COMPUTING CLASSICAL MODULAR FORMS

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

1. Data status

Third, I'd like to give you an update on where things stand. It looks like we now have a compl

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 newfo

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.or

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

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 gett

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

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 pr
  -> 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 abeli

In database entry:
```

Date: September 25, 2018.

```
  - 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. CM and inner twist

8. 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?
```

9. Weight 1

```
> 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.
```

10. JC comments on representing newforms

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

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!

11. 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?
>

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.
```

References