Classification of Pseudo-cuspidal Points for Integrable Systems with Delta-Potential

Tai Lam
Physics Department, UMass Boston

Professor Steven Jackson
Thesis Advisor
Math Department, UMass Boston

May 24, 2019

Abstract

For exactly solvable quantum integrable systems with delta potential, we use the language of kaleidoscopes to classify the fully determined stable solutions, which we call pseudo-cupsidal points, for any finite Coxeter group G, using an algorithm. The algorithm is written in SageMath and the pseudo-cupsidal points for Coxeter group types A_1, C_2, and F_4 are classified as test cases for the algorithm.

1 Introduction

1.1 Background

Ansatz is a German word that means "beginning". In mathematics, an ansatz is a conjecture of the solution; subsequent work demonstrates that the the conjecture is true. For applicable physical systems, the main idea of the Bethe ansatz is that the total solution is locally a finite a finite superposition of plane waves. Interactions between many particles that can be reduced to analyzing a collection two-body interactions, which is used to convert many-body 1-D systems in into a one body problem in high dimensions. The result is the constraint of the wavefunction is given by the Bethe ansatz equations. The Bethe ansatz is closely related with integrability, which means in classical mechanics all quantities of motion can be solved for. For all known physical systems which are solvable via the Bethe ansatz, the physical system is integrable. By extension, the Bethe ansatz and quantum interference are important to quantum integrable many-body systems, since the Bethe ansatz has been used to solve these problems when perturbation and mean-field theories cannot be used. [Bat07]

The Bethe ansatz was first used by Hans Bethe in 1931. In 1963, the first major application of the Bethe ansatz since the 1930s was the Lieb-Liniger model of the Bose gas.

It is a line of spinless interacting bosons with finite, zero-range repulsive delta potentials, which are allowed to move anywhere on a 1-D line. This was one of the first many-body integrable systems which had eigenfunctions that were exactly solvable via the Bethe ansatz. [Bat07, LL63]

In 1964, McGuire used the Bethe ansatz to investigate three cases of exactly solvable 1-D N-body problems for the following types of systems: 3 particles of arbitrary mass on a 1-D line with repulsive, infinite-strength delta potentials; 3 particles of equal mass with by finite, equal-strength delta potentials; and N particles of equal mass with finite, equal-strength delta potentials. [McG64]

In 1971, Gaudin studied the Lieb-Liniger model in a 1-D box with vanishing boundary conditions, instead of circle boundary conditions used in the original Lieb-Liniger model. It was the first time that finite reflection groups were mentioned in the mathematics of a variant of the Lieb-Liniger model, namely the Coxeter groups A_n, B_n, C_n, D_n, and G_2. Gaudin associated the group A_n with the Lieb-Liniger model. The author acknowledged the Bethe ansatz method may be applied to systems represented by the other classical finite Coxeter groups beyond A_n. [Gau71]

Gutkin gave the earliest rigorous mathematical treatment in 1982 by proving the Bethe ansatz is valid for quantum integrable systems with delta potential and how quantum mechanical systems of delta potentials mathematically correspond to systems of finite sets of affine hyperplanes, which can be analyzed using reflection groups. The hamiltonians for these types of systems have the form of

$$\hat{H} = \hat{\triangle} - 2\sum_{h\in R} c(h)\delta(h) \tag{1}$$

with potential

$$\hat{V} = -2\sum_{h\in R} c(h)\delta(h) \tag{2}$$

Such quantum systems were methodically categorized as being specific cases of both classical and exceptional finite Coxeter groups of the types: A_n, B_n, D_n, E_6, E_7, E_8, F_4, G_2, H_3, H_4, and I_2^n. Gutkin notes Euclidean A_n and affine $\tilde{A}_n$ correspond to the Lieb-Liniger gas. [Gut82]

In 1997, Heckman and Opdam utilized the tools of root systems and Hecke algebra to mathematically examine quantum integrable systems corresponding to finite Coxeter groups, which were named, "particle system with delta-function interaction." [HO97a, HO97b]. The concept of coupling parameter, which is the constant of interaction strength between each hyperplane, is introduced and the authors considered systems for repulsive, arbitrary, and attractive coupling constants. The authors also introduced the concept of spherical cupsidal points, which would help find the bound states of the kaleidoscopic systems. However, the authors were unable to classify the spherical cupsidal points of all irreducible root systems, only having limited success with B_n and H_4. [HO97a]

In 2006, Emsiz, Opdam, and Stokman generalized periodic quantum integrable systems with delta potential and periodic boundary conditions as periodic boundary value problems involving affine root systems, Hecke algebra, and Dunkl operators. In this context, the authors derived the associated Bethe ansatz equations and eigenfunctions. [EOS06]

In 2015, Olshanii and Jackson demonstrated it is possible for an exceptional Coxeter group to represent a physically realistic quantum integrable system with delta potential for the first time, namely affine $\tilde{F}_4$, using the language of kaleidoscopes. This system is made of four particles with different masses in a 1-D hard-wall box and the bound solutions are exactly solvable using the Bethe ansatz. [OJ15]

1.2 Overview

In the present work, we direct our attention to quantum integrable systems with arbitrary attractive and finite delta-potentials. These physical systems may be mathematically represented interchangeably by finite reflection groups, finite Coxeter groups (both classical and exceptional), and kaleidoscopes. We seek to find the bound states of the specified kaleidoscopic systems which are square integrable. The conditions that admit square integrable solutions are given by spectral values $\vec{\lambda}$ that are regular and purely real, which we call *pseudo-cupsidal points*. Because an approach to find pseudo-cupsidal points that works in all generality is desired but the process of finding such spectral values is impractical to do by hand for large Coxeter groups, we utilize an algorithmic approach with a script written in computer algebra system SageMath in order to find pseudo-cupisal points for any general Coxeter group. Coxeter group types A_1 and C_2 were used as small test cases for the algorithm and type F_4 was used as a large test case. For the following test cases, the number of pseudo-cupsidal points in the output of the algorithm were: 1 for type A_1, 12 for type C_2, and 13,717 for type F_4.

2 Definitions, Conventions, and Notation

Hyperplanes are objects of $(n-1)$ dimensions in $\mathbb{R}^n$, also called *mirrors*, in the context of kaleidoscopes. A finite collection of hyperplanes is called a *kaleidoscope*. The hyperplanes divide $\mathbb{R}^d$ into a finite set of connected regions called *chambers*.

G is a finite reflection group, or equivalently a finite Coxeter group. Associated with each Coxeter group is the *root system* of G, or the set of unit normals to the reflection hyperplanes and $\Delta \in \mathbb{R}^n$.

Fix a vector $\vec{v}$ not orthogonal to any root, which is a single unit normal from the root system. The members of the set

$$\Delta^+ = \{\vec{\alpha} \in \Delta \,|\, \vec{v} \cdot \vec{\alpha} > 0\} \tag{3}$$

are the *positive roots*, while the remaining roots are the *negative roots*.

The root space is $\mathbb{R}\Delta$, which is the span of Δ. For example, the root space for Coxeter group type $G = A_1$, represented in $\mathbb{R}^2$, is

$$A_1 = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \right\} \tag{4}$$

In this example, according to Figure 1, $\mathbb{R}\Delta$ is the horizontal x-axis (but not all of $\mathbb{R}^2$).

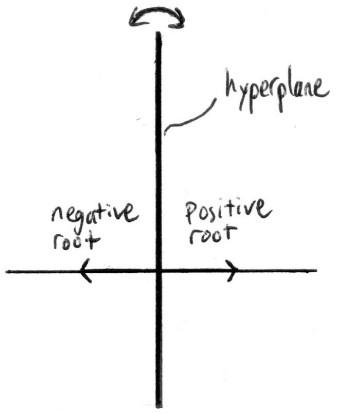

Figure 1: Type A_1 represented in $\mathbb{R}^2$.

$\vec{\lambda} \in \mathbb{C}^n$ is said to be *regular* if $\vec{\alpha} \cdot \vec{\lambda} \neq 0 \ \forall \ \vec{\alpha} \in \Delta$. For example, in Figure 1, any vector with non-zero component along the x-axis is a regular vector in the previous example. Otherwise, $\vec{\lambda}$ is said to be *singular*.

There is a special chamber called the *dominant chamber*, which is illustrated by Figure 2 and given by the formula

$$C_0 = \left(\vec{\mathbf{v}} \in \mathbb{R}^d \mid \vec{\mathbf{v}} \cdot \vec{\alpha} > 0 \ \forall \ \vec{\alpha} \in \Delta^+\right) \tag{5}$$

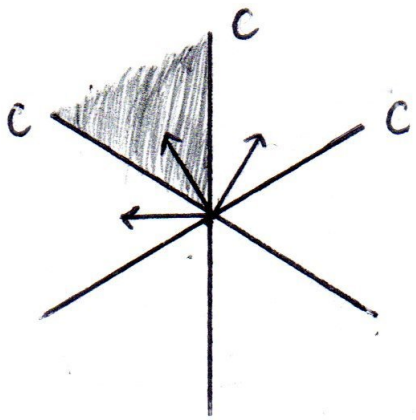

Figure 2: Dominant chamber C_0 and positive roots Δ^+ in a generic kaleidoscope.

For a fixed regular spectral value $\vec{\lambda} \in \mathbb{C}^n$, let the *spherical wavefunction* $\sigma_{\vec{\lambda}}^G : \mathbb{R}^n \to \mathbb{C}$ be the unique G-invariant function given in the dominant chamber by the formula

$$\sigma_{\vec{\lambda}}^G(\vec{\mathbf{x}}) = \frac{1}{|G|} \sum_{g \in G} \prod_{\vec{\alpha} \in \Delta^+} \frac{g\vec{\lambda} \cdot \vec{\alpha} + c_{\vec{\alpha}} |\vec{\alpha}|}{g\vec{\lambda} \cdot \vec{\alpha}} \, \mathrm{e}^{g\vec{\lambda} \cdot \vec{\alpha}} \quad \forall \ \vec{\mathbf{x}} \in C_0 \tag{6}$$

The spherical solution comes from the Bethe ansatz, which has been applied to kaleidoscopic systems. $\sigma_{\vec{\lambda}}^G$ is always well-defined, since $\vec{\lambda}$ is regular.

However, for which $\vec{\lambda}$ will $\sigma_{\vec{\lambda}}^G$ be square-integrable? Square-integrable means

$$\int \left|\sigma_{\vec{\lambda}}^G\right|^2 \, \mathrm{d}x < \infty \tag{7}$$

Square-integrable solutions will be a necessary requirement for bound states of kaleidoscopic systems.

Observe that for any $g \in G$ and $\vec{\lambda} \in \mathbb{C}^n$, the following is true:

$$\sigma_{g\vec{\lambda}}^G = \sigma_{\vec{\lambda}}^G \tag{8}$$

which exhibits *conjugacy/group invariance* for $\sigma_{\vec{\lambda}}^G$ in G.

Two matrices M_1 and M_2 are said to be *conjugate* (under a group G) if $\exists\, g \in G$ such that $gM_1 = M_2$ and the equivalence relation is denoted as $M_1 \sim_G M_2$. If the matrices M_1 and M_2 are both vectors, then conjugacy also applies to vectors. *Coupling constants* are constants associated with each Dirac delta potential, which indicates the interaction strength between conjugate classes of mirrors in kaleidoscopic systems.

For regular $\vec{\lambda} \in \mathbb{C}^n$, a related object called the *support* is defined as:

$$\mathrm{supp}(\vec{\lambda}) = \left\{ g\vec{\lambda} \;\middle|\; \prod_{\vec{\alpha}\in\Delta^+} \frac{g\vec{\lambda}\cdot\vec{\alpha} + c_{\vec{\alpha}}|\vec{\alpha}|}{g\vec{\lambda}\cdot\vec{\alpha}} \neq 0 \;\forall\, g \in G \right\} \tag{9}$$

In this paper, $\mathrm{supp}(\vec{\lambda})$ will finite because G is finite.

Another related term is the following: for any regular $\vec{\lambda} \in \mathbb{C}^n$, the *binding roots* of the set $\mathcal{B}(\vec{\lambda})$ are defined as:

$$\mathcal{B}(\vec{\lambda}) = \left\{ \vec{\alpha} \in \Delta \;\middle|\; \vec{\lambda}\cdot\vec{\alpha} + c_{\vec{\alpha}}|\vec{\alpha}| = 0 \right\} \tag{10}$$

Now, we define *pseudo-cupsidal point*, the main object of investigation for this paper.

Definition 1. *$\vec{\lambda} \in \mathbb{C}^n$ is said to be a* pseudo-cupsidal point *if $\mathcal{B}(\vec{\lambda})$ spans $\mathbb{R}\Delta$.*

Binding roots lead to zero numerators in the spherical solution coefficients in Eqn 6. This leads to the following theorem.

Theorem 2.1. *$g\vec{\lambda} \in supp(\vec{\lambda})$ iff every binding root of $g\vec{\lambda}$ is negative.*

Proof. For

$$\sigma_{\vec{\lambda}}^G(\vec{\mathbf{x}}) = \frac{1}{|G|} \sum_{g\in G} \prod_{\vec{\alpha}\in\Delta^+} \frac{g\vec{\lambda}\cdot\vec{\alpha} + c_{\vec{\alpha}}|\vec{\alpha}|}{g\vec{\lambda}\cdot\vec{\alpha}} \, \mathrm{e}^{g\vec{\lambda}\cdot\vec{\alpha}} \tag{11}$$

and

$$\mathrm{supp}(\vec{\lambda}) = \left\{ g\vec{\lambda} \;\middle|\; \prod_{\vec{\alpha}\in\Delta^+} \frac{g\vec{\lambda}\cdot\vec{\alpha} + c_{\vec{\alpha}}|\vec{\alpha}|}{g\vec{\lambda}\cdot\vec{\alpha}} \neq 0 \;\forall\, g \in G \right\} \tag{12}$$

it is true that

$$g\vec{\lambda} \in \mathrm{supp}(\vec{\lambda}) \Leftrightarrow g\vec{\lambda}\cdot\vec{\alpha} + c_{\vec{\alpha}}|\vec{\alpha}| \neq 0 \;\forall\, \vec{\alpha} \in \Delta^+ \tag{13}$$

Recall the definition of binding roots

$$\mathcal{B}(\vec{\lambda}) = \left\{ \vec{\alpha} \in \Delta \;\middle|\; \vec{\lambda} \cdot \vec{\alpha} + c_{\vec{\alpha}} |\vec{\alpha}| = 0 \right\} \tag{14}$$

So,

$$g\vec{\lambda} \in \text{supp}(\vec{\lambda}) \Leftrightarrow \mathcal{B}(g\vec{\lambda}) \cap \Delta^+ = 0 \tag{15}$$

$$g\vec{\lambda} \in \text{supp}(\vec{\lambda}) \Leftrightarrow \mathcal{B}(g\vec{\lambda}) \subseteq \Delta^- \tag{16}$$

□

The *negative root cone* $\mathcal{C}^-$ denotes the set

$$\mathcal{C}^- = \left\{ \sum_{\alpha \in \Delta^+} a_{\vec{\alpha}} \, \vec{\alpha} \;\middle|\; a_{\vec{\alpha}} < 0 \right\} \tag{17}$$

Figure 3 shows an example of a negative root cone in A_2.

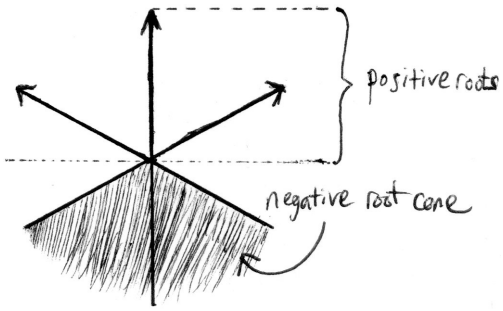

Figure 3: Negative root cone of type A_2

This leads to the following theorem.

Theorem 2.2. *$\sigma^G_{\vec{\lambda}}$ is square-integrable iff supp$(\vec{\lambda}) \subseteq \mathcal{C}^-$.*

3 Algorithms

3.1 The Setup

In $\mathbb{R}^d$, there is a finite system of hyperplanes, or a kaleidoscope, represented by the potential

$$\hat{V} = 2 \sum_p c_p \delta_p \tag{18}$$

where p stands for each hyperplane and c_p represents the coupling constant of each plane p. The assignment of coupling constants is invariant under the group. Figure 4 depicts a generic drawing of a finite system of hyperplanes in $\mathbb{R}^d$.
For the hamiltonian $\hat{H}$ of kaleidoscopic systems, the hamiltonian is represented by

$$\hat{H} = \hat{\triangle} - \hat{V} \tag{19}$$

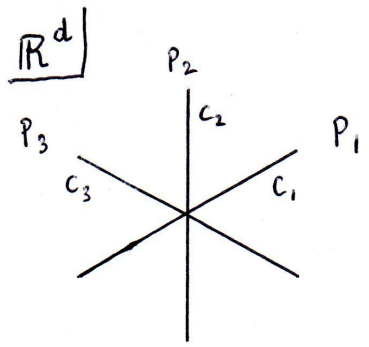

Figure 4: Generic finite system of hyperplanes

Using the time-independent Schrödinger equation

$$\hat{H}\,|\Psi\rangle = \lambda\,|\Psi\rangle \tag{20}$$

we seek to find all eigenvalues of $\hat{H}$, for a suitable meanings of "all" and "eigenvalues". For the Bethe ansatz, assume the following: first, the hyperplanes are the mirrors of a kaleidoscope. (Reflections generate a finite group G.) Secondly, coupling constants are invariant under G (i.e., if hyperplanes P_1 and P_2 are conjugate hyperplanes, then $c_1 = c_2$). The second point guarantees that G is a group of symmetries of the system.

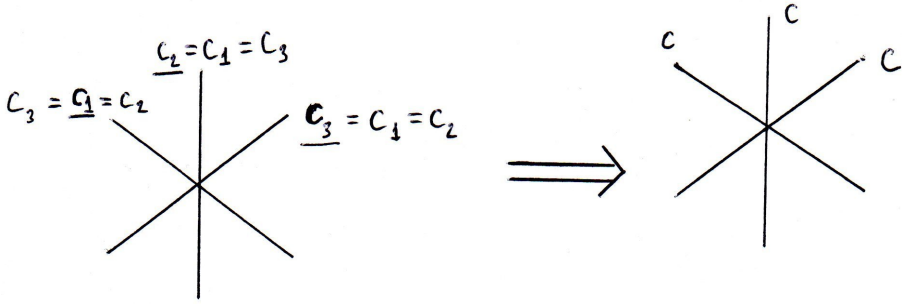

Figure 5: Group invariance guarantees conjugate mirrors/hyperplanes have identical coupling constants/interaction strengths.

According to the Bethe ansatz, eigenfunctions λ have the following form: for each chamber C and each $g \in G$, choose a complex number $a_{c,g}$. Put

$$\Psi(\vec{\mathbf{x}}) = \sum_{g\in G} a_{c,g}\, \mathrm{e}^{(g\vec{\lambda})\cdot\vec{\mathbf{x}}} \qquad \vec{\mathbf{x}} \in C, \text{ each chamber} \tag{21}$$

for some fixed $\vec{\lambda} \in \mathbb{C}^d$. For the $a_{c,g}$'s, one particular solution is the spherical solution, namely

$$\sigma^G_{\vec{\lambda}}(\vec{\mathbf{x}}) = \frac{1}{|G|} \sum_{g\in G} \prod_{\vec{\alpha}\in\Delta^+} \frac{g\vec{\lambda}\cdot\vec{\alpha} + c_{\vec{\alpha}}|\vec{\alpha}|}{g\vec{\lambda}\cdot\vec{\alpha}}\, \mathrm{e}^{g\vec{\lambda}\cdot\vec{\alpha}} \tag{22}$$

However, there is an issue of whether solution will uncontrollably diverge to infinity for most values of $\vec{\lambda}$. The spectral problem is: for which spectral values of $\vec{\lambda}$ will make $\sigma^G_{\vec{\lambda}}$ a bounded function? It turns out that if $\vec{\lambda}$ is *purely real*, then $\sigma^G_{\vec{\lambda}}$ is bounded.

Yet, there could be an issue of dividing by 0 in the denominator of the coefficient

$$\frac{g\vec{\lambda}\cdot\vec{\alpha}+c_{\vec{\alpha}}|\vec{\alpha}|}{g\vec{\lambda}\cdot\vec{\alpha}} \tag{23}$$

However, in this paper, we study *regular* pseudo-cupsidal points, for which the denominator never vanishes.

Due to this phenomenon, we use a SageMath algorithm to find $\vec{\lambda}$ which are regular.

3.2 Algorithm Pseudocode

Algorithm 1: Algorithm to produce `GoldElixir`

Input: Coxeter group G
Output: `GoldElixir`
Result: each matrix M in `GoldElixir` is used to find each pseudo-cupsideal point $\vec{\lambda}$
Let G be the Coxeter group
Let l be the set of positive roots of G
Define a function to determine conjugacy in Coxeter group G as
def `IsConjugate(`G, $\vec{\mathbf{v}}_1$, $\vec{\mathbf{v}}_2$`):`
  for $g \in G$ **do**
    if $\vec{\mathbf{v}}_1 \sim_G \vec{\mathbf{v}}_2$ **then**
      return *True*
      break
    end
  end
  return *False*
Let b be the rank of Coxeter group G
Define a function, using `IsConjugate(`G, $\vec{\mathbf{v}}_1$, $\vec{\mathbf{v}}_2$`)`, to form conjugate bases out of combinations of b positive roots from Δ^+ as
def `MakeConjClasses(`l, G`):`
  return *a list of* `ConjClasses` *of roots*
Let s be the number of conjugate classes of roots in ConjClasses
foreach *set of* $(\vec{\alpha}_1,\ldots,\vec{\alpha}_b)$ *of* b *roots* **do**
  Form binding roots to build the following matrix:

$$M = \left[\begin{array}{c|ccc} \vec{\alpha}_1^{\intercal} & & & \\ \hline \vdots & \vec{\mathbf{k}}_1 & \cdots & \vec{\mathbf{k}}_s \\ \hline \vec{\alpha}_n^{\intercal} & & & \end{array}\right] \tag{24}$$

  where $\vec{\mathbf{k}}_j$ is the column vector whose i^{th} entry is 1 if $\vec{\alpha}_i$ belongs to conjugacy class $i \in s = \{1,\ldots,s\}$, or 0 otherwise
  Store each M in a list `ComplicatedMatrix`
  return `ComplicatedMatrix`
end

Algorithm 1: (continued)

Let the `LHS` and `RHS` respectively be

$$\texttt{LHS} = \begin{bmatrix} \vec{\alpha}_1^{\intercal} \\ \hline \vdots \\ \hline \vec{\alpha}_n^{\intercal} \end{bmatrix} \qquad \texttt{RHS} = \begin{bmatrix} | & & | \\ \vec{\mathbf{k}}_1 & \dots & \vec{\mathbf{k}}_s \\ | & & | \end{bmatrix} \tag{25}$$

foreach M *in* `ComplicatedMatrix` **do**
  Calculate rref(M)
  if *rref(M) has full rank* **then**
    Store the `RHS` of rref(M) in list `StoredRHS`
  end
  return `StoredRHS`
end
`/* Store only the first representatives of the conjugate classes */`
foreach `RHS` *in* `StoredRHS` **do**
  if `RHS` *is not conjugate to any earlier matrix in* `StoredRHS` **then**
    Store `RHS` in new list `GoldElixir`
  end
end
return `GoldElixir`

Raw Code of the Algorithm The raw code of the algorithm implemented in SageMath may be found in Appendix A.

How Algorithm Relates to Physics Problem In the output of `GoldElixir`, let the matrix M' represent an element of `GoldElixir`, where M' is the $(n \times s)$ matrix, n is the dimension of the ambient space of G, and s is the number of conjugacy classes or the number of independent coupling constants.

Suppose the coupling constants are $c_1, \dots, c_s$ and the associate root lengths are $l_1, \dots, l_s$. Thus, each M is encoding a $\vec{\lambda}$, where each $\vec{\lambda}$ is given by

$$\vec{\lambda} = M \begin{bmatrix} c_1 l_1 \\ \vdots \\ c_s l_s \end{bmatrix} \tag{26}$$

The coupling constants $c_1, \dots, c_s$ for any given general Coxeter group G, are arbitrary so that the algorithm can help calculate them.

3.3 Methodology

So, how do we find pseudo-cupsidal points? Suppose $\vec{\lambda}$ is pseudo-cupsidal. Then, $\mathcal{B}(\vec{\lambda})$ spans $\mathbb{R}\Delta$. By dropping redundant vectors, we get a basis for $\mathbb{R}\Delta$ contained in $\mathcal{B}(\vec{\lambda}) \subseteq \Delta$. These are the steps:

1. Search for bases contained in Δ.

2. Decompose $\vec{\lambda}$ into a component $\text{proj}_{\mathbb{R}\Delta}(\vec{\lambda})$ in $\mathbb{R}\Delta$ and perpendicular component $\vec{\lambda^{\perp}}$ in $(\mathbb{R}\Delta)^{\perp}$. In the test cases used, it happened that $\mathbb{R}\Delta = \mathbb{R}^n$, so $\text{proj}_{\mathbb{R}\Delta}(\vec{\lambda}) = \vec{\lambda}$.

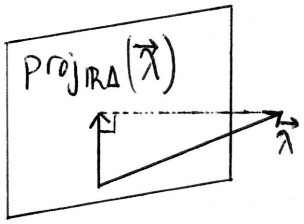

Figure 6: Picture of $\text{proj}(\vec{\lambda})$ in $\mathbb{R}\Delta$

3. Suppose $(\alpha_1, \ldots, \alpha_n)$ is a basis contained in $\mathcal{B}(\vec{\lambda})$. Then compute $\vec{\lambda}$ by row reducing:

$$\begin{cases} \vec{\alpha}_1 \cdot \vec{\lambda} = -c_{\vec{\alpha}_1} |\vec{\alpha}_1| \\ \vec{\alpha}_2 \cdot \vec{\lambda} = -c_{\vec{\alpha}_2} |\vec{\alpha}_2| \\ \qquad \vdots \\ \vec{\alpha}_n \cdot \vec{\lambda} = -c_{\vec{\alpha}_n} |\vec{\alpha}_n| \end{cases} \tag{27}$$

This is the matrix called `ComplicatedMatrix` from the algorithm, which was

$$\texttt{ComplicatedMatrix} = \left[\begin{array}{c|ccc} \vec{\alpha}_1^{\intercal} & & & \\ \hline \vdots & \vec{\mathbf{c}}_1 & \cdots & \vec{\mathbf{c}}_n \\ \hline \vec{\alpha}_n^{\intercal} & & & \end{array}\right] \tag{28}$$

4 Results

4.1 Small Test Cases

Type A_1 For the INPUT, we have $s = 1$, $\Delta = \{-1, 1\}$ or $\Delta^+ = \{1\}$, $g = \{\iota, \sigma\}$, and associate length $l_1 = 1$. The OUTPUT[1] is the vector (1). (There is only 1 conjugacy class. In this case, there happens to be only 1 mirror.) Thus, the pseudo-cupsidal point $\vec{\lambda}$ is

$$\vec{\lambda} = M[c_1 l_1] = [1][c_1 \cdot 1] \tag{29}$$

$$\vec{\lambda} = [c_1] \tag{30}$$

Recall the spherical solution for A_1 (and $c_1 \neq 0$), which would be

$$\sigma_{\vec{\lambda}}^{A_1} = \frac{1}{2}\left[\frac{c_1 + c_1}{c_1} \mathrm{e}^{c_1 x} + \frac{-c_1 + c_1}{c_1} \mathrm{e}^{-c_1 x}\right] \tag{31}$$

$$\Rightarrow \sigma_{\vec{\lambda}}^{A_1} = \frac{1}{2} \cdot 2\mathrm{e}^{c_1 x} \tag{32}$$

$$\Rightarrow \sigma_{\vec{\lambda}}^{A_1} = \mathrm{e}^{c_1 x} \tag{33}$$

[1] which may be found on GitHub (please ignore GitHub's attempt to output the Sage object in ASCII)

This is a valid bound state in the dominant chamber only if $c_1 < 0 \; \forall \; x > 0$.

Reflecting over the y-axis, the result is $c_1 > 0 \; \forall \; x < 0$ for the remaining chamber in A_1. The entire result is

$$\sigma_{\vec{\lambda}}^{A_1}(x) = \begin{cases} \mathrm{e}^{c_1 x} & \forall \; x < 0 \\ \mathrm{e}^{-c_1 x} & \forall \; x > 0 \end{cases} \quad \text{for some positive constant } c_1 \tag{34}$$

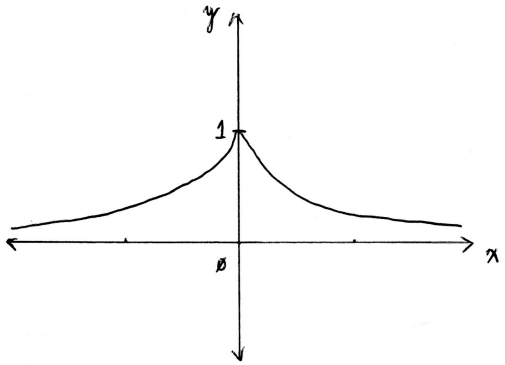

Figure 7: The spherical solution $\sigma_{\vec{\lambda}}^{G}(x)$ in Coxeter group A_1 from Eqn 34

The OUTPUT for A_1 is used to obtain $\sigma_{\vec{\lambda}}^{G}(x)$, which is the same as the solution for the bound state of the attractive Dirac-delta potential found in most undergraduate quantum mechanics textbooks.

Type C_2 The INPUT is the Coxeter group type C_2. The OUTPUT[2] is a (2×2) matrix list with 12 matrices, the associate lengths are $l_1 = 1$ and $l_2 = \sqrt{2}$, and $a = \sqrt{2}$.

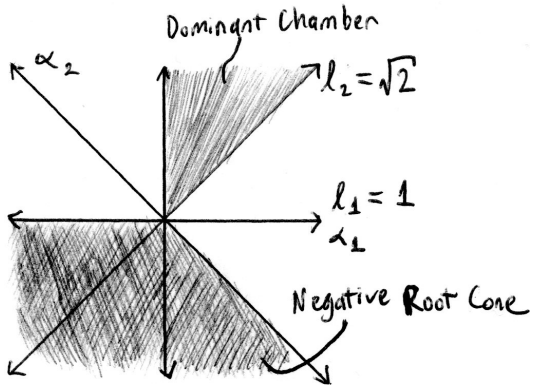

Figure 8: Illustration of type C_2

Take the first element of `GoldElixir` as the OUTPUT, say M_1, where

$$M_1 = \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\sqrt{2} & 1 \end{bmatrix} \tag{35}$$

[2]which may be found on GitHub

There are 2 conjugate root classes with associate root lengths $l_1 = 1$ and $l_2 = \sqrt{2}$, which correspond to the first and second columns of M_1, respectively. The pseudo-cupsidal point $\vec{\lambda}$ associated with M_1 is

$$\vec{\lambda} = M_1 \begin{bmatrix} c_1 l_1 \\ c_2 l_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix} \begin{bmatrix} c_1 \cdot 1 \\ c_2 \cdot \sqrt{2} \end{bmatrix} \tag{36}$$

$$\vec{\lambda} = \begin{bmatrix} c_1 \\ -\sqrt{2}c_1 + \sqrt{2}c_2 \end{bmatrix} \tag{37}$$

This may be rewritten in a form easier for general Coxeter group G as follows: since there is a slight complication of Sage using the coordinate system of $\vec{\alpha}_1$ and $\vec{\alpha}_2$, the pseudo-cupsidal point can undergo a change of basis and $\vec{\lambda}$ in the standard basis will be

$$\vec{\lambda} = c_1 l_1 (1\vec{\alpha}_1 - \sqrt{2}\vec{\alpha}_2) + c_2 l_2 (0\vec{\alpha}_1 - \vec{\alpha}_2) \tag{38}$$

The resulting $\vec{\lambda}$, along with the constraint of $x_2 \leqslant 0 \cap x_1 + x_2 \leqslant 0$ imposed by the negative root cone in Figure 8, may be substituted in the spherical wavefunction of Eqn 6 to obtain the spherical solution for M_1.

4.2 Large Test Case

For the large test case, the Coxeter group used was type F_4 as the INPUT. The OUTPUT was a list of 13,717 pseudo-cupsidal points. The Sage object file for `GoldElixir` may be found on GitHub.

References

[Bat07] Murray T. Batchelor, *The Bethe ansatz after 75 years*, Physics Today **60** (2007), no. 1, 36–40.

[EOS06] E. Emsiz, E. M. Opdam, and J. V. Stokman, *Periodic integrable systems with delta-potentials*, Comm. Math. Phys. **264** (2006), no. 1, 191–225. MR 2212221

[Gau71] M. Gaudin, *Boundary energy of a Bose gas in one dimension*, Phys. Rev. A **4** (1971), 386–394.

[Gut82] Eugene Gutkin, *Integrable systems with delta-potential*, Duke Math. J. **49** (1982), no. 1, 1–21. MR 650365

[HO97a] G. J. Heckman and E. M. Opdam, *Erratum: "Yang's system of particles and Hecke algebras"*, Ann. of Math. (2) **146** (1997), no. 3, 749–750. MR 1491452

[HO97b] ———, *Yang's system of particles and Hecke algebras*, Ann. of Math. (2) **145** (1997), no. 1, 139–173. MR 1432038

[LL63] Elliott H. Lieb and Werner Liniger, *Exact analysis of an interacting Bose gas. I. The general solution and the ground state*, Phys. Rev. (2) **130** (1963), 1605–1616. MR 0156630

[McG64] J. B. McGuire, *Study of exactly soluble one-dimensional N-body problems*, J. Mathematical Phys. **5** (1964), 622–636. MR 0161667

[OJ15] Maxim Olshanii and Steven G Jackson, *An exactly solvable quantum four-body problem associated with the symmetries of an octacube*, New Journal of Physics **17** (2015), no. 10, 105005.

A Pseudo-cupsidal Point Algorithm

A.1 Note

The following raw code of the algorithm may also be found on GitHub.

A.2 Raw Code of the Algorithm

```
# Initialize F4 - may be replaced with any other Coxeter group
G = CoxeterGroup(['F',4], implementation='reflection')
l = list(G.roots()) # List of all roots (in G)

# Define a conjugacy function in Coxeter group G
def IsConjugate(G, v1, v2):
    for g in G:
        if g * v1 == v2:
            return True
    return False

# Define function for forming conjugate classes in G
def MakeConjClasses(l, G):
    classes = []
    for x in l:
        found = False
        for c in classes:
            if IsConjugate(G, x, c[0]):
                c.append(x)
                found = True
                break
        if not found:
            classes.append([x])
    return classes

# Form the conjugacy classes from list of roots l in G
ConjClasses = MakeConjClasses(l, G)

d = {} # Initialize empty dictionary
# Create look-up dictionary (for speed)
```

```
for r in l:
    for c in ConjClasses:
        if r in c:
            d[tuple(r)] = c

def ObtainMatOfRHS(G, ConjClasses):  # Define function to obtain desired
↪ RHS's
    RedRootComb = Combinations(list(G.roots()), G.rank()).list() # This
    ↪ list will contain redun root comb's
    PreMatList = [] # Initialize empty list for preprocessed matrices
    for i in range(0, len(RedRootComb)):
        temp = matrix(RedRootComb[i]) # Creating a matrix out of rows
        PreMatList.append(temp) # Appending temp to PreMatList list
    def FormComplicatedMatrix(G, PreMatList):
        ComplicatedMatList = [] # Initialize empty list
        for m in PreMatList:
            TempVectList = [] # Initialize to store
            for c in ConjClasses:
                TempScalar = [] # Initialize empty list
                for i in range(0, m.nrows()):
                    if d[tuple(m.row(i))] == c: # If row is in conj class
                        TempScalar.append(1) # i-th entry is 1
                    else:
                        TempScalar.append(0) # i-th entry is 0
                TempVect = vector(G.base_ring(), TempScalar) # Form a
                ↪ vector from 0's & 1's
                TempVectList.append(TempVect) # Append TempVect on the RHS
                ↪ on deck
            for v in TempVectList: # for each v in TempVectList
                m = m.augment(v) # Augment v to  RHS of ea. matrix m in
                ↪ PreMatList
            ComplicatedMatList.append(m) # Add modified m to
            ↪ ComplicatedMatList
        return ComplicatedMatList
    ComplicatedMatList = FormComplicatedMatrix(G, PreMatList) # Form this
    RowReducedList = [] # Initialize empty list for next part
    for m in ComplicatedMatList:
        if m.rref().submatrix(0, 0, G.rank(), G.rank()) ==
        ↪ identity_matrix(G.rank()): # If LHS part has full rank
            RowReducedList.append(m.rref()) # Then store the rref(m)
    StoredRHS = [] # Initialize empty list
    for m in RowReducedList:
        StoredRHS.append(m.submatrix(0, G.rank(), G.rank(),
        ↪ len(ConjClasses))) # Storing only the RHS's of each m in
        ↪ RowReducedList
```

```
    return StoredRHS

# Use function to obtain RHS's
StoredRHS = ObtainMatOfRHS(G, ConjClasses)

# Function to store only the 1st conj representatives of list l (in G)
def ConClassReps(l, G):
    result = []
    while len(l)>0:
        print "Found %d classes so far."%len(result) # Display messages to
        ↪  indicate progress
        print "%d elements left to classify."%len(l)
        print
        result.append(l[0])
        i = 0
        for g in G:
            print i, len(l)
            i += 1
            the_conjugate = g * result[-1]
            newlist = [x for x in l if x != the_conjugate]
            l = newlist
    return result

GoldElixir = ConClassReps(StoredRHS, G) # Return the conj class
↪  representatives
save(GoldElixir, "GoldElixirF4") # Saves the specified Sage object as
↪  "filename.sobj"
```