COMPUTING CLASSICAL MODULAR FORMS

Abstract. We discuss the practical aspects of computing classical modular forms.

Contents

1. Data status

```
OK, the data for Nk^2 <= 4000 is now online.  We now cover a total of
247,438 spaces, of which 30,738 are non-empty, and we have a total of
67,180 newforms, giving rise to a total of 9,966,498 embedded modular
forms (so we should have just shy of 10 million CMF L-functions).

The boss form is
http://cmfs.lmfdb.xyz/ModularForm/GL2/Q/holomorphic/983/2/c/a/, with
dimension 39690.
Third, I'd like to give you an update on where things stand.  It looks like we now have a compl
```

Date: December 3, 2018.

Just to give some stats on the data we do have (excluding k=1), this range covers 5533 (Galois

Among the 4843 newforms, 3707 have coefficient fields of degree <= 20, and we now have canonica

For all 3707 of the newforms with coefficient field degree <= 20 we have computed algebraic a_n

To compare to the current Mongo DB database of modular forms, within this range only 2373 newf

On the other hand the current database contains a lot of newforms that are well outside our Nk

We would still be missing some forms that are in the current database, e.g. http://www.lmfdb.o

The one place where I could imagine wanting to go past Nk^2 <= 4000 is for trivial character,

Running on 64 cores the magma code now takes about 1.5 hours to compute all the data for Nk^2

I guess my main question for AB and JB is this: what do you think about the feasibility of get

In my view the data we have for Nk^2 <= 1000 is already clearly better than what is currently

Best,

Drew

2. Notes from discussions

* Magma doesn't compute Atkin-Lehner eigenvalues for quadratic character
* Maybe later, it would be nice if we could also compute exact Hecke data using pari
* Sage code for Conrey labels
* Polredabs the polys in mffield_500.txt
  -> Sort decomp by trace up to absolute degree <= 20
* Compute L-function data whenever feasible (and then they might not have labels), including p
  -> Compute L-hash for product L-function
  -> is CM?, has inner twist (what is relationship to Galois conjugates?), Sato-Tate group
  -> special to weight 2: numerical (geometric) endomorphism algebra, database of modular abel

In database entry:
  - at least a_n's up to Sturm bound

Two boxes:
  k*N <= 1000
  k = 2, N = larger bound

JV:
* a_n or a_p

Bober has:
  * N <= 999, k <= 4 : labeling for decomp (Conrey label), a_{p^n}'s for p^n < 2000
  * N <= 99, k <= 12

```
  * N <= 30, k <= 30

polydb:
  * N <= 434, k <= 3,4
  * N*k <= 390, k <= 30

* Make exact matches for Galois orbits



=====

* T_1 + T_p + T_q linear combinations to pick out quadratic subspaces, k = 2 and chi = triv
```

3. Introduction

4. Running time

4.1. **In theory.**

```
This is a bit optimistic, but typically OK, yes :

1) you assume that the weight is fixed (otherwise the size of the matrix
entries must be taken into account); and that the Nebentypus has fixed order
as well [ otherwise you need to work in cyclotomic fields or large degree, which
increases the cost of "base field" computations ]

2) splitting the space may need many (linear combinations of) T_p
[ I don't know anything better than the Sturm bound to guarantee that the
T_p, p <= B, generate the Hecke algebra ]. So O~(d^4) would be a
worst case [ given assumption 1) ]

>   * To get further eigenvalues, you typically only need one row of
> T_p, but you still need to multiply this row by each eigenvector, so
> it ends up being basically soft-O(d*p) again.

For the trace formula, here's a quick back-of-the-enveloppe computation.
Will check this with Henri in september :-) :

1) We must first build the space S_k^new:

1.a) we pick successive forms T_j Tr^new until they generate the space.
Assuming the first O~(d) values of j are enough [ heuristic for now but it may
be possible to prove this; it's true in practice ], this requires
expanding those O~(d) forms up to Sturm bound ( O~(d) ). So will need
O~(d * max(j)) = O~(d^2) coeffs of Tr^new.

1.b) all Hurwitz class numbers of index O~(d * max(j)) are precomputed
[ cost O~(d^3) ]; the coefficient Tr(n) [ = trace of T_n on the space S_k]
costs O(sqrt(n)). I am assuming that the weight and Nebentypus are
```

fixed, otherwise we need to take into account the "size" of coefficients.

So computing all Tr(n) up to O~(d^2) costs O~(d^3). The Tr^new(n) are
simple convolutions of the Tr(n) with Moebius function and the like and
costs the same up to log factors (sums over divisors etc.).

1.c) we compute the rank of the matrix made up by the coefficients of
the T_j Tr^new, and hope to get maximal rank in O(1) trials with O~(d)
forms: O(d^3) [or whatever exponent: no soft-Oh because we expect to
detect the rank by projecting Z[\chi] to a small finite field]

1.d) we precompute base change matrices from and to Miller's basis: at
least O~(d^\omega+1) [the T_j Tr^new form a somewhat random basis and the
coefficients in the original -> Miller base change matrix are huge]

Total [heuristic] cost for this phase: O~(d^\omega+1)

2) To compute the matrix of T_p on our basis for S_k^new, we now need
coefficients of Tr^new up to O~(d * max(j) * p). The Hurwitz class
number precomputation and subsequent coefficients computation jumps to
O~(d^3 p^{3/2}).

3) Then it's the same as in the other methods: characteristic polynomial,
factorization over Q(\chi), splitting, etc.

Thus, in theory, I would expect the trace formula to be slower than modular
symbols because of

- the cost to convert to Miller basis (or to express a random form in
  terms of the T_j Tr^new basis)

- the extra costs (extra coefficients) involved in hitting T_j Tr^new by T_p

In practice, as long as p doesn't get too large (and the linear algebra
involved in converting T_j Tr^new -> Miller basis doesn't get dominant),
I'm not sure at all that this is the case. It also depends on how you
get S_k^new from modular symbols when N is highly composite : kernels of
degeneracy maps can get expensive since they apply on "huge" S_k (of
dimension D), not "tiny" S_k^new (of dimension d).

I'm *very* interested in data points if you compare the above guesstimates
with Sage or Magma running times. :-)

4.2. **In practice.**

Thanks for this! I notice that in fact you computed a lot more spaces than N*k <= 1000. I ex

The Magma run has completed all the spaces with N*k <= 500, and I get an exact match with your

I uploaded the files mfdecomp_500.m.txt and mfdecomp_500.gp.txt to your repo which contain cor

5. How to deal with spaces that are too big

I propose we run Magma on a range of weights, levels, and characters, but keeping only Hecke orbits of dimension <= 4. The 4 is arbitrary, it says we'll e.g. be interested in fourfolds but not fivefolds; I think that's reasonable for where we're at now. Here's what it would look like in pari:

```
? for(i=1,#L, T = gettime(); Snew = mfinit([N,4,[G,L[i]]], 0); [vF,vK]
= mfsplit(Snew, bnd); print1(mfdim(Snew)); print1(" ");
print(gettime()/1000.);)
```

This already seems to take forever for me in a space with N = 220; I think the linear algebra over cyclotomic fields has not been optimized in Pari.

My proposed strategy, for weight k >= 3:
  choose a large prime p split in the cyclotomic field,
  factor a Hecke polynomial mod p,
  for the combinations of factors that give dimension <= 4, find lifted polynomials that are q-Weil polynomials,
  and for these, find the exact eigenspace, and then compute the remaining Hecke eigenvalues over the cyclotomic field

Variant: try several large primes to find one with minimal splitting; or take a prime which is not necessarily split but of approximately the same norm.

For weight k >= 3, my expectation is that there are few small eigenspaces, most will be discarded, and there will not be a combinatorial explosion in the third step.

OTOH, for weight k = 2, we should instead loop over p-Weil polynomials with character and repeated split the space, just like Cremona does for elliptic curve--I would expect many eigenforms.

5.1. Polredabs and polredbest.

Really important: take version of Pari = blank, Sage 8.3.

6. Atkin–Lehner operators

JV is right, the A-L operators W_M for M||N map S_k(N,chi) to S_k(N,chi') with some explicit de

7. Issues with haracters

At the same time I am going to add some additional data that I want to display on the newform home pages, which is an explicit specification of the character values on generators of (Z/NZ)* as elements of the coefficient field (in terms of our nice basis). This is actually a non-trivial computation and is what took most of the time when I was processing your weight 1 data (I need this in order to compute the complex embedding data and L-function labels).

It is not enough just to embed the character field into the coefficient field (which is already hard when the coefficient field is big), you need to embed it in such a way that the character values are compatible with the Hecke action, and it can take a non-trivial amount of time to work this out (magma gives character values in terms of some zeta_n, but you need to identify this zeta_n with the correct nth root of unity in the coefficient field).

You might argue that we could avoid this by keeping track of the coefficient field as a relative extension of a cyclotomic field as we compute the forms, but I think it is better to keep everything in absolute terms, especially when we are making comparisons across multiple computer algebra systems, and we ultimately want to display our eigenvalues as elements of our LLL-reduced basis of the ring of integers of the coefficient field.

Having done so, it will be a big help to people who want to play with our data to be able to construct the character directly in terms of our basis (it also means they don't have to mess around with Conrey labels or character orbit labels, we can specify the character directly as a map from Z to Q(f) using a nice representation of Q(f) -- when the character degree gets large this is actually much faster.

8. CM and inner twist

Ciaran Schembri and I have been making use of the new database & web pages, in particular looki

If we fix: weight 2, trivial character, dimension 2, and search for forms with inner teist we s

http://cmfs.lmfdb.xyz/ModularForm/GL2/Q/holomorphic/?weight=2&prime_quantifier=exact&char_order

and comparing, I note the following:

(1) the form at level 169 is missing from my table. I must have made a mistake in 1984, which

(2) I include forms at level 225 and 256 (with coefficients fields sqrt(5) and sqrt(2) respecti

Are we disagreeing with the definition of inner twist, or have I found a bug in our code for de

John

CM forms are inner twists of themselves, but we consider these "trivial" inner twists, since twisting a CM form by the CM character doesn't change the form.

Note that the page says that they have no *non-trivial* inner twists. I could be convinced to include CM forms in inner twists if people thought that made more sense. Alternatively, we could change the search drop-down option to include options that specify whether CM forms are to be included or not.

This will of course be explained in the inner twist knowl that I have not gotten around to writing. Working on knowls is next on my todo list, just as soon as I finish computing projective images, which is turning out to be a lot more work than I hoped it would be, determining these by identifying the corresponding artin reps does not look like it is feasible once the level gets large.

Drew

9. Precision in short vector

Hi John,

FYI, the try/catch code you put around the call to MinkowskiLattice in heigs.m is not sufficient to guarantee that we will be able to get 1 as a shortest vector, this was causing an assert failure when checked later on (I hit this in the corresponding path in zbasis.m for some spaces in the Nk^2 <= 4000 computations)

The problem is that the call using the default precision might succeed even though it doesn't find a basis vector that is a root of unity. I changed the code in zbasis.m to always use precision that is at list as large as the discriminant, and this fixes the problem. This is in the update to zbasis.m that I just merged, but I have not changed the heigs.m code (this code is no longer used, so could remove it, but I left it there in case you want to look at it and/or change it).

Examples of spaces where this problem shows up are 14:16:3 and 961:2:7 (the space 14:16:3 can be decomposed quite quickly into to forms with dimension 10, the problem shows up for the first form in lex order, which is the second form in the order returned by magma).

```
> Attach("chars.m");
> Attach("heigs.m");
> chi := CharacterOrbitReps(14)[3];
> S :=
> NewSubspace(NewSubspace(CuspidalSubspace(ModularSymbols(chi,16,-1))));
> F := NewformDecomposition(S);
> ExactHeckeEigenvalues(F[2]);

ExactHeckeEigenvalues(
Vf: Modular symbols space of weight 16 and dimension 5 over Cycl...
)
In file "/home/drew/Dropbox/dev/cmf/heigs.m", line 214, column 5:
>>     assert ind ne 0;
       ^
Runtime error in assert: Assertion failed
```

Drew

10. Questions and observations

> In order to identify conjugate forms that Magma erroneously lists, I am
> comparing absolute traces of a_n for n up to the Sturm bound. Stupid
> question: this is obviously necessary, but is it sufficient? Comparing
> minpolys would certainly be enough, and traces up to some bound is certainly
> enough, the question is whether the Sturm bound works. In any case
> comparing the results with Pari should catch any problems this might cause.
>
> Sure, but can non-conjugate forms give rise to the same isogeny class of
> abelian varieties over Q? If not then there is some B such that checking
> traces of a_n for n <= B is enough, and then the questions is whether B
> is the Sturm bound or larger. Or are you are telling me that non-conjugate
> forms can define the same AV over Q (in other words, non-conjugate modular
> AVs with isogenous restrictions of scalars)? Do you know any examples?

Sorry, no, I was trying to say that I don't see a way to use the Sturm
bound for this purpose.

The non-empty spaces of level 2 always seem to decompose as [floor(d/2),ceil(d/2)]. I know thi

I would guess this is just the Maeda conjecture in level 2 after you
decompose the space under the Atkin-Lehner operator--so far from a
theorem.

But is it at least a theorem that
Atkin-Lehner will split the space as evenly as possible in level 2?

11. Weight 1

> The last space 3900:1:24 completed in around 215h and over 250gb;
> after all that this space is 0-dimensional. Weight 1 forms are weird.

```
> chi := Generators(FullDirichletGroup(383))[1];
> M := ModularForms(chi,1);
> Dimension(M);
190
> HeckeOperator(M,5);

>> HeckeOperator(M,5);
                   ^
Runtime error: Hecke operator computation currently only supported for spaces
with a single character that takes values +/-1.
```

Regarding your comment about the Hecke cutters, these are actually
minpolys of T_p, and for the big spaces Magma can be really slow at
computing them, even just for T_2 (which for the 4760,4760 example would
already be enough, in fact the trace of a_2 down to Q already
distinguishes these spaces).

I see -- presumably magma had done a similar computation to find the splitting but does not gi

A while back I reported on one case I had where the dimension was 162*80, so the Hecke matrice

It would be nice of Magma (and the others) would use such modular methods automatically.

* In the k > 1 and 3000 < Nk^2 <= 4000 computation there is only one
space
left (the 4760,4760 one), and I expect this one to finish today or
tomorrow.

* I have classified the projective Galois images for all the wt 1 forms
of level up to 3000, and I have kernel polys for the projective reps
for all the A4,S4,A5, and Dn cases with n <= 10, except for a dozen or
so S4 cases where the kernel field is not in the LMFDB. I am going to
work today on computing these S4 fields. I was surprised by the
fact that even though all the A5 fields are there (which is good because
these would be a pain to compute on the fly), some of the S4 fields
are not, e.g. the quartc field x^4 - 401*x - 8421 with Galois closure S4
arises for a weight 1 form of level 1203 = 3 * 401, but this field is
not in the LMFDB (it does have a sextic sibling in the LMFDB, but other
cases don't have any siblings either).

* I have computed exact Artin reps for all the weight one forms with
projective image D2 (some of which can have very large kernel fields,
as high as degree 88). I am going to try using the same method to lift
all the projective reps I have, but I only expect to be able to lift
those that have a lift of reasonably small degree (e.g. <= 20).

12. JC comments on representing newforms

The following occurred to me while driving up to Scotland 10 days ago, so I hope it still make

We think of each d-dimensional newform as one object f representing a Galois orbit of newforms

Now, f and its Galois conjugates span a d-dimensional complex vector space V_C; also the Q-ba

So that gives another view of what we have been doing. There is a well-defined Q-vector spac

I don't think that this viewpoint helps us in our computations, but (for me at least) it helps

Now I will get back to actually doing some computations!

Here's some interesting information about a space I mentioned recently, 3901:1:10, whose outp

Recall that this space has dimension 200 splitting as 40+160, the degree of Q(chi) is 40 (it i

It did take over 5 hours to compute the inverse. But if we were to store the inverse rather t

I am tempted to write a script which takes all out text data files and replaces the basis matr

13. Comments on pari

```
> Two more things:  (1) I had been under the impression that
> polredbest(),
> unlike polredabs(),  was faster because it did not need to factor any
> discriminants.  But the documentation suggests otherwise.  the docs for
> polredabs() at http://pari.math.u-bordeaux.fr/dochtml/html-stable/ are
> full
> of warnings regarding factorization; the docs for polredbest() say
> nothing
> like that but do say "This routine computes an LLL-reduced basis for
> the
> ring of integers of ?[X]/(T), then ..."   which implies that the full
> ring
> of integers is known along the way.  Perhaps I should ask on pari-users
> about that?
>

> Is the documentation wrong?  Or does polredbest() indeed do the hard work?

This would be interesting to know, I had always assumed it was "safe" to
call polredbest() with a polynomial whose discriminant cannot be readily
factored.

Yes, this is imprecise. The documentation should read "LLL-reduced basis
for an order in Q[X]/(T)...". [ It now does, in 'master'. ]

The function correctly states that it runs in polynomial time (wrt. the
size of its input).

Cheers,

    K.B.
```

14. Timings

```
I finally have computed the data for N*k^2 <= 1000 complete using gp (+sage for later processi

Here are some stats:


$ wc mfdata_1000.m.txt
    5533     5533 82782397 mfdata_1000.m.txt
$ wc mfdata_1000.g.txt
    5533     5533 82955633 mfdata_1000.g.txt

So my output is not quite as compact as yours, though they are close.  In more detail:


sage: gdata = read_dtp("t1000")
```

```
Read 5533 spaces of which 2653 are nontrivial; 4843 Galois orbits.
3707 orbits have dimension <=20
largest three dimsensions: [1404, 1824, 2016]
Total time = 120960.448
Max time = 11638.884 for space (237, 2, 14)
Average time (all spaces)      = 21.862
Average time (nonzero spaces)  = 45.327

sage: mdata = read_dtp("mfdata_1000.m.txt")
Read 5533 spaces of which 2653 are nontrivial; 4843 Galois orbits.
3707 orbits have dimension <=20
largest three dimsensions: [1404, 1824, 2016]
Total time = 158823.160
Max time = 2685.130 for space (227, 2, 3)
Average time (all spaces)      = 28.705
Average time (nonzero spaces)  = 59.853

so the current gp+sage code is faster.  However there are some individual spaces for which the

The wall time for that run was about 3 hours on 37 cores but the vast majority were done much

(237,2,14): magma 283s,  gp 11638s !!!  dims are [24,576] so we don't want any a_n but I was c

There are some similar ones.   I have no trouble when the whole space is irreducible (see exam

(227,2,3): magma 2685s, gp 66s.

I think it is worth asking Karim for any way to get the traces where the space is not irreduci

John

PS It takes 20 minutes to compare the data in the two files, checking all field isomorphisms e
```

15. Difficult spaces to crack

```
> Certainly characters can go into the Birch method.  My student Jeff
> Hein told me that he didn't see how to get odd weight, and I guess I
> have some questions about the representation theory.  But if the hard
> case is k = 2 (with character), we should be able to crush this along
> the lines you suggest: compute the charpoly modulo p^m to avoid
> coefficient blowup and to a precision m that we can still detect
> irreducibility/factor in the same way as standard algorithms for
> factoring polynomials; or maybe instead modulo several primes p to
> show that the Galois group must be transitive.  I bet this will extend
> the reach of our computations.

Cool, I think this is certainly worth doing, although I will note that
while the weight 2 cases tend to make up a lot of the hardest cases,
some
of the weight 3 spaces are worse.  Below is a list of the 50 spaces in
```

```
the
Nk^2 <= 4000 run that took more than a day of CPU time to decompose
(format is N:k:o:t:dims, where o is the character orbit index and t is
time in secs).  You will see that they are almost all irreducible, and
mostly weight 2, but the four worst are weight 3.

I should note that this list does not include spaces where magma
crashed,
so there may be worse ones.  In this list the largest dimension is
23664.

659:2:7:86131.530:[14904]
647:2:7:86704.960:[15264]
913:2:13:88332.850:[13120]
997:2:9:88743.530:[13448]
997:2:11:88932.440:[13612]
995:2:36:95736.300:[11760]
367:3:8:95768.880:[7200]
807:2:12:96388.560:[11616]
879:2:11:96784.550:[13824]
317:3:6:98371.470:[8112]
941:2:10:98799.170:[14168]
932:2:16:102256.440:[112,12768]
859:2:15:109402.970:[17040]
797:2:4:113738.590:[12870]
867:2:20:113915.770:[12800]
683:2:7:119791.410:[16800]
857:2:7:127255.000:[15052]
809:2:7:127876.620:[13400]
409:3:16:132082.350:[8576]
883:2:17:136203.910:[18396]
959:2:32:143112.970:[11520]
955:2:24:144263.650:[13536]
979:2:31:154626.130:[14080]
353:3:12:156767.980:[9280]
977:2:9:160131.810:[19440]
433:3:20:175136.130:[10224]
929:2:11:180191.070:[17248]
799:2:19:190632.330:[12320]
967:2:15:206656.620:[21120]
943:2:31:208300.560:[13120]
347:3:4:210818.510:[9804]
913:2:16:214055.630:[13120]
911:2:15:220080.060:[21600]
827:2:7:222318.230:[23664]
919:2:15:223166.620:[21888]
835:2:12:229866.700:[13448]
401:3:15:231949.390:[10560]
907:2:7:233859.160:[22500]
```

```
439:3:8:233959.950:[10368]
787:2:7:247444.260:[16900]
951:2:12:263325.890:[16224]
961:2:15:267155.290:[19680]
359:3:4:333982.220:[10502]
823:2:7:335100.710:[18496]
389:3:6:354783.950:[12288]
895:2:12:356818.500:[15488]
383:3:4:382833.470:[11970]
431:3:8:421735.840:[11928]
419:3:8:477776.880:[12420]
443:3:8:845077.270:[14016]
```

16. Issues with trusting computer output

```
chi:=FullDirichletGroup(7).1^2;
A:=ModularForms(chi,2);
assert Dimension(NewSubspace(A)) eq 4;
```

```
this will always succeed as it should, but if insert one extra line
checking the dimension of the Eisenstein subspace before checking the
dimension of the new subspace, Magma will return the wrong dimension for
the new subspace.
```

```
chi:=FullDirichletGroup(7).1^2;
A:=ModularForms(chi,2);
assert Dimension(EisensteinSubspace(A)) eq 4;
assert Dimension(NewSubspace(A)) eq 4;
```

17. More linear algebra, in Magma

```
Andrew,

> I can't guarantee this will hit the same code path internally, but
> externally it is doing exactly the same thing:
>
>> chi := FullDirichletGroup(29).1^2;
>> k := 2;
>> time S :=
>> NewformDecomposition(NewSubspace(CuspidalSubspace(ModularSymbols(chi,k,-1))));
> Time: 0.020
>> S;
> [
> Modular symbols space of level 29, weight 2, character $.1^2, and
> dimension 2 over Cyclotomic Field of order 28 and degree 12
> ]

Thanks for that.

The reason for the crash is now understood and a patch is being
```

developed.

My colleague Allan Steel looked at the slowness of your computation
and found (not surprisingly) that it arises from linear algebra over
Q or a number field. In particular, in this computation the bottlenecks
were the computation of the min polynomial of an element of a Hecke
algebra and then computing a huge resultant in order to factor the
min polynomial (mainly the latter). Allan hacked the code for this
example introducing some parallelism and this reduced the runtime
down to 40 minutes using 16 cores. The answer is

```
[
     Modular symbols space of level 743, weight 2, character $.1^2, and
dimension
     61 over Cyclotomic Field of order 742 and degree 312
]
```

It would be nice to make improvements for the general case but this
will take some serious development.

18. Running time

I finished weight 1 to level 2000, data uploaded. Approximately twice as much stuff as for 1-1

```
sage: gdata1 = read_dtp("mfdata_wt1_1000.gp.txt")
Read 26852 spaces of which 1368 are nontrivial; 2130 Galois orbits.
2088 orbits have dimension <=20
largest three dimsensions: [42, 46, 52]
Total time = 155525.304
Max time = 1869.929 for space (975, 1, 62)
Average time (all spaces)      = 5.792
Average time (nonzero spaces)  = 47.931

sage: gdata2 = read_dtp("mfdata_wt1_1001-2000.gp.txt")
Read 47348 spaces of which 2287 are nontrivial; 4309 Galois orbits.
4050 orbits have dimension <=20
largest three dimsensions: [106, 112, 130]
Total time = 1592922.855
Max time = 29630.354 for space (1950, 1, 63)
Average time (all spaces)      = 33.643
Average time (nonzero spaces)  = 210.930
```

The bulk of the weight>1 for Nk^2 up to 2000 are done, but as usual a few hard cases are holdi

John

19. Interesting examples

>> Hi John,
>>

```
>> While uploading the data I ran into a problem computing the trace
>> bound
>> for the space
>>
>>      1500:1:12:379.805:[16,16]
>>
>> because the two forms have the same tr(a_n) for n up to 1000.  I'd be
>> curious to have you compute more traces to see if this persists or
>> not.
```

References