QUADRATIC ISOGENY PRIMES

USER'S GUIDE

BARINDER S. BANWAIT

Abstract. This document gives an overview of the implementation of *Quadratic Isogeny Primes*, a software package which computes, for a given quadratic field K which is not imaginary quadratic of class number one, a tight superset of the set of *isogeny primes for* K, namely, the primes p for which there exists an elliptic curve over K admitting a K-rational p-isogeny. This *user's guide* is to be considered a companion to the author's article *Explicit isogenies of prime degree over quadratic fields* which explains the theory upon which the algorithms and implementation are based. Some extended examples explaining how one may exactly compute the set of isogeny primes from the superset output by the program for specific Ks is also included here to illustrate the utility of the package.

1. Introduction

A key notion in the study of elliptic curves is that of an **isogeny**, defined as a surjective morphism $\phi : E \to E'$ between elliptic curves which maps the identity element of E to that of E'; or equivalently, precisely those surjective morphisms of curves which induce a group homomorphism between the underlying group of geometric points on E and E'. The **degree of** ϕ is defined as its degree when viewed as a morphism of curves - that is, the degree of the extension of function fields induced by the pullback of ϕ - and ϕ is said to be **separable** if this field extension is separable. If N is the degree of ϕ, one says that ϕ is an N-isogeny. If E and E' are both defined over the same field K, ϕ is said to be K**-rational** if it commutes with the natural Galois action on E and E'. It follows from the basic Galois theory of elliptic function fields that a separable K-rational N-isogeny can be decomposed as a chain of K-rational p-isogenies for p prime, so the isogenies of prime degree carry an elevated significance.

We henceforth take K to be a number field - so in particular all isogenies are separable. It is a corollary of a result of Shafarevich from 1962 [20] that an elliptic curve E/K not admitting any complex multiplication over K admits only finitely many K-rational isogenies of prime degree. In the base case of $K = \mathbb{Q}$, no elliptic curve admits complex multiplication over $\mathbb{Q}$, so every rational elliptic curve admits only finitely many prime degree isogenies.

In a seminal paper involving a delicate analysis of the Néron model of the Eisenstein quotient of the Jacobian of the modular curve $X_0(N)$, Mazur succeeded in 1978 in proving an *explicit uniform version* of the above corollary of Shafarevich for $K = \mathbb{Q}$.

2010 *Mathematics Subject Classification.* 11G05 (primary), 11Y60, 11G15. (secondary).

Theorem 1.1 (Mazur, Theorem 1 in [15])**.** *Let $E/\mathbb{Q}$ be an elliptic curve possessing a $\mathbb{Q}$-rational p-isogeny. Then p belongs to the set*

$$\mathsf{IsogPrimeDeg}(\mathbb{Q}) := \{2, 3, 5, 7, 11, 13, 17, 19, 37, 43, 67, 163\}\,.$$

It is worth stressing that, prior to this theorem, it was not even known that the set $\mathsf{IsogPrimeDeg}(\mathbb{Q})$ of primes p for which there exists an elliptic curve over $\mathbb{Q}$ possessing a $\mathbb{Q}$-rational p-isogeny was even finite!

Naturally one is motivated to determine the uniform set $\mathsf{IsogPrimeDeg}(K)$ of **isogeny primes for** K for other number fields. In general $\mathsf{IsogPrimeDeg}(K)$ may not be finite due to the possible existence of elliptic curves admitting complex multiplication over K. If such curves exist over K, then necessarily $\mathsf{IsogPrimeDeg}(K)$ is infinite, and it is a theorem of Momose [16, Theorem B] that, for quadratic number fields, this is the only way which $\mathsf{IsogPrimeDeg}(K)$ can be infinite. From the arithmetic theory of elliptic curves possessing complex multiplication, this means that, for K a quadratic number field, finiteness of $\mathsf{IsogPrimeDeg}(K)$ is equivalent to K not being one of the nine imaginary quadratic fields of class number one.

In our recent paper [4] we provide - assuming the Generalised Riemann Hypothesis - the first instances of the determination of $\mathsf{IsogPrimeDeg}(K)$ for $K \neq \mathbb{Q}$ since Mazur's work.

Theorem 1.2 ([4])**.** *Assuming GRH, we have the following.*

$$\begin{aligned}\mathsf{IsogPrimeDeg}(\mathbb{Q}(\sqrt{7})) &= \mathsf{IsogPrimeDeg}(\mathbb{Q})\\ \mathsf{IsogPrimeDeg}(\mathbb{Q}(\sqrt{-10})) &= \mathsf{IsogPrimeDeg}(\mathbb{Q})\\ \mathsf{IsogPrimeDeg}(\mathbb{Q}(\sqrt{5})) &= \mathsf{IsogPrimeDeg}(\mathbb{Q}) \cup \{23, 47\}\,.\end{aligned}$$

This is actually obtained as a corollary of the following more general algorithmic result.

Algorithm 1.3. *Let K be a quadratic field of discriminant Δ_K which is not imaginary quadratic of class number 1. Then there is an algorithm which computes a superset of $\mathsf{IsogPrimeDeg}(K)$ as the union of four explicitly computable sets:*

$$\begin{aligned}\mathsf{IsogPrimeDeg}(K) \subseteq\ &\mathsf{NotTypeOneTwoPrimes}(K) \cup \mathsf{TypeOnePrimes}(K)\\ &\cup\ \mathsf{TypeTwoPrimes}(K) \cup \mathsf{Supp}(\Delta_K).\end{aligned}$$

The termination of the algorithm relies on the Generalised Riemann Hypothesis.

It is the Sage [23] and PARI/GP [22] implementation of this algorithm that constitutes the software package *Quadratic Isogeny Primes*, and the present document is to be considered the *user's guide* for this package, the general ideas having been described in [4].

This document is organised as follows. In Section 2 we give a very quick overview of the algorithm, which is based on studying the *isogeny types*. The cases of isogenies of **Type 1** and **Type 2** require separate handling; these constitute Section 4 and Section 5 respectively. A generic and quick upper bound is shown in Section 6, and Section 7 explains the testing of the package. Finally in Section 8 we illustrate the use of the package to determine $\mathsf{IsogPrimeDeg}(K)$ in the specific case of $K = \mathbb{Q}(\sqrt{5})$. Readers wishing to simply use the package for themselves are welcome to skip the implementation details and jump straight to this final section.

2. Overview of the algorithm

This section gives an overview of *Quadratic Isogeny Primes.* More details may be found in [4].

Let E/K be an elliptic curve over a number field possessing a K-rational p-isogeny for some prime $p \geq 5$; write V for the kernel of the isogeny, a one-dimensional G_K-stable module, where $G_K := \mathrm{Gal}(\overline{K}/K)$ denotes the absolute Galois group of K, and denote by λ the *isogeny character*:

$$\lambda : G_K \longrightarrow \mathrm{Aut}\, V(\overline{K}) \cong \mathbb{F}_p^\times.$$

The study of the isogeny character was initiated by Mazur [15, Section 5], and developed by Momose, who provided the following classification, in which θ_p denotes the mod-p cyclotomic character of G_K.

Theorem 2.1 (Momose, Theorem 1 in [16])**.** *Let K be a number field. Then there exists an effective constant $C_0 = C_0(K)$ such that for any prime $p > C_0$, and for any elliptic curve admitting a K-rational p-isogeny, the isogeny character λ falls into one of the following three types:*

Type 1. λ^{12} *or* $(\lambda\theta_p^{-1})^{12}$ *is unramified.*
Type 2. $\lambda^{12} = \theta_p^6$ *and* $p \equiv 3 \pmod 4$.
Type 3. *K contains the Hilbert class field H_L of an imaginary quadratic field L. The rational prime p splits in L:*

$$p\mathcal{O}_L = \mathfrak{p}\bar{\mathfrak{p}}.$$

For any prime $\mathfrak{q}$ of K prime to $\mathfrak{p}$, with Frobenius automorphism $\sigma_{\mathfrak{q}}$,

$$\lambda^{12}(\sigma_{\mathfrak{q}}) = \alpha^{12} \pmod{\mathfrak{p}}$$

for any $\alpha \in K^\times$ with $\alpha\mathcal{O}_L = \mathrm{Nm}_{K/L}(\mathfrak{q})$.

Therefore, for K a quadratic field not imaginary quadratic of class number one, Type 3 does not arise, so Momose's theorem may be reinterpreted in this case as saying that, outside of a finite set $\mathsf{NotTypeOneTwoPrimes}(K)$, the isogeny character must be of Type 1 or Type 2. While Momose did not make his constant C_0 effective, our algorithm constructs a tight superset of $\mathsf{NotTypeOneTwoPrimes}(K)$, which the rest of this section illustrates.

The first step towards the proof of Momose's theorem is a description of how λ^{12} acts on ideals of K coprime to p (identifying λ with a character of the ideal group $I_K(p)$).

Lemma 2.2 (Momose, Lemma 1 of *loc. cit.*)**.** *Assume that K is Galois over $\mathbb{Q}$, and that p is unramified in K. Then for a fixed prime $\mathfrak{p}$ of K lying over p, there exist integers a_σ satisfying $0 \leq a_\sigma \leq 12$, for $\sigma \in \mathrm{Gal}(K/\mathbb{Q})$ such that*

$$\lambda^{12}((\alpha)) \equiv \alpha^\varepsilon \pmod{\mathfrak{p}}$$

for $\varepsilon = \sum_\sigma a_\sigma \sigma$ and $\alpha \in K^\times$ prime to p.

Thus, λ^{12} acts (modulo $\mathfrak{p}$) via an element ε of the group ring $\mathbb{Z}[\mathrm{Gal}(K/\mathbb{Q})]$, for which there are only finitely many possibilities. In our case of quadratic K, we may denote ε as a pair (a, b) of integers, referring to $a \cdot \mathrm{id} + b \cdot \sigma$, with σ being the non-trivial Galois automorphism.

As explained in Section 2 of [4], each of a and b must be valued in the set $\{0, 4, 6, 8, 12\}$; one therefore obtains 25 possible values for ε.

Lemma 2.3. *The possible values of the group ring character ϵ for a quadratic field K are as follows:*

(1) **Quadratic** *ε: the 4 pairs $(12a, 12b)$ for $a, b \in \{0, 1\}$;*
(2) **Quartic** *ε: the 5 pairs $(6a, 6b)$ for $a, b \in \{0, 1, 2\}$, excluding the 4 quadratic pairs;*
(3) **Sextic** *ε: the 12 pairs $(4a, 4b)$ for $a, b \in \{0, 1, 2, 3\}$, excluding the 4 quadratic pairs;*
(4) **Mixed** *ε: the 4 pairs $(4, 6)$, $(8, 6)$, $(6, 4)$, $(6, 8)$.*

Following Freitas and Siksek in [10] (see the discussion just before Proposition 2.2 in *loc. cit.*) we henceforth refer to ε as the **signature of the isogeny character** λ, or sometimes simply the **isogeny signature**.

Therefore, to bound the isogeny primes for a quadratic field K not imaginary quadratic of class number one, one needs to obtain a bound on the primes where the isogeny character acts via each of these 25 signatures.

We observe that we in fact do not need to consider all 25 signatures. As explained in Remark 2 of [16], if an elliptic curve E/K has a K-rational p-isogeny with character of signature $\varepsilon = (a, b)$, then the dual isogeny (which corresponds to the action of the Fricke involution w_p on $X_0(p)$) will be a K-rational p-isogeny of signature $(12 - a, 12 - b)$. Since we are interested in bounding the p that can possible arise, we need only consider signatures up to "dualising" under $\varepsilon \mapsto 12 - \varepsilon$. This brings the number of signatures to consider down to 13.

Furthermore, since swapping the integers in the signature corresponds in the moduli interpretation to sending a K-point x on $X_0(p)$ to its Galois conjugate, we need additionally only consider signatures up to this swapping. This further brings the number of signatures to consider down to a mere 10.

The signatures $\varepsilon = (0, 0)$ or $(12, 12)$ correspond to Momose's Type 1 in Theorem 2.1 above; and $\varepsilon = (6, 6)$ is related (but not exactly equivalent) to Momose's Type 2. These three signatures will be dealt with separately in the subsequent two sections of this paper, so for the rest of this section we exclude them from consideration.

The 8 remaining possiblities for εs are implemented in a global Python dictionary, with keys corresponding to the εs, and values giving the type.

```python
EPSILONS_NOT_TYPE_1_2 = {
    (0, 12): "quadratic",
    (0, 4): "sextic",
    (0, 8): "sextic",
    (4, 4): "sextic-constant",
    (4, 8): "sextic",
    (4, 12): "sextic",
    (4, 6): "mixed",
    (0, 6): "quartic",
}
```

Keeping track of the type of ε allows us to impose further conditions on any possible isogeny primes that arise from the algorithm, as explained at the end of Section 3 of [4]. Here is a snapshot of this 'filtering':

```
def filter_ABC_primes(K, prime_list, eps_type):

    if eps_type == "quadratic":
        # no additional restrictions
        return prime_list

    elif eps_type == "quartic":
        # prime must split, and be congruent to 3 mod 4
        output_list = []

        for p in prime_list:
            if p % 4 == 3:
                if not K.ideal(p).is_prime():
                    output_list.append(p)
        return output_list
```

The main idea for computing a tight superset of $\mathsf{NotTypeOneTwoPrimes}(K)$ is that of **auxiliary primes**; these are prime ideals $\mathfrak{q}$ of K of residue degree 1 lying over a rational prime q different to p, the assumed isogeny degree. In short, each auxiliary prime gives rise to a non-zero integer which p must divide; and by taking several auxiliary primes and taking the greatest common divisor of the resulting integers, one can significantly reduce the size of $\mathsf{NotTypeOneTwoPrimes}(K)$.

More precisely, we argue as follows. Let E/K be an elliptic curve over a quadratic field K admitting a K-rational p-isogeny, where the isogeny character acts via the group ring character ε not of Type 1 or 2; and let $\mathfrak{q}$ be an auxiliary prime as above. Considering the reduction type of E at $\mathfrak{q}$, we know that either E has potentially multiplicative reduction at $\mathfrak{q}$, or it has potentially good reduction at $\mathfrak{q}$.

In the potentially multiplicative reduction case, it follows - essentially from Tate's algorithm - that p must divide one of the following two non-zero integers (non-zero because ε is not of Type 1):

$$A(\varepsilon, \mathfrak{q}) := \mathrm{Nm}_{K/\mathbb{Q}}(\alpha^\varepsilon - 1) \quad \text{or} \quad B(\varepsilon, \mathfrak{q}) := \mathrm{Nm}_{K/\mathbb{Q}}(\alpha^\varepsilon - q^{12h_\mathfrak{q}}),$$

where $h_\mathfrak{q}$ denotes the order $\mathfrak{q}$ in the class group of K, and α is a generator of the ideal $\mathfrak{q}^{h_\mathfrak{q}}$. We refer to primes in the support of these two integers as **Type A** and **Type B** primes, respectively. Observe that each of these integers depends only on ε and $\mathfrak{q}$.

In the potentially good reduction case, Momose shows (Lemma 2 of [16]; see also the discussion in Section 2 of [4]), under a mild restriction on $\mathfrak{q}$ to be described below, that $\alpha^\varepsilon \neq \beta^{12h_K}$ for any root β of the characteristic polynomial of Frobenius of any elliptic curve over $\mathbb{F}_q$; consequently p must divide the following non-zero integer depending only on ε and $\mathfrak{q}$:

$$C(\varepsilon, \mathfrak{q}) := \mathrm{lcm}(\{\mathrm{Nm}_{K(\beta)/\mathbb{Q}}(\alpha^\varepsilon - \beta^{12h_K}) \mid \beta \text{ is a Frobenius root over } \mathbb{F}_q\}).$$

We refer to primes dividing this integer as **Type C** primes. Note that this LCM is being taken over all characteristic polynomials of Frobenius for elliptic curves over $\mathbb{F}_q$.

The mild restriction alluded to above is that, if K is imaginary quadratic, then $\mathfrak{q}$ must be non-principal in order to obtain that $C(\varepsilon, \mathfrak{q})$ is non-zero.

We write $ABC(\varepsilon, \mathfrak{q})$ for the lowest common multiple of the four integers $A(\varepsilon, \mathfrak{q})$, $B(\varepsilon, \mathfrak{q})$, $C(\varepsilon, \mathfrak{q})$, and q; we include q here since the possible isogeny primes of type

A, B or C were deemed to be distinct from q. For Aux a finite set of auxiliary primes, we may thus define the set of **not-Type 1 or 2** primes for K as

$$\mathsf{NotTypeOneTwoPrimes}(K) := \bigcup_{\varepsilon} \mathsf{Supp}\left(\gcd_{\mathfrak{q}\in\mathsf{Aux}}\left(ABC(\varepsilon,\mathfrak{q})\right)\right)$$

where the union is taken over the 8 pairs declared in the global dictionary `EPSILONS_NOT_TYPE_1_2`.

3. $\mathsf{NotTypeOneTwoPrimes}(K)$

This section makes some remarks on the Sage implementation of $\mathsf{NotTypeOneTwoPrimes}(K)$, which is carried out by the function `get_not_type_one_two_primes`. The requested number of auxiliary primes is taken, which in the imaginary quadratic case are further checked to be non-principal ideals:

```
def get_pre_type_one_two_primes(K, aux_prime_count=3, loop_curves=False):

    if K.is_totally_real():
        aux_primes = K.primes_of_degree_one_list(aux_prime_count)
    else:
        it = K.primes_of_degree_one_iter()
        aux_primes = []
        while len(aux_primes) < aux_prime_count:
            aux_prime_candidate = next(it)
            if not aux_prime_candidate.is_principal():
                aux_primes.append(aux_prime_candidate)
```

The main part of the function is the following block, which computes, for each auxiliary prime $\mathfrak{q}$, a dictionary, whose keys are the 8 εs, and the value for each ε is the integer $ABC(\varepsilon,\mathfrak{q})$ defined in the previous section; i.e., the lowest common multiple of the four integers $A(\varepsilon,\mathfrak{q})$, $B(\varepsilon,\mathfrak{q})$, $C(\varepsilon,\mathfrak{q})$, and q.

```
for q in aux_primes:
    q_class_group_order = C_K(q).multiplicative_order()
    # these will be dicts with keys the epsilons, values sets of primes
    AB_primes_dict = get_AB_primes(K,q,epsilons, q_class_group_order)
    C_primes_dict = get_C_primes(K, q, epsilons, q_class_group_order,
                                 loop_curves)
    unified_dict = {}
    q_rat = Integer(q.norm())
    assert q_rat.is_prime()
    for eps in epsilons:
        unified_dict[eps]=lcm([q_rat, AB_primes_dict[eps], C_primes_dict[eps]])
    tracking_dict[q] = unified_dict
```

The computation of the integers $A(\varepsilon,\mathfrak{q})$, $B(\varepsilon,\mathfrak{q})$, and $C(\varepsilon,\mathfrak{q})$ are delegated to the inner methods `get_AB_primes` and `get_C_primes`.

The first of these methods is as follows:

```
def get_AB_primes(K, q, epsilons, q_class_group_order):

    output_dict_AB = {}
    alphas = (q ** q_class_group_order).gens_reduced()
    assert len(alphas) == 1, "q^q_class_group_order not principal, which is bad"
    alpha = alphas[0]
    rat_q = ZZ(q.norm())
    assert rat_q.is_prime(), "somehow the degree 1 prime is not prime"
    for eps in epsilons:
        alpha_to_eps = group_ring_exp(alpha,eps)
        A = (alpha_to_eps - 1).norm()
        B = (alpha_to_eps - (rat_q ** (12 * q_class_group_order))).norm()
        output_dict_AB[eps] = lcm(A,B)
    return output_dict_AB
```

`get_C_primes` requires one to loop over all possible characteristic polynomials of Frobenius of elliptic curves defined over `residue_field`. This set of polynomials is contained in the set of **Weil polynomials** of degree 2 whose complex roots have absolute value equal to the square-root of the cardinality of `residue_field`. Such polynomials are implemented in Sage as `weil_polynomials`, and as such, obtaining this potentially larger set of polynomials is faster than looping over all elliptic curves over `residue_field` and computing the frobenius polynomials. The boolean flag `loop_curves` switches between these two sets of polynomials.

```
if loop_curves:
    frob_polys_to_loop = get_weil_polys(residue_field)
else:
    frob_polys_to_loop = R.weil_polynomials(2, residue_field.cardinality())
```

While looping through all polynomials in `frob_polys_to_loop`, the possible roots β are extracted, and for each β and ε, the quantity $\mathrm{Nm}_{K(\beta)/\mathbb{Q}}(\alpha^{\varepsilon} - \beta^{12\,\mathrm{ord}_{C_K}(\mathfrak{q})})$ is computed and stored in a "growing LCM" integer which itself is ultimately stored in a dictionary with key ε:

```
if frob_poly.is_irreducible():
    frob_poly_root_field = frob_poly.root_field('a')
    _, K_into_KL, L_into_KL, _ = K.composite_fields(frob_poly_root_field, 'c',
                                        both_maps=True)[0]
else:
    frob_poly_root_field = IntegerRing()
roots_of_frob = frob_poly.roots(frob_poly_root_field)
betas = [r for r,e in roots_of_frob]

for beta in betas:
    if beta in K:
        for eps in epsilons:
            N = (group_ring_exp(alpha, eps)
                    - beta ** (12*q_class_group_order)).absolute_norm()
            N = ZZ(N)
            output_dict_C[eps] = lcm(output_dict_C[eps], N)
    else:
        for eps in epsilons:
            N = (K_into_KL(group_ring_exp(alpha, eps))
              - L_into_KL(beta ** (12*q_class_group_order))).absolute_norm()
            N = ZZ(N)
            output_dict_C[eps] = lcm(output_dict_C[eps], N)
return output_dict_C
```

This allows one to only compute factorisations of Weil polynomials of all elliptic curves over $\kappa(\mathfrak{q})$ once for each $\mathfrak{q}$, rather than every time for each ε.

The dictionary thus created is collapsed and inverted to yield a dictionary with keys the εs and values the GCDs, taken over all $\mathfrak{q} \in \mathsf{Aux}$, of the integers $ABC(\varepsilon, \mathfrak{q})$:

```
    tracking_dict_inv_collapsed = {}
    for eps in epsilons:
        q_dict = {}
        for q in aux_primes:
            q_dict[q] = tracking_dict[q][eps]
        q_dict_collapsed = gcd(list(q_dict.values()))
        tracking_dict_inv_collapsed[eps] = q_dict_collapsed
```

The function then ends by taking the prime divisors of the integers thus far obtained, passing them through the `filter_ABC_primes` method shown above, and finally taking the union over all declared εs, completing the determination of $\mathsf{NotTypeOneTwoPrimes}$:

```
    final_split_dict = {}

    for eps_type in set(epsilons.values()):
        eps_type_tracking_dict_inv = {eps:ZZ(tracking_dict_inv_collapsed[eps])
                                      for eps in epsilons if epsilons[eps] == eps_type}
        eps_type_output = lcm(list(eps_type_tracking_dict_inv.values()))
        eps_type_output = eps_type_output.prime_divisors()
        eps_type_output = filter_ABC_primes(K, eps_type_output, eps_type)
        final_split_dict[eps_type] = set(eps_type_output)

    output = set.union(*(val for val in final_split_dict.values()))
    output = list(output)
    output.sort()
    return output
```

4. $\mathsf{TypeOnePrimes}(K)$

The determination of $\mathsf{TypeOnePrimes}$ is very similar to $\mathsf{NotTypeOneTwoPrimes}$: one uses auxiliary primes $\mathfrak{q}$ to compute non-zero integers which an isogeny prime must divide, and one takes the GCD of the resulting integers. More precisely, for an elliptic curve E/K over an isogeny-finite quadratic field admitting a K-rational p-isogeny of Type 1, as before one considers the two types of potential reduction E may have at $\mathfrak{q}$. The summary is as follows.

- If E has potentially multiplicative reduction at $\mathfrak{q}$, then p divides $\mathrm{Nm}(\mathfrak{q})^{12h_{\mathfrak{q}}} - 1$.
- If E has potentially good reduction at $\mathfrak{q}$, then p must divide the integer $C((0,0),\mathfrak{q})$. This integer is non-zero provided $\mathfrak{q}$ is a split prime of K.

This however requires that a certain morphism

$$f : X^{(2)}_{/S} \to \tilde{J}_{/S}$$

from the symmetric square of the modular curve $X_0(p)$ to the Eisenstein quotient of $J_0(p)$ with base $S = \mathrm{Spec}\,\mathbb{Z}[1/p]$ is a *formal immersion along* (∞,∞) away from characteristics 2, 3, and 5. We do not discuss the notion of formal immersion here;

it suffices to say that Kamienny proved [13, Proposition 3.2] that one has a formal immersion as above whenever $p \geq 73$ or $p = 67$.

Defining

$$D(\mathfrak{q}) := \operatorname{lcm}\left(\operatorname{Nm}(\mathfrak{q}), \operatorname{Nm}(\mathfrak{q})^{12h_{\mathfrak{q}}} - 1, C((0,0),\mathfrak{q})\right),$$

one therefore has the following superset for the Type 1 primes (again, for Aux a finite set of auxiliary primes):

$$\mathsf{TypeOnePrimes}(K) = \mathsf{PrimesUpTo}(61) \cup \{71\} \cup \mathsf{Supp}\left(\gcd_{\mathfrak{q}\in\mathsf{Aux}}\left(D(\mathfrak{q})\right)\right).$$

The function which computes the Type 1 primes begins as follows:

```
Q_2 = 7
def get_type_1_primes(K, aux_prime_count=3, loop_curves=False):
    """Compute the type 1 primes"""

    C_K = K.class_group()
    aux_primes = [Q_2]
    prime_to_append = Q_2
    for _ in range(1,aux_prime_count):
        prime_to_append = next_prime(prime_to_append)
        aux_primes.append(prime_to_append)
```

`aux_prime_count` is a parameter denoting the number of auxiliary primes to take, and `Q_2` is the global value of 7 denoting what Momose calls $q_{(2)}$; the code thus far takes `aux_prime_count` number of rational primes larger than or equal to 7.

Note that taking inert primes as well (which Momose does in his proof) allows us to get a potentially tighter superset; the resulting integers are not guaranteed to be non-zero, but if they are, they become a multiplicative upper bound. To guarantee termination of the algorithm, we therefore need to include at least one prime which splits in K; this is appended to `aux_primes` if not already there:

```
my_legendre_symbols = set([legendre_symbol(K._D, p) for p in aux_primes])
if 1 not in my_legendre_symbols:
    prime_to_append = next_prime(prime_to_append)
    while legendre_symbol(K._D, prime_to_append) != 1:
        prime_to_append = next_prime(prime_to_append)
    aux_primes.append(prime_to_append)
```

We initialise a dictionary to store the integers $D(\mathfrak{q})$ for each q, and then begin the loop over all such auxiliary primes, initially computing various entities associated with the residue field and order of $\mathfrak{q}$ in the class group:

```
    D_dict = {}
    R = PolynomialRing(Rationals(), 'x')

    for q in aux_primes:
        frak_q = K.primes_above(q)[0]
        residue_field = frak_q.residue_field(names='z')
        residue_field_card = residue_field.cardinality()
        frak_q_class_group_order = C_K(frak_q).multiplicative_order()
        exponent = 12 * frak_q_class_group_order
```

At this point we loop over the Weil polynomials.

```
running_D = q
if loop_curves:
    weil_polys = get_weil_polys(residue_field)
else:
    weil_polys = R.weil_polynomials(2, residue_field_card)

for wp in weil_polys:
    D = get_C00(wp, residue_field_card, exponent)
    D = Integer(D)
```

If q is a rational prime which splits in K, then we know that this `D` must be non-zero if `wp` actually comes from an elliptic curve; (recalling that, if the user has `loop_curves=False`, the set of `frob_polys_to_loop` may be larger than strictly necessary); we may therefore insist on only adding non-zero Ds to the running LCM:

```
    if legendre_symbol(K._D, q) == 1:
        if D != 0:
            running_D = lcm(running_D, D)
    else:
        running_D = lcm(running_D, D)
```

Otherwise we simply add `D` to the running LCM. This could mean that `running_D` is zero for this `q`; but since we end up taking GCDs of a list of integers not all of which are zero, this will not matter.

We add the other terms defining $D(\mathfrak{q})$ to `running_D`:

```
    running_D = lcm(residue_field_card ** exponent - 1, running_D)
    D_dict[q] = running_D
```

The computation of the inner method `get_C00` then proceeds via computing the roots of the characteristic polynomial of Frobenius:

```
def get_C00(frob_poly, residue_field_card, exponent):
    """This computes the integer C((0,0), frak_q)"""

    if frob_poly.is_irreducible():
        frob_poly_root_field = frob_poly.root_field("a")
    else:
        frob_poly_root_field = IntegerRing()
    roots_of_frob = frob_poly.roots(frob_poly_root_field)
    if len(roots_of_frob) == 1:
        assert roots_of_frob[0][1] == 2
        beta = roots_of_frob[0][0]
        return 1 + residue_field_card ** exponent - 2 * beta ** exponent
    else:
        beta, beta_bar = [r for r, e in roots_of_frob]
        return (
            1 + residue_field_card ** exponent - beta ** exponent
                                            - beta_bar ** exponent)
```

The code for this section then concludes by taking the greatest common divisor of the set of $D(\mathfrak{q})$ for the various `q` the program considered, finding the prime divisors of this GCD, and appending the *bad formal immersion primes*:

```
    output = gcd(list(D_dict.values()))
    output = set(output.prime_divisors())

    # Add the bad formal immersion primes
    output = output.union(set(prime_range(62)), {71})

    # Sort and return
    output = list(output)
    output.sort()
    return output
```

5. TypeTwoPrimes(K)

This section deals with isogeny characters of signature $(6, 6)$. An isogeny prime with this signature is not necessarily of Type 2 as in Theorem 2.1; however as explained in Section 5 of [4], one may compute a finite set of primes outside of which this is the case. Namely, writing Gen for a set of generators of the class group of K, assumed to be split and coprime to p, one obtains a multiplicative upper bound as $\operatorname{lcm}_{\mathfrak{q} \in \mathsf{Gen}} (A(\varepsilon, \mathfrak{q}), B(\varepsilon, \mathfrak{q}), q)$.

Taking therefore a finite set AuxGen of such generating sets of Cl_K, then as before this allows one to construct a multiplicative sieve for isogeny primes p whose isogeny signature is $(6, 6)$ but for which the isogeny character is not of Momose Type 2:

$$\mathsf{TypeTwoNotMomosePrimes}(K) := \mathsf{Supp}\left(\gcd_{\mathsf{Gen} \in \mathsf{AuxGen}} \left(\operatorname{lcm}_{\mathfrak{q} \in \mathsf{Gen}} (A(\varepsilon, \mathfrak{q}), B(\varepsilon, \mathfrak{q}), q)\right)\right).$$

This is implemented in the function `get_type_2_not_momose`:

```
def get_type_2_not_momose(K, aux_prime_count):
    """Compute a superset of TypeTwoNotMomosePrimes"""

    C_K = K.class_group()
    h_K = C_K.order()

    if h_K == 1:
        return set()

    class_group_gens = list(C_K.gens())

    it = K.primes_of_degree_one_iter()

    aux_gen_list = [
        get_one_aux_gen_list(C_K, class_group_gens, it) for _ in
        range(aux_prime_count)
    ]

    running_gcd = 0
    for gen_list in aux_gen_list:
        the_lcm = get_the_lcm(C_K, gen_list)
        running_gcd = gcd(the_lcm, running_gcd)

    return set(ZZ(running_gcd).prime_divisors())
```

Note that if the class group is trivial, then one may directly go from saying that λ/χ_p^6 is unramified everywhere to saying that it is trivial; i.e., there are no Type 2 primes which are not Momose Type 2 primes.

We are then left to deal with the primes of Momose Type 2. Momose gives a necessary condition that an isogeny prime of Momose Type 2 must satisfy, which in [4] is expressed as follows.

Condition CC. Let K be a quadratic field, and E/K an elliptic curve admitting a K-rational p-isogeny, with p of Type 2. Let q be a rational prime $< p/4$ such that $q^2+q+1 \not\equiv 0 \pmod{p}$. Then the following implication holds:

$$\text{if } q \text{ splits or ramifies in } K, \text{ then } q \text{ does not split in } \mathbb{Q}(\sqrt{-p}).$$

This condition may be expressed concretely via the legendre symbols $\left(\frac{D}{q}\right)$ and $\left(\frac{q}{p}\right)$, so may be checked for any prime p as follows:

```
def satisfies_condition_CC(K,p):
    for q in prime_range(p/4):
        if (q**2 + q + 1) % p != 0:
            if not K.ideal(q).is_prime():
                if legendre_symbol(q,p) == 1:  # i.e. not inert
                    return False
    return True
```

The question is then of how far we need to check: can we find an upper bound on Type 2 primes?

It was Larson and Vaintrob who found an effective bound on Type 2 primes assuming GRH, and modulo the determination of an effectively computable absolute constant c_7 (which alas is not effectively computed!), which appears in [14, Corollary 6.3]. Tracing through their proof, and combining it with the best possible bounds in the Effective Chebotarev Density Theorem due to Bach and Sorenson [2, Theorem 5.1], we are able to offer a modest improvement on their bound which removes the dependence on c_7 in our case.

Proposition 5.1. *Assume GRH. Let K be a quadratic field, and E/K an elliptic curve possessing a K-rational p-isogeny, for p a Type 2 prime. Then p satisfies*

$$p \leq (16\log p + 16\log(12\Delta_K) + 26)^4.$$

In particular, there are only finitely many primes p as above.

See [4, Proposition 5.6] for the proof. For $K = \mathbb{Q}(\sqrt{5})$, this bound on Type 2 primes is approximately 5.65×10^{10}. The algorithm therefore needs to check all primes up to this large bound.

The Sage script has implemented this check. We may check all primes up to the Type 2 bound by running the following with `bound=None`:

```
def get_type_2_primes(K, bound=None):
    """Compute a list containing the type 2 primes"""

    # First get the bound
    if bound is None:
        bound = get_type_2_bound(K)
        print("type_2_bound = {}".format(bound))
    # We need to include all primes up to 25
    # see Larson/Vaintrob's proof of Theorem 6.4
    output = set(prime_range(25))

    for p in pari.primes(25, bound):
        p_int = Integer(p)
        if p_int % 4 == 3:  # Type 2 primes necessarily congruent to 3 mod 4
            if satisfies_condition_CC(K,p_int):
                output.add(p_int)
    return output
```

However, on our version of Sage, executing this with `bound=None` quickly yields a PARI memory error. The issue is precomputing very large sets of prime numbers, which a typical architecture may not be able to handle.

We therefore translate the above routing directly into PARI/GP code. The following shows the code in the case of $K = \mathbb{Q}(\sqrt{-5})$; the other cases are similar.

```
D=-5;  \\ change this to desired value
typetwobound=56546719183;  \\ change this to corresponding bound
export(D)

\\check if condition CC is satisfied
satisfiesCC(p) =
{
  forprime(q = 7,p/4,
    if((q^2 + q + 1) % p != 0,
      if(kronecker(D,q) != -1,
        \\ don't memoize the next call
        if(kronecker(q,p) == 1,
          return(0)
        );
      );
    );
  );
  return(1);
}
export(satisfiesCC)

\\print to stdout if p satisfies condition CC
print_satisfiesCC(p) =
{
  if(satisfiesCC(p),
    print(p," is a type 2 prime")
  );
}
export(print_satisfiesCC)

\\ for D=-6,-5,6,10
congruence_condition_main(p) =
{
    if(p%24 == 19,
      x=p%5;
      if((x==2)||(x==3),
        return(1);
      return(0);
      );
    return(0);
    );
}
export(congruence_condition_main)

blockSize=100000;
export(blockSize)

checktypetwo(pBeg) =
{
    my(p,cond);
    forprime(p = pBeg*blockSize, (pBeg+1)*blockSize-1,
             cond=custom_congruence_condition(p,D);
             if(cond,print_satisfiesCC(p)));
}
export(checktypetwo)

howMany=floor(typetwobound/blockSize);

parapply(checktypetwo,[0..howMany]);
```

Note that here we have computed by hand the Kronecker symbol $\left(\frac{D}{q}\right)$ for $q = 2, 3, 5$; this allows us to start the q loop at 7, and to restrict to primes in certain congruence classes modulo 24 and 5[1]. The other cases of K are similarly optimised.

The above code checks all primes in blocks of size 100,000; this is to mitigate the overhead required in the communication between the primary and secondary threads (see [21, Section 2.4] for more on this overhead issue).

We initialise PARI/GP to precompute the requisite number of primes:

```
gp --primelimit 56546719183 --stacksize 20000000
```

Remark 5.2. The above code requires a version of PARI/GP which allows for parallel computing. Running it for `D=-5` took just over an hour of wall time with 8 threads running on an Intel Core i5 processor.

Remark 5.3. The largest Type 2 prime we have encountered for any isogeny-finite quadratic field is 163. There is a connection between Type 2 primes and an anaglogue of the class number one problem, as discussed in Goldfeld's Appendix to Mazur's paper [15]. Indeed, it is mentioned there that Joe Buhler also ran a similar computation, and also found that 163 was the largest prime observed for the quadratic fields not of class number one.

6. The DLMV bound

In this section we present the Sage code for calculating the DLMV bound for an isogeny-finite quadratic field K.

We begin by computing the bound on TypeTwoPrimes, using Sage's `find_root` function.

[1]every little helps!

```
GENERIC_UPPER_BOUND = 10 ** 30
def get_type_2_bound(K):

    # The Bach and Sorenson parameters
    A = 4
    B = 2.5
    C = 5

    n_K = K.degree()
    delta_K = K.discriminant().abs()

    D = 2 * A * n_K
    E = 4 * A * log(delta_K) + 2 * A * n_K * log(12) + 4 * B * n_K + C + 1

    R = PolynomialRing(Rationals(), 'x')
    x = R.gen()
    f = x - (D*log(x) + E) ** 4

    try:
        bound = find_root(f,10,GENERIC_UPPER_BOUND)
        return ceil(bound)
    except RuntimeError:
        warning_msg = ("Warning: Type 2 bound for quadratic field with "
        "discriminant {} failed. Returning generic upper bound").format(delta_K)
        print(warning_msg)
        return GENERIC_UPPER_BOUND
```

This searches for a root of the function

$$p - (16\log p + 16\log(12\Delta_K) + 26)^4$$

in the range $[10, 10^{30}]$, and if it doesn't find one, returns a generic bound of 10^{30}; for all of our examples, a root was found in this interval.

Using this function, we obtain the following implementation of the DLMV bound. We make one minor improvement: David took $C(K, 2(\Delta_K^{Ah_K}))$ (for A an effectively computable absolute constant which was recently actually effectively computed to be 12577 [1]). However, since we are in any case assuming GRH, we may replace the $2(\Delta_K^{Ah_K})$ with $(4\log|\Delta_K|^{h_K} + 5h_K + 5)^2$, using Theorem 5.1 of [2] (applied with $E = H_K$, the Hilbert class field of K.)

```
def DLMV(K):
    """Compute the DLMV bound"""

    # First compute David's C_0

    Delta_K = K.discriminant().abs()
    h_K = K.class_number()
    R_K = K.regulator()
    r_K = K.unit_group().rank()
    delta_K = log(2)/(r_K + 1)
    C_1_K = r_K ** (r_K + 1) * delta_K**(-(r_K - 1)) / 2
    C_2_K = exp(24 * C_1_K * R_K)
    CHEB_DEN_BOUND = (4*log(Delta_K**h_K) + 5*h_K + 5)**2
    C_0 = ((CHEB_DEN_BOUND**(12*h_K))*C_2_K + CHEB_DEN_BOUND**(6*h_K))**4
    # Now the Type 1 and 2 bounds

    type_1_bound = (1 + 3**(12 * h_K))**2
    type_2_bound = get_type_2_bound(K)

    return max(C_0, type_1_bound, type_2_bound)
```

7. Verification Testing

In order to have some testing of the code, we require examples of isogeny primes larger than 71 over specific quadratic fields. From the survey article [12], together with one example from Box's Section 4.7 [6], we have the six examples shown in Table 7.1

Prime	Is Isogeny Prime over	Found by
73	$\mathbb{Q}(\sqrt{-127})$	Galbraith [11]
73	$\mathbb{Q}(\sqrt{-31})$	Box [6]
103	$\mathbb{Q}(\sqrt{5 \cdot 577})$	Galbraith (*loc. cit.*)
137	$\mathbb{Q}(\sqrt{-31159})$	Galbraith (*loc. cit.*)
191	$\mathbb{Q}(\sqrt{61 \cdot 229 \cdot 145757})$	Elkies [9]
311	$\mathbb{Q}(\sqrt{11 \cdot 17 \cdot 9011 \cdot 23629})$	Galbraith (*loc. cit.*)

Table 7.1. Instances of isogeny primes in the literature which are rational over isogeny-finite quadratic fields.

These examples serve as the basis of a unit testing framework, which has been implemented in `test_quadratic_isogeny_primes.py`. Moreover, for each case, we check that $\mathsf{IsogPrimeDeg}(K)$ contains $\mathsf{IsogPrimeDeg}(\mathbb{Q})$. Here is a snapshot of the tests:

```
AUX_PRIME_COUNT = 2
class TestQuadraticIsogenyPrimes(unittest.TestCase):

    def test_73(self):
        K = QuadraticField(-127)
        superset = get_isogeny_primes(K, AUX_PRIME_COUNT)
        self.assertTrue(set(superset).issuperset(EC_Q_ISOGENY_PRIMES))
        self.assertIn(73, superset)

    def test_103(self):
        K = QuadraticField(5 * 577)
        superset = get_isogeny_primes(K, AUX_PRIME_COUNT)
        self.assertTrue(set(superset).issuperset(EC_Q_ISOGENY_PRIMES))
        self.assertIn(103, superset)

    def test_311(self):
        K = QuadraticField(11*17*9011*23629)
        superset = get_isogeny_primes(K, AUX_PRIME_COUNT)
        self.assertTrue(set(superset).issuperset(EC_Q_ISOGENY_PRIMES))
        self.assertIn(311, superset)
```

Furthermore, the inclusion $\mathsf{IsogPrimeDeg}(\mathbb{Q}) \subseteq \mathsf{IsogPrimeDeg}(K)$ is checked for each quadratic field $K = \mathbb{Q}(\sqrt{D})$ for $|D| \leq 100$.

```
def test_interval(self):

    R = 100

    for D in range(-R, R + 1):
        if Integer(D).is_squarefree():
            if not D in CLASS_NUMBER_ONE_DISCS:
                if D != 1:
                    K = QuadraticField(D)
                    superset = get_isogeny_primes(K, AUX_PRIME_COUNT)
                    self.assertTrue(set(superset).issuperset(EC_Q_ISOGENY_PRIMES
```

Running all of these tests takes less than three minutes on an old laptop.

8. An example: $\mathsf{IsogPrimeDeg}(\mathbb{Q}(\sqrt{5}))$

We illustrate how the *Quadratic Isogeny Primes* package may be used to exactly determine the set of isogeny primes $\mathsf{IsogPrimeDeg}(K)$ for a given quadratic field which is not imaginary quadratic of class number one. In this final section we show this for $K = \mathbb{Q}(\sqrt{5})$.

Having cloned the Git repository, the main file in the package is `quadratic_isogeny_primes.py`, which takes one required argument - the integer D for which $K = \mathbb{Q}(\sqrt{D})$ - and several optional arguments, including `--aux_prime_count`, which determines the number of auxiliary primes to take. Increasing this number will reduce the size of the final superset, but will take longer to run. In this example we take 6 auxiliary primes:

```
sage quadratic_isogeny_primes.py 5 --aux_prime_count 6
```

Running this on an old laptop takes about 20 seconds, and shows that the superset for $\mathsf{IsogPrimeDeg}(K)$ is the following:

```
superset = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
67, 71, 73, 79, 163]
```

Note that we are also warned about the following:

```
WARNING: Only checking Type 2 primes up to 1000.
```

As explained in Section 5, Sage encounters a memory error when attempting to check all primes up to the bound on Type 2 primes, which may be attempted by running the script with the option `--rigorous`; this results in the following (truncated) error:

```
type_2_bound = 56546719183
Traceback (most recent call last):
...
cypari2.handle_error.PariError: the PARI stack overflows (current size: 1000000;
 maximum size: 2596085760)
```

To check all primes up to this bound of 56546719183, we navigate to the `gp_scripts` folder, and edit the file `partype2primes.gp` for our desired D and the Type 2 bound:

```
D=5;  \\ change this to desired value
typetwobound=56546719183;
export(D)
```

We initialise PARI/GP with the requisite number of precomputed primes:

```
gp --primelimit 56546719183
```

The user may wish to confirm that their version of PARI/GP has been configured for parallel computing; if so, the number of available threads is declared upon starting the PARI/GP calculator:

```
parisize = 400003072, primelimit = 56546719183, nbthreads = 8
```

One then loads the script `partype2primes.gp` and executes the main 'parallel for loop', which checks all primes up to 56546719183 for whether they are indeed of Type 2; if so, they are printed on screen. Doing this for `D = 5` took just under 70 minutes on an old laptop:

```
? read("partype2primes.gp");
? parapply(checktypetwo,[0..howMany]);
3 is a type 2 prime
7 is a type 2 prime
23 is a type 2 prime
43 is a type 2 prime
47 is a type 2 prime
67 is a type 2 prime
163 is a type 2 prime
cpu time = 9h, 18min, 26,461 ms, real time = 1h, 9min, 55,918 ms.
?
```

Therefore, conditional upon the Generalised Riemann Hypothesis, the superset found by `quadratic_isogeny_primes.py` is indeed a superset, and no Type 2 primes larger than 1000 need to be considered. Since an elliptic curve over $\mathbb{Q}$ may be considered as an elliptic curve over any number field K, we have $\mathsf{IsogPrimeDeg}(\mathbb{Q}) \subseteq \mathsf{IsogPrimeDeg}(K)$, so we need to decide whether or not the following are isogeny primes for $\mathbb{Q}(\sqrt{5})$:

$$\{23, 29, 31, 41, 47, 53, 59, 61, 71, 73, 79\}.$$

This is equivalent to asking, for each p in this list, whether or not the modular curve $X_0(p)$ admits a non-cuspidal $\mathbb{Q}(\sqrt{5})$-rational point.

Such questions are notoriously difficult; but fortunately, many of these ps we need to consider are such that the genus of $X_0(p)$ is fairly small; and most importantly, there has been in recent years much progress in the study of *quadratic points on low-genus modular curves*. Of particular significance are the works of Bruin and Najman [8], Özman and Siksek [19], and most recently Box [7]; taken together, these three works give a complete determination of the quadratic points on $X_0(N)$ when it has genus 2, 3, 4 or 5. In essence, for each such N, there are only finitely many quadratic points which do not correspond to elliptic $\mathbb{Q}$-curves; and these finitely many points are determined explicitly. All of these works utilise Magma [5] in a significant way.

By combining these results with earlier work of Özman in determining local solubility of quadratic twists of $X_0(N)$ [18], together with a fair amount of Sage and Magma computation - notably the Chabauty package - one is able to decide that, out of the primes in the above set, only the primes 23 and 47 are isogeny primes for $\mathbb{Q}(\sqrt{5})$, thereby proving the final assertion of Theorem 1.2. This is summarised in the final section - *Weeding out the Pretenders* - of [4]; here we provide some more details of which Magma commands to run.

Prime 23. We construct the modular curve in Magma:

```
> R<x> := PolynomialRing(Rationals());
> K<a> := NumberField(R![-1, -1, 1]);
> N:=23;
> X := SmallModularCurve(N,K);
```

We search for rational points, and find a pair of points over K:

```
> rats := RationalPoints(X : Bound:=10);
> rats;
{@ (1 : 0 : 0), (1 : 1 : 0), (-3 : 23*a - 26 : 1), (-3 : -23*a - 3 : 1) @}
```

We can obtain the j-invariant of the corresponding non-cuspidal points:

```
> P := rats[3];
> PConj := rats[4];
> jInvariant(P,23);
191346871173120*a - 309605741199360
> jInvariant(PConj,23);
-191346871173120*a - 118258870026240
```

Therefore, $23 \in \mathsf{IsogPrimeDeg}(K)$. We call this method the **Magma Point Search Method**.

Prime 29. We attempt the Magma Point Search Method, but find no new points. We increase the `Bound`, but still find no new points:

```
> N:=29;
> X := SmallModularCurve(N,K);
> rats := RationalPoints(X : Bound:=1000000);
> rats;
{@ (1 : 0 : 0), (1 : 1 : 0) @}
```

We suspect therefore that such points do not exist, which we now seek to prove.

This modular curve is one of the 18 values of N for which the quadratic points on $X_0(N)$ have been classified by Bruin and Najman [8] - these are the values for which $X_0(N)$ is hyperelliptic, as originally found by Ogg [17], but excluding the notorious 37 (which was dealt with by Box, see below). These modular curves have infinitely many quadratic points which the authors call **non-exceptional**: namely those arising from preimages of $\mathbb{P}^1(\mathbb{Q})$ under the hyperelliptic map $X_0(N) \to \mathbb{P}^1$. There can only be finitely many quadratic points not of this sort; these are called

exceptional, and these are completely determined by Bruin and Najman; they also prove that the non-exceptional quadratic points correspond to quadratic $\mathbb{Q}$-curves.

We find (Table 5 of *loc. cit.*) that there are no exceptional points defined over K; therefore, if $X_0(29)$ has a non-cuspidal K-point, then it must arise as the pullback of a rational point under the hyperelliptic map. Furnishing $X_0(29)$ with a simplified model (of the form $y^2 =$ sextic), this means that a possible K-point (x, y) on this simplified model must have $x \in \mathbb{Q}$, and therefore $y = \sqrt{5} \cdot Y$ for $Y \in \mathbb{Q}$. This would yield a $\mathbb{Q}$-rational point on the 5-twist of $X_0(29)$.

We check whether this twist has any $\mathbb{Q}$-rational points:

```
> N:=29;
> X := SmallModularCurve(N);
> X_simp, f := SimplifiedModel(X);
> X5 := QuadraticTwist(X_simp,5);
> RationalPoints(X5 : Bound:=1000);
{@ @}
```

We are therefore reduced to proving that this twisted curve `X5` does not have any rational points.

We check the rank of the Jacobian:

```
> J5 := Jacobian(X5);
> RankBound(J5);
0
```

We thus have a genus 2 curve with rank zero; so to determine the rational points we can try to use `Chabauty0`:

```
> Chabauty0(J5);
{@ @}
```

This shows that `X5` does not have any rational points, and that therefore $29 \notin \mathsf{IsogPrimeDeg}(K)$. We call this method the **Twist-Chabauty0** method.

Prime 31. We apply the Twist-Chabauty0 method with `N := 31` to see that $31 \notin \mathsf{IsogPrimeDeg}(K)$.

Prime 41. Applying the Twist-Chabauty0 method here fails at the point of computing rank bounds:

```
> RankBound(J5);
Runtime error: Upper bound is not obtainable.
Upper bound for the rank of Pic^0(X)/2*Pic^0(X) is 1.
```

This suggests that the rank is 1. Since the genus of this hyperelliptic curve is 3, Chabauty's method should still in principle work to determine the points. We present here another method.

These twists of the modular curves $X_0(N)$ were studied by Özman [18], who determined for each p whether or not $X^d(\mathbb{Q}_p)$ is empty; that is, if there is a **local obstruction** at a given place. We show, using part (5) of Özman's Theorem 1.1 in *loc. cit.*, that $X^5(41)(\mathbb{Q}_5)$ is empty.

We thus adopt the notation of Section 4 of *loc. cit.*. Let $\mathbb{K}$ denote the quadratic field $\mathbb{Q}(\sqrt{5})$, ν the prime of $\mathbb{K}$ lying over $p = 5$, and R the ring of integers of the completion $\mathbb{K}_\nu$. We take $N = 41$, and denote by $\mathcal{X}_0(N)$ the Deligne-Rapoport model of $X_0(N)$, which is smooth and regular over $\mathbb{Z}[1/N]$. Letting w_N be the Atkin-Lehner involution, Özman defines the set S_N to be the primes p for which there is a w_N-fixed, $\mathbb{F}_p$-rational point on the special fiber of $\mathcal{X}_0(N)_{/R}$, and relates this to the local solubility of $X^d(N)$ as follows.

Theorem 8.1 (Özman, Theorem 4.10 in [18])**.** *Let p be a prime ramified in $\mathbb{Q}(\sqrt{d})$ and N a square-free integer such that p is unramified in $\mathbb{Q}(\sqrt{-N})$. Then $X^d(N)(\mathbb{Q}_p) \neq \emptyset$ if and only if p is in the set S_N.*

Thus, for our purposes, we would like to show that $5 \notin S_{41}$.

Özman gives a concrete description of the primes in S_N. To state it, let us make the following definitions:

$$\begin{aligned}
\mathbb{M} &= \mathbb{Q}(\sqrt{-N})\\
\mathcal{O} &= \mathbb{Z}[\sqrt{-N}]\\
j(\mathcal{O}) &= j\text{-invariant of the order } \mathcal{O}\\
\mathbb{B} &= \mathbb{Q}(j(\mathcal{O}))\\
\mathbb{H} &= \text{Hilbert class field of } \mathbb{M}.
\end{aligned}$$

Then Özman proves the following.

Proposition 8.2 (Özman, Proposition 4.6 in *loc. cit.*)**.** *Let p be an odd prime and let $\mathcal{P}$ be a prime of $\mathbb{M}$ lying over p. Then p is in S_N if and only if there exists a prime ν of $\mathbb{B}$ lying over p such that the residue degree $f(\nu|p) = 1$ and $\mathcal{P}$ totally splits in $\mathbb{H}/\mathbb{M}$.*

Although this proposition is stated as only applying for odd primes, it also holds for $p = 2$, as proved by Özman in a separate argument appearing between Remark 4.8 and Theorem 4.10 in *loc. cit.*.

This very concrete description of S_N can be implemented in Sage:

```
def oezman_sieve(p,N):
    """Returns True iff p is in S_N"""

    M.<a>=QuadraticField(-N)
    h_M = M.class_number()
    H = M.hilbert_class_field('b')
    primes_above_p = M.primes_above(p)

    primes_tot_split_in_hcf = []

    for P in primes_above_p:
        if len(H.primes_above(P)) == h_M:
            primes_tot_split_in_hcf.append(P)

    if not primes_tot_split_in_hcf:
        return False

    R.<x> = QQ[]
    f = R(hilbert_class_polynomial(M.discriminant()))
    B.<t> = NumberField(f)
    assert B.degree() == h_M  # from Oezman's proof

    possible_nus = B.primes_above(p)

    for nu in possible_nus:
        if nu.residue_class_degree() == 1:
            return True

    return False
```

Running `oezman_sieve(5,41)` yields `False`, and hence $41 \notin \mathsf{IsogPrimeDeg}(K)$. We call this method the **Özman sieve**.

Remark 8.3. Although Theorem 8.1 requires the prime p to be unramified in $\mathbb{Q}(\sqrt{-N})$, the implication

$$p \notin S_N \implies X^d(N)(\mathbb{Q}_p) = \emptyset$$

still holds if p ramifies in both $\mathbb{Q}(\sqrt{d})$ and $\mathbb{Q}(\sqrt{-N})$; this is Part (6) of Theorem 1.1 of [18]. Since we shall only ever be concerned with showing that $X^d(N)(\mathbb{Q}_p) = \emptyset$, we may apply `oezman_sieve(p,N)` for any p that ramifies in $\mathbb{Q}(\sqrt{d})$.

Remark 8.4. Özman already used this method in Example 4.11 of *loc. cit.* to show that $29 \notin \mathsf{IsogPrimeDeg}(K)$.

Prime 47. The Magma Point Search Method yields that $47 \in \mathsf{IsogPrimeDeg}(K)$:

```
> N:=47;
> X := SmallModularCurve(N,K);
> rats := RationalPoints(X : Bound:=100);
> P := rats[3];
> PConj := rats[4];
> jInvariant(P,N);
-368136133486354759680*a - 2275206429815355801 60
> jInvariant(PConj,N);
368136133486354759680*a - 595656776467890339840
```

Prime 53. This is the first prime that is not in the list of Bruin and Najman.

Fortunately, Box [6] has determined (Section 4.2 of *loc. cit.*) that all quadratic points on $X_0(53)$ are non-exceptional, although in a different sense to that of Bruin and Najman; namely, that all quadratic points arise from the pullback of $\mathbb{Q}$-points on $X_0^+(53)$ via the natural quotient map $\rho : X_0(53) \to X_0^+(53)$; moreover all such points correspond to $\mathbb{Q}$-curves.

We may thus apply the Özman sieve to show that $53 \notin \mathsf{IsogPrimeDeg}(K)$.

Note that this case also follows from Theorem 2.5 in [12], since $53 \equiv 1 \pmod 4$, but is not a norm of K.

Prime 59. Table 17 of [8] shows that $X_0(59)$ does not contain any exceptional quadratic points. The Özman sieve then shows that $59 \notin \mathsf{IsogPrimeDeg}(K)$.

Prime 61. Section 4.3 of [6] shows that $X_0(61)$ does not contain any exceptional quadratic points. The Özman sieve then shows that $61 \notin \mathsf{IsogPrimeDeg}(K)$.

Prime 71. Table 18 of [8] shows that $X_0(71)$ does not contain any exceptional quadratic points. The Özman sieve then shows that $71 \notin \mathsf{IsogPrimeDeg}(K)$.

One may also attempt the Twist-Chabauty0 method here, but the last step of calling `Chabauty0` fails, since `TorsionSubgroup` is (at the time of writing) only implemented for Jacobians of genus 2 curves over the rationals.

However, in this case, we can show that the torsion subgroup of the Jacobian of $X^5(71)$ is trivial by computing the number of points over various reductions and showing that their greatest common divisor is 1:

```
> my_BadPrimes := BadPrimes(X5);
> set:={};
> for p in PrimesUpTo(15) do
for>     if not p in my_BadPrimes then
for|if>         J5_red := BaseChange(J5, GF(p));
for|if>         set:=set join {#J5_red};
for|if>         printf "prime = %o,  point size = %o\n",p, #J5_red;
for|if>     end if;
for> end for;
prime = 3,  point size = 1159
prime = 7,  point size = 238144
prime = 11,  point size = 2721600
prime = 13,  point size = 8817984
> GCD(set);
1
```

We are thus in the following situation:

Lemma 8.5. *Let X be a hyperelliptic curve over a number field K of genus ≥ 2, none of whose Weierstrass points are rational over K. Suppose also that the Mordell-Weil group of the Jacobian,* $\mathrm{Jac}(X)(K)$*, is trivial. Then $X(K)$ is empty.*

Proof. Suppose for a contradiction that $P \in X(K)$. We use this as the base-point for the Abel-Jacobi embedding $\iota_P : X \hookrightarrow \mathrm{Jac}(X)$. Write $X : y^2 = f(x)$ for f of even degree. Choose $a \in K$, define the degree 2 rational divisor $D := (a, \sqrt{f(a)}) + (a, -\sqrt{f(a)})$, and consider the degree 0 divisor $D - 2P$. Since $\mathrm{Jac}(X)(K)$ is trivial, we have that D is linearly equivalent to $2P$, so there exists a rational function g such that $2P = D + \mathrm{div}(g)$; thus g belongs to $\mathcal{O}(D)$, the Riemann-Roch space of D. This is a 2-dimensional K-vector space, so we may take the functions 1 (constant function) and $1/(x-a)$ as a basis over K. Therefore we have that $g = (x-b)/(x-a)$, from which it follows that $2P = (b, \sqrt{f(b)}) + (b, -\sqrt{f(b)})$. But since P is K-rational, we must have that $f(b) = 0$, i.e. that $(b, 0)$ is a K-rational Weierstrass point, contradicting our hypothesis. □

We are grateful to Samir Siksek who explained to us the proof of the above lemma.

Prime 73. Section 4.7 of [6] determines all 11 quadratic points on $X_0(73)$ (besides the cusps), and none are defined over K. Note that this result relies on the determination of the $\mathbb{Q}$-points on $X_0^+(73)$, completed by Balakrishnan, Best, Bianchi, Lawrence, Müller, Triantafillou and Vonk (Theorem 6.3 in [3]).

Prime 79. This is dealt with in the appendix to [4], written jointly with Derickx.

Using the methods described in the above example, we may now prove the remaining two cases of Theorem 1.2 more succinctly.

Proof of Theorem 1.2. Table 8.1 collects the relevant information for showing that the primes in the set

$$\{23, 29, 31, 41, 47, 53, 59, 61, 71, 73\}$$

do not arise as isogeny primes for $\mathbb{Q}(\sqrt{7})$ or $\mathbb{Q}(\sqrt{-10})$. The relevant result from the tables of Bruin-Najman and Box are cited, which show that, for each p as above, if p is an isogeny prime, then it must arise as a non-exceptional K-point; equivalently, arises as a quadratic $\mathbb{Q}$-curve. We also take this opportunity to summarise the case of $\mathbb{Q}(\sqrt{5})$, allowing all three instances to be summarised in one table. The entries in the table are then one of the following:

(1) TC0 - referring to the Twist-Chabauty0 method which deals with that case;
(2) the value `q` at which `oezman_sieve(q,p)` yields `False`, showing that no such quadratic $\mathbb{Q}$-curve exists. Note that $\mathbb{Q}(\sqrt{7})$ or $\mathbb{Q}(\sqrt{-10})$ admit 2 ramified primes, giving us two opportunities to apply the Özman sieve in these cases (Recall Remark 8.3);
(3) ✗ - referring to the cited reference allowing us to conclude negatively immediately;
(4) ✓ - referring to that prime being an isogeny prime for that quadratic field;
(5) — - referring to that case not being relevant (this only applies to the prime 79, where for $\mathbb{Q}(\sqrt{-10})$ and $\mathbb{Q}(\sqrt{7})$ we know that 79 is not a possible isogeny prime).

Running `sage quadratic_isogeny_primes D --aux_prime_count 25`, for $D = 7$ and -10 then shows that 163 is the only isogeny prime greater than 73, and the proof of Theorem 1.2 is complete. □

References

1. Jeoung-Hwan Ahn and Soun-Hi Kwon, *An explicit upper bound for the least prime ideal in the Chebotarev density theorem*, Annales de l'Institut Fourier **69** (2019), no. 3, 1411–1458.
2. Eric Bach and Jonathan Sorenson, *Explicit bounds for primes in residue classes*, Mathematics of Computation **65** (1996), no. 216, 1717–1735.
3. Jennifer S Balakrishnan, Alex J Best, Francesca Bianchi, Brian Lawrence, J Steffen Müller, Nicholas Triantafillou, and Jan Vonk, *Two recent p-adic approaches towards the (effective) Mordell conjecture*, Preprint available online at `https://arxiv.org/abs/1910.12755`, 2019.
4. Barinder S. Banwait, *Explicit isogenies of prime degree over quadratic fields*, Submitted, preprint available online at `https://arxiv.org/abs/2101.02673`, 2021.
5. Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system. I. The user language*, J. Symbolic Comput. **24** (1997), no. 3-4, 235–265, Computational algebra and number theory (London, 1993). MR MR1484478
6. Josha Box, *Quadratic points on modular curves with infinite Mordell–Weil group*, submitted. Preprint available online at `https://arxiv.org/abs/1906.05206`, 2019.
7. ———, *Quadratic points on modular curves with infinite Mordell–Weil group*, Mathematics of Computation **90** (2021), 321–343.
8. Peter Bruin and Filip Najman, *Hyperelliptic modular curves $X_0(n)$ and isogenies of elliptic curves over quadratic fields*, LMS Journal of Computation and Mathematics **18** (2015), no. 1, 578–602.

p	Reference	$\mathbb{Q}(\sqrt{-10})$	$\mathbb{Q}(\sqrt{5})$	$\mathbb{Q}(\sqrt{7})$
23	Table 2 of Bruin-Najman	2	✓	2
29	Table 5 of Bruin-Najman	TC0	TC0	2
31	Table 7 of Bruin-Najman	TC0	TC0	TC0
41	Table 12 of Bruin-Najman	5	5	7
47	Table 14 of Bruin-Najman	2	✓	7
53	Section 4.2 of Box	5	5	2
59	Table 17 of Bruin-Najman	5	5	7
61	Section 4.3 of Box	5	5	7
71	Table 18 of Bruin-Najman	5	5	2
73	Section 4.7 of Box	✗	✗	✗
79	Appendix A of [4]	—	✗	—

TABLE 8.1. Deciding on possible isogeny primes. The numbers in the table correspond to primes where the Özman sieve was applied. "TC0" refers to an application of the Twist-Chabauty0 method, ✗ means the reference cited is enough to decide negatively on that prime, ✓ means the prime is an isogeny prime for that field, and — means the corresponding case is not relevant. Bruin-Najman refers to [8] and Box to [6]. In all cases apart from where ✓ is present, this table is ruling out primes.

9. Noam D Elkies, *Elliptic and modular curves over finite fields and related computational issues*, Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A.O.L. Atkin (Providence, R.I.) (D.A. Buell and J.T. Teitelbaum, eds.), American Mathematical Society, 1998, pp. 21–76.
10. Nuno Freitas and Samir Siksek, *Criteria for Irreducibility of mod p Representations of Frey Curves*, Journal de Théorie des Nombres de Bordeaux **27** (2015), no. 1, 67–76 (en). MR 3346965
11. Steven D. Galbraith, *Rational points on $X_0^+(p)$*, Experiment. Math. **8** (1999), no. 4, 311–318.
12. Josep González, Joan-Carles Lario, and Jordi Quer, *Arithmetic of $\mathbb{Q}$-curves*, Modular Curves and Abelian Varieties (Basel) (John E. Cremona, Joan-Carles Lario, Jordi Quer, and Kenneth A. Ribet, eds.), Birkhäuser Basel, 2004, pp. 125–139.
13. Sheldon Kamienny, *Torsion points on elliptic curves and q-coefficients of modular forms*, Inventiones mathematicae **109** (1992), no. 1, 221–229.
14. Eric Larson and Dmitry Vaintrob, *Determinants of subquotients of Galois representations associated with abelian varieties*, Journal of the Institute of Mathematics of Jussieu **13** (2014), no. 3, 517559.
15. Barry Mazur, *Rational isogenies of prime degree*, Inventiones mathematicae **44** (1978), no. 2, 129–162, With an appendix by Dorian Goldfeld.
16. Fumiyuki Momose, *Isogenies of prime degree over number fields*, Compositio Mathematica **97** (1995), no. 3, 329–348.
17. Andrew Ogg, *Abelian curves of 2-power conductor*, Proceedings of the Cambridge Philosophical Society **62** (1966), 143–148.
18. Ekin Özman, *Points on quadratic twists of $X_0(N)$*, Acta Arithmetica **152** (2012), 323–348.
19. Ekin Özman and Samir Siksek, *Quadratic points on modular curves*, Mathematics of Computation **88** (2019), no. 319, 2461–2484.
20. Igor R. Shafarevich, *Algebraic number fields*, AMS Translations **31** (1962), 25–39, first appeared in Proceedings of the International Congress of Mathematicians, Stockholm.
21. The PARI Group, Univ. Bordeaux, *Introduction to parallel GP*, 2021, available from `https://pari.math.u-bordeaux.fr/pub/pari/manuals/2.12.0/`.

22. The PARI Group, Univ. Bordeaux, *PARI/GP version 2.14.0*, 2021, available from `http://pari.math.u-bordeaux.fr/`.
23. The Sage Developers, *Sagemath, the Sage Mathematics Software System (Version 9.2)*, 2020, `https://www.sagemath.org`.

Barinder S. Banwait, Harish-Chandra Research Institute, Prayagraj, India
Email address: `barinder.s.banwait@gmail.com`