Progress on Hejhal's Algorithm

Alex Karlovitz

May 3 2019

I ran Hejhal's algorithm today as described in my Learning Seminar talk (so no special tricks for numerical stability). I input the following three r values:

- $r = 9$
- $r = 9.53369526135355727092246524989604949951171875$ (this is approximately the true eigenvalue)
- $r = 10$

I used parameters

- $D = 50$ (digits of precision)
- $M_0 = 30$ (this is the value suggested by BSV for $D = 50$)
- $Q = 100$ (still not sure how to appropriately choose Q)
- $Y_1 = 1/2$ and $Y_2 = 1/10$ (these are the two heights at which I take the sample points along a closed horocycle)

To check multiplicativity, I printed the Fourier coefficients a_2, a_3, and a_6 for each r. For $r = 9.533695...$, I got

$$a_2 = -1.06833 \qquad a_3 = -0.45619 \qquad a_6 = 0.48737$$

from height 1/2 and

$$a_2 = -1.06833 \qquad a_3 = -0.45619 \qquad a_6 = 0.48737$$

from height 1/10. Notice that these are exactly the same! The code printed out 50 digits for each of these. You have to go about 14 digits to start to see a difference. Moreover, the digits that I have provided exactly match those given by BSV on Strömbergsson's website.

I also printed out the errors. That is, for each r, I computed the vectors of Fourier coefficients suggested by the sample points at Y_1 and Y_2. Then I printed out the 2-norm of the difference between these vectors. The errors were as follows:

- for $r = 9$, error was about 2.8×10^{32}
- for $r = 9.533695...$, error was about 4.6×10^{19}
- for $r = 10$, error was about 2.06×10^{33}

The good news is that the smallest error was given by the r closest to the true eigenvalue. The bad news is that these errors are huge compared to the size of the Fourier coefficients.

May 4 2019

Today, I ran Hejhal's algorithm for real; that is, I let it run the iterative process to zero in on an eigenvalue. I used the parameters

- $D = 50$ (digits of precision)
- $M_0 = 30$ (this is the value suggested by BSV for $D = 50$)
- $Q = 100$ (still not sure how to appropriately choose Q)
- $Y_1 = 1/2$ and $Y_2 = 1/10$ (these are the two heights at which I take the sample points along a closed horocycle)

and I started with r values $9, 9.1, 9.2, \ldots, 9.9, 10$. After 14 iterations, the output was

$$r = 9.53369526135356$$

This agrees to 14 decimal places with the computations by BSV. Next, I plan to let the process run a few more iterations. Judging by yesterday's results, I expect the r value given by my algorithm to start to diverge from the true eigenvalue at around the 15^{th} decimal place.

May 5 2019

Starting with

$$r = 9.53369526135356$$

which was the result from last time, I ran 8 more iterations and got

$$r = 9.5336952613535580388596044418811798095703125$$

At around the 14^{th} or 15^{th} digit, this starts to diverge from the true eigenvalue. So it appears that computing everything to $D = 50$ digits of precision allows for 14 to 15 digits of precision in the final answer.

The 2^{nd}, 3^{rd}, and 6^{th} Fourier coefficients given when r is input the 14 decimal places of accuracy are as follows:

$$a_2 = -1.0683335512232188680742305108912588987488213870376$$

$$a_3 = -0.45619735450604610015637972856710657995012991007215$$

$$a_6 = 0.48737093979348801248710895531524614714300871266411$$

These agree with the values on Strömbergsson's website to 12 digits, 12 digits, and 10 digits, respectively.

June 25 2019

I repeated the test with the same inputs as above except with $Y_1 = 1/300$ and $Y_2 = 1/50,000$. This immediately went awry, choosing $r = 10.3$ at the first step instead of the true $r = 9.5$. This surprised me, since lower values of Y_1 and Y_2 will cause the sample points to hit many more fundamental domains. I expected this to cause the linear system to have more information, and hence cause the algorithm to converge more quickly.

I have three guesses for why this test didn't work as expected:

- I didn't use mpmath right away when I initialized Y_1 and Y_2; perhaps this caused more rounding error because of the small numbers.
- Maybe $1/300$ and $1/50,000$ are too close together for the linear system to have a lot of information.
- Perhaps there is an error in my above reasoning.

June 26 2019

Regarding the guesses listed above, I was able to quickly rule out the first guess. Next, to think about the second guess, I input the heights $Y_1 = 1/2$ and $Y_2 = 1/50,000$, expecting this to do better than any previous experiment. The output was

$$r = 9.53369526135355616036361440503969788551330566 40625$$

This is correct to about the 13^{th} or 14^{th} decimal place, which is one digit worse than the original $Y_1 = 1/2$ and $Y_2 = 1/10$ case.

July 7 2019

I ran Hejhal's algorithm today for a new group; namely, the group

$$\Gamma_4 = \left\langle \begin{pmatrix} 1 & \sqrt{2} \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \right\rangle$$

To make Fourier expansion at the cusp the same is in the $SL(2, \mathbb{Z})$ case, I in fact used a group conjugate to Γ_4:

$$\left\langle \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & -\frac{1}{\sqrt{2}} \\ \sqrt{2} & 0 \end{pmatrix} \right\rangle$$

According to a paper by Hejhal ("On Eigenvalues of the Laplacian for Hecke Triangle Groups"), Γ_4 has $\lambda = 1/4 + r^2$ as an eigenvalue where $r = 11.317680\ldots$.

I ran the algorithm with parameters

- $D = 50$
- $M_0 = 30$
- $Q = 100$
- $Y_1 = 1/2$ and $Y_2 = 1/10$

This did not work. Next, I tried running the code to a higher precision and with more sample points (along with different choices for Y_1 and Y_2). Specifically, I used the parameters

- $D = 80$
- $M_0 = 80$
- $Q = 100$
- $Y_1 = 1/3$ and $Y_2 = 1/20$

This *did* work! It found $r = 11.317680$ to those 6 decimal places, which is the same precision to which Hejhal found the eigenvalue.

I plan to mess around with the precision and number of sample points to see how high I need to go to make the code work.

July 8 2019

I reran yesterday's test with the parameters

- $D = 50$
- $M_0 = 50$

- $Q = 100$
- $Y_1 = 1/3$ and $Y_2 = 1/20$

This worked!

This time, I was looking for the eigenvalue $\lambda = 1/4 + r^2$ where $r = 7.220872\ldots$. Again, the code found this eigenvalue to 6 places of accuracy.

July 16 2019

I realized that I am unsure why the number of test points must be larger than the size of the final system. In fact, I reran the tests from July 8 with the following parameters

- $D = 50$
- $M_0 = 50$
- $Q = 10$
- $Y_1 = 1/3$ and $Y_2 = 1/20$

This worked to a reasonable degree!

In six steps, it found the eigenvalue $\lambda = 1/4 + r^2$ where $r = 7.220872\ldots$ to 4 decimal places. Of course, using less points gives less information. So it is not surprising that this worked to less decimal places.