

Re: Simple answer, surrogate factoring

Source: <http://sci.tech-archive.net/Archive/sci.math/2005-03/1468.html>

From: Tim Peters (*tim.one_at_comcast.net*)

Date: 03/04/05

Date: Thu, 3 Mar 2005 23:46:26 -0500

[JSH]

[...]

> where $f_1 f_2 = Tj^2$, and you iterate through all possible integer f_1

> and f_2 , and take the gcd of Ax with M .

>

> If Ax has M as a factor, then you calculate that ratio of z to x .

>

> The numerator is just

>

> $(+/- (f_1 - f_2) + 2j^2 +/- (f_1 + f_2))$

I couldn't follow the derivation, so will just ask: the redundant pair of outermost parentheses here is surprising, so is this really what you meant to type?

> while the denominator is

>

> $2M^2 - 2(+/- (f_1 - f_2) + 2j^2)$

This one is missing a right parenthesis; don't know whether

$$2M^2 - 2(+/- (f_1 - f_2)) + 2j^2$$

or

$$2M^2 - 2(+/- (f_1 - f_2) + 2j^2)$$

was intended.

> and you divide out any factors in common between them, and then take

> the gcd of the denominator with M , and for at least one case for any

> non-zero j coprime to M , you will have a prime factor of M .

>

> The basic algorithm is now perfect.

I tried all plausible (to me) ways of fleshing out an implementation from this, and they all failed to factor some 3-digit composites.

With the

$$2M^2 - 2(+/-)(f_1 - f_2) + 2j^2$$

guess, only 2 of 1000 products of two random 20-bit primes factored, always using $j=1$.

With the

$$2M^2 - 2(+/-)(f_1 - f_2) + 2j^2$$

guess, again sticking to $j=1$, 1 of 1000 factored.

All of this is consistent with the hypothesis that the thing that matters the most in these algorithms is how many splittings are tried; Tj^2 generally has few splittings, and especially when sticking to $j=1$.

*> I think there are far faster more elegant algorithms out there, but
> at this point, it's proof of concept time.*

Well, this is by far the fastest non-factoring algorithm of this kind I've tested to date <wink>.