# NUCUBE

Caleb Smith

August 2020

## Introduction

Given a Binary Quadratic Form, $f$, and an integer $T$, computing $f^{2^T}$ is conjectured to require $T$ sequential squarings. Currently, the state of the art for composing a positive definite form with itself is the NUDUPL algorithm [JVDP02].

An alternative to repeatedly squaring is to repeatedly cube, using an algorithm such as NUCUBE, as described in [IJJS10]. The obvious benefit to this approach is that you would need to perform fewer than $T$ sequential cubing operations, specifically, one would only need to compute $T' = \log_3(2) * T \approx 0.6309 * T$ cubings. Note that although we cannot hit the precise value of $2^T$ via $3^{T'}$, once we arrive at a form, $f'$, which is close, but does not exceed, $f^{2^T}$, we can compute $f''$ such that $f' * f'' = f^{2^T}$ in parallel while computing $f'$.

## Benchmarking

Slava provided me with some code as a starting point to benchmark NUCUBE and NUDUPL, which is dependent on libararies by Maxwell Sayles [Say14b][Say14a], who provides the implementations of both NUDUPL and NUCUBE. Although this is not an ideal comparison, because it does not appear that Sayles maximally parallelized the algorithms, it should be sufficient to get an idea of performance.

In my tests, I selected a number of squaring iterations and matched the approximately equivalent number of cubings iterations (e.g. 1,000,000 squarings vs 630,929 cubings). There were three different iteration counts I measured (1,000,000, 10,000,000, and 30,000,000 squarings and their cubing counterparts) and I also repeated the computations twice, where once we reduced after every iteration, and the other where we never reduced. Finally, this was done over 128, 256, 512, 1024, 2048, and 4096-bit discriminants. The table below shows the results for different discriminant sizes, where the gap is defined as the factor of how much longer cubing takes to finish compared to squaring. The list of discriminants used will be at the end of the document.

| Bits | Gap |
|---|---|
| 128 | 1.020 |
| 256 | 1.034 |
| 512 | 1.057 |
| 1024 | 1.124 |
| 2048 | 1.303 |
| 4096 | 1.474 |

I believe the reason as to why cubing is slower is because in [Say13], algorithm 2.4, line 4, you see that modulo reduction is with a modulus twice the bit-length of NUDUPL ([Say13] algorithm 2.3, line 3). In the original NUCUBE paper, the authors mention exploring more specialized algorithms for higher powers, such as quintupling. If any new results in this domain are published, it would probably be worthwhile investigating their performance measures, given how surprisingly close NUCUBE is to NUDUPL.

Included in libq.c is commented out sections that will measure the accumulated time spent in the squaring/cubing operation and/or reductions. After looking into these, the reduction operation accounts for a fairly small percentage of the overall computation. It did account for a larger percentage of the computation for cubing compared to squaring, so a fast implementation of reducing would likely shrink the gap even further.

# References


[IJJS10] Laurent Imbert, Michael J Jacobson Jr, and Arthur Schmidt. Fast ideal cubing in imaginary quadratic number and function fields. *Advances in Mathematics of Communications*, 4(2):237, 2010.

[JVDP02] Michael J Jacobson and Alfred J Van Der Poorten. Computational aspects of nucomp. In *International Algorithmic Number Theory Symposium*, pages 120–133. Springer, 2002.

[Say13] Maxwell Sayles. Improved arithmetic in the ideal class group of imaginary quadratic number fields with an application to integer factoring. Master's thesis, Graduate Studies, 2013.

[Say14a] Maxwell Sayles. *Liboptarith*, 2014. `https://github.com/maxwellsayles/liboptarith`.

[Say14b] Maxwell Sayles. *Libqform*, 2014. `https://github.com/maxwellsayles/libqform`.

## Discriminants

128 $bit$ : $-309361850110207312103360594127882960623$

256 $bit$ : $-102522683075961319165911365929379221882824975317180317196283357569799441166543$

512 $bit$ : $-900033056416639750932759042388903806271587957296221292145057048595614285498590105180567690327394033139568645809185843039000228390237927816822308995941451 1$

1024 $bit$ : $-14114031779479266886294333265685651937848229142872728741331872208921644856715573709476890364371640451754971538566416336031629628415531005898098437377051739849295186016171796036887422747366933654181857516683920922868475581107141637638455190214978018453208688168357607147964649960133082425926064595251720552667 9$

2048 $bit$ : $-27793237001305170698656509497755844092586386872496295629779837507971117559354691523581890478176914693359675630301087242227995550958743592046124379271342523117944672201940224307563498623282396336043787901362415643315794359898423628976229444385132461815183120709043651974544581523175645485793883889543615011559151806508862786144831212673747194337362835767292592316842392285011597319847853278165631070684231627247677822273593999074723433871243030768063363559931194419142809976257675348498631355931756891791994467558601521208397038790339158400191945339609224183625935566879000287625158572127711542235803324639115306136663$

4096 $bit$ : −9695426998390542133727565590819102614415724220493882796250571579142025733604762905388150846970164625333880187358573212755064723841162404317856512168429597001282079431539837024747711024173870673479543460103567530167817277185748441324576198186981348153716412870442798478462757811762106032654542270562011585525733737733085615366626533424339020351239581543199684932516656000241152684007889386440940459946236524448642897716655674126164057691359356204523821816510240120032572312987803509366063716678779881533529609516340134872350841850365016773858539129332201629742379104991097175158663158519829980508118895884671435984875812053677253855322505493524162453061641871762087119164653527322767596794389023181647352872917484413122256906987114843132346049368358261061501051793754976174190623937685085623668090537111250234428701792929177142784045003886526958146926027979074990145133385981099254967812561723909220503804079950048800957581535001803758516834625549926155146793037685392697786356451353063065597941613113517197182509809147863232723555743773050774011389160266742401569953692223891099020089736468436702327665166015169523298890139777792213082666680075032726461229326228812016814617099758827864112526819937199827074874773661285852677094058795