Lectures Notes for AIM/Atlas of Lie Groups Workshop

John R. Stembridge ⟨jrs@umich.edu⟩

Department of Mathematics
University of Michigan
Ann Arbor, Michigan 48109–1109

23–27 July 2003

Topics

I. The Spherical Unitary Dual: Computational Overview

Or at least, this is my understanding of what the problem is...

Let R be a root system with V the ambient real vector space, along with the usual choices (simples, positive roots,...) and W the Weyl group.

Given a reduced expression for the longest element of the Weyl group, say $w_0 = s_{\alpha_1} \cdots s_{\alpha_l}$, there is an induced ordering $\beta_1, \dots, \beta_l$ of the positive roots; namely,

$$\beta_l = \alpha_l,\ \beta_{l-1} = s_l \alpha_{l-1},\ \dots,\ \beta_1 = s_l \cdots s_2 \alpha_1.$$

Now for each $\nu \in V$, let $A(\nu)$ denote the following element of the real group algebra of W:

$$A(\nu) := (1 + \langle \nu, \beta_1^\vee \rangle s_{\alpha_1}) \cdots (1 + \langle \nu, \beta_l^\vee \rangle s_{\alpha_l}).$$

Elementary Facts 1–3.

(1) *$A(\nu)$ is independent of the choice of reduced expression.*

(2) *If $w_0\nu = -\nu$, then $A(\nu)$ is Hermitian in every unitary representation of W.*

(3) *$A(\nu)$ is invertible if and only if $\langle \nu, \beta^\vee \rangle \neq \pm 1$ for all roots β.*

The proof of (2) boils down to the observation that the root ordering corresponding to the reduced expression $w_0 = s_{\alpha_l} \cdots s_{\alpha_1}$ is $-w_0\beta_l, \dots, -w_0\beta_1$.

Let V_0 denote the subspace of V fixed by $-w_0$.

The Problem. *For (potentially) all unitary irreps $\sigma \in \widehat{W}$, and all $\nu \in V_0^+$, determine when $\sigma(A(\nu))$ is positive semi-definite (psd). More ambitiously, compute the signature of every $\sigma(A(\nu))$ as well.*

For the p-adic case, we are primarily interested only in knowing, for a given ν, whether $A(\nu)$ is psd in *every* irrep of W or not. Here, we are asking for considerably more data, on the expectation that it will be useful for other applications (e.g., the real case).

It is natural to carve up V into cells according to where a given point ν may sit relative to each hyperplane $\langle \cdot, \beta^\vee \rangle = \pm 1$ (i.e., where the singularities occur in each factor of $A(\nu)$). Given that we are interested only in the case of dominant ν, we will discard the hyperplanes $\langle \nu, \beta^\vee \rangle = -1$ (for positive roots β). With this in mind, we define a *cell* C to be any of the (non-empty) sets obtained by selecting one of

$$\{\nu \in V : \langle \nu, \beta^\vee \rangle > 1\},\ \{\nu \in V : \langle \nu, \beta^\vee \rangle = 1\},\ \{\nu \in V : \langle \nu, \beta^\vee \rangle < 1\}$$

for each positive root β, and taking the intersection. Note that each cell is polyhedral and open relative to its affine span.

It is important to note that cells will often include points that are not in the dominant chamber; there even exist cells that are entirely disjoint from the dominant chamber. A second complication is that we are only interested in points where $w_0\nu = -\nu$. We say that C is a *dominant cell* if $C \cap V_0^+$ is non-empty.

[One good picture, perhaps for B_2, should make this clear.]

Of course, our primary interest is in the dominant cells.

Hypothesis. *For each unitary W-rep σ, the signature of $\sigma(A(\nu))$ is constant within each (dominant) cell.*

This may follow from what is known about the spherical unitary dual in the p-adic case, but in any case it is not an elementary fact that the above hypothesis is true. Indeed, it is possible to construct similar-looking Hermitian products of operators-with-each-factor-having-linear-parameters such that the signature can change without crossing the boundary walls defined by the singularities of each factor. (However, it is elementary to show that signatures must be constant on the big open cells.)

In order to implement a computer solution of The Problem for a given root system R (given the Hypothesis), there are four main subproblems:

A. Generate a unitary model (i.e., explicit matrices for each simple reflection) for each (desired) irrep σ of W.

B. Devise an efficient way to visit each dominant cell C, and select a representative point ν from C.

C. Evaluate the matrix $\sigma(A(\nu))$.

D. Test whether $\sigma(A(\nu))$ is psd, or (if desired) computed the signature.

Some numbers to help keep things in perspective:

In $W(E_7)$ the largest irrep has degree 512 (median 144).

In $W(E_8)$ the largest irrep has degree 7168 (median 1296).

With 1GB of memory, a matrix with $(7168)^2$ entries has about 20 bytes (50 digits) available per entry.

In E_7, there are

- 4160 dominant big cells
- 46 of these are positive for the reflection representation
- there are 227 dominant vertices (i.e., 0-dimensional cells)
- there are $113,100$ dominant cells total.

In E_8, there are

- 25080 dominant big cells
- 205 of these are positive for the reflection representation
- there are 739 dominant vertices (i.e., 0-dimensional cells)
- there are $1,070,716$ dominant cells total.

Some comments about the subproblems:

A. is tricky, not solved yet (to our knowledge). Jeff has some interesting approaches that have worked for E_6 and most of E_7. Energy/brainpower directed at this subproblem—minimizing the matrix entries, $\mathbb{Z}$-forms, sparsity—has potential for big payoffs, possibly making the difference between E_8 being feasible or not. (See C and D.)

B. is mostly solved. The cells have nice explicit descriptions, and I have good, fast, efficient code for visiting each one. To select the point ν, one can use linear programming; alternatively, there is a canonical point one can select: there is a unique alcove face in each cell that is minimal in Bruhat order (and of the same dimension as the given cell). Take its barycenter. Given the tendency for messy coordinates in ν to create monstrous matrices for $\sigma(A(\nu))$, having a canonical choice may be preferable to the alternatives. (But this is just a belief.) We will discuss this in II in more detail.

C. should be easy: do it one column at a time. There could be a space issue unless the representing matrices for the simple reflections are sparse and are stored as such. Otherwise, in the worst case, we need to store 8 (or 9) matrices of size $(7168)^2$.

D. is theoretically easy: (roughly) equivalent to computing one determinant of order $\deg(\sigma)$, using the LU-factorization. But this is the computational bottleneck. Testing whether a matrix is psd requires (AFAIK) exact, rational (or at least integer), arbitrary precision arithmetic. In fact, given the extra expense of $\mathbb{Q}$-arithmetic (addition is as costly as multiplication), it may be necessary to optimize for models and matrices with small denominators, clear these denominators, and then do fraction-free Gaussian operations.

Another bottleneck might be in A.

II. Cell Combinatorics

For simplicity, let's assume $w_0 = -1$, or equivalently, $V = V_0$. There are a few more details to worry about when $w_0 \neq -1$, but they are under control. Roughly speaking, the fix one needs to apply in every statement below is to add the requirement that the objects in question are stable under the action of $-w_0$.

A. The big cells: where $A(\nu)$ is nonsingular.

Let $P(R^\vee)$ denote the partial ordering of the positive co-roots: $\gamma^\vee > \beta^\vee$ if $\gamma^\vee - \beta^\vee$ is a sum of positive co-roots. Given $\nu \in V^+$, the set

$$F(\nu) = \{\beta^\vee : \langle \nu, \beta^\vee \rangle > 1\}$$

is an order-filter (i.e., upward-closed) subset of $P(R^\vee)$. Furthermore, $F(\nu)$ must be constant within a given cell.

Theorem. *Conversely, every order filter of $P(R^\vee)$ occurs as F(some dominant cell). In other words, the big dominant cells are in bijection with order filters of $P(R^\vee)$.*

Consequently, a minimal set of defining inequalities for the big cell indexed by F is

$$\langle \nu, \beta^\vee \rangle > 1 \quad \text{for minimal } \beta^\vee \in F,$$
$$\langle \nu, \beta^\vee \rangle < 1 \quad \text{for maximal } \beta^\vee \notin F.$$

Another consequence of the above characterization is that the cell indexed by F is bounded if and only if F contains no simple roots.

Based on J.-K.'s work, it looks like it would be interesting to have a characterization of the big cells C such that $C \cap V^+$ is an alcove of the affine Weyl group associated to $R^\vee$.

Although we don't (yet) have an answer, this might be nicely characterized by features of F.

An interesting bit of numerology:

THEOREM (not due to me). *The number of order filters of $P(R^\vee)$ is*

$$(h + d_1) \cdots (h + d_n)/|W|,$$

where h is the Coxeter number and the d_i's are the degrees of the basic invariants of W.

B. The lower cells.

Given dominant $\nu \in V^+$, set

$$E(\nu) = \{\beta^\vee : \langle \nu, \beta^\vee \rangle = 1\}.$$

Note that $E(\nu) \cup F(\nu)$ must also be an order-filter of $P(R^\vee)$, and $E(\nu)$ must be constant within each cell. Notice also that if there are two related elements in $E(\nu)$, say $\gamma^\vee > \beta^\vee$, then $\langle \nu, \gamma^\vee - \beta^\vee \rangle = 0$, and hence $\langle \nu, \alpha^\vee \rangle = 0$ for every simple co-root $\alpha^\vee$ that appears in the support of $\beta^\vee - \gamma^\vee$ (since ν is dominant). Thus the cell containing ν may be confined to certain walls of V_0^+.

Let $J = \{\alpha^\vee : \langle \nu, \alpha^\vee \rangle = 0\}$ denote the set of simple roots indexing the confining walls.

Then $\langle \nu, \beta^\vee \rangle$ is constant within each equivalence class of positive co-roots $\beta^\vee$ that differ by a root in the parabolic subsystem R_J, and is 0 on R_J itself.

Let $P(R^\vee)/J$ denote the induced partial ordering on the equivalence classes of $(R^\vee)^+$, excluding the trivial class of roots supported on J. Then $E(\nu) \cup F(\nu)$ is an order filter of $P(R^\vee)/J$ and $E(\nu)$ must be a subset of the minimal elements of this order filter.

It is important to remember that we did not freely choose J; it was determined from $E(\nu)$ as the set of simple co-roots that occur as differences between pairs of elements of $E(\nu)$. Nevertheless, for purposes of computation it is often easier to choose J first, and consider the possible corresponding choices for E; to describe this in a shorthand way, let us say that E is *J-colored.*

THEOREM. *Conversely, for fixed J, every disjoint pair (E, F) such that $E \cup F$ is an order filter of $P(R^\vee)/J$ and E is a J-colored subset of the minimal elements of $E \cup F$ arises as the defining data for some dominant cell.*

Thus we now have a name $C = C(J, E, F)$ for each dominant cell (remember that the

J is redundant, but convenient), and a minimal set of defining equations for the cell C is

$$\begin{aligned}
\langle \nu, \beta^\vee \rangle &> 1 \ \text{ for minimal } \beta^\vee \in F, \\
\langle \nu, \beta^\vee \rangle &< 1 \ \text{ for maximal } \beta^\vee \notin E \cup F, \\
\langle \nu, \alpha^\vee \rangle &= 0 \ \text{ for } \alpha^\vee \in J, \\
\langle \nu, \beta^\vee \rangle &= 1 \ \text{ for one } \beta^\vee \in E \text{ per } J\text{-equivalence class.}
\end{aligned}$$

In particular, the co-dimension of C is $|J| + |E/J|$.

REMARK. There is a nice polytope $Q(R^\vee)$ (the dual of the Fomin-Zelevinsky generalized associahedron) such that the number of k-dimensional faces of $Q(R^\vee)$ equals the number of k-dimensional dominant cells of the form $C(\varnothing, E, F)$.

C. Canonical cell representatives.

Given a dominant cell $C = C(J, E, F)$, we now want to consider the problem of selecting a point ν from the cell. Given the above defining equations for the cell, it is easy to use linear programming methods to produce a point in the cell.

This is what my Maple program currently does.

However, since the niceness of the coordinates of ν can have a significant impact on the niceness of the matrix entries of $\sigma(A(\nu))$, it may be desirable to locate a *canonical* point in the cell, one that we may hope to have nice coordinates. Of course, we have no choice in the case of the 0-dimensional cells.

Note that the hyperplanes $\langle \cdot, \beta^\vee \rangle = 1$ are a subset of the hyperplanes of the affine Weyl group $\widetilde{W}$ associated to $R^\vee$. Thus each big cell is a union of alcoves (ignoring issues of measure zero), and each cell is a union of alcove faces (i.e., $\widetilde{W}$-images of faces of the fundamental alcove A_0). Let us refer to the alcove faces of maximum dimension contained in a given cell C as *alcove facets* of C. So if C is a big cell, its alcove facets are alcoves.

THEOREM. *Each dominant cell has a unique alcove facet that is closest to A_0.*

Closeness is measured by the number of hyperplanes of the full affine arrangement that separate the given alcove face from A_0.

Idea of Proof. Given a cell $C = C(J, E, F)$, there are certain hyperplanes of the full affine arrangement that "obviously" separate C from the fundamental alcove, and others that "obviously" contain C. If you write down the obvious lists, you can prove that the lists satisfy the correct convexity properties that force them to be the set of *all* separating and containing hyperplanes for some alcove face. Any other alcove face would have to cross additional hyperplanes, or cut down the dimension too far, so this is the unique alcove facet of C that is closest to A_0. $\square$

Remarks. (1) The special case corresponding to the big cells follows from the work of Shi on the Kazhdan-Lusztig cells in affine Weyl groups.

(2) There is a standard way that separating sets of hyperplanes encode elements of (affine) Weyl groups. So given the list of hyperplanes discussed in the above proof, we can recover the affine Weyl group element that maps some face of A_0 to the nearest alcove facet of $C(J, E, F)$. By following the image of the barycenter of that face over to C, we obtain a (dominant) canonical point ν from C.

(3) The above proof also proves the previous theorems characterizing the dominant cells. The point is that the argument works if one starts with any datum (J, E, F) satisfying the necessary conditions, thereby proving that the datum defines a non-void region in V.

I have not yet implemented this approach to producing cell representatives.

D. The Bottom-up approach.

Aside from the linear irreps of W, we expect the psd cells to be rare.

Considering that there are roughly 10^6 dominant cells in E_8, we definitely don't want to naively visit *every* cell and test whether it is psd in every irrep of W.

Another approach, suggested by J.-K. Yu, is to start with the lower dimensional cells and work our way up. Since psd-ness is a closed condition, a cell C can be psd only if every cell on the boundary of C is also psd.

Note: dominant cell $C(J, E, F)$ is contained in the closure of $C(J', E', F')$ iff $F \subseteq F'$ and $E \cup F \supseteq E' \cup F'$. (These conditions automatically force J to contain J' and E to contain E'.)

One complication is that the dominant cells extend beyond the fundamental chamber; in particular, the closure of a dominant cell may be a union of both dominant and non-dominant cells. It is possible that the non-dominant cells also have nice descriptions generalizing the above, but at the moment I don't know the recipe.

But we don't necessarily have to address this issue. A given dominant cell C of dimension k must have at least one dominant boundary cell of dimension $k - 1$. Having determined all dominant psd cells of dimension $k - 1$, it is easy to generate the names $C(J, E, F)$ of all dominant k-dimensional cells whose dominant boundary cells are all psd. These are the ones that we test for psd-ness. There may be cells that we would have skipped if we had included tests involving non-dominant boundary cells, but it seems plausible that the gain from this is negligible.

This approach has the added advantage that it helps us track the maximal psd cells—a more compact description of the psd cell data.

III. Software

Some issues to keep in mind:

Q1. Do we want multiple solutions? Redundancy is nice in mathematics, but even more valuable with software. Sanity checks are easier and maybe good enough.

Q2. How much programming/web support do we need?

Remarks.

• The main constraint we have in the spherical unitary dual problem is the need to do exact Gaussian-type transformations on large $(7168)^2$ matrices with potentially huge rational entries. If we can avoid using the very largest irreps, this might make the problem much more tractable, and require less in the way of both software and hardware.

• Everything else can be done easily e.g. in Maple. A working implementation is at

⟨`http://www.math.lsa.umich.edu/~jrs/data/unitary`⟩

It is mostly complete, but lacks models for most W-irreps.

• The big W-irreps probably cannot be handled internally by Maple or Mathematica (what about "gridMathematica"?).

⟨`http://www.wolfram.com/products/gridmathematica/`⟩

Note that Maple (and Mathematica?) can call external C programs.

• Is Magma feasible? One issue is the lack of friendly licensing. (Maple and Mathematica can be annoying too, but with University-wide licenses, this is less of a problem.)

⟨`http://magma.maths.usyd.edu.au/magma/`⟩

• Gnu Multi-Precision (GMP) library (for linking from C or C++ code)

- ubiquitous, found e.g. in most Linux distributions
- includes arbitrary precision rational arithmetic
- friendlier languages (see Pari/GP below, maybe python) have hooks that can link into GMP

⟨`http://http://www.swox.com/gmp/`⟩

• (Objective) Caml - a general-purpose language developed at INRIA

- has an interactive system and a compiler,
- available in many linux distributions,
- has built-in library for arbitrary precision rational arithmetic, but doesn't seem to have higher-level LinAlg functions built-in.

⟨http://http://caml.inria.fr/⟩

- Java does have a java.math package:

  ”Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecimal). BigInteger is analogous to Java’s primitive integer types except that it provides arbitrary precision, hence operations on BigIntegers do not overflow or lose precision. In addition to standard arithmetic operations, BigInteger provides modular arithmetic, GCD calculation, primality testing, prime generation, bit manipulation, and a few other miscellaneous operations.”

- Lidia - a C++ library developed by German academics (uses GMP)
  - has many high-level functions for number theory and linear algebra

  ⟨http://http://www.informatik.tu-darmstadt.de/TI/LiDIA/⟩

- Pari/GP - a number theory package developed by Henri Cohen et. al.
  - has both a command interpreter (GP) for easy development, and a programmer’s C library for building compiled applications.
  - has a preprocessor for converting GP code to C.
  - includes many high-level matrix ops, (LLL, Smith Normal,...)
  - very fast. Even the interpreter is much faster than Maple.

  ⟨http://http://www.parigp-home.de/⟩

Comparison of the Pari/GP interpreter and Maple V.3 running on a 1GHz P3, computing the determinant of an $n \times n$ Hilbert matrix (i.e., $H_n = [1/(i + j - 1)]_{1 \leqslant i,j \leqslant n}$).

n	size(det(H_n))	Maple (sec)	GP (sec)
15	128	.03	0
30	523	.23 ($1MB$)	.03
60	2125	3.47 ($1.6MB$)	.46
120	8579	180.87 ($5.0MB$)	7.45

The ”catch” is that the matrix operations in GP seem to be very RAM hungry. The $n = 120$ case had a 64MB stack allocated to it. For $n = 130$, 64MB is not big enough! In contrast, while Maple slows down considerably, it appears to have more sophisticated memory management and garbage collection. It would be interesting to see how well an implementation of a psd-testing program would do when compiled with Pari. Would it still be RAM-hungry? Also of interest would be a comparison with Caml.

The psd-testing problem seems at least coarsely parallelizable; e.g., take a block decomposition of the matrix, and do Gaussian operations over a ground ring of $m \times m$ matrices. This is a well-defined problem that could likely to be farmed out to a programmer using some combination of Pari/GP, GMP, C, C++, java,...

IV. Hardware

A big advantage of using GMP, Caml, java, Pari/GP,... over Maple, Mathematica, Magma is that they are distributed with source. They can be installed and take advantage of almost any hardware we buy (32bit vs. 64bit, clusters,...). No licensing worries either.

Assuming that we can't solve the biggest problems on a single machine (1GB might be needed simply to store the matrix), this suggests using a cluster of commodity PC's.

In the linux world, there is a popular (and free) clustering project called "openMosix".

⟨`http://openmosix.sourceforge.net/`⟩

It automatically distributes the load of separate computations across multiple networked machines. It is even available on bootable CDs that come loaded with complete linux distributions and tons of software. For example, "Quantian" is a bootable linux CD with openMosix and has Maxima and Pari/GP preloaded.

⟨`http://dirk.eddelbuettel.com/quantian.html`⟩

A possible solution: an array of (16, 24,...) mid-line intel PC's each with 1GB RAM and 2GHz P4's, and perhaps one larger machine with more RAM that would act as the controller dishing out the subproblems. These PC's should cost around $1000 or so each.

Example. Dell Optiplex SX260 ultra small form factor

- P4 2.0GHz
- 400FSB
- 1GB RAM
- 40GB HD
- CD-ROM
- no monitor
- $1017

⟨`http://www.dell.com/us/en/hied/products/model_optix_optix_sx260.htm`⟩

Also needed: a 100Mb switch — probably \$1000–\$1500, perhaps also also a UPS and one or more racks.

Note that we would need administrative and institutional support. This many PC's will likely use noticeable electrical power, air conditioning, etc. A really cheap approach would be to co-opt the CPU cycles of an instructional lab during off-hours.