mRemainders as Special Numbers

by *Euclid, Fermat, Euler, Lagrange,..., and Resh*

pdf , Ed. 1.2

Frrraction app om iPad, by GRS Enterprises, NLC

mRemainders as Special Numbers

pdf , Ed. 1.2

Jim Resh, GRS Enterprises NLC

September 23-Oct 9, 2011

mRemainders as Special Numbers

pdf , Ed. 1.2

Summary

The note begins with a brief survey of mRemainders[1] as numbers, then introduces mRemainder arithmetic, and finally tells how to use *Frrraction*[2] to do one of the harder bits of mRemainder arithmetic: calculating their reciprocals.

mRemainders: Are they integers?

No. With names like 0, 1, 817, and the like[3], they *look* like integers, but...unlike actual integers, they can divide each other and the result always looks like another integer. (Makes you wonder about mRemainder fractions, doesn't it—since their ratios dissolve back into pseudo-integers? *e.g.* 2/3 is 4 (mod 5).) The things are formally called "congruence classes modulo *m*", where *m* itself really is an integer. In Frrraction Guide we'll just call them what they are: *mRemainders mod m*.

Reminder. Division of two positive integers is defined this way: Dividing *a* by *b* creates unique **quotient** *q* and **remainder** *r* which satisfy the equation: $a/b = q + r/b$ where $0 <= r < b$. (Without the limits on *r* the numbers aren't unique.) This is what *Frrraction* calls converting from "pure fraction" to "mixed fraction" form. The quotient *q* is called "the integer part of *a*/*b*" and *r*/*b* is called "the fractional part of *a*/*b*". The defining equation can also be written without fractions as $a = q \cdot b + r$.

---

1 mRemainders are properly called "residue classes", but I find that term to be a bit less intuitive.

2 *Frrraction* is a fraction calculator app for the Apple iPhone that emphasizes continued fractions. It's currently undergoing beta testing.

3 Sometimes over-bars or notations like $(0)_m$ $(1)_m$ $(817)_m$ *etc.* are used to emphasize that mRemainders are not integers, but that quickly gets tiresome to read. It's simpler to just remember: Not integers.

Definition. The ***mod-m mRemainder c,*** is just an integer in the range $0 \le c < m$. For a given m, there are only a finite number of mRemainders. For instance, there are only two mRemainders mod 2. They are 0 and 1. As with mod-2, there are always exactly m mRemainders mod m. Their common names are: 0, 1, 2, ..., $m-1$.

Definition. ***Addition***, ***subtraction***, and ***multiplication*** of mRemainders mod m are defined as extensions of ordinary integer arithmetic: If a and b are mRemainders mod m then add, subtract, or multiply them as though they were integers, then replace the integer result by the remainder it produces when divided by m. (Often, the quick way to do that replacement step is to simply add or subtract m.)

Example addition: Addition on a 12 hour clock is an example of mRemainder arithmetic. 5 hours after 8:00 is 1:00. Using mRemainder arithmetic, we can write this as $8 + 5 = 13 \equiv 1 \pmod{12}$ (because 13/12 = 1+**1**/12, *i.e.* the remainder of 13/12 is 1).

Note. In expressions involving mRemainders, the combination "≡" and "(mod)" work together replacing the simple equals-sign "=" to distinguish mRemainder arithmetic from normal integer arithmetic. We owe this notation to Karl Gauss. The basic rules of arithmetic still apply: $a + b \equiv b + a \pmod{m}$; if $a \equiv b \pmod{m}$ and $b \equiv c \pmod{m}$ then $a \equiv c \pmod{m}$, and so forth.

Example addition. The sum of 3 + 2 mod 5 is calculated as: 3 + 2 = 5 and the remainder of 5/5 is 0. So we write: $3 + 2 \equiv 0 \pmod{5}$.

Reminder. The **negative** of a number is a number that yields 0 when added to the first. Within the integers, all numbers have negatives. The **reciprocal** of a number is a number that yields 1 when *multiplied* by the first. Within the integers, only 1 and −1 have reciprocals.

Note: Since the result of 3 + 2 (mod 5) is 0, we can say that 2 is the negative of 3 or, equally, that 3 is the negative of 2 (mod 5).

Example multiplication. The product 3 · 2 mod 5 is calculated as: 3 · 2 = 6 and the remainder of 6/5 is 1. So we write: 3 · 2 ≡ 1 (mod 5).

Note: Since the result of the above example multiplication is 1, we can say that 2 is the reciprocal of 3 or, equally, that 3 is the reciprocal of 2 (mod 5). This reveals **the central oddity of these integer-like mRemainder things**: If the modulus m is a prime number (*i.e.* divisible only by 1 and itself) then all nonzero mod-m mRemainders have reciprocals.

Definition. Division $a \div b$ of two mRemainders modulo m is defined as multiplying by the reciprocal of the denominator. Thus, if $c \equiv b^{-1} \pmod{m}$ then $a \div b \equiv a \cdot c \pmod{m}$.

Example. Table 1 shows the addition, multiplication, negative, and reciprocal tables for mod 5.

Table 1.

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

·	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

c	−c
0	0
1	4
2	3
3	2
4	1

c	1/c
0	-
1	1
2	3
3	2
4	4

Exercise. Write addition, multiplication, negative, and reciprocal tables for mod 6 in Table 2.

Table 2.

+	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

x	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

c	−c
0	
1	
2	
3	
4	
5	

c	1/c
0	
1	
2	
3	
4	
5	

The absence of many inverses in the mod-6 exercise is because 6 is not a prime number. Try it with mod 7 or 11 (or mod p for any *prime* number p) and you'll find that *all* their nonzero elements will have inverses.

Exercise. Fill Table 3, for mod 7.

Table 3.

Example division. $2 \div 5 \equiv 2 \cdot 5^{-1} \equiv 2 \cdot 3 \equiv 6 \pmod 7$. ■

Calculating reciprocals, mod *m*

Finding mod-*m* **negatives** is **easy**: The negative of mRemainder *c* (mod *m*) is *m*−*c*.

Finding mod-*m* **reciprocals** is **hard to do unless *m* is small**—small enough to build the multiplication table for modulus *m*. Hunt in the table for a 1-entry in the row of interest; that 1's column-label is then the desired reciprocal. For instance, find 1/3 in Table 1 by noticing that the column-2 entry of row 3 is 1, so $2 \equiv 1/3 \pmod 5$.

The tabular approach is impractical when the modulus is large. How about forming the table for $m = 2039$? For instance, can you find $1000^{-1} \pmod{2039}$? [It is 1307.]

The standard literature of Number Theory has two other inversion methods, called Euclid's u·c+v·m method[4] and Euler's totient method[5]. Both are quite involved, not particularly easy. Below are four new methods, which work well with *Frrraction*.

An elementary inversion algorithm: The +++ Method

Put 0+*m*/*c* into *Fraction F1*, put 0+0/1 into *Fraction F2*, and leave *F2num* active. Repeatedly tap + (add) while watching *F2num*. Stop when *F2num* contains c−1. At that time, the desired mod-*m* reciprocal 1/*c* is 1+*F2int*.

Example. Use the +++ Method to find $10^{-1} \pmod{2047}$.

Solution: Put 0+2047/10 into *F1*, 0+0/1 into *F2*, and leave *F2* active.

---

4 http://www.math.ou.edu/~kmartin/nti/chap2.pdf Section 2.3

5 http://www.math.ou.edu/~kmartin/nti/chap3.pdf Section 1.5

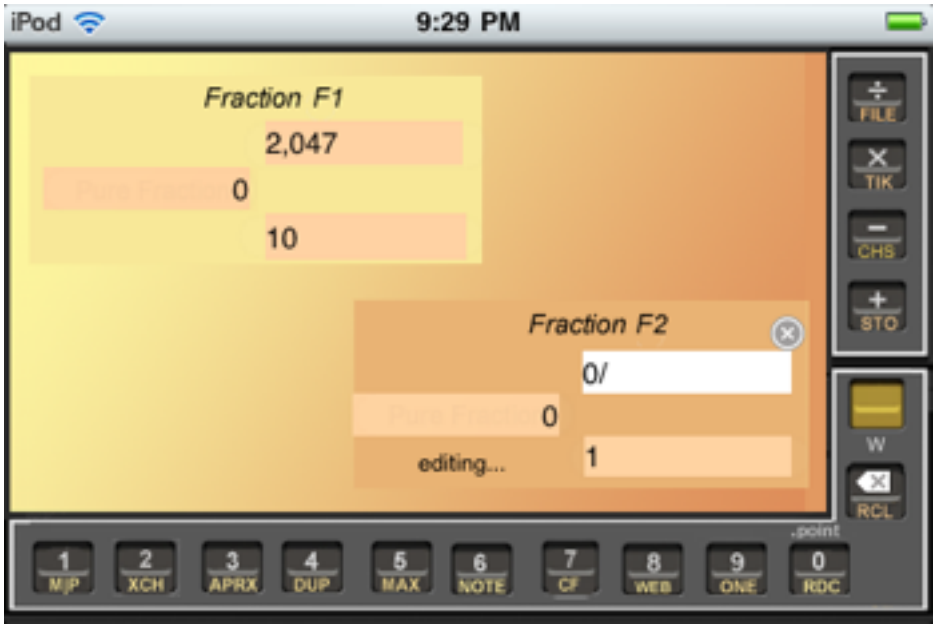

Figure 1a. *Frrraction setup for finding the reciprocal of* 10 *(mod* 2047*)*

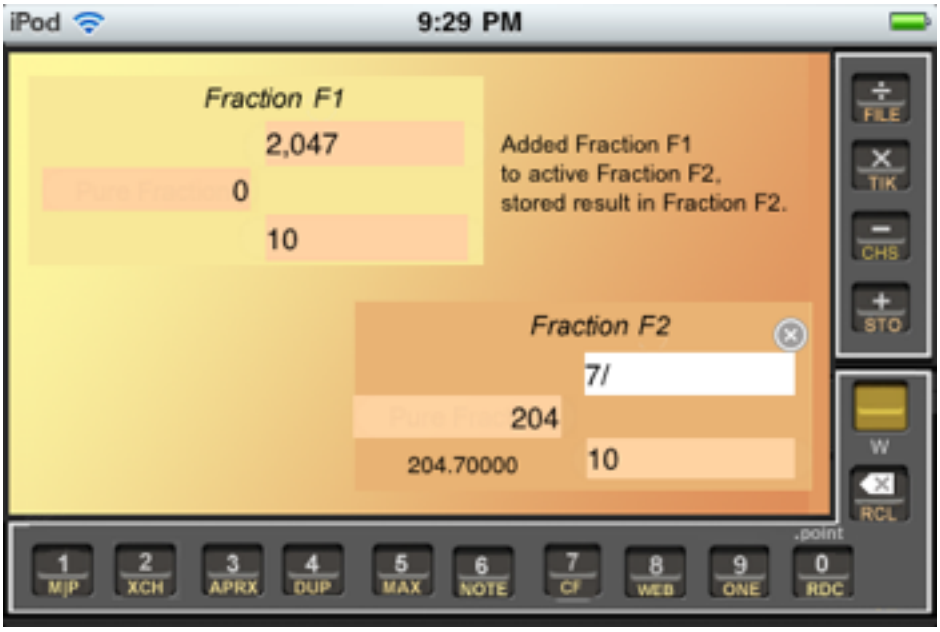

Figure 1b. *After the first addition in the +++ Method for* 1/10 *(mod* 2047*)*

Tapping +,+,+... repeatedly generates the following in *F2*:

204+7/10	409+2/5	614+1/10	818+4/5	1023+1/2	1228+1/5	1432+9/10

As shown in Fig.1c, the 7th result shows that $1433 \equiv 10^{-1} \pmod{2047}$.

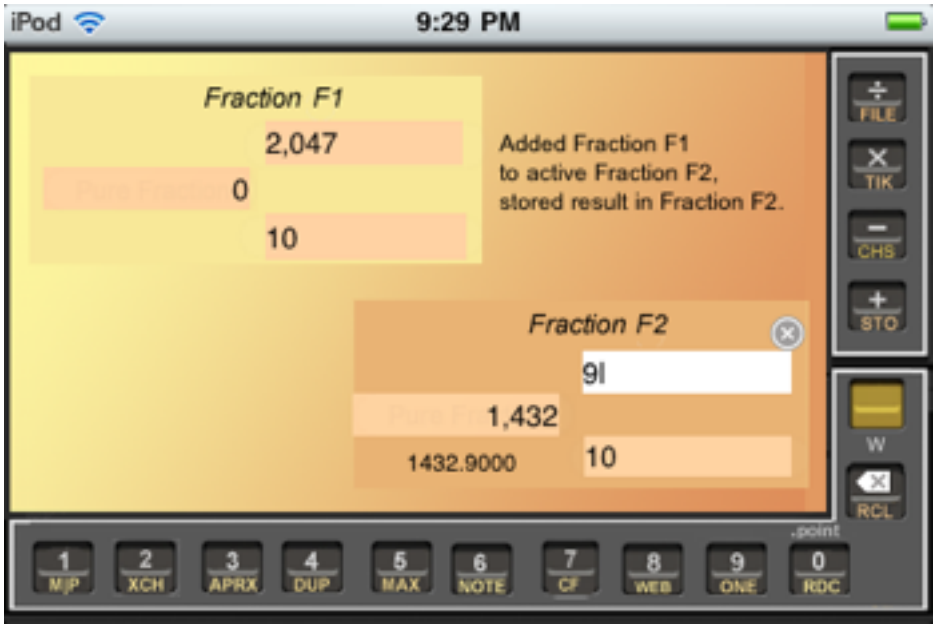

Figure 1c. *After the 7th addition in the +++ Method for* 1/10 *(mod* 2047*)*

To confirm the result, put 0+10/2047 into F1, 1433+0/1 into F2, as in Fig.1d:

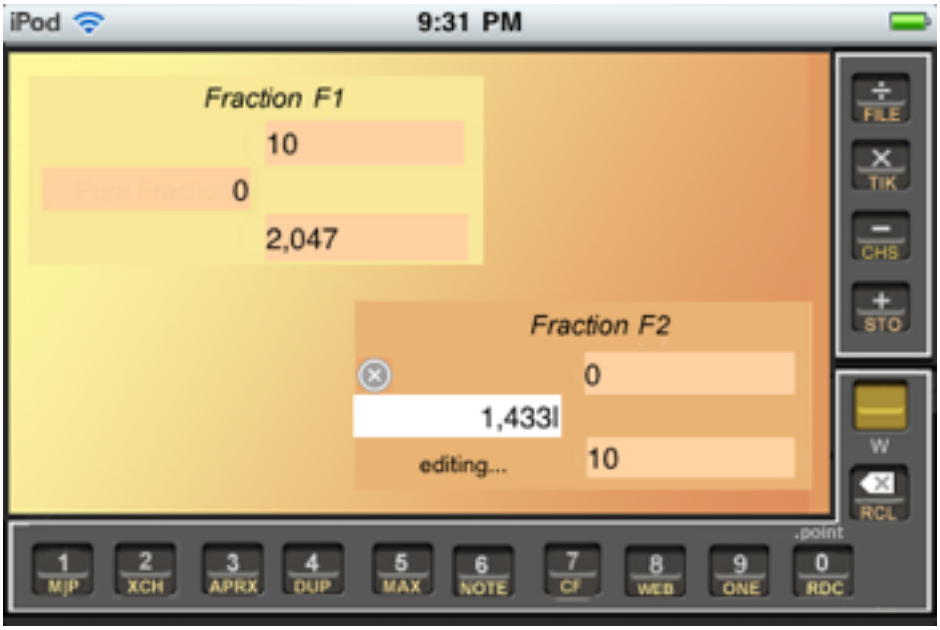

Figure 1d. *Setup for verification of* 1/10 *(mod* 2047*)*

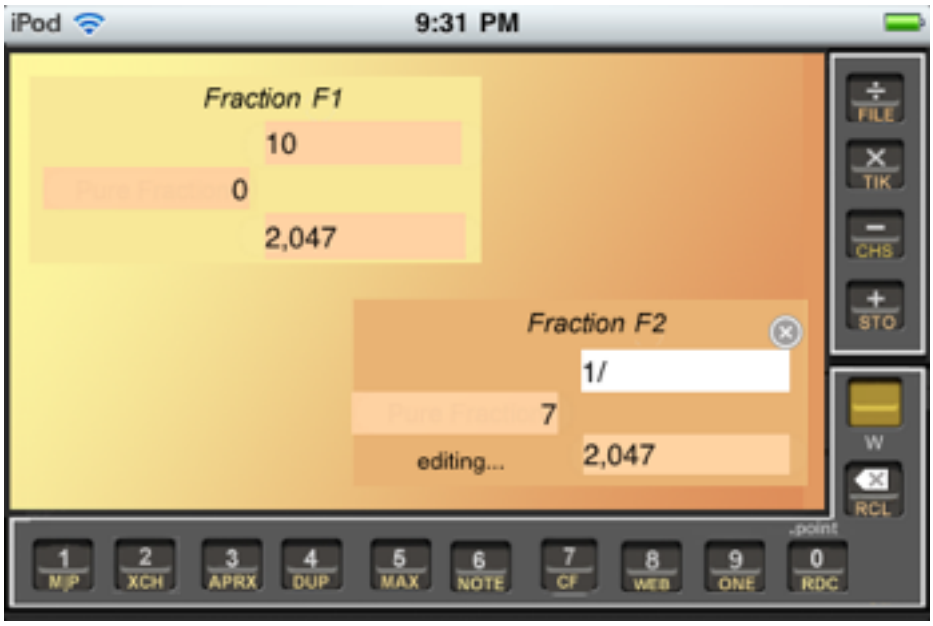

Figure 1e. *Result of verification of* 1/10 (*mod* 2047)

Then multiply. As shown in Fig.1e, the product is 7+1/2047. The 1 in *F2num* shows that $10 \cdot 1433 \equiv 1 \pmod{2047}$, confirming that $1433 \equiv 1/10 \pmod{2047}$. Pretty slick, yes? ▣

Why the +++ Method works. The following statements are all true if and only if $d \cdot c \equiv 1 \pmod{m}$:

(i) The remainder after dividing $d \cdot c$ by m is 1, *i.e.*

(ii) $d \cdot c / m = e + 1/m$ for some integer e

(iii) $d = e \cdot m / c + 1/c$

(iv) $e \cdot m/c = d - 1/c$ [Interesting: e and $-m$ are mod-c reciprocals]

(v) $e \cdot m/c = d - 1 + 1 - 1/c$

(vi) $e \cdot m/c = (d - 1) + (c - 1)/c$

So, if we add m/c to 0 getting $n_1 + k_1/c$; then add m/c to that result getting $n_2 + k_2/c$; and continue adding m/c to the previous result getting $n_i + k_i/c$ for $i = 1, 2, \ldots$ until we reach a result of the form $p + (c-1)/c$. At that point, p is clearly the d

− 1 of Eq.(vi) so the desired d is $p + 1$. And what we really did was multiply m/c by e.▣

Side note. If you already know d, Eq.(ii) offers a way to predict the number of additions the +++ Method would require: Besides being the multiplier of m/c in Eq.(vi), ***e* is the integer part of the fraction *d* ·*c* /*m* —**which is the arithmetic **utilized to verify** a proposed mod-m reciprocal of c.

The +++ Method is, more often than not, cumbersome for large moduli (modulae? modulusses?). If you applied it to find 1000^{-1} (mod 2047) the method would require 617 additions in the +++ sequence! Few number-lovers would have the patience to tap the + button twice a second for five minutes in order to find that out! (It's easy to simulate the 617 additions: Just multiply 2047/1000 by 617 to get the same result: 1262 + 999/1000.) [In Version 1.1 of this article, the result was incorrectly reported as 1432+999/1000, a mis-carryover from the previous example?]

Reminder: Every finite continued fraction has a twin. For any fraction a/b there are two distinct continued fraction expansions: The traditional shallower one, $[q_0; q_1, q_2,..., q_{n-1}, q_n]$ with bottom quotient $q_n > 1$. In those terms the other twin, deeper by one, is $[q_0; q_1, q_2,..., q_{n-1}, q_n - 1, 1]$. **One of the twins has even depth, the other odd.** The twin that ends in 1 is traditionally deprecated in favor of the shallower one, with the thought that the two were totally equivalent. As you see below, this was shortsighted: They certainly are *not* totally equivalent. Converting either to the other is easy.

The RCF (Reversed Continued Fraction) inversion method

This truly is a bit of Continued Fraction magic: To find the numeric value $d \equiv 1/c \pmod{m}$, start by constructing the CF

$m/c = [q_0; q_1, q_2,..., q_{n-1}, q_n]$ whose depth n is even*. Reverse the order of its quotients to obtain RCF = $[q_n; q_{n-1},..., q_2, q_1, q_0]$. The stacked fraction for RCF is m/d, whose denominator is the desired reciprocal of c.

*If you use the twin whose n is odd, it produces the *negative* of the reciprocal of c.

Example. Find the reciprocal of 2000 (mod 2039).
Solution. The CF of 2039/2000 with even depth is [1;51,3,1,1,4,1]. Reversing it yields RCF = [1;4,1,1,3,51,1] whose stacked form is 2039/1673, so the desired mod-2039 reciprocal of 2000 is 1673. ■

Another bit of Continued Fraction magic:

The SCF (Shortened Continued Fraction) inversion method

To find the numeric value $d \equiv 1/c \pmod m$, start by constructing the CF $m/c = [q_0; q_1, q_2,..., q_{n-1}, q_n]$ whose depth n is even*. Shorten it by omitting the bottom quotient q_n to obtain

$$[q_0; q_1, q_2,..., q_{n-1}] = d/e$$

whose numerator is the desired mod-m reciprocal of c. That's all there is to it.

*If you use the twin, whose n is odd, it produces the *negative* of the reciprocal of c.

Isn't that amazing?! Could it get any easier? Well, yes, it can get a little bit easier, if you're doing the calculations using the iPhone Frrraction app: See the ISCF method below.

Example. Use the SCF Method to find and verify the reciprocal of 2000 (mod 2039).
Solution. The CF of 2039/2000 with even depth is [1;51,3,1,1,4,1]. Thus the desired reciprocal is the numerator of d/e = [1;51,3,1,1,4] = 1673/1641, *i.e.* d = 1673.■

Verify this by multiplying 1673 by 2000/2039 to get 1641+1/2039. The remainder 1 verifies that 1673x2000 ≡ 1 (mod 2039).

(That neglected SCF denominator e = 1641 also has significant meaning: it happens to be the number of additions the +++ Method would take in order to generate d = 1673 as the desired reciprocal. Try it: That multiple of 2039/2000 equals 1672+1999/2000, which is the +++ Method's proof that 1672+1 is the mod-2039 reciprocal of 2000.)■

The ISCF (Inverted Shortened Continued Fraction) method

The ISCF method is similar to the SCF method, but starts by constructing the CF for c/m not for m/c. (That's why the method is called "inverted.")

1. Produce the CF c/m = $[0; q_1, q_2,..., q_{n-1}, q_n]$. (Note that the integer part is 0 because c is always less than m. That forces us to choose the CF-twin for which n is an *odd* integer.
2. Shorten it by removing q_n to get: $[0; q_1, q_2,..., q_{n-1}] = e/d$
3. Finally, shift all the remaining quotients to the left by one position to get:

$$[q_1; q_2,..., q_{n-1}] = d/e$$

whose numerator d is the desired inverse of c (mod m).

Note: Step 2 (shorten) and Step 3 (shift) are easily combined into a single step.

Example. Use the ISCF Method in *Frrraction* to find and verify the reciprocal of 2000 (mod 2039).
Solution. The CF of 2000/2039 (notice the inverted data, relative to doing this same example using the SCF method) with odd depth is [0;1,51,3,1,1,4,1]. You can find this out using *Frrraction* this way: Put 0+2000/2039 into *F1*, put an empty square bracket [] into Notes; Fig.2a shows the setup:

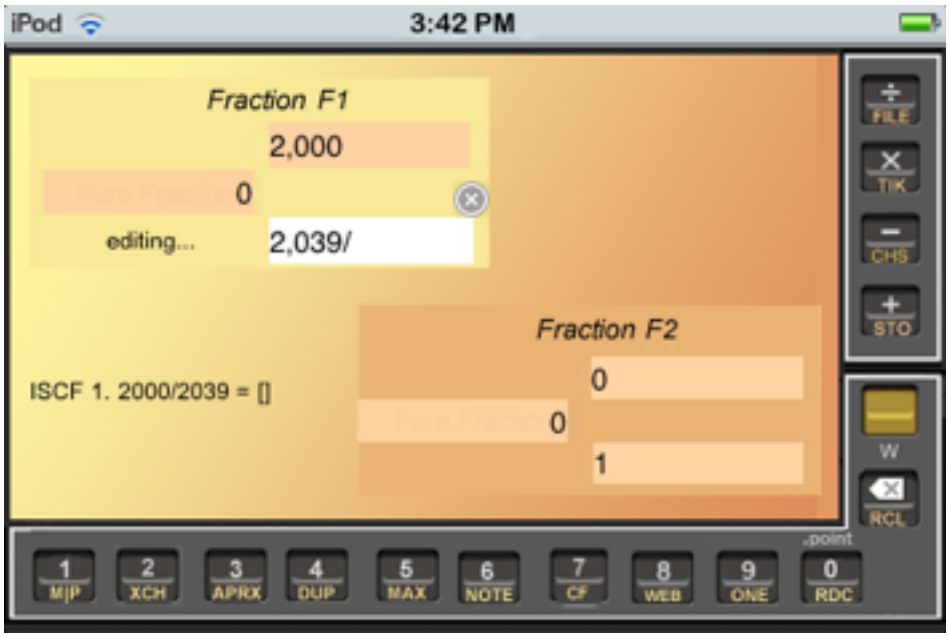

Figure 2a. *Setup for Step 1*, ISCF example in *Frrraction*

Then yTap the CF command. This computes the CF [0;1,51,3,1,1,4,1] into Notes, as shown in Fig. 2b (which also

shows the setup for Step 2):

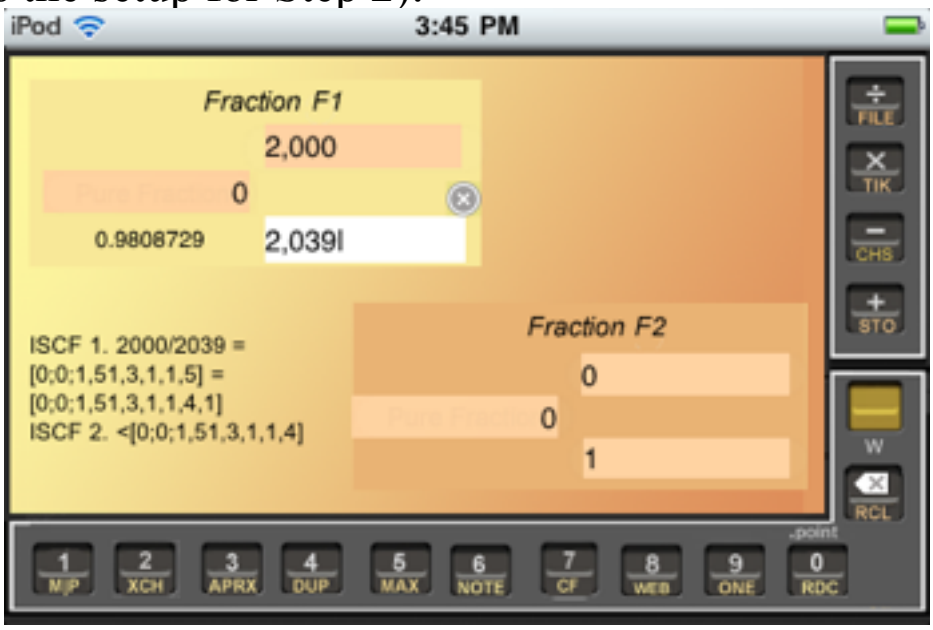

Figure 2b. Result of Step 1, setup for Step 2, ISCF example

For Step 2, go into Notes, copy and paste the CF that Step 1 produced, edit the copy to produce the shortened bracket command <[0;1,51,3,1,1,4]—shown in Fig.2b—then make *F2* be active and use the CF command again; this computes 0+1,641/1,673 into *F2*.

Start Step 3 by going back into Notes and edit it to produce the shifted bracket command <[1;51,3,1,1,4]. Screen shot Fig. 2c shows the edit being completed:

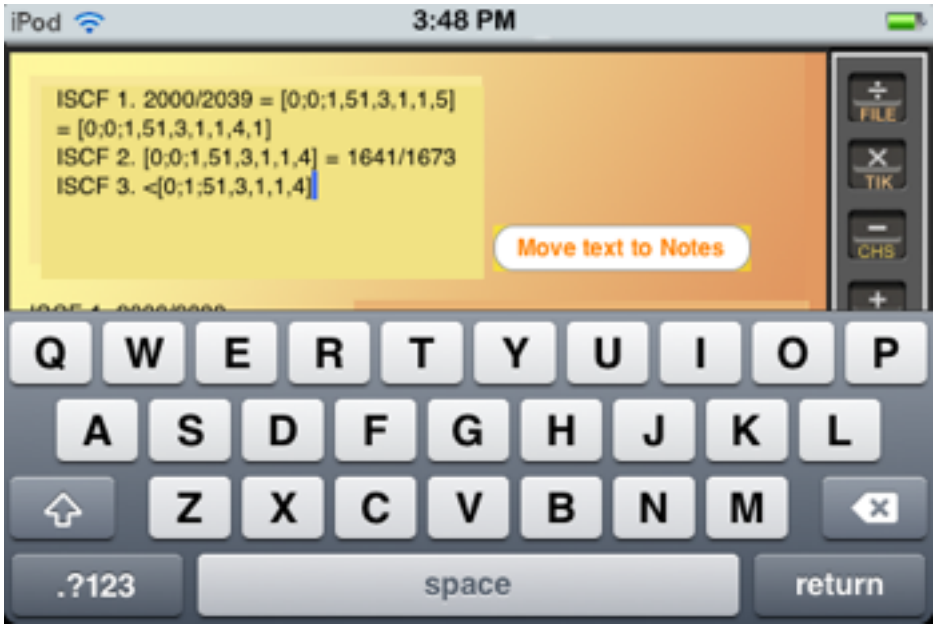

Figure 2c. *Step 3 setup*, ISCF example

Finally, move the edited text into Notes, then use the CF command to compute 0+1,673/1,641 = *d*/*e* into F2, showing that *d* is 1,673. the result is shown in Fig.2d:

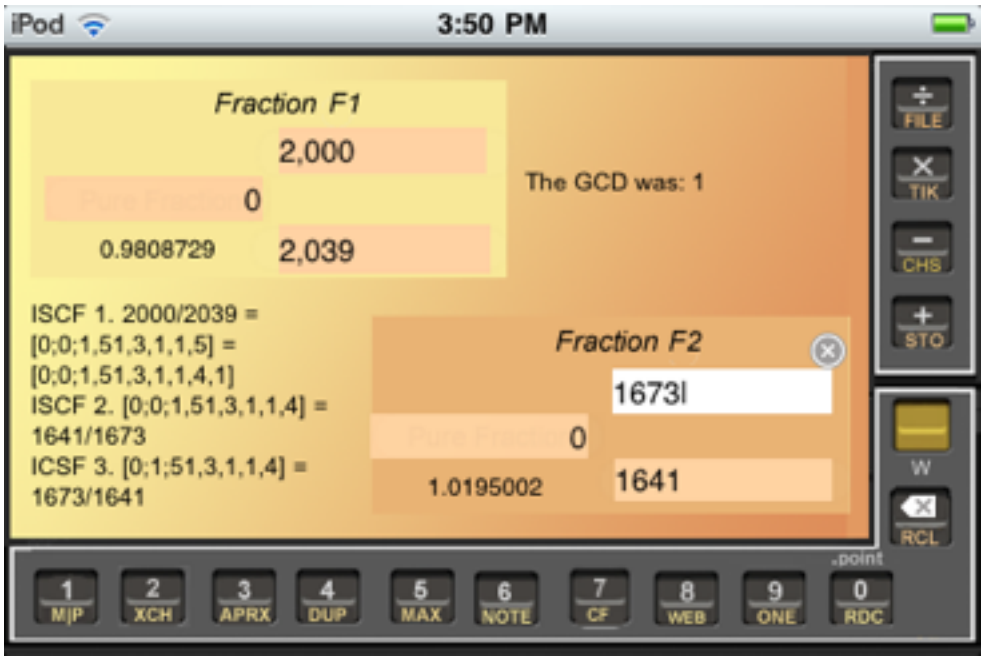

Figure 2d. *Step 3 completed*, ISCF example

Verify the result: Here's where the ISCF method is better than SCF for *Frrraction*: To verify, just replace *e* (1,641) by 1 in *F2den*, then multiply. The setup is shown in Fig.2e:

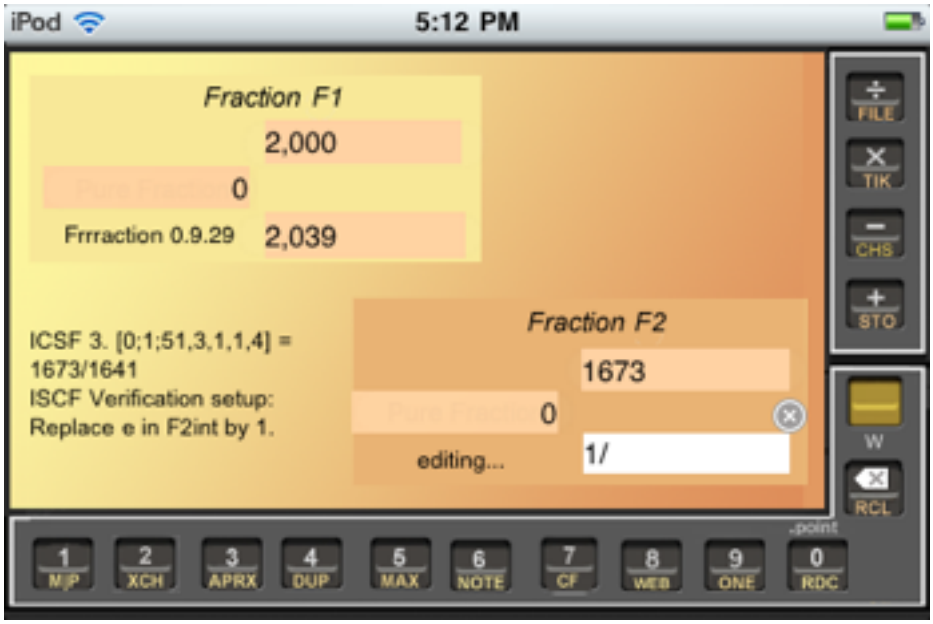

Figure 1e. *Verification setup,* ISCF example

(The non-inverted SCF method would require an awkward replacement of *m*/*c* by *c*/*m* in *F1*, and a similarly awkward replacement of *e*/*d* by *d*/1 in *F2*.) Fig.1f shows the verification result in this example:

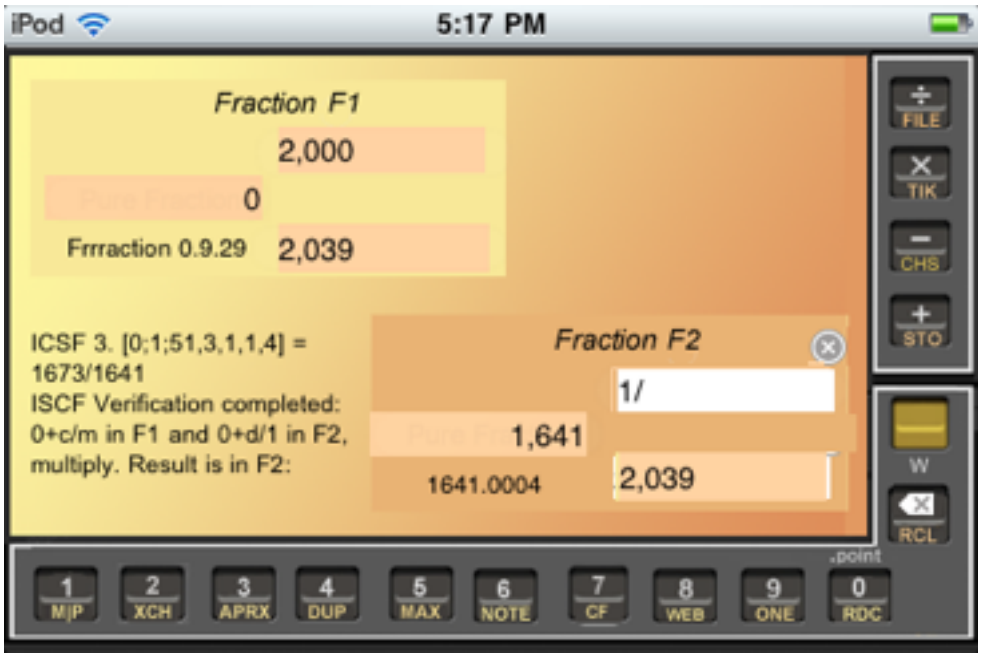

Figure 1f. *Verification completed*, ISCF example

The remainder $r = 1$, in *F2num*, verifies that $1673 \cdot 2000 \equiv 1 \pmod{2039}$. ■

Examples this size are simple enough to perform entirely by hand, but *Frrraction* makes them a lot less tedious.

Conclusion

To the best of my knowledge, the RCF, SCF, and ISCF methods are new ways to compute modular reciprocals. Of course there's a lot of literature that I haven't yet read, so there's still a good chance that they aren't really new. If you spot some relevant Number Theory, please let me know, jresh@frrraction.com.