section 3).

1.1 Prior Support and Related Proposals

Stein received NSF grant DMS-0653968 from the ANTC program to support his personal research for 2007–2010 on the Birch and Swinnerton-Dyer conjecture. Stein also received NSF grant DMS-0703583 from the COMPMATH program to support one postdoc for three years, who will work on developing linear algebra algorithms and implementations for Sage; his work will be important for some of the student projects, and he will also serve as a mentor. Stein has also applied for an NSF FRG grant jointly with Booker, Elkies, Conrey, Rubinstein, and Sarnak to provide more postdoctoral and student support for a related project that involves invariants of modular forms and L-functions.

The co-PI Doran submitted in November 2007 a proposal to NSF on "Geometry, Periods, and Moduli of Calabi-Yau Manifolds". This proposal would provide summer support for two graduate students, Ursula Whitcher and Jacob Lewis, who would make use of the proposed hardware in their research at the interface of geometry, physics, and number theory.

Stein and Doran have also submitted an NSF proposal in October 2007 to the CSUMS program entitled "Undergraduate Computational Research in Arithmetic Geometry", which would support a cadre of 6–8 undergraduates who would work on research that would use the computational resources that we are requesting in this proposal.

Rubinstein is currently partly supported by an NSERC Discovery Grant at University of Waterloo.

The purpose of the present proposal is to complement the above NSF-funded research projects with an extensive computational presence. This will also have a significant positive impact on the training of students in the use of serious computational techniques in mathematical research.

2 Proposed Research

2.1 The Representation Theory of Lie Groups Project

A *Lie group* is a group G endowed with the structure of a differentiable manifold such that both group multiplication and group inversion are smooth maps. The additive group $\mathbb{R}$, the group $O(n)$ of rotations in an n-dimensional Euclidean space, the group $\mathrm{GL}(n,\mathbb{R})$ of invertible $n \times n$ real matrices, and the Lorentz group of special relativity are all examples of Lie groups. A *representation* of a Lie group is a pair (π, V) consisting of a complex vector space V and a continuous homomorphism

$$\pi : G \to \mathrm{Aut}(V).$$

A *unitary representation* of a Lie group G is a representation $(\pi, \mathcal{H})$ where $\mathcal{H}$ is a complex Hilbert space and the homomorphisms $\pi(g)$, $g \in G$, preserve the inner product of $\mathcal{H}$. A representation (π, V) is *irreducible* if the vector space V contains no proper closed G-invariant subspace. The *unitary dual* $\widehat{G}_u$ of G is the set of all (of all equivalence classes) of irreducible unitary representations of G.

Since the beginning of the last century it has been a fundamental problem to determine the unitary dual $\widehat{G}_u$ for an arbitrary real Lie group G ([W],[Ki], [AK], [Vo1]). There are

two distinct, yet equally fundamental, motivations for this problem. First of all, the unitary irreducible representations of a Lie group G constitute the basic building blocks of noncommutative harmonic analysis, the natural 20th century successor to classical Fourier analysis. Secondly, by a mere change of interpretation, a unitary irreducible representation $(\pi, \mathcal{H})$ corresponds to an elementary quantum mechanical system carrying an action of a continuous symmetry group G. Indeed, the parameterization of the irreducible unitary representations of the symmetry group of space-time achieved by Wigner [Wi] in the late 1930's was simultaneously the classification of all possible elementary particles in Minkowski space. During the last century, the unitary dual problem was solved for several important general cases; and in fact, by the 1970's, the outstanding part of the general case had been reduced to the case of real reductive Lie groups. However, to this day, it remains an open problem to parameterize the set of irreducible unitary representations of a real reductive Lie group.

On the other hand, it is also known that for any fixed real reductive group G, the problem of determining $\widehat{G}_u$ is in principle achievable by a finite computation. Moreover, there is a method of reduction that offers some hope for a completely general solution via a finite computation. The simple Lie groups (the basic building blocks of reductive groups) naturally break up into two families: the real forms of the *classical* Lie groups and the real forms of five *exceptional* Lie groups. The classical Lie groups, $\mathrm{SL}(n)$, $\mathrm{SO}(n)$ and $\mathrm{Sp}(2n)$, arising as they do as groups of linear transformations preserving certain invariant bilinear forms, are very similar in character and lend themselves, quite naturally, to uniform, inductive methods of analysis. On the other hand, the exceptional Lie groups, virtually by tautology, are not amenable to uniform, inductive methodologies. Thus, a basic plan of attack for reductive groups has been to settle the unitary dual question for the exceptional groups via a (finite) case-by-case analysis, and then to try to finish off the rest of the (remaining classical) groups with uniform inductive arguments. The problem, however, was that the finite computation involved was known only in principle, and then only to a small cadre of experts in the field. Moreover, it was not even clear that the necessary computations could be carried out within the confines of present day computer technology.

Nevertheless, in 2002, Jeff Adams (U. Maryland) began to organize a team of representation theorists, combinatorists, and computer scientists for a direct computational assault on the unitary dual of at least the simple Lie groups of rank at most 8. Achieving this goal would not only settle the unitary dual problem completely for the real forms of the exceptional groups, it would also provide an ample supply of examples of the classical groups from which the inductive arguments leading to general cases might be developed.

By 2006 the Atlas team had grown to a group of nineteen specialists and its principal programmer, Fokko du Cloux, had written software that could deftly handle the many arcane subtleties of the structure theory of the reductive Lie groups. In particular, the software could in principle compute the Kazhdan-Lusztig-Vogan (KLV) polynomials ([LV]) of any reductive algebraic group. However, it is here that the Atlas group also met a frontier of computability. For the computation of the KLV polynomials for the largest exceptional group, E_8, was huge and perhaps computationally inaccessible. Indeed, the basic object to be computed amounted to a 453060×453060 matrix whose entries were polynomials with integer coefficients of degree ≤ 31; potentially (using 4-byte integers) 2.6×10^4 gigabytes worth of data. Nevertheless, by

employing a series of compression and reduction techniques the Atlas team computed the KLV polynomials for E_8 using a mere 64 GB of RAM on the Sage computer at the University of Washington (see [Vo3]). This result was widely heralded as a major computational *tour de force*.

2.1.1 Open problems and technological needs

Further KLV computations

Mathematically, the E_8 computation was important because E_8 is the largest of the exceptional Lie groups, and so the "hardest nut to crack" in the direct computational approach to the unitary dual problem. For the classical Lie groups, there is no such largest group; but, as remarked above, it is expected that uniform, inductive methodologies will provide the best approach to the classical cases. Nevertheless, in order to develop, check, and initialize inductive methods, it will be crucial to have an explicit computational grasp of as many low lying cases as possible.

We also point out that interest in KLV polynomials is by no means limited to the Atlas project. The original Kazhdan-Lusztig polynomials [KL1] were actually defined for Coxeter groups, and so, in particular, for the Weyl groups of complex simple Lie algebras. These KL polynomials have applications and interpretations far outside pure representation theory. For example, the KL polynomials for the symmetric group $\mathfrak{S}_n$ are used to compute the intersection cohomology of Schubert varieties [KL2]. Yet while distinct from the KLV polynomials used in the Atlas program, KL polynomials can nevertheless be computed using the Atlas software. Suppose W is the Weyl group of a simple complex Lie algebra $\mathfrak{g}$. It turns out that if one regards $\mathfrak{g}$ as a real Lie algebra (by restriction of scalars), then the KLV polynomials for $\mathfrak{g}$ coincide with the KL polynomials for W. In this manner we have computed the KL polynomials for $\mathfrak{S}_{10}$ (a finite group of order 3,628,800) by computing the KLV polynomials for $\mathrm{SL}(10,\mathbb{C})$. Thus, independently of the goals of the Atlas project, it would be of great interest to compute the KLV polynomials for as many simple complex Lie groups as possible.

The computational servers we request in this proposal would permit us to compute the KLV polynomials for all the real and complex simple classical Lie groups at least up to rank 9.

Analysis of Atlas output

The Atlas software actually produces more than KLV polynomials. For example, as a by-product of the KLV computation, the software can also compute the full W-graph of a translation family of irreducible admissible representations. This W-graph is a weighted directed graph whose vertices correspond to irreducible representations of a fixed infinitesimal character and whose vertex weights correspond to the τ-invariant of the corresponding irreducible representations. An edge from a vertex x to a vertex y with multiplicity m indicates that the representation y occurs with multiplicity m in the Jordan-Hölder series for the tensor product $x \otimes \mathfrak{g}$, where $\mathfrak{g}$ is the adjoint representation.

In recent months we have made great strides in extracting from the W-graphs important representation theoretical invariants of the corresponding representations. For example, by grouping together the representations which can be connected in both directions by directed

paths in the W-graph, one obtains a partitioning of the set of irreducible admissible representations into *cells*. It is fairly easy to see by formal considerations alone that the annihilators of the representations in the same cell share the same associated variety. But it also turns out that the collection of all the tau invariants of the representations that occur in that cell allow us to determine that associated variety completely. But what is most striking about this result is the use of combinatorial data associated to group representations into cells in order to determine a common geometric invariant.

We stress that to make a discovery like this it was absolutely necessary to first see the W-graph data *in toto* and that that was only possible by employing computers to do the actual looking.

Here is another example of computer-enabled discovery. Since the 1980's tau invariants ([BJ], [D], [Vo2], [G]) have served as fundamental invariants in the theory of the primitive ideals of the universal enveloping algebra of a Lie algebra $\mathfrak{g}$. However, tau invariants by themselves fail to separate primitive ideals. Yet the Atlas software tells us not only the tau invariant attached to (the annihilator of) an irreducible representation but also the tau invariants of its nearest neighbors in the W-graph. This led us to introduce the notion of an nth-order tau invariant of a representation x as the set of tau invariants of the vertices within n-steps of π in the W-graph. Explicit computations then showed that these higher order tau invariants actually separate the set of irreducible admissible representations of regular integral infinitesimal character into equivalence classes with the same primitive ideal.

The development and exploitation of such combinatorial invariants of the W-graphs is an entirely new frontier for representation theory. Indeed, arising as they do as particularly intricate and symmetrical digraphs, the W-graphs produced by the Atlas software promise to be fertile ground for discovery in pure combinatorics as well. However, these frontiers are completely inaccessible without computers with a huge amount of RAM, ample storage and rapid I/O. The machines proposed with direct access to the Atlas data would be an indispensable tool for this purpose.

Storage and Dissemination of Results

The E_8 data surely comprises one of the most intricate and complex combinatorial structures ever discovered. As such, it and the results of other KLV computations should be made readily available not only to other representation theorists, but also combinatorists, algebraic geometers, computer scientists and others.

However, just to store the "answer" to the E_8 KLV computation (in fact, a highly compressed version of the "answer") requires 60 GB of disk space. Moreover, even on a machine with a huge amount of RAM, it still requires a high degree of programming skill in order to tweak the code so that the computation runs within the available RAM in a reasonable amount of time. For example, Noam Elkies suggested one of the key ideas used to squeeze the E_8 computation onto the Sage computer, which was to carry out the computations modulo 256, 255 and 253, then use the Chinese remainder theorem to reconstruct the answer over $\mathbb{Z}$. Thus, lacking exceptional computers, storage space, and expertise, for many researchers the results of the Atlas computations are quite inaccessible.

We remark also that sometimes what might be of most interest to an outside researcher is not the entire set of KLV data but rather the answer to a simple question about that data. For

example, we have been asked by algebraic geometers if it ever happened that the coefficient of the highest possible degree of a KL polynomial is greater than one? One of the purposes of the proposed machines would be to provide an online front end to the Atlas data, where such questions might be posed and answered. In this regard, we note that the University of Washington, where the proposed machines are to reside, is an ideal location for such a data storage and retrieval center since the University of Washington is directly connected to both the Abeline and National Lambda Rail internet backbones.

2.2 Number Theory: The Modular Forms and L-functions Database

Modular forms, L-functions, and Galois representations underlie much of twentieth century number theory and are connected to the practical applications of number theory in physics, geometry, and cryptography.

We propose to carry out several major computations of modular forms and their associated L-functions that go far beyond anything that has been computed before. The projects described below would move forward the theory of modular forms, and improve fundamental algorithms in linear algebra and parallel computation that are likely to have an impact outside of number theory. For example, we intend to compute extensive data about all weight 2 newforms on $\Gamma_0(N)$ for all levels $N \leq 10{,}000$. So far the best that has ever been done is to compute most forms up to level 7,000, and even then the PI (Stein) computed *very little* about each form, and the data is not in an easy-to-use form. Computing data about a modular forms means (a) computing explicit data that uniquely determines the modular form, (b) computing a large number of coefficients of the q-expansion of the modular form, (c) computing special values and zeros of L-functions associated to the modular form, (d) computing information about Galois representations attached to the modular form, and (e) computing Birch and Swinnerton-Dyer invariants of the corresponding abelian variety or motive.

The PI's work so far has led to much beautiful synergy between explicit computation, conjecture, and theory. An exciting example of this is Emerton's deep proof [Eme03] of Stein's refined Eisenstein conjecture. This conjecture was made by the PI based on exactly the above computations for all prime levels up to 767. Also, Stein's joint work with Barry Mazur and others on ranks of elliptic curves [BMSW07], the Sato-Tate conjecture, and Shafarevich-Tate groups [DWS03] was all firmly motivated by such calculations.

Pulling this project off will require doing cutting edge research on asymptotically fast linear algebra algorithms in order to greatly speed up code the PI has already written, then implementing, running, and tuning the resulting algorithms. Fortunately the PI (Stein) has extensive experience working on exactly these sorts of computation for over a decade (and has written much relevant software), and he just hired a postdoc who is likely the world's top expert on applying exactly linear algebra to challenge problems in mathematical research.

Below we describe general background on modular forms and L-functions, then enumerate several very specific important computations that we intend to do using the two proposed servers, which were configured optimally for solving difficult exact linear algebra problems.

2.2.1 General Background on Modular Forms and L-Functions

A *modular form* of integer level N and integer weight k is a holomorphic function $f(z)$ on the complex upper half plane $\mathfrak{h}$ such that for all integer 2×2 matrices $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ with determinant 1 and $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \equiv \left(\begin{smallmatrix} 1 & * \\ 0 & 1 \end{smallmatrix}\right) \pmod{N}$ we have

$$f\left(\frac{az+b}{cz+d}\right) = (cz+d)^k f(z),$$

along with certain technical convergence conditions "at the cusps". The closely related L-functions, such as the famous Riemann Zeta function

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \text{ prime}} \frac{1}{1-p^{-s}},$$

are certain types of holomorphic functions on a subset of the complex plane attached to arithmetic data. A famous example of the relationship between modular forms and L-functions is Andrew Wiles's proof of Fermat's Last Theorem, which involves proving that certain L-functions attached to elliptic curves arise from weight 2 modular forms.

The Delta function

$$\Delta(q) = q\left(\prod_{n=1}^{\infty}(1-q^n)\right)^{24} = \sum_{n=1}^{\infty} \tau(n) q^n,$$

is a weight 12 modular form, where $q = e^{2\pi i z}$. Ramanujan conjectured that for all primes p,

$$|\tau(p)| \leq 2p^{11/2},$$

and Deligne later proved this (and more – winning a Fields Medal) in part by showing that there are associated *Galois representations*

$$\mathrm{Gal}(\overline{\mathbf{Q}}/\mathbf{Q}) \to \mathrm{GL}_2(\mathbf{Q}_\ell),$$

for each prime ℓ.

Two of the seven Clay Mathematics Million Dollar Millennium Problems deal with properties of L-functions and modular forms, namely the Riemann Hypothesis and the Birch and Swinnerton-Dyer conjecture. The Riemann Hypothesis concerns the distribution of prime numbers; it asserts that all the zeros in the critical strip $0 < \Re(s) < 1$ of the unique meromorphic continuation of $\zeta(s)$ lie on the line $\Re(s) = \frac{1}{2}$. The correctness of the fastest used algorithms for constructing large prime numbers, which are used by the public-key cryptosystems that everybody who uses the Internet relies on daily, depends on the truth of a generalized version of this 150-year-old unsolved problem.

Virtually all branches of number theory and arithmetic geometry have been touched by L-functions and modular forms. Besides containing deep information concerning the distribution of prime numbers and the structure of elliptic curves and abelian varieties, they appear, for example in Tunnell's conjectural classification of congruent numbers, a problem first posed by

Arab mathematicians over one thousand years ago, which asks which integers are the area of a right triangle with rational sides [Kob84].

The proposed projects will result in the creation of a vast amount of data about a wide range of modular forms and L-functions, which will far surpass in range and depth anything computed before in this area. We will generate this data in a systematic fashion and organize it in a freely available online data archive, along with the actual programs that were used to generate these tables. Our archive will be a rich searchable source of examples and tools for researchers working on L-functions and modular forms for years to come, and will allow for future updates and expansions.

Modular forms encode an extraordinary amount of deep arithmetic information. For example, newforms play a central role in Wiles's proof of Fermat's Last Theorem, and much research toward generalizing his methods to attack other Diophantine equations (see [Dar97, Mer99, Che05, SC03]) depends on computations and tables of modular forms. The modularity theorem of Wiles et al. also forges a deep link between modular forms and the Birch and Swinnerton-Dyer conjecture.

A major conjecture of Serre, which was proved during the last 3 years (by forthcoming work of Dieulefait, Khare, Kisin, and Wintenberger), asserts that all 2-dimensional odd representations of the Galois group of $\mathbb{Q}$ arise from classical modular forms via a construction of Deligne. Also modular forms are used to construct optimal expander graphs that arise in information theory, construct optimal codes, are important in counting points on varieties over finite fields, and arise as generating functions.

2.2.2 Specific Computations and Data

Below we describe several major computations of modular forms and their associated L-functions. In each case we discuss the computation, how it could be carried out, and some specific ways in which it would impact theoretical work.

Compute extensive data about all weight 2 newforms on $\Gamma_0(N)$ for all levels $N \leq 10{,}000$, and all weight 2 newforms with quadratic character and level $N \leq 1{,}000$. We intend to do this mainly using modular symbols [Cre97, Ste07]. The complexity is dominated by exact linear algebra, computation of sparse kernels, dense characteristic polynomials and decompositions of modules as direct sums of simple modules (rational canonical form). C. Pernet, who is a postdoc at University of Washington 2008–2010 funded by NSF grant DMS-0713225, is doing cutting edge research on asymptotically fast linear algebra algorithms that are critical to this problem.

A construction of Shimura associates to each newform a modular abelian variety. Extending much previous work of Stein, we intend to compute much about the Birch and Swinnerton-Dyer invariants attached to these abelian varieties, then make conjectures and prove theorems based on the resulting data.

Enumerate all weight 2 newforms with rational Fourier coefficients on $\Gamma_0(N)$ for $N \leq 234{,}446$. John Cremona has computed all such newforms for $N \leq 130{,}000$ as the result of much work using his optimized software (which is part of the Sage software) [Cre97, Cre]. This enumeration has ground to a halt, and in collaboration with Cremona, Dembele, and others we intend to push the calculation further using new techniques coming out of combining the

massive search of Stein-Watkins [SW02, BMSW07] with better algorithms for computing Hecke operators that involve quaternion algebras and ternary quadratic forms (forthcoming work of G. Tornaria), and sparse linear algebra techniques. The first known elliptic curve of rank 4 has composite conductor 234,446, so this calculation would also provide a complete enumeration of all elliptic curves up to the first of rank 4 (Stein and two undergraduates enumerated all prime-conductor curves up to level 234,446, and none had rank 4). This computation will involve mainly sparse exact linear algebra over $\mathbf{Q}$ in vector spaces of dimension up to around 60,000.

Compute all weight 2 newforms whose coefficients generate a field of degree 2 *on* $\Gamma_0(N)$ *for* $N \leq 100{,}000$. Such forms correspond to 2-dimensional abelian varieties. We hope to pave the way for future collaborations with researchers on $\mathbf{Q}$-curves, modular genus 2 curves, and quaternionic multiplication surfaces (Bjorn Poonen, Jordan Ellenberg, Jordi Quer, Peter Clark, etc.).

Compute all $\mathbf{Q}$*-curves of level* $N \leq 100{,}000$. Now that Serre's modularity conjecture is known, it is possible to compute every $\mathbf{Q}$-curve of given conductor using modular symbols. We intend to use modular symbols to compute the modular forms associated to each $\mathbf{Q}$-curve up to level 100,000, then in many cases find a corresponding Weierstrass equation.

Compute all weight 4 newforms on $\Gamma_0(N)$ *with level* $N \leq 1000$. This computation would involve the same techniques as the weight 2 calculation, except that the linear algebra would involve much larger rational numbers, albeit on matrices with fewer rows and columns. This will involve dense linear algebra over $\mathbf{Q}$ in spaces of dimension up to 632 with fairly small numbers; in particular, we will have to compute and factor characteristic polynomials of dense matrices of this size.

Compute data about modular motives. To each weight 4 newform there is a corresponding motive (a higher weight generalization of an abelian variety), and there are conjectures about that motive due to Beilinson, Bloch, and Kato that generalize the Birch and Swinnerton-Dyer conjecture. Extending the work started in [DWS03], we hope to compute extensive data about each such motive, enumerate the results in tables, formulate conjectures, compute algebraic parts of L-values, and more.

Compute all newforms on $\Gamma_0(N)$ *with level* $N \leq 100$ *and weight* $k \leq 100$. To do this computation we let $R(N) = \oplus_{k \geq 0} M_k(\Gamma_0(N))$ be the ring of modular forms of level N, and use modular symbols to compute $M_k(\Gamma_0(N))$ for the first few k, then use this low-weight data to find explicit algebra generators for $R(N)$ for each $N \leq 100$. We then generate a q-expansion basis to high precision for the spaces of cusp forms for each weight $k \leq 100$, compute a Hecke operator on it, and use it to write down newforms. This computation will involve explicit very dense linear algebra over $\mathbf{Q}$ in vector spaces of dimension up to 1774.

We will also carry out a similar computation for $\Gamma_1(N)$. The resulting data will be useful for investigating questions about images of Galois representations and properties of p-adic modular forms. The recent exciting work of Bill Hart and David Harvey in which they have revolutionized algorithms for univariate polynomial arithmetic, as well as Clement Pernet's work [DPW05] on linear algebra, will both be crucial for making this computation feasible.

2.3 Mathematical Physics: Mapping the Supersymmetry Genome

Supersymmetry in mathematical physics is a symmetry between the two fundamental classes of particles found in nature: bosons and fermions. From a physics perspective, interest in supersymmetry stems from the fact that many promising theories like superstring theory and M-theory are supersymmetric and, if applicable to the physical world[1], supersymmetry would explain many puzzling features of known laws of nature. Furthermore, insights gained from studying supersymmetric quantum field theories may lead to better understanding of quantum field theories in general.

2.3.1 Background

The ultimate goal of this project is to completely classify supersymmetric theories. In many cases, supersymmetry is understood only "on-shell", i.e., the correspondence between bosons and fermions becomes apparent only after restricting to the solutions of the equations of motion. Rather, we are interested in determining "off-shell" formulations of supersymmetry, where the correspondence between bosons and fermions is manifest independently of the dynamics of the particles. We want to establish the *mathematics* of supersymmetry, how the supersymmetry is represented algebraically, independently of the *physics*, which involves the imposition of a Lagrangian. In [GLPR02], Gates asserts that this "fundamental supersymmetry challenge" is key to any theory providing a description of our universe, and he laments that it has gone unanswered since the early days of supersymmetry over three decades ago.

Supersymmetry is well understood in both on-shell and off-shell formulations for low dimensional spacetimes, but in higher dimensions, such as the 10 and 11 dimensions required for string theory and M-theory, the supersymmetry appears only on-shell. On-shell theories are significantly more difficult to quantize than off-shell theories, and the standard methods of quantization work only for off-shell formulations. A complete classification and construction of off-shell supersymmetric theories in 10 dimensions could give rise to off-shell formulations of string theory, and a construction of 11 dimensional off-shell supersymmetry could yield a covariant formulation of M-theory, which has thus far eluded the physics community. These off-shell theories may turn out to be extraordinarily complicated, with thousands of degrees of freedom and accessible only with the aid of a computer, which would explain why they have not yet been discovered by the theoretical physics community.

Restricting to low dimensional supersymmetric theories simplifies the process of constructing and classifying them. However, it also gives a glimpse into the complexity of higher dimensional theories. Via a procedure called "dimensional reduction", every high dimensional theory admits a low dimensional shadow. With a complete classification of low dimensional theories, we can study the shadows of the known high dimensional theories. Then reversing this process, determining when and how low dimensional theories extend to higher dimensions, can provide our desired constructive classification of high dimensional theories.

Faux and Gates have shown [FG05] that the data required to specify one-dimensional off-shell supersymmetric theories can be encoded in directed bipartite graphs called "Adinkras."

[1]The Large Hadron Collider (LHC) at CERN will begin to search for experimental evidence for superpartners in 2008.

These Adinkras are analogous to weight diagrams for representations of compact Lie groups and their Lie algebras, except that the edges of Adinkras are further decorated with signs and height assignments (see Figure 1).

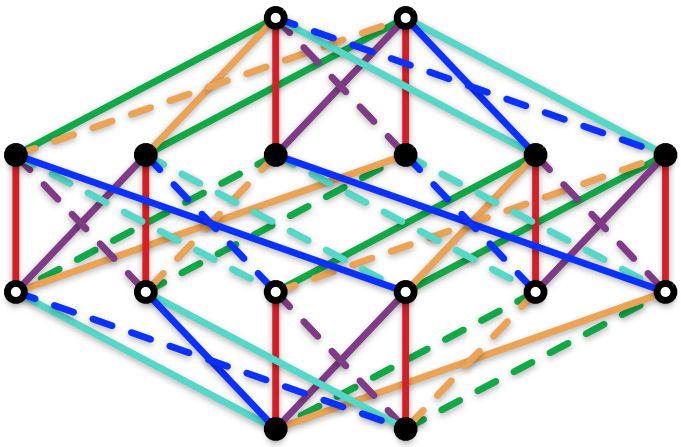

Figure 1: An irreducible 6-regular Adinkra

In [DFGHIL] the co-PI Doran and his collaborators show that every Adinkra topology is obtained by taking the 1-skeleton of an N-dimensional cube, then taking the quotient by a binary linear error correcting code. The relevant codes are *doubly even*, hence the classification of Adinkra topologies is equivalent to the classification of doubly even binary linear error correcting codes.

A binary code of type $[n, k]$ is a linear subspace $C \subset {\mathbb{F}_2}^n$ of dimension k, fixing the standard basis for ${\mathbb{F}_2}^n$ (we call n the degree of C). Linear codes are usually represented as the rowspace of a matrix. Such a code is doubly even if every word has Hamming weight (number of ones) divisible by four. Under the standard inner product, doubly even codes are self-orthogonal, a valuable property in coding theory. Linear codes are typically used to provide information redundancy when transmitting over channels with noise, thus the vectors (with respect to the standard basis) are called words. The Hamming weight is a metric, and given a small number of errors, the distance from the transmitted word to the original will be minimal, and the original word can be recovered. In particular, doubly even codes are used in constructions of exotic codes [Koc90], and when $8 \mid n$, all self-dual binary linear codes are doubly even.

2.3.2 Specific Computations and Data

Compute all isomorphism types of doubly even binary codes with degree $n \leq 32$. We intend to carry this out using generation by canonical augmentation [McK98, KÖ06]. Codes are generated from smaller codes by augmentation or gluing, the addition of a vector to the basis. A computable definition of when an augmentation is canonical is given, so that using only canonical augmentations, the isomorphism types are exhaustively enumerated without repeats. This definition is given in terms of a canonical labeling process, in which a unique

representative of an isomorphism class is computed. This process dominates the complexity of the computation.

We compute the canonical representative of the isomorphism class using partition refinement [Sim71, McK81, Leo91]. A labeling of the code is an ordering of the columns, which determines a representative of the isomorphism class. We interpret ordered partitions as partial orderings of the columns, and a partition consisting of singletons (discrete) as a full ordering, hence a labeling of the code. The algorithm traverses a tree consisting of progressive refinements of ordered partitions, searching for the canonical label. It uses information about the automorphism group as it is collected to prune the search tree; the automorphism group is also computed by the algorithm.

The complexity of computing canonical representatives depends strongly on the combinatorial structure of the object. The large amount of symmetry leads to larger trees to search; however, more information about the automorphism group allows for more pruning of the tree. Cases bearing fewer symmetries lead to smaller trees, but more of the tree must be searched. We achieve optimum running times with codes that have "medium" amounts of symmetry.

The main bottleneck of the labeling algorithm is the refinement process itself. A supercomputer is not a valid solution to this problem because the problem is not easy to parallelize; it consists of sorting and comparison, repeatedly on a relatively small set of data. Rather this computation should be run in parallel with other computations on a machine with few nodes, but very fast processors, such as the one we are asking for.

The other factor affecting total computation time is the sheer number of objects to be generated. For example, there are at least

$$619{,}287{,}158{,}132 \approx \frac{m}{32!}$$

distinct classes of $[32, 10]$ doubly even codes, where m is obtained by Gaborit's formulas for the total number of such codes [Gab96]. The computation to generate the codes is massively parallelizable–each code in the list that has not been augmented can be investigated independently. Aside from the speed of computing canonical representatives, the number of processes and the overhead between them will determine the total running time. Large amounts of RAM are helpful in speeding up communication, allowing for fewer interactions with the hard drives. This will give the presence of many processors even more influence over speed.

This computation has been attempted on the machine purchased under NSF Grant DMS-0555776, by Robert Miller who is a graduate student of PIs Doran and Stein. Miller used DSage to control the generation of codes, with Magma performing the sophisticated coding theory computations. In the process, Miller discovered several serious bugs in Magma. The Magma functions in question were adapted from software written by Leon [Leo91], which has massive memory leak problems. Since Magma is closed source, it was impossible for Miller to fix the bugs, and the computation ground to a halt. Miller has mostly finished implementing in Sage his own new version of the algorithm, which does not leak memory and in our tests is as much as five times faster than Magma. Moreover, the new implementation Miller is creating is free and open source, so other researchers can improve and extend his code.

Due to the combinatorial nature of the codes, the results of this computation will require a large amount of space to store, on the terabyte scale. By having the storage on the same

machine as the computation, much of the communication overhead can be eliminated, and in addition, the whole process can be fine tuned for optimum performance.

The results of this computation will be of interest to physicists and coding theorists. Given the astronomical size of the expected output, a lookup table will not suffice; initial estimates using [Gab96] indicate that a lookup table would require at least 28 terabytes of disk space. Thus instead of using a lookup table we intend to create a database that stores only recursive instructions for building the codes, and a very fast Internet interface to quickly query for and retrieve these codes. Thus the requested server machines would also be ideal for hosting the results, which would include data such as weight distributions and automorphism groups.

3 Sage: Computing and Disseminating Data

Much of the computation and dissemination of data that comes out of this project will be done in the context of Sage, which is an open source software project started by one of the PI's (Stein) in 2005 to support advanced mathematical research. Now that a much larger group of people are using and contributing to Sage, the scope of Sage has broadened enormously. The Sage software will be used in numerous ways in order to support the research described elsewhere in this proposal:

1. **Implementing Software:** Sage has a very rich library of functionality that builds on decades of work, hence provides an excellent environment for implementing the code we will need to do the research described in this proposal. Morever, Sage has integrated support for robust task farming (DSage).

2. **Distributing Software:** Many implementations of algorithms that come out of this project will have an automated test suite and be included with Sage, so they are extremely easy for other researchers to make use of and contribute to.

3. **Distributing Databases:** Sage has a sophisticated web-based interface, which will facilitate sharing and distributing results of calculations. Sage also includes relational and object-oriented databases.

Sage is the only serious general purpose mathematics software that uses a mainstream programing language as the user language. The programming language used for working with Sage is Python, which is a powerful modern interpreted programming language.

> "Google has made no secret of the fact they use Python a lot for a number of internal projects. Even knowing that, once I was an employee, I was amazed at how much Python code there actually is in the Google source code system.", said Guido van Rossum, Google, creator of Python.
>
> "Python plays a key role in our production pipeline. Without it a project the size of Star Wars: Episode II would have been very difficult to pull off. From crowd rendering to batch processing to compositing, Python binds all things together," said Tommy Burnette, Senior Technical Director, Industrial Light & Magic.

Sage is *free and open source*, so it is flexible and extensible. In particular, everybody is allowed to view and modify absolutely all of the source code of their copy of Sage, change their copy however they want for their needs, and freely share an unlimited number of copies with others. This makes adopting Sage for our project a good long-term investment.

Instead of reinventing the wheel, Sage combines many of the best existing open source libraries that have been developed over the last 40 years (about *5 million lines of code*) with over *200,000 lines of new code*.

In November 2007, Sage won first place in the scientific category of the Trophées du Libre, (`http://www.tropheesdulibre.org/?lang=en`) which is a major international free software competition. There are several thousand Sage users and over 100 active developers.

4 Undergraduate Impact

There are six undergraduates at the University of Washington that work with the PI's regularly, some funded by an NSF VIGRE grant. They would make use of the proposed hardware in the context of research and Sage development.

The equipment requested in this proposal will also have an immediate and strong impact on two other important undergraduate activities. Since 1988, the University of Washington has been host to an REU program that has produced many striking results in the area of discrete inverse problems. Students have found new phenomena (so called n-to-1 electrical networks), have created new methods for studying inverse problems (star-K transformations), and have developed many new algorithms. A demand has arisen for "REU software" which will serve as an asset for researchers everywhere. For example students now have created the currently most efficient methods for solving the inverse problem for critical circular planar networks. They have also developed algorithms (and software) to find the medial graph of a graph embedded in a Riemann surface. These are only a few of the examples of types of routines that can be developed and publicized using Sage.

The University of Washington has also been prominent in the Mathematical Contest in Modeling. This contest asks teams of students to precisely formulate and solve vaguely given problems over a four-day period. The winning papers often draw wide attention – for example the recent problem of fairly drawing Congressional districts (the Gerrymandering Problem). UW teams have been declared Outstanding Winners seven times in the last six years. Even so, some teams have been held back by not being able to produce definitive results under the four-day limit because of slow computing times. Having large storage space and fast computation will make a big difference to their ability to attack hard problems. They will also benefit greatly by the central repository of the most useful algorithms. This asset would of course be available to any university, not just the University of Washington since the software is open source.