Numerical verification of Maass cusp forms

Andrei Seymour-Howell

University of Bristol

April 4, 2021

Maass cusp forms

Let $\mathbb{H} = \{z = x + iy | y > 0\}$ denote the upper half-plane. We define the Hecke congruence subgroup $\Gamma_0(N) < \mathrm{SL}(2, \mathbb{Z})$ by

$$\Gamma_0(N) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{SL}(2, \mathbb{Z}) | c \equiv 0 \pmod{N} \right\}$$

for $N > 0$. This group acts on $\mathbb{H}$ by linear fractional transformations, i.e

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} z = \frac{az + b}{cz + d} \qquad \forall \gamma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \Gamma_0(N), z \in \mathbb{H}.$$

The modular surface $X = \Gamma_0(N) \backslash \mathbb{H}$ is a finite volume non-compact surface with Laplacian

$$\Delta = -y^2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right).$$

We also have the measure

$$\frac{dx\, dy}{y^2}.$$

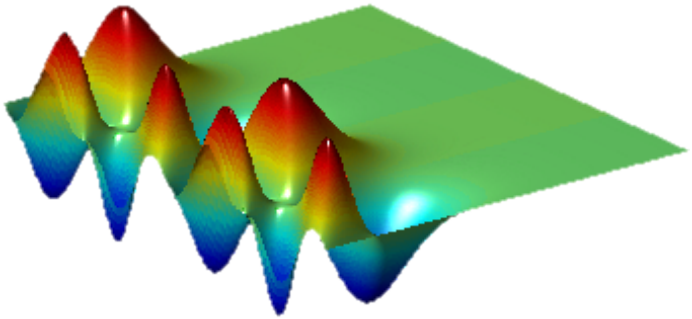
Maass cusp forms

We call a function $f : \mathbb{H} \to \mathbb{C}$ a **Maass cusp form** of level N (trivial character) if

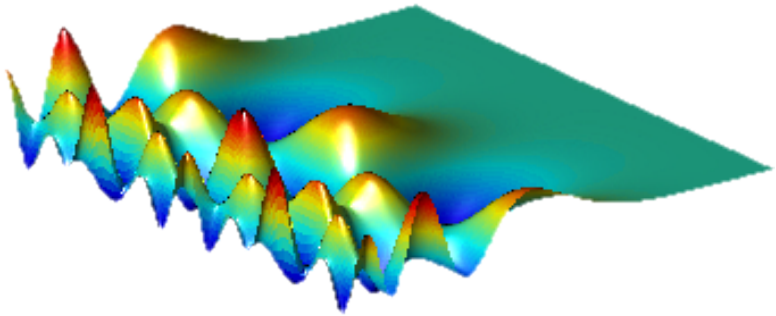
1. f is an eigenfunction of the Laplacian, $\Delta f = \lambda f, \lambda \geq 0$,
2. f is automorphic, $f(\gamma z) = f(z)$ for all $\gamma \in \Gamma_0(N)$,
3. $f \in L^2(X)$, i.e f is square-integrable,
4. f vanishes at all of the cusps of X.

We will denote the space of Maass cusp forms of level N and Laplace eigenvalue λ by $\mathcal{S}_\lambda(N)$.
The set of functions that just satisfy points (2), (3) and (4) we shall denote as $L^2_{\text{cusp}}(X)$.

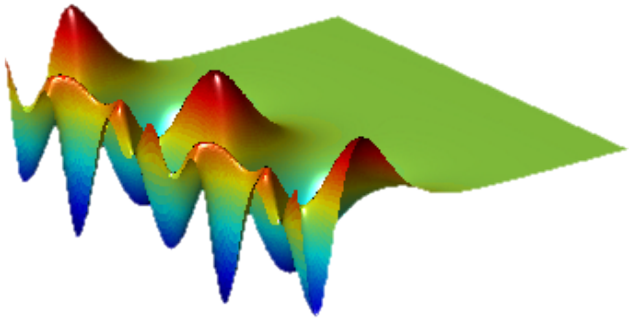
Pictures of Maass forms



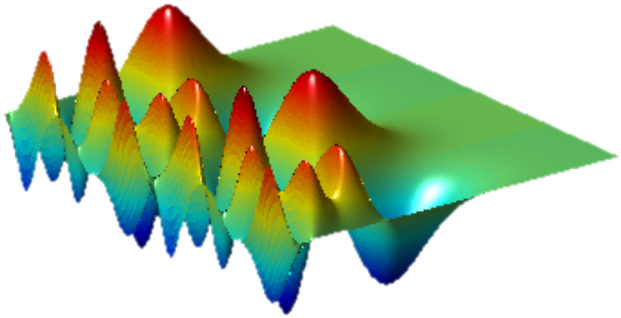
(a) Level 1, $\lambda = 91.141345\ldots$



(b) Level 1, $\lambda = 190.131547\ldots$



(c) Level 2, $\lambda = 79.867724\ldots$



(d) Level 3, $\lambda = 182.713668\ldots$

Figure: Images of Maass forms from the LMFDB.

Hecke operators

For any $f \in \mathcal{S}_\lambda(N)$ and any non-zero integer n coprime to N, we define the **Hecke operator** T_n by

$$T_n f(z) = \frac{1}{\sqrt{|n|}} \sum_{\substack{ad=n \\ (a,N)=1 \\ d>0}} \sum_{j=0}^{d-1} \begin{cases} f\left(\dfrac{az+j}{d}\right) & \text{if } n > 0, \\ f\left(\dfrac{a\bar{z}+j}{d}\right) & \text{if } n < 0. \end{cases}$$

This will map $\mathcal{S}_\lambda(N) \to \mathcal{S}_\lambda(N)$.

Hecke operators

For any $f \in \mathcal{S}_\lambda(N)$ and any non-zero integer n coprime to N, we define the **Hecke operator** T_n by

$$T_n f(z) = \frac{1}{\sqrt{|n|}} \sum_{\substack{ad=n \\ (a,N)=1 \\ d>0}} \sum_{j=0}^{d-1} \begin{cases} f\left(\dfrac{az+j}{d}\right) & \text{if } n > 0, \\ f\left(\dfrac{a\bar{z}+j}{d}\right) & \text{if } n < 0. \end{cases}$$

This will map $\mathcal{S}_\lambda(N) \to \mathcal{S}_\lambda(N)$.
Now a famous result then tells us that there exists an orthogonal basis $\{f_j\}$ in $L^2_{\text{cusp}}(X)$ consisting of eigenfunctions to all Hecke operators T_n with $(n, N) = 1$.

Hecke eigenvalues

A Maass cusp form f of level N and with Laplace eigenvalue $\lambda = \frac{1}{4} + R^2$ has a Fourier expansion (at ∞) of the form

$$f(z) = \sum_{n \neq 0} a(n) \sqrt{y} K_{iR}(2\pi |n| y) e(nx)$$

where $e(nx) = \exp(2\pi i n x)$ and $K_\nu(u)$ is the K-Bessel function.

Hecke eigenvalues

A Maass cusp form f of level N and with Laplace eigenvalue $\lambda = \frac{1}{4} + R^2$ has a Fourier expansion (at ∞) of the form

$$f(z) = \sum_{n \neq 0} a(n) \sqrt{y} K_{iR}(2\pi |n| y) e(nx)$$

where $e(nx) = \exp(2\pi i n x)$ and $K_\nu(u)$ is the K-Bessel function. We call a Maass form f **even** if $a(-n) = a(n)$ or **odd** if $a(-n) = -a(n)$.

Hecke eigenvalues

A Maass cusp form f of level N and with Laplace eigenvalue $\lambda = \frac{1}{4} + R^2$ has a Fourier expansion (at ∞) of the form

$$f(z) = \sum_{n \neq 0} a(n)\sqrt{y} K_{iR}(2\pi|n|y)e(nx)$$

where $e(nx) = \exp(2\pi i n x)$ and $K_\nu(u)$ is the K-Bessel function.
We call a Maass form f **even** if $a(-n) = a(n)$ or **odd** if $a(-n) = -a(n)$.
If f is also a Hecke eigenfunction for all Hecke operators T_n with $(n, N) = 1$, i.e $T_n f = \lambda(n) f$, then we can normalise such that $a(1) = 1$ and we have

$$a(n) = \lambda(n)$$
$$a(-n) = \varepsilon\lambda(n)$$

where ε is 1 is f is even and -1 if f is odd.

Hejhal's Algorithm

There are a few methods known for computing Maass forms, the most widely used is an algorithm due to Hejhal from the 1990's. The algorithm goes in the following steps

Hejhal's Algorithm

There are a few methods known for computing Maass forms, the most widely used is an algorithm due to Hejhal from the 1990's. The algorithm goes in the following steps

1. Truncate the Fourier series and treat it like a discrete Fourier series.

Hejhal's Algorithm

There are a few methods known for computing Maass forms, the most widely used is an algorithm due to Hejhal from the 1990's. The algorithm goes in the following steps

1. Truncate the Fourier series and treat it like a discrete Fourier series.
2. Do an inverse Fourier transform along a certain horocycle of points away from the fundamental domain.

Hejhal's Algorithm

There are a few methods known for computing Maass forms, the most widely used is an algorithm due to Hejhal from the 1990's. The algorithm goes in the following steps

1. Truncate the Fourier series and treat it like a discrete Fourier series.
2. Do an inverse Fourier transform along a certain horocycle of points away from the fundamental domain.
3. This will give an expression for the Fourier coefficients, however to make it a non-tautology, we use the automorphy of the Maass form to produce a linear system for the Fourier coefficients.

Hejhal's Algorithm

There are a few methods known for computing Maass forms, the most widely used is an algorithm due to Hejhal from the 1990's. The algorithm goes in the following steps

1. Truncate the Fourier series and treat it like a discrete Fourier series.
2. Do an inverse Fourier transform along a certain horocycle of points away from the fundamental domain.
3. This will give an expression for the Fourier coefficients, however to make it a non-tautology, we use the automorphy of the Maass form to produce a linear system for the Fourier coefficients.
4. We then use a non-linear search strategy to zoom in on an Laplace eigenvalue.

Hejhal's Algorithm

There are a few methods known for computing Maass forms, the most widely used is an algorithm due to Hejhal from the 1990's. The algorithm goes in the following steps

1. Truncate the Fourier series and treat it like a discrete Fourier series.
2. Do an inverse Fourier transform along a certain horocycle of points away from the fundamental domain.
3. This will give an expression for the Fourier coefficients, however to make it a non-tautology, we use the automorphy of the Maass form to produce a linear system for the Fourier coefficients.
4. We then use a non-linear search strategy to zoom in on an Laplace eigenvalue.

This method is heuristic, so we require another method to certify if the data produced is correct.

Verification methods

- In 2006, Booker, Strömbergsson and Venkatesh proved that it is possible to certify whether a candidate Maass form is "close" to a true Maass form. Roughly, suppose you have a computed eigenvalue $\tilde{\lambda} = \frac{1}{4} + \tilde{R}^2$ and the coefficients of a suspected Maass form $\tilde{f}$. Then they showed that if $\tilde{f}$ is "almost automorphic", then $\tilde{f}$ is "close" to a true Maass cusp form f. They only showed this for level 1, i.e $\mathrm{SL}(2, \mathbb{Z})$ and computed and verified the first few Maass cusp forms to a hundred digits.

Verification methods

- In 2006, Booker, Strömbergsson and Venkatesh proved that it is possible to certify whether a candidate Maass form is "close" to a true Maass form. Roughly, suppose you have a computed eigenvalue $\tilde{\lambda} = \frac{1}{4} + \tilde{R}^2$ and the coefficients of a suspected Maass form $\tilde{f}$. Then they showed that if $\tilde{f}$ is "almost automorphic", then $\tilde{f}$ is "close" to a true Maass cusp form f. They only showed this for level 1, i.e $\mathrm{SL}(2, \mathbb{Z})$ and computed and verified the first few Maass cusp forms to a hundred digits.
- In 2007, Booker and Strömbergsson devised a different way to rigorously compute Maass cusp forms. The main tool they used for this was an explicit Selberg trace formula that was for suitable for numerical computation.

Selberg trace formula

The Selberg trace formula allows one to consider the whole spectrum of Maass cusp forms for a fixed level N. Selberg derived this in the 1950's to prove the existence of Maass cusp forms.

Selberg trace formula

The Selberg trace formula allows one to consider the whole spectrum of Maass cusp forms for a fixed level N. Selberg derived this in the 1950's to prove the existence of Maass cusp forms. In our case, if we have a Hecke eigenbasis $\{f_j\}$ of $L^2_{\text{cusp}}(X)$ with respective Laplace eigenvalues λ_j, the Selberg trace formula allows us to compare

$$\text{(Spectral side)} \sum_{j=1}^{\infty} H(\lambda_j) = \text{(Geometric side)}$$

for some nice (analytic) test function H. The RHS will be a collection of terms that will all be computable and can give us information on the LHS. In our case we will also consider a trace formula with Hecke operators on the LHS.

Aside to modular forms

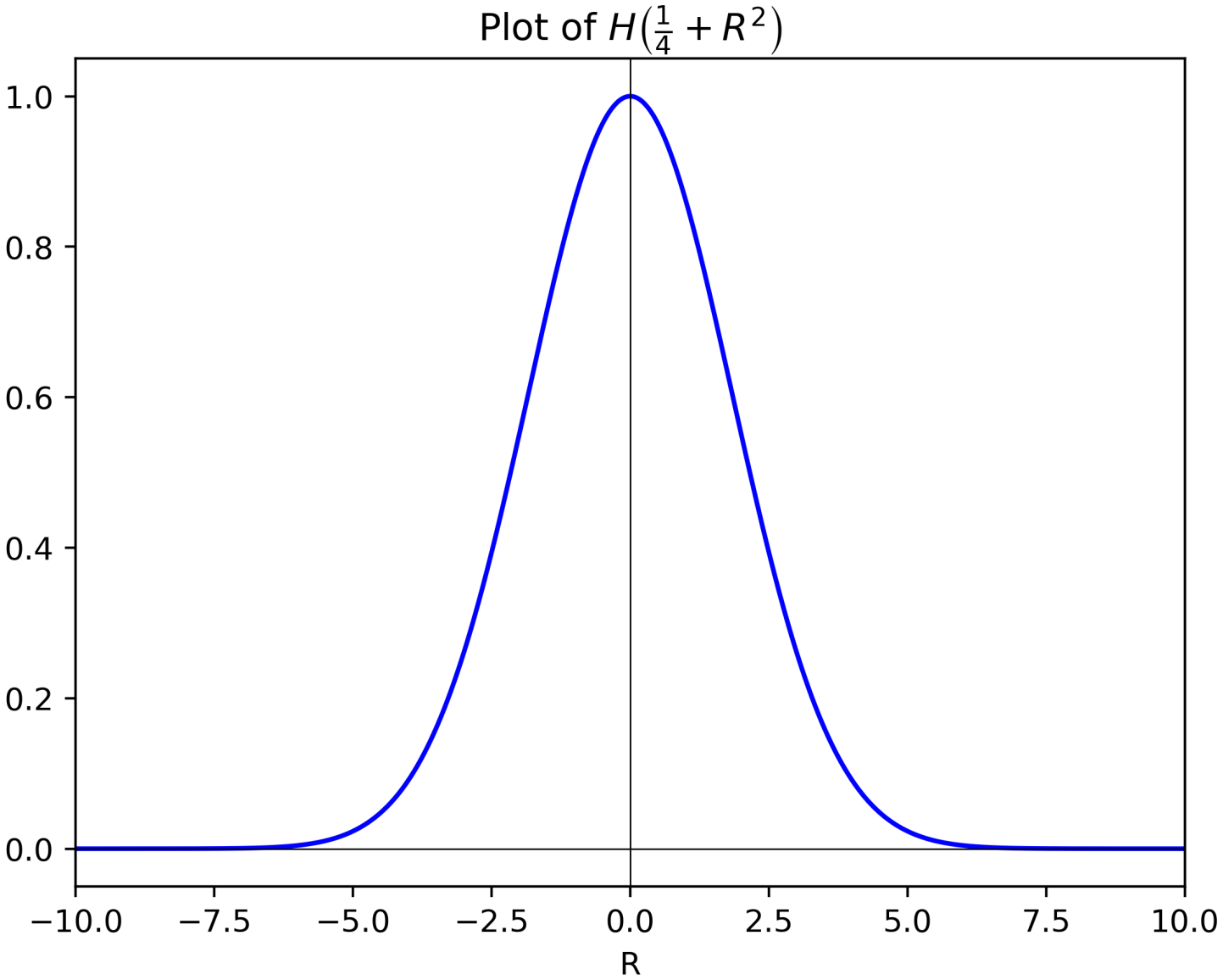
Explicit versions of trace formulas for modular forms have been used to compute basis elements of the spaces of modular forms, for example in the pari command `mfeigenbasis`. Since these spaces are finite dimensional, one can use the Hecke operators and linear algebra to extract the Fourier coefficients of the basis elements.

Aside to modular forms

Explicit versions of trace formulas for modular forms have been used to compute basis elements of the spaces of modular forms, for example in the pari command `mfeigenbasis`. Since these spaces are finite dimensional, one can use the Hecke operators and linear algebra to extract the Fourier coefficients of the basis elements.

In the Maass form case, the spaces we will be working with are infinite dimensional, hence the requirement for the test function in the trace formula. Our idea is to mimic the holomorphic form case by choosing a test function such that the contribution from the larger eigenvalues is negligible and we can then treat the problem as a finite linear algebra one.

Example of a test function



Verification using the Selberg trace formula

Fix a level N. Let $\{f_j\}$ be a Hecke eigenbasis of $L^2_{\text{cusp}}(X)$ with respective Laplace eigenvalues λ_j such that $\lambda_1 \leq \lambda_2 \leq \ldots$. Let $a_j(n)$ be the Hecke eigenvalues of f_j, that is $T_n f_j = a_j(n) f_j$ for $(n, N) = 1$.

Verification using the Selberg trace formula

Fix a level N. Let $\{f_j\}$ be a Hecke eigenbasis of $L^2_{\text{cusp}}(X)$ with respective Laplace eigenvalues λ_j such that $\lambda_1 \leq \lambda_2 \leq \ldots$. Let $a_j(n)$ be the Hecke eigenvalues of f_j, that is $T_n f_j = a_j(n) f_j$ for $(n, N) = 1$.

We will fix a sufficiently nice test function H that is positive and monotonically decreasing for $\lambda > 0$. The Selberg trace formula allows us to compute

$$t(n, H) := \sum_{j=1}^{\infty} a_j(n) H(\lambda_j)$$

for any $n \neq 0$ and $(n, N) = 1$.

Verification using the Selberg trace formula

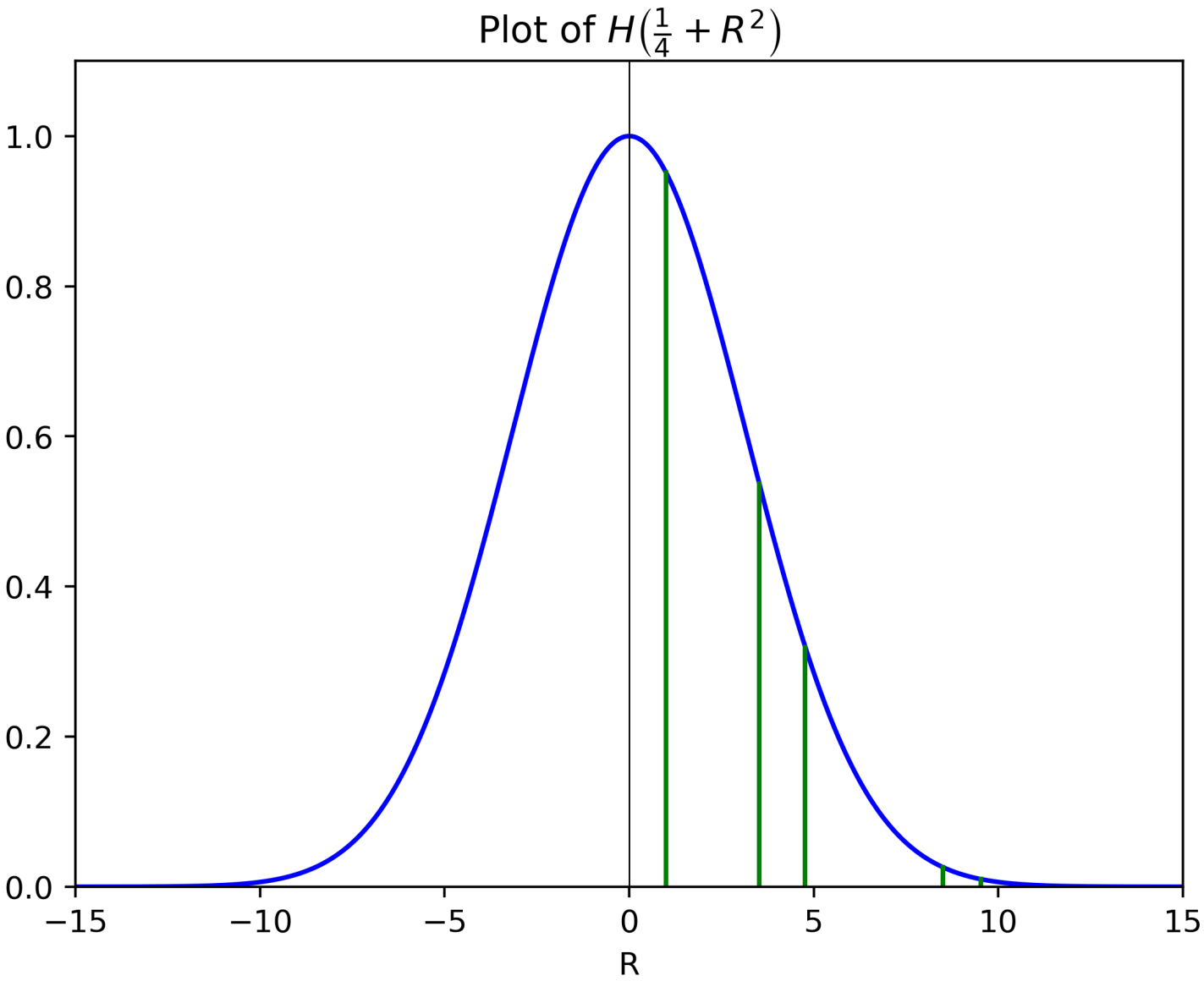
Fix a level N. Let $\{f_j\}$ be a Hecke eigenbasis of $L^2_{\text{cusp}}(X)$ with respective Laplace eigenvalues λ_j such that $\lambda_1 \leq \lambda_2 \leq \ldots$. Let $a_j(n)$ be the Hecke eigenvalues of f_j, that is $T_n f_j = a_j(n) f_j$ for $(n, N) = 1$.

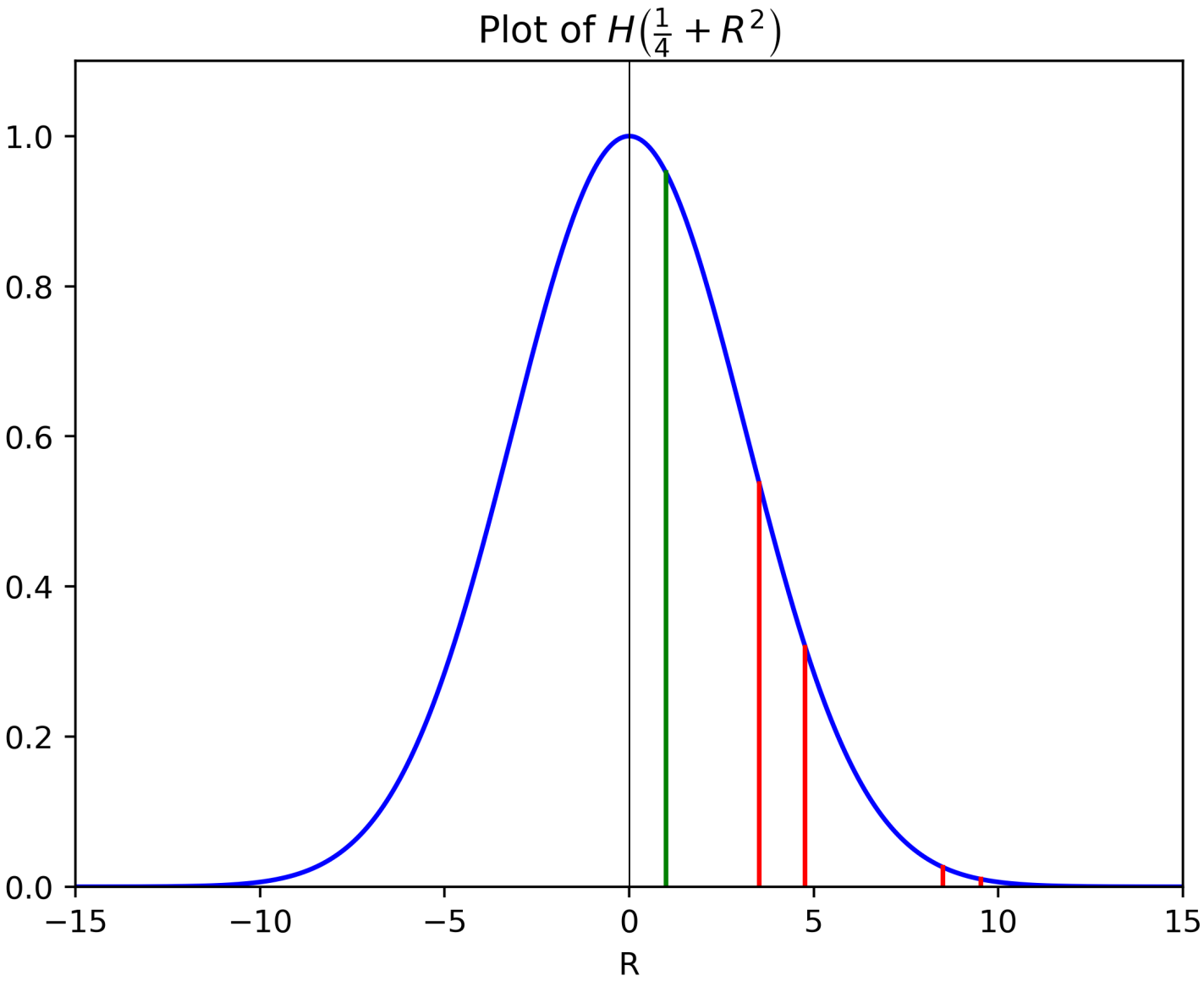
We will fix a sufficiently nice test function H that is positive and monotonically decreasing for $\lambda > 0$. The Selberg trace formula allows us to compute

$$t(n, H) := \sum_{j=1}^{\infty} a_j(n) H(\lambda_j)$$

for any $n \neq 0$ and $(n, N) = 1$.

The main idea here is to use linear algebra to remove the contribution of all the forms up to some limit and isolate just 1 form. We then use our approximation to this form (say computed using Hejhal's algorithm) to see how well the approximation removes the remaining contribution.


Plot of $H\left(\frac{1}{4}+R^2\right)$
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15
R


Plot of $H\left(\frac{1}{4}+R^2\right)$
1.0
0.8
0.6
0.4
0.2
0.0
−15
−10
−5
0
5
10
15
R

The details

Using the Hecke relations, we have for any real sequence of numbers $\{c(m)\}_{m=1}^{M}$ satisfying $c(m) = 0$ if $(m, N) > 1$, that

$$\left(\sum_{m=1}^{M} c(m)a_j(m)\right)^2 = \sum_{m_1=1}^{M}\sum_{m_2=1}^{M} c(m_1)c(m_2) \sum_{d|(m_1,m_2)} a_j\left(\frac{m_1 m_2}{d^2}\right).$$

The details

Using the Hecke relations, we have for any real sequence of numbers $\{c(m)\}_{m=1}^{M}$ satisfying $c(m) = 0$ if $(m, N) > 1$, that

$$\left(\sum_{m=1}^{M} c(m)a_j(m)\right)^2 = \sum_{m_1=1}^{M}\sum_{m_2=1}^{M} c(m_1)c(m_2)\sum_{d|(m_1,m_2)} a_j\left(\frac{m_1 m_2}{d^2}\right).$$

We define

$$\begin{aligned} Q(c, H) &:= \sum_{j=1}^{\infty}\left(\sum_{m=1}^{M} c(m)a_j(m)\right)^2 H(\lambda_j) \\ &= \sum_{m_1=1}^{M}\sum_{m_2=1}^{M} c(m_1)c(m_2)\sum_{d|(m_1,m_2)}\sum_{j=1}^{\infty} a_j\left(\frac{m_1 m_2}{d^2}\right) H(\lambda_j) \\ &= \sum_{m_1=1}^{M}\sum_{m_2=1}^{M} c(m_1)c(m_2)\sum_{d|(m_1,m_2)} t\left(\frac{m_1 m_2}{d^2}, H\right). \end{aligned}$$

Intuition

Suppose that we have putative numerical approximations $\tilde{\lambda}_j, \tilde{a}_j(m)$ to $\lambda_j, a_j(m)$. Then we want to choose numbers $c_i(m)$ such that $c_i(m) = 0$ if $(m, N) > 1$ and

$$\sum_{m=1}^{M} c_i(m)\tilde{a}_j(m) = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases}$$

Intuition

Suppose that we have putative numerical approximations $\tilde{\lambda}_j, \tilde{a}_j(m)$ to $\lambda_j, a_j(m)$. Then we want to choose numbers $c_i(m)$ such that $c_i(m) = 0$ if $(m, N) > 1$ and

$$\sum_{m=1}^{M} c_i(m)\tilde{a}_j(m) = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\tilde{H}_i(\lambda) = H(\lambda)(\lambda - \tilde{\lambda}_i)^2$. For the verification we shall prove that there exists a Laplace eigenvalue near $\tilde{\lambda}_i$. For this, we use the definition of Q to compute

$$\varepsilon_i := \sqrt{\frac{Q(c_i, \tilde{H}_i)}{Q(c_i, H)}}.$$

Then there exists a cuspidal eigenvalue $\lambda \in [\tilde{\lambda}_i - \varepsilon_i, \tilde{\lambda}_i + \varepsilon_i]$.

In practice

Suppose that we have putative numerical approximations $\tilde{\lambda}_j$. Define

$$\varepsilon_i := \sqrt{\frac{Q(c_i, \tilde{H}_i)}{Q(c_i, H)}},$$

for some real sequence of numbers $c_i(m)$ with $c_i(m) = 0$ if $(m, N) > 1$. The RHS is just a ratio of two positive definite quadratic forms, so we can just minimise this ratio. This will then give us the $c_i(m)$ that can be used to calculate ε_i.

Completeness

For completeness of the eigenvalues we have that $H(\lambda) \geq H(\tilde{\lambda}_i + \varepsilon_i)$ for all $\lambda \in [\tilde{\lambda}_i - \varepsilon_i, \tilde{\lambda}_i + \varepsilon_i]$. Hence any eigenvalue λ that is not contained in $\bigcup_i [\tilde{\lambda}_i - \varepsilon_i, \tilde{\lambda}_i + \varepsilon_i]$ must satisfy

$$H(\lambda) \leq t(1, H) - \sum_i H(\tilde{\lambda}_i + \varepsilon_i),$$

where the second sum ranges over all such i such that $[\tilde{\lambda}_i - \varepsilon_i, \tilde{\lambda}_i + \varepsilon_i]$ does not overlap the corresponding interval for any smaller value of i.

Completeness

For completeness of the eigenvalues we have that $H(\lambda) \geq H(\tilde{\lambda}_i + \varepsilon_i)$ for all $\lambda \in [\tilde{\lambda}_i - \varepsilon_i, \tilde{\lambda}_i + \varepsilon_i]$. Hence any eigenvalue λ that is not contained in $\bigcup_i [\tilde{\lambda}_i - \varepsilon_i, \tilde{\lambda}_i + \varepsilon_i]$ must satisfy

$$H(\lambda) \leq t(1, H) - \sum_i H(\tilde{\lambda}_i + \varepsilon_i),$$

where the second sum ranges over all such i such that $[\tilde{\lambda}_i - \varepsilon_i, \tilde{\lambda}_i + \varepsilon_i]$ does not overlap the corresponding interval for any smaller value of i.
Since H is monotonic, this determines numbers $\delta_i > 0$ such that $|\lambda_i - \tilde{\lambda}_i| \leq \varepsilon_i$ and $|\lambda_j - \tilde{\lambda}_i| \geq \delta_i$ for $j \in \mathbb{N} \backslash \{i\}$.

Hecke eigenvalues

Once we have a proven bound for the Laplace eigenvalue and proven bounds for the spacing of the Laplace eigenvalues, we can consider the Hecke eigenvalues.
The rough idea here is that with the previous bounds we can give bounds on how well

$$\sum_{m=1}^{M} c(m)a_j(m)$$

approximates δ_{ij}. Then using the fact that for our forms $a(1) = 1$, we can use our $c_i(m)$ to isolate the Hecke eigenvalues of one of the forms.

Computing the Laplace eigenvalues

With the same H as before, we define $\tilde{H}(\lambda) = \lambda H(\lambda)$. Let Q and $\tilde{Q}$ denote the respective matrices of the quadratic forms $Q(c, H)$ and $Q(c, \tilde{H})$. To find the eigenvalues λ_j, we seek solutions to the generalised symmetric eigenvalue problem

$$\tilde{Q}x = \lambda Q x.$$

The matrices Q and $\tilde{Q}$ are already computed when doing the verification, so this adds little extra time to the code and allows it to be fully self-contained.

Computational remarks

- The computations are done in C using a library called ARB, which does rigorous real and complex arithmetic with arbitrary precision. This is done using ball arithmetic, a form of interval arithmetic, which represents its numbers by a midpoint and radius.

Computational remarks

- The computations are done in C using a library called ARB, which does rigorous real and complex arithmetic with arbitrary precision. This is done using ball arithmetic, a form of interval arithmetic, which represents its numbers by a midpoint and radius.
- The minimisation of the quadratic forms is done using the PC “long double” type, which has a 64-bit mantissa. Since this is the slowest part of the code, this gives a substantial speed increase.

Computational remarks

- The computations are done in C using a library called ARB, which does rigorous real and complex arithmetic with arbitrary precision. This is done using ball arithmetic, a form of interval arithmetic, which represents its numbers by a midpoint and radius.
- The minimisation of the quadratic forms is done using the PC “long double” type, which has a 64-bit mantissa. Since this is the slowest part of the code, this gives a substantial speed increase.
- This code is probably unfeasible for levels larger than a few hundred due to large increase in the number of forms required to be removed to achieve good precision.

Computational remarks

- The computations are done in C using a library called ARB, which does rigorous real and complex arithmetic with arbitrary precision. This is done using ball arithmetic, a form of interval arithmetic, which represents its numbers by a midpoint and radius.
- The minimisation of the quadratic forms is done using the PC "long double" type, which has a 64-bit mantissa. Since this is the slowest part of the code, this gives a substantial speed increase.
- This code is probably unfeasible for levels larger than a few hundred due to large increase in the number of forms required to be removed to achieve good precision.
- Currently this method has only been implemented for square-free level N due to availability of explicit forms of the Selberg trace formulas with Hecke operators.

Thanks for listening!

Example of trace formula, composite square-free level

$$\frac{\mu(N)\sigma_1(|n|)}{\sqrt{|n|}} h\left(\frac{i}{2}\right) + \sum_{j=1}^{\infty} h(R_j) a_j(n)$$

$$= \sum_{\substack{t\in\mathbb{Z} \\ \sqrt{D}=\sqrt{t^2-4n}\notin\mathbb{Q} \\ D>0}} c_N(D) \cdot g\left(\log\left(\frac{(|t|+\sqrt{D})^2}{4|n|}\right)\right)$$

$$+ \sum_{\substack{t\in\mathbb{Z} \\ \sqrt{D}=\sqrt{t^2-4n}\notin\mathbb{Q} \\ D<0}} c_N(D) \cdot \frac{\sqrt{|D/4n|}}{2\pi} \int_{-\infty}^{\infty} \frac{g(u)\cosh(u/2)}{\sinh^2(u/2)+|D/4n|} du$$

$$+ \left[\text{if } \sqrt{n}\in\mathbb{Z} : \frac{\prod_{p|N}(p-1)}{12\sqrt{n}} \int_{-\infty}^{\infty} \frac{g'(u)}{\sinh\left(\frac{u}{2}\right)} du\right].$$

$$c_N(D) = \frac{L(1,\psi_d)}{l} \prod_{p|N}(\psi_d(p)-1) \prod_{p|l}\left[1+(p-\psi_d(p))\frac{(l,p^\infty)-1}{p-1}\right].$$