Carnegie Mellon University

Progress in Formalising Sphere Packing in Dimension 8

International Congress on Mathematical Software 2026

Sidharth Hariharan
based on joint work with Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Auguste Poiroux, Maryna Viazovska, and community contributors

23 July, 2026

sidharthhariharan@cmu.edu

Contents

Setting Up a Formalisation Project

What & Why

Formal Verification

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formalisation

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formalisation

What & Why

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formalisation

To me, ‘formalising mathematics’ means strictly more than ‘formally verifying mathematics’: it involves organising it into an extensible, reusable artefact that communicates insight.

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formalisation

To me, ‘formalising mathematics’ means strictly more than ‘formally verifying mathematics’: it involves organising it into an extensible, reusable artefact that communicates insight.

The best example is `Mathlib`, an extensive community-driven library of mathematics formalised in Lean. But there are many more examples, most of which treat Mathlib as a dependency.

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formalisation

To me, ‘formalising mathematics’ means strictly more than ‘formally verifying mathematics’: it involves organising it into an extensible, reusable artefact that communicates insight.

The best example is `Mathlib`, an extensive community-driven library of mathematics formalised in Lean. But there are many more examples, most of which treat Mathlib as a dependency.

Why?

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formalisation

To me, ‘formalising mathematics’ means strictly more than ‘formally verifying mathematics’: it involves organising it into an extensible, reusable artefact that communicates insight.

The best example is `Mathlib`, an extensive community-driven library of mathematics formalised in Lean. But there are many more examples, most of which treat Mathlib as a dependency.

Why?

In undergrad, I had a very interesting professor who wears even more interesting trousers.

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formalisation

To me, ‘formalising mathematics’ means strictly more than ‘formally verifying mathematics’: it involves organising it into an extensible, reusable artefact that communicates insight.

The best example is `Mathlib`, an extensive community-driven library of mathematics formalised in Lean. But there are many more examples, most of which treat Mathlib as a dependency.

Why?

In undergrad, I had a very interesting professor who wears even more interesting trousers.

He once told me,

Formal Verification

Interactive theorem provers like Lean, Rocq, Isabelle, Agda, Metamath etc have a long history of being used to check the correctness of mathematical arguments.

Proofs are represented as code, which (at least in Lean) is checked by a small, trusted kernel.

Formalisation

To me, 'formalising mathematics' means strictly more than 'formally verifying mathematics': it involves organising it into an extensible, reusable artefact that communicates insight.

The best example is `Mathlib`, an extensive community-driven library of mathematics formalised in Lean. But there are many more examples, most of which treat Mathlib as a dependency.

Why?

In undergrad, I had a very interesting professor who wears even more interesting trousers.

He once told me,

> "Formalising mathematics is a great way to learn it."

Setting up the Sphere Packing Project

Setting up the Sphere Packing Project

28 February 2024 — First commit: SH and MV launch project

Setting up the Sphere Packing Project

28 February 2024 — First commit: SH and MV launch project

Late July 2024 — Stable definition of a sphere packing

Setting up the Sphere Packing Project

28 February 2024 — First commit: SH and MV launch project

Late July 2024 — Stable definition of a sphere packing

December 2024 — Many key steps understood (non-formally)

Setting up the Sphere Packing Project

28 February 2024	First commit: SH and MV launch project
Late July 2024	Stable definition of a sphere packing
December 2024	Many key steps understood (non-formally)
April 2025	Many things stated, some proven. First dabble in metaprogramming.

Setting up the Sphere Packing Project

28 February 2024	First commit: SH and MV launch project
Late July 2024	Stable definition of a sphere packing
December 2024	Many key steps understood (non-formally)
April 2025	Many things stated, some proven. First dabble in metaprogramming.
13 June 2025	Project opened to public contributions

Setting up the Sphere Packing Project

28 February 2024	First commit: SH and MV launch project
Late July 2024	Stable definition of a sphere packing
December 2024	Many key steps understood (non-formally)
April 2025	Many things stated, some proven. First dabble in metaprogramming.
13 June 2025	Project opened to public contributions
December 2025	Steady community engagement, significant progress, interest from AI companies

Setting up the Sphere Packing Project

28 February 2024	First commit: SH and MV launch project
Late July 2024	Stable definition of a sphere packing
December 2024	Many key steps understood (non-formally)
April 2025	Many things stated, some proven. First dabble in metaprogramming.
13 June 2025	Project opened to public contributions
December 2025	Steady community engagement, significant progress, interest from AI companies
February 2026	**Project milestone: formal verification**

Division of Labour

Division of Labour

We have different maintainers for different parts of the project:

Division of Labour

We have different maintainers for different parts of the project:

1. Sphere packing theory: SH, GM, BM

Division of Labour

We have different maintainers for different parts of the project:

1. Sphere packing theory: SH, GM, BM
2. Fourier analysis and lattices: SH, BM

Division of Labour

We have different maintainers for different parts of the project:

1. Sphere packing theory: SH, GM, BM
2. Fourier analysis and lattices: SH, BM
3. Modular form theory: CB, SL

Division of Labour

We have different maintainers for different parts of the project:

1. Sphere packing theory: SH, GM, BM
2. Fourier analysis and lattices: SH, BM
3. Modular form theory: CB, SL
4. Construction of the magic function: SH, SL, BM

Division of Labour

We have different maintainers for different parts of the project:

1. Sphere packing theory: SH, GM, BM
2. Fourier analysis and lattices: SH, BM
3. Modular form theory: CB, SL
4. Construction of the magic function: SH, SL, BM
5. Quasimodular forms and modular form inequalities: SL

Key Design Choices

What is a sphere packing, really?

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024):

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024):

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

Second attempt (4 July, 2024):

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

Second attempt (4 July, 2024):

```
def Discrete (X : Set V) : Prop := (Countable X) ∧ (∀ x ∈ X, ∃ ε > 0, ∀ y ∈ X, y ≠ x →
    Euclidean.dist x y > ε)
def Centres (X : Set V) : Prop := (Discrete d X) ∧ (∀ x ∈ X, ∀ y ∈ X, x ≠ y →
    Euclidean.dist x y ≥ 2)
def Packing (X P : Set V) : Prop := (Centres d X) ∧ (P = ⋃ x ∈ X, {y | Euclidean.dist x y <
    1})
def isPacking (P : Set V) : Prop := ∃ X, Packing d X P
def LatticePacking ( P : Set V) : Prop := (Packing d  P) ∧ (is_lattice )
def isLatticePacking (P : Set V) : Prop := ∃ , LatticePacking d  P
```

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

Second attempt (4 July, 2024): A soup of unbundled predicates!

```
def Discrete (X : Set V) : Prop := (Countable X) ∧ (∀ x ∈ X, ∃ ε > 0, ∀ y ∈ X, y ≠ x →
    Euclidean.dist x y > ε)
def Centres (X : Set V) : Prop := (Discrete d X) ∧ (∀ x ∈ X, ∀ y ∈ X, x ≠ y →
    Euclidean.dist x y ≥ 2)
def Packing (X P : Set V) : Prop := (Centres d X) ∧ (P = ⋃ x ∈ X, {y | Euclidean.dist x y <
    1})
def isPacking (P : Set V) : Prop := ∃ X, Packing d X P
def LatticePacking ( P : Set V) : Prop := (Packing d  P) ∧ (is_lattice )
def isLatticePacking (P : Set V) : Prop := ∃ , LatticePacking d  P
```

What is a sphere packing, really?

Here's the definition of a sphere packing from the original paper.

> *Let $X \subseteq \mathbb{R}^d$ be a discrete set of points such that $\|x - y\| \geq 2$ for any distinct $x, y \in X$. Then the union $\mathcal{P} = \bigcup_{x \in X} B_d(x, 1)$ is a sphere packing.*

First attempt (1 March, 2024): A packing forgets its own centres and radius!

```
def SpherePacking {P : Set V} (hP : Countable P) {r : ℝ} (hr : r > 0)
  (hPr : ∀ p₁ p₂ : V, p₁ ∈ P → p₂ ∈ P → p₁ ≠ p₂ → Euclidean.dist p₁ p₂ > 2*r) : Set V :=
  {y | ∃ p : V, p ∈ P ∧ y ∈ ball p r}
```

Second attempt (4 July, 2024): A soup of unbundled predicates! Centres *still* not recoverable!

```
def Discrete (X : Set V) : Prop := (Countable X) ∧ (∀ x ∈ X, ∃ ε > 0, ∀ y ∈ X, y ≠ x →
    Euclidean.dist x y > ε)
def Centres (X : Set V) : Prop := (Discrete d X) ∧ (∀ x ∈ X, ∀ y ∈ X, x ≠ y →
    Euclidean.dist x y ≥ 2)
def Packing (X P : Set V) : Prop := (Centres d X) ∧ (P = ⋃ x ∈ X, {y | Euclidean.dist x y <
    1})
def isPacking (P : Set V) : Prop := ∃ X, Packing d X P
def LatticePacking ( P : Set V) : Prop := (Packing d  P) ∧ (is_lattice )
def isLatticePacking (P : Set V) : Prop := ∃ , LatticePacking d  P
```

What is a sphere packing, really?

What is a sphere packing, really?

Third attempt (15 July, 2024):

What is a sphere packing, really?

Third attempt (15 July, 2024):

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
    SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { :
    AddSubgroup V} [DiscreteTopology ] [isLattice' ] where
  periodic : ∀ x ∈ X, ∀ y ∈ , x + y ∈ X
```

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
    SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { :
    AddSubgroup V} [DiscreteTopology ] [isLattice' ] where
  periodic : ∀ x ∈ X, ∀ y ∈ , x + y ∈ X
```

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
     SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { :
     AddSubgroup V} [DiscreteTopology ] [isLattice' ] where
  periodic : ∀ x ∈ X, ∀ y ∈ , x + y ∈ X
```

Fourth attempt (30 July, 2024):

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
    SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { :
    AddSubgroup V} [DiscreteTopology ] [isLattice' ] where
  periodic : ∀ x ∈ X, ∀ y ∈ , x + y ∈ X
```

Fourth attempt (30 July, 2024):

```
structure SpherePacking (d : ℕ) where
  centers : Set (EuclideanSpace ℝ (Fin d))
  radius : ℝ
  radius_pos : 0 < radius
  centers_dist : Pairwise (radius ≤ dist · · : centers → centers → Prop)
structure PeriodicSpherePacking (d : ℕ) extends SpherePacking d where
   : AddSubgroup (EuclideanSpace ℝ (Fin d))
  _discrete : DiscreteTopology  := by infer_instance
  _lattice : IsZlattice ℝ  := by infer_instance
  _action : ∀ {|x y|}, x ∈  → y ∈ centers → x + y ∈ centers
```

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
    SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { :
    AddSubgroup V} [DiscreteTopology ] [isLattice' ] where
  periodic : ∀ x ∈ X, ∀ y ∈ , x + y ∈ X
```

Fourth attempt (30 July, 2024): Our first stable definition!

```
structure SpherePacking (d : ℕ) where
  centers : Set (EuclideanSpace ℝ (Fin d))
  radius : ℝ
  radius_pos : 0 < radius
  centers_dist : Pairwise (radius ≤ dist · · : centers → centers → Prop)
structure PeriodicSpherePacking (d : ℕ) extends SpherePacking d where
   : AddSubgroup (EuclideanSpace ℝ (Fin d))
  _discrete : DiscreteTopology  := by infer_instance
  _lattice : IsZlattice ℝ  := by infer_instance
  _action : ∀ {|x y|}, x ∈  → y ∈ centers → x + y ∈ centers
```

What is a sphere packing, really?

Third attempt (15 July, 2024): There's no canonical packing structure on a set of centres!

```
class SpherePackingCentres (X : Set V) [DiscreteTopology X] where
  nonoverlapping : ∀ x ∈ X, ∀ y ∈ X, x ≠ y → Euclidean.dist x y ≥ 2
class LatticePackingCentres (X : AddSubgroup V) [DiscreteTopology X] [isLattice' X] extends
    SpherePackingCentres d X
class PeriodicPackingCentres (X : Set V) [DiscreteTopology X] [SpherePackingCentres d X] { :
    AddSubgroup V} [DiscreteTopology ] [isLattice' ] where
  periodic : ∀ x ∈ X, ∀ y ∈ , x + y ∈ X
```

Fourth attempt (30 July, 2024): Our first stable definition! Thank you Gareth Ma!

```
structure SpherePacking (d : ℕ) where
  centers : Set (EuclideanSpace ℝ (Fin d))
  radius : ℝ
  radius_pos : 0 < radius
  centers_dist : Pairwise (radius ≤ dist · · : centers → centers → Prop)
structure PeriodicSpherePacking (d : ℕ) extends SpherePacking d where
   : AddSubgroup (EuclideanSpace ℝ (Fin d))
  _discrete : DiscreteTopology  := by infer_instance
  _lattice : IsZlattice ℝ  := by infer_instance
  _action : ∀ {|x y|}, x ∈  → y ∈ centers → x + y ∈ centers
```

Defining the Magic Function

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument.

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

$$a(x) := \int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right) (z+1)^2 \, e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right) (z-1)^2 \, e^{\pi i \|x\|^2 z} \, dz$$
$$- 2 \int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz + 2 \int_{i}^{i\infty} \phi_0(z) \, e^{\pi i \|x\|^2 z} \, dz$$

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

$$
\begin{aligned}
a(x) := &\int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right) (z+1)^2 \, e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right) (z-1)^2 \, e^{\pi i \|x\|^2 z} \, dz \\
&- 2\int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz + 2\int_{i}^{i\infty} \phi_0(z) \, e^{\pi i \|x\|^2 z} \, dz \\
b(x) := &\int_{-1}^{i} \psi_S\left(\frac{-1}{z+1}\right) (z+1)^2 \, e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \psi_S\left(\frac{-1}{z-1}\right) (z-1)^2 \, e^{\pi i \|x\|^2 z} \, dz \\
&- 2\int_{0}^{i} \psi_S\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz - 2\int_{i}^{i\infty} \psi_S(z) \, e^{\pi i \|x\|^2 z} \, dz
\end{aligned}
$$

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

$$\begin{aligned} a(x) := &\int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right)(z+1)^2\, e^{\pi i\|x\|^2 z}\, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right)(z-1)^2\, e^{\pi i\|x\|^2 z}\, dz \\ &- 2\int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2\, e^{\pi i\|x\|^2 z}\, dz + 2\int_{i}^{i\infty} \phi_0(z)\, e^{\pi i\|x\|^2 z}\, dz \\ b(x) := &\int_{-1}^{i} \psi_S\left(\frac{-1}{z+1}\right)(z+1)^2\, e^{\pi i\|x\|^2 z}\, dz + \int_{1}^{i} \psi_S\left(\frac{-1}{z-1}\right)(z-1)^2\, e^{\pi i\|x\|^2 z}\, dz \\ &- 2\int_{0}^{i} \psi_S\left(\frac{-1}{z}\right) z^2\, e^{\pi i\|x\|^2 z}\, dz - 2\int_{i}^{i\infty} \psi_S(z)\, e^{\pi i\|x\|^2 z}\, dz \end{aligned}$$

a and b have three very special properties: they are **radial and Schwartz**, they are (respectively) **Fourier ± 1-eigenfunctions**, and they have **double zeroes at E_8 lattice points**.

Defining the Magic Function

The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

$$
\begin{aligned}
a(x) := &\int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right) (z+1)^2 \, e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right) (z-1)^2 \, e^{\pi i \|x\|^2 z} \, dz \\
&- 2\int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz + 2\int_{i}^{i\infty} \phi_0(z) \, e^{\pi i \|x\|^2 z} \, dz \\
b(x) := &\int_{-1}^{i} \psi_S\left(\frac{-1}{z+1}\right) (z+1)^2 \, e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \psi_S\left(\frac{-1}{z-1}\right) (z-1)^2 \, e^{\pi i \|x\|^2 z} \, dz \\
&- 2\int_{0}^{i} \psi_S\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz - 2\int_{i}^{i\infty} \psi_S(z) \, e^{\pi i \|x\|^2 z} \, dz \\
g(x) := &\frac{\pi i}{8640} a(x) + \frac{i}{240\pi} b(x)
\end{aligned}
$$

a and b have three very special properties: they are **radial and Schwartz**, they are (respectively) **Fourier ± 1-eigenfunctions**, and they have **double zeroes at E_8 lattice points**.

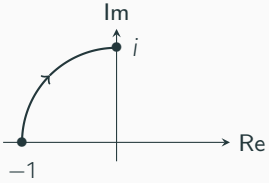
The magic function $g : \mathbb{R}^8 \to \mathbb{R}$ is a very special function that is crucial to Viazovska's argument. g is defined as a linear combination of two functions a and b, which are defined in terms of quasimodular forms.

$$\begin{aligned} a(x) := &\int_{-1}^{i} \phi_0\left(\frac{-1}{z+1}\right) (z+1)^2 \, e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \phi_0\left(\frac{-1}{z-1}\right) (z-1)^2 \, e^{\pi i \|x\|^2 z} \, dz \\ &- 2\int_{0}^{i} \phi_0\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz + 2\int_{i}^{i\infty} \phi_0(z) \, e^{\pi i \|x\|^2 z} \, dz \\ b(x) := &\int_{-1}^{i} \psi_S\left(\frac{-1}{z+1}\right) (z+1)^2 \, e^{\pi i \|x\|^2 z} \, dz + \int_{1}^{i} \psi_S\left(\frac{-1}{z-1}\right) (z-1)^2 \, e^{\pi i \|x\|^2 z} \, dz \\ &- 2\int_{0}^{i} \psi_S\left(\frac{-1}{z}\right) z^2 \, e^{\pi i \|x\|^2 z} \, dz - 2\int_{i}^{i\infty} \psi_S(z) \, e^{\pi i \|x\|^2 z} \, dz \\ g(x) := &\frac{\pi i}{8640} a(x) + \frac{i}{240\pi} b(x) \end{aligned}$$

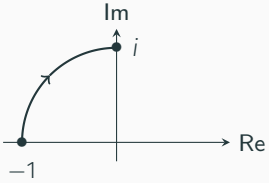
a and b have three very special properties: they are **radial and Schwartz**, they are (respectively) **Fourier ± 1-eigenfunctions**, and they have **double zeroes at E_8 lattice points**. This endows g with unique, 'magic' properties.

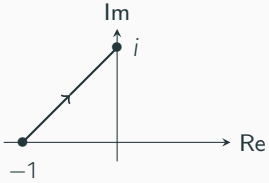
Choosing Contours

Choosing Contours

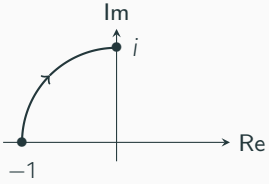


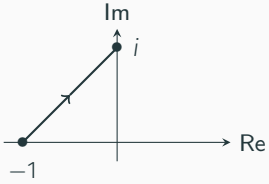
Choosing Contours

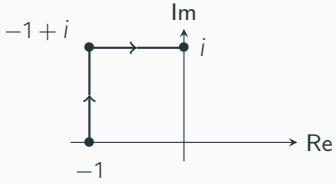




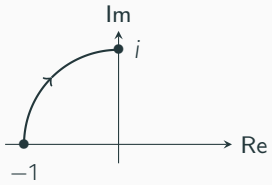
Choosing Contours

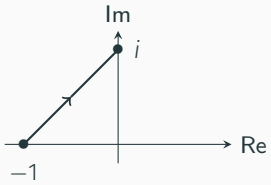


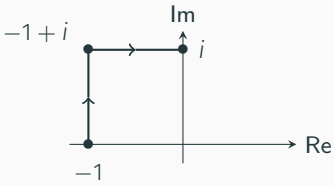




Choosing Contours







theorem `Complex.integral_boundary_rect_eq_zero_of_differentiable_on_off_countable` source

```
    {E : Type u} [NormedAddCommGroup E] [NormedSpace ℂ E] (f : ℂ → E) (z w : ℂ) (s : Set ℂ)
    (hs : s.Countable) (Hc : ContinuousOn f (Set.uIcc z.re w.re ×ℂ Set.uIcc z.im w.im))
    (Hd :

     ∀ x ∈ Set.Ioo (min z.re w.re) (max z.re w.re) ×ℂ Set.Ioo (min z.im w.im) (max z.im w.im) \ s,
      DifferentiableAt ℂ f x
     )
    :
(((∫ (x : ℝ) in z.re..w.re, f (↑x + ↑z.im * I)) - ∫ (x : ℝ) in z.re..w.re, f (↑x + ↑w.im * I)) + I •
    ∫ (y : ℝ) in z.im..w.im, f (↑w.re + ↑y * I)) - I • ∫ (y : ℝ) in z.im..w.im, f (↑z.re + ↑y * I)
    = 0
```

Cauchy-Goursat theorem for a rectangle: the integral of a complex differentiable function over the boundary of a rectangle equals zero. More precisely, if `f` is continuous on a closed rectangle and is complex differentiable at all but countably many points of the corresponding open rectangle, then its integral over the boundary of the rectangle equals zero.

Defining the Magic Function in Lean

Defining the Magic Function in Lean

```
def ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp (π * I * r * (z : ℂ))
                                    ⋮
def ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp (π * I * r * (z : ℂ))
```

Defining the Magic Function in Lean

```
def ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp (π * I * r * (z : ℂ))
                                    ⋮
def ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp (π * I * r * (z : ℂ))
```

```
def ₁ : ℝ → ℂ := fun t ↦ I * ₁' r (z₁' t)
                    ⋮
def ₆ : ℝ → ℂ := fun t ↦ I * ₆' r (z₆' t)
```

Defining the Magic Function in Lean

```
def ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp (π * I * r * (z : ℂ))
                                    ⋮
def ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp (π * I * r * (z : ℂ))
```

```
def ₁ : ℝ → ℂ := fun t ↦ I * ₁' r (z₁' t)
                    ⋮
def ₆ : ℝ → ℂ := fun t ↦ I * ₆' r (z₆' t)
```

```
def I₁' : ℝ → ℂ := fun x ↦ ∫ t in (0 : ℝ)..1, ₁ x t
                    ⋮
def I₆' : ℝ → ℂ := fun x ↦ 2 * ∫ t in Ici (1 : ℝ), ₆ x t
```

Defining the Magic Function in Lean

```
def ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp (π * I * r * (z : ℂ))
                                    ⋮
def ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp (π * I * r * (z : ℂ))
```

```
def ₁ : ℝ → ℂ := fun t ↦ I * ₁' r (z₁' t)
                    ⋮
def ₆ : ℝ → ℂ := fun t ↦ I * ₆' r (z₆' t)
```

```
def I₁' : ℝ → ℂ := fun x ↦ ∫ t in (0 : ℝ)..1, ₁ x t
                    ⋮
def I₆' : ℝ → ℂ := fun x ↦ 2 * ∫ t in Ici (1 : ℝ), ₆ x t
```

```
def I₁ : V → ℂ := fun x ↦ I₁' (‖x‖ ^ 2)
                    ⋮
def I₆ : V → ℂ := fun x ↦ I₆' (‖x‖ ^ 2)
```

Defining the Magic Function in Lean

```
def φ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp (π * I * r * (z : ℂ))
                                    ⋮
def φ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp (π * I * r * (z : ℂ))
```

```
def Φ₁ : ℝ → ℂ := fun t ↦ I * φ₁' r (z₁' t)
                                    ⋮
def Φ₆ : ℝ → ℂ := fun t ↦ I * φ₆' r (z₆' t)
```

```
def I₁' : ℝ → ℂ := fun x ↦ ∫ t in (0 : ℝ)..1, Φ₁ x t
                                    ⋮
def I₆' : ℝ → ℂ := fun x ↦ 2 * ∫ t in Ici (1 : ℝ), Φ₆ x t
```

```
def I₁ : V → ℂ := fun x ↦ I₁' (‖x‖ ^ 2)
                                    ⋮
def I₆ : V → ℂ := fun x ↦ I₆' (‖x‖ ^ 2)
```

```
def a' : ℝ → ℂ := fun x ↦ I₁' x + I₂' x + I₃' x + I₄' x +
    I₅' x + I₆' x
```

Defining the Magic Function in Lean

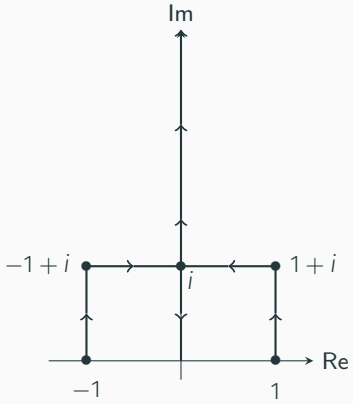
```
def ₁' : ℂ → ℂ := fun z ↦ φ₀'' (-1 / (z + 1)) * (z + 1) ^ 2 * cexp (π * I * r * (z : ℂ))
                                    ⋮
def ₆' : ℂ → ℂ := fun z ↦ φ₀'' (z) * cexp (π * I * r * (z : ℂ))
```

```
def ₁ : ℝ → ℂ := fun t ↦ I * ₁' r (z₁' t)
                    ⋮
def ₆ : ℝ → ℂ := fun t ↦ I * ₆' r (z₆' t)
```

```
def I₁' : ℝ → ℂ := fun x ↦ ∫ t in (0 : ℝ)..1, ₁ x t
                    ⋮
def I₆' : ℝ → ℂ := fun x ↦ 2 * ∫ t in Ici (1 : ℝ), ₆ x t
```

```
def I₁ : V → ℂ := fun x ↦ I₁' (‖x‖ ^ 2)
                    ⋮
def I₆ : V → ℂ := fun x ↦ I₆' (‖x‖ ^ 2)
```

```
def a' : ℝ → ℂ := fun x ↦ I₁' x + I₂' x + I₃' x + I₄' x +
    I₅' x + I₆' x
```



Outcomes and Milestones

Modularity

Modularity

Sphere Packing Theory (SH, GM, BM)

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

Quasimodular Forms and Inequalities (SL)

Modularity

Sphere Packing Theory (SH, GM, BM)

- Definitions of sphere packings, periodic packings, density, sphere packing constant, periodic sphere packing constant
- Density formula for periodic packings
- The theory of the E_8 lattice and the associated sphere packing
- Progress on Cohn-Elkies [2, Theorem 3.1]

Modular Forms (CB, SL)

- Fixing the definition of E_2
- q-expansions of the Eisenstein Series
- H-functions (4th powers of Jacobi Θ-functions) as modular forms
- Definition/API for Δ, relation to η
- Dimension formula for level one

Construction of the Magic Function (SH, BM)

- Definitions and API for a, b, g, their integrands and parametrisations
- Proving `PolyFourierCoeffBound`, a tool to analyse asymptotic behaviour
- Systematically bounding the integrands of a
- Proving slash action relations for ψ_I, ψ_S, ψ_T
- Cauchy-Goursat for unbounded rectangles

Quasimodular Forms and Inequalities (SL)

- Serre derivatives: definitions and API, equivariance and invariance
- Serre derivatives of E_2, E_4, E_6
- Defining F and G, restriction to the imaginary axis, positivity of restrictions
- Modular linear differential equations

Metaprogramming

A `norm_num` extension for $\mathbb{C}$

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta
- Proves `Tendsto (fun z => expr(`f_1 `z, ..., `f_n `z)) l (nhds c)` where `Tendsto` f_i `l (nhds` a_i`)` are known and the expression is continuous

A `norm_num` extension for $\mathbb{C}$

- Developed mainly by Edison Xie with inputs from Heather Macbeth, Bhavik Mehta, SH
- Initially a `conv`-tactic, `norm_numI`
- Simplifies expressions in $\mathbb{C}$ by splitting to real and imaginary parts, running `norm_num` on each part, and recombining them

```
example : (1 + 3.5 + I) * (1 + I) = 7 / 2 +
    11 / 2 * I := by norm_num1
example : (3 + 4 * I)⁻¹ * (3 + 4 * I) = 1 :=
    by norm_num1
example : 1 + I ≠ 0 := by norm_num1
example : 1 + I ≠ 1 + 2 * I := by norm_num1
example : im ((1 + 3 * I)⁻¹) = -0.3 := by
    norm_num1
example : 10 * re ((1 + 3 * I)⁻¹) = 1 := by
    norm_num1
example : (conj (3 + 4 * I) : ℂ) * (3 + 4 *
    I) = 25 := by norm_num1
```

Proving `Tendsto` given atomic limits

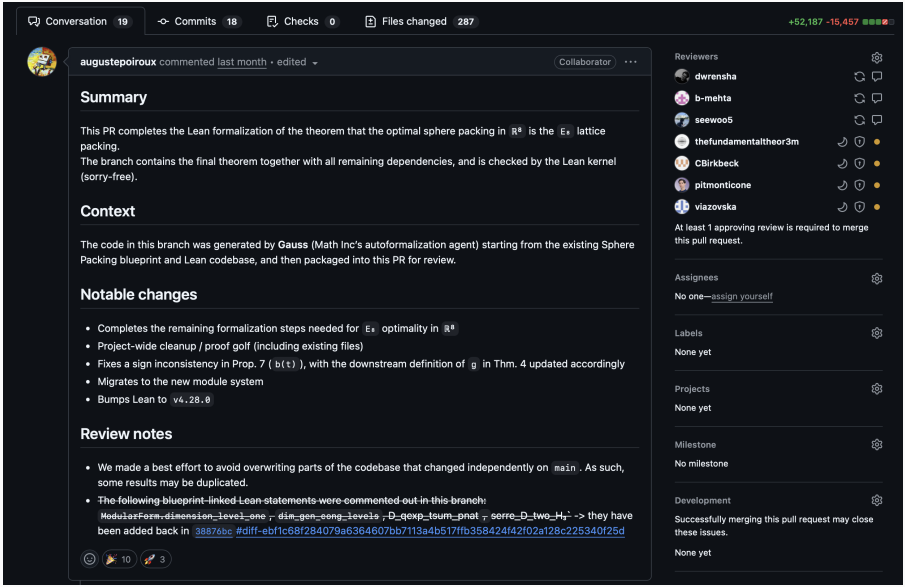
- Developed mainly by Cameron Freer with inputs from Seewoo Lee and Bhavik Mehta
- Proves `Tendsto (fun z => expr(`f_1 `z, ..., `f_n `z)) l (nhds c)` where `Tendsto` f_i `l (nhds` a_i`)` are known and the expression is continuous

```
-- Two atoms, polynomial
example (h₁ : Tendsto f atTop (nhds 0)) (h₂ :
    Tendsto g atTop (nhds 1)) :
    Tendsto (fun z => f z ^ 2 + f z * g z + g
    z ^ 2) atTop (nhds 1) := by tendsto_cont

-- Unused hypotheses don't interfere
example (h₁ : Tendsto f atTop (nhds 0)) (h₂ :
    Tendsto g atTop (nhds 1))
    (unrelated : Tendsto f atBot (nhds 5)) :
    Tendsto (fun z => f z * g z) atTop (nhds
    0) := by tendsto_cont
```

Formal Verification

Formal Verification



Conversation 19 · Commits 18 · Checks 0 · Files changed 287 · +52,187 -15,457

augustepoiroux commented last month • edited — Collaborator

Summary

This PR completes the Lean formalization of the theorem that the optimal sphere packing in $\mathbb{R}^8$ is the E_8 lattice packing.
The branch contains the final theorem together with all remaining dependencies, and is checked by the Lean kernel (sorry-free).

Context

The code in this branch was generated by **Gauss** (Math Inc's autoformalization agent) starting from the existing Sphere Packing blueprint and Lean codebase, and then packaged into this PR for review.

Notable changes

- Completes the remaining formalization steps needed for E_8 optimality in $\mathbb{R}^8$
- Project-wide cleanup / proof golf (including existing files)
- Fixes a sign inconsistency in Prop. 7 (`b(t)`), with the downstream definition of `g` in Thm. 4 updated accordingly
- Migrates to the new module system
- Bumps Lean to `v4.28.0`

Review notes

- We made a best effort to avoid overwriting parts of the codebase that changed independently on `main` . As such, some results may be duplicated.
- ~~The following blueprint-linked Lean statements were commented out in this branch:~~
  ~~`ModularForm.dimension_level_one`~~ , ~~`dim_gen_cong_levels`~~ , ~~`D_qexp_tsum_pnat`~~ , ~~`serre_D_two_H`~~ -> they have been added back in 3887ebc #diff-ebf1c68f284079a6364607bb7113a4b517ffb358424f42f02a128c225340f25d

🎉 10 · 🚀 3

Reviewers
- dwrensha
- b-mehta
- seewoo5
- thefundamentaltheor3m
- CBirkbeck
- pitmonticone
- viazovska

At least 1 approving review is required to merge this pull request.

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close these issues.
None yet

Is the project finished?

Is the project finished?

NO!

Remaining Work

The Virtues of Formalisation

The Virtues of Formalisation

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The Virtues of Formalisation

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The first four were both my greatest motivators: **verifying theorems**, **correcting mistakes** (in my understanding), **gaining insight**, and **building libraries**.

The Virtues of Formalisation

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The first four were both my greatest motivators: **verifying theorems**, **correcting mistakes** (in my understanding), **gaining insight**, and **building libraries**.

The fifth (**searching for definitions and tools**) and fourteenth (**improving access**) greatly benefitted me in my journey.

The Virtues of Formalisation

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The first four were both my greatest motivators: **verifying theorems**, **correcting mistakes** (in my understanding), **gaining insight**, and **building libraries**.

The fifth (**searching for definitions and tools**) and fourteenth (**improving access**) greatly benefitted me in my journey.

The eighth (**engineering concepts**), ninth (**communicating**), and tenth (**collaborating**) have been key outcomes.

The Virtues of Formalisation

In “Mathematics and the Formal Turn” [1], Avigad effectively presents 18 positives of the advent of formal methods.

The first four were both my greatest motivators: **verifying theorems**, **correcting mistakes** (in my understanding), **gaining insight**, and **building libraries**.

The fifth (**searching for definitions and tools**) and fourteenth (**improving access**) greatly benefitted me in my journey.

The eighth (**engineering concepts**), ninth (**communicating**), and tenth (**collaborating**) have been key outcomes.

The sixteenth (**using automated reasoning**) and seventeenth (**using artificial intelligence**) have both been greatly impactful to this project, and will remain a big part of its future.

Towards a Formalisation

Towards a Formalisation

There are two approaches: manual formalisation and cleaning up the Gauss code.

Towards a Formalisation

There are two approaches: manual formalisation and cleaning up the Gauss code.

For several months now, we've been trying hard to combine the two approaches/make them meet in the middle.

Towards a Formalisation

There are two approaches: manual formalisation and cleaning up the Gauss code.

For several months now, we've been trying hard to combine the two approaches/make them meet in the middle.

But it turns out that cleaning up large volumes of LLM-generated code carries great costs in time and resources.

Towards a Formalisation

There are two approaches: manual formalisation and cleaning up the Gauss code.

For several months now, we've been trying hard to combine the two approaches/make them meet in the middle.

But it turns out that cleaning up large volumes of LLM-generated code carries great costs in time and resources.

Moreover, its effectiveness plateaus after a point.

Towards a Formalisation

There are two approaches: manual formalisation and cleaning up the Gauss code.

For several months now, we've been trying hard to combine the two approaches/make them meet in the middle.

But it turns out that cleaning up large volumes of LLM-generated code carries great costs in time and resources.

Moreover, its effectiveness plateaus after a point.

There is a lot more riding on the manual approach than we anticipated.

Towards a Formalisation

There are two approaches: manual formalisation and cleaning up the Gauss code.

For several months now, we've been trying hard to combine the two approaches/make them meet in the middle.

But it turns out that cleaning up large volumes of LLM-generated code carries great costs in time and resources.

Moreover, its effectiveness plateaus after a point.

There is a lot more riding on the manual approach than we anticipated.

We're just going to have to roll up our sleeves and get it done.

Acknowledgements

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi

Acknowledgements

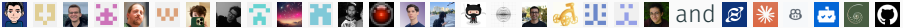
- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)

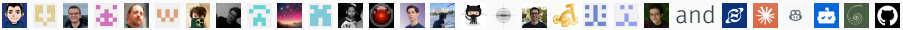
Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)



Acknowledgements

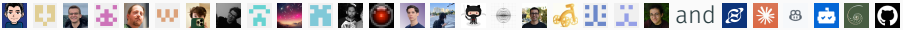
- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)



- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers

Acknowledgements

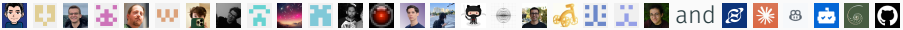
- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)



- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers
- Hoskinson Center for Formal Mathematics

Acknowledgements

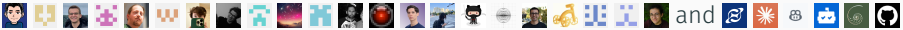
- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)



- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers
- Hoskinson Center for Formal Mathematics
- Institute for Computer-Aided Reasoning in Mathematics

Acknowledgements

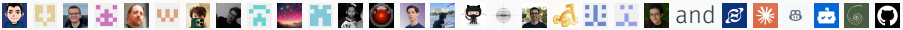
- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)



- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers
- Hoskinson Center for Formal Mathematics
- Institute for Computer-Aided Reasoning in Mathematics
- G-Research

Acknowledgements

- Chris Birkbeck, Seewoo Lee, Gareth Ma, Bhavik Mehta, Maryna Viazovska
- Matthew Cushman, Cameron Freer, Auguste Poiroux, Aayush Rajasekaran, David Renshaw, Tito Sacchi
- David Loeffler, Pietro Monticone, Utensil Song
- All our contributors! (Human and non-human alike)



- Jeremy Avigad, Kevin Buzzard, Simon DeDeo, Leo de Moura, Emily Riehl, the Mathlib maintainers
- Hoskinson Center for Formal Mathematics
- Institute for Computer-Aided Reasoning in Mathematics
- G-Research

Thank you very much! Merci beaucoup!

Questions?

References i

[1] J. Avigad.
Mathematics and the formal turn.
arXiv preprint arXiv:2311.00007, 2023.

[2] H. Cohn and N. Elkies.
New Upper Bounds on Sphere Packings I.
Annals of Mathematics, 157(2):689–714, 2003.