# Chapter 6
# Linear Congruences

Suppose a certain airline is consistently 25 hours late in departure and arrival (this has happened, but no names will be mentioned) while another one, flying the same route, is only 2 to 3 hours late. If you were in a hurry, which airline would you fly – food, lack of leg room and all else being equal? Obviously, being 25 hours late is as good (or bad) as being only 1 hour late. In other words, in a *daily recurring* event an extra day, or even several, makes no difference. The mathematics that deals with this kind of situation is called *modular arithmetic*, because only remainders *modulo* a given integer matter.

Another application of modular arithmetic occurs in wave interference phenomena such as ripples on a lake or patterns of light and dark in a hologram. In all these cases a path difference of, say, half a wavelength is indistinguishable from a path difference of one and a half or two and a half, etc., wavelengths.

And of course, there are many applications in mathematics proper. For example, few people care what $n^{560}$ is, but the remainder of $n^{560}$ when divided by 561 for all $n$ coprime to 561 is a question of some actuality. (As it happens, all these remainders are 1 – a terrible thing to happen, as we shall see.) But first we have to know the ground rules of modular arithmetic, so that $n^{560}$ for, say, $n = 500$, a 1512-digit monster of a number, cannot frighten us.

## 6.1 Residues

When $c$ divided by $m$ leaves the remainder $b$ (not necessarily positive) we write (following Gauss)

$$c \equiv b \pmod{m}; \tag{6.1}$$

read: $c$ is congruent $b$ modulo $m$.

More generally, we define the above congruence as meaning

$$m | (c - b); \tag{6.2}$$

read: $m$ divides $c$ minus $b$ (without remainder).

*Example:* $16 \equiv -2 (\bmod 9)$ implies $9 | (16+2)$.

Together with $c$, all

$$b = mq + c; \qquad q = \text{integer} \tag{6.3}$$

belong to the same *residue class* modulo $m$ [6.1].

A *complete residue system* modulo $m$ consists of $m$ integers, one representative each from each residue class. The most common residue systems are the *least nonnegative* residue system modulo $m$, consisting of the integers $0, 1, 2, \ldots, m-1$, and the *least absolute* residue system, consisting of the integers $0, \pm 1, \pm 2, \ldots, \pm(m-1)/2$ (for odd $m$).

For many purposes, one calculates with the congruence sign for a given modulus as if it were an equal sign.

Addition:

$$\begin{array}{ll} \textit{Example:} & 13 \equiv 4 \ (\bmod 9) \\ & \underline{16 \equiv -2 \ (\bmod 9)} \\ & 29 \equiv 2 \ (\bmod 9) \qquad \text{Check!} \end{array} \tag{6.4}$$

Multiplication of the two upper congruences results in

$$208 = -8 \ (\bmod 9) \qquad \text{Check!} \tag{6.5}$$

The congruence

$$c \equiv 5 \ (\bmod m) \tag{6.6}$$

can be "cancelled" by the GCD of $c$, $b$ and $m$. With $(c, b, m) = d$, we may write

$$\frac{c}{d} \equiv \frac{b}{d} \left( \bmod \frac{m}{d} \right). \tag{6.7}$$

*Example:* $28 \equiv 4 \ (\bmod 6)$ can be converted to $14 \equiv 2 \ (\bmod 3)$.

Another useful rule is the following. If

$$mc \equiv mb \ (\bmod n) \tag{6.8}$$

and the GCD $(m, n) = d$, then

$$c \equiv b \ \left( \bmod \frac{n}{d} \right). \tag{6.9}$$

*Example:* $28 \equiv 4 \ (\bmod 6)$ implies $7 \equiv 1 \ (\bmod 3)$.

Among the many useful applications of linear congruences is the ancient error-detecting algorithm sailing under the name of "casting out 9's" [6.2]. If we add two decimal numbers column by column, then if in any one column the sum exceeds

9, we reduce the result modulo 10 and add 1 (or 2 or 3, etc.) to the next column. Thus, in terms of the sum of the decimal digits, we have added 1 (or 2 or 3, etc.) and subtracted 10 (or 20 or 30, etc.). We have therefore changed the sum of the digits by a multiple of 9; in other words, the sum of the digits has not changed modulo 9.

*Example:*

$$\begin{array}{rll} 86 & \text{sum of digits} = 14, & \text{sum of sum of digits} = 5 \\ +\ \ 57 & \text{sum of digits} = 12, & \text{sum of sum of digits} = 3 \\ \hline = 143 & \text{sum of digits} = 8, & \end{array}$$

Check: $14 + 12 = 26 \equiv 8 (\operatorname{mod} 9)$. Check! Of course in the check we can apply the same rule and consider sums of sums of digits: Check: $5 + 3 = 8 \equiv 8 (\operatorname{mod} 9)$. Check!

The same rule also holds for multiplication.

*Example:*

$$\begin{array}{rl} 15 & \text{sum of digits} = 6, \\ \times\ \ 17 & \text{sum of digits} = 8, \\ \hline =\ \ 255 & \text{sum of digits} = 12, \text{ sum of sum of digits} = 3\,. \end{array}$$

Check: $6 \times 8 = 48$, sum of digits $= 12$, sum of sum digits $= 3$. Check!

The reason why sums of digits when multiplied give the same result modulo 9 as the numbers themselves is that, trivially, any power of 10 is congruent 1 modulo 9:

$$10 \equiv 1 \pmod 9 \quad \text{and therefore} \quad 10^k \equiv 1 \pmod 9 \quad \text{for any} \quad k \geq 0.$$

The only problem with this ancient error-detecting code is that it can fail to signal an error. In fact, for random errors, about 10 % of the errors go undetected. Fortunately, though, the casting-out-9's is not restricted to the decimal system; it works for any base $b$, casting out $(b - 1)$'s. Specifically, it also works for a much neglected (and very simple) number system: the base-100 or "hectic" system. One of the advantages of the hectic system is that it needs no new digits. For example, the year of Gauss's birth in hectic notation looks like this:

$$17\ 77, \quad \text{sum of digits} = 94,$$

and that of his death

$$18\ 55, \quad \text{sum of digits} = 73,$$

and the difference of these two hectic numbers (his age when he died) is

$$00\ 78, \quad \text{sum of digits} = 78.$$

Check: $73 - 94 = -21 \equiv 78 \pmod{99}$. Check!

This example is perhaps too simple, but with the hectic error-detecting algorithm the undetected error rate has dropped to about 1 %.

A simple rule exists also for divisibility by 11, used in the International Standard Book Numbers (ISBN). It follows from $10 \equiv -1 \bmod 11$ and $100 \equiv 1 \bmod 11$ that divisibility by 11 of an integer and its digital sum, taken with *alternating* signs, are equivalent. Thus 517, for example, is immediately seen to be divisible by 11 because $5 - 1 + 7 = 11$. If the result of this operation is itself a large number, the operation can of course be repeated until manageable numbers, like 0 or 11, are reached.

Divisibility checks also exist for 7, 13, 17, 19 etc., but they are not as efficient. Thus, to check $n$ for a factor 7, one writes

$$n = 10a + b$$

and

$$m = a - 2b\,.$$

Now if $m \equiv 0 \bmod 7$, it follows that $a \equiv 2b$ and $10a \equiv 20b \equiv 6b$. Thus $n \equiv 6b + b \equiv 7b \equiv 0 \bmod 7$. Of course the rule can be iterated.

The same approach tells us that, for divisibility by 13, we should check $a + 4b$. Thus, for example, for $n = 91$, we have $9 + 4 = 13$, implying that 91 is divisible by 13.

Similarly, for $n = 17$, we should look at $a - 5b$ and for $n = 19$, the test is $a + 2b$.

## 6.2 Some Simple Fields

Complete residue systems modulo a prime form a *field*, i. e., a set of numbers for which addition, subtraction, multiplication and division (except by 0) are defined and for which the usual commutative, associative and distributive laws apply [6.3].

For the least nonnegative residue system modulo 2, consisting of 0 and 1 (perhaps the most important one in this computer age) we have the addition table:

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array} \tag{6.10}$$

which is isomorphic both to the logical operation "exclusive or" and to multiplication of signs (if we identify 0 with + and 1 with −).

The multiplication table for 0 and 1:

$$\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array} \tag{6.11}$$

is isomorphic to the logical "and" and the set-theoretic "intersection".

Multiplication for a complete residue system modulo a composite number has no inverse for some of its members, as can be seen from the multiplication table modulo 4:

|  | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 |
| 2 | 0 | 2 | 0 | 2 |
| 3 | 0 | 3 | 2 | 1 |

(6.12)

Thus, there is no number which when multiplied by 2 gives 1, i. e., 2 has no inverse. Also, division by 2 is not unique: 2 divided by 2 could be either 1 or 3.

This grave defect is rectified by *prime* residue systems which consist only of those residue classes that are coprime to the modulus. Thus, the least nonnegative prime residue system modulo 10 consists of the integers 1, 3, 7 and 9, and their multiplication table is well behaved:

|  | 1 | 3 | 7 | 9 |
|---|---|---|---|---|
| 1 | 1 | 3 | 7 | 9 |
| 3 | 3 | 9 | 1 | 7 |
| 7 | 7 | 1 | 9 | 3 |
| 9 | 9 | 7 | 3 | 1 |

(6.13)

If $f$ is a polynomial with integer coefficients, then $a \equiv b \pmod{m}$ implies

$$f(a) \equiv f(b) \pmod{m}, \tag{6.14}$$

For $(m,n) = 1$ if $x$ and $y$ run through complete residue systems modulo $n$ and $m$, respectively, then $mx + ny$ runs through a complete residue system modulo $mn$.

*Example:* $n = 5$, $m = 3$:

|  | $x = 0$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $y = 0$ | 0 | 3 | 6 | 9 | 12 |
| 1 | 5 | 8 | 11 | 14 | 2 |
| 2 | 10 | 13 | 1 | 4 | 7 |

We will encounter this kind of decomposition of a residue class modulo a product of coprimes again when we discuss simultaneous congruences and the Chinese Remainder Theorem with its numerous applications in Chap. 17.

## 6.3 Powers and Congruences

What is $2^{340} \pmod{341}$? Obviously, the ancient Chinese did not know or they would not have formulated their primality test (Sect. 2.3). With the aid of the congruence notation we may rewrite the Chinese test as

$$2^n \equiv 2 \pmod{n}, \tag{6.15}$$

if *and only if n* is prime. Here the first "if" is all right (see Fermat's Theorem, Chap. 8) but the "and only if" is wrong, as we shall presently demonstrate with the composite $n = 341 = 11 \cdot 31$. For odd $n$ we may write, because of (6.9),

$$2^{n-1} \equiv 1 \pmod{n}.$$

Of course, it is foolish actually to calculate $2^{340}$, a 103-digit number, if we are interested only in the remainder modulo 341. Instead we will decompose 340 into $10 \cdot 34$ and first raise 2 to the 10th power, giving 1024. Dividing 1024 by 341 leaves the remainder 1:

$$2^{10} \equiv 1 \pmod{341}. \tag{6.16}$$

Now raising the result to the 34th power is easy because $1^{34} = 1$. Thus,

$$2^{340} \equiv 1 \pmod{341}, \quad \text{or} \tag{6.17}$$

$$341 \mid (2^{341} - 2), \tag{6.18}$$

in spite of the fact that 341 is composite. Woe to the Chinese and three cheers for the congruence notation! We could even have done this in our heads, without recourse to pencil and paper, and thereby demolished a false "theorem" which had stood undisputed for so many centuries.

Composite numbers which masquerade as primes vis-a-vis Fermat's theorem are called *pseudoprimes*. 341 is actually the smallest pseudoprime to the base 2. In a certain sense, pseudoprimes have become almost as important as actual primes in modern digital encryption. We will hear more about pseudoprimes and even *absolute* and *strong pseudoprimes* in Chap. 20.

Of course, in calculating $2^{340} \pmod{341}$ we were lucky, because $2^{10}$ already gave the remainder 1. In general we will not be so lucky, and we need a universal algorithm that will see us through any base $b$ or exponent $n$.

More formally, to calculate $b^n \pmod{m}$, we first find the binary decomposition of $n$:

$$n = \sum_{k=0}^{\lfloor \log_2 n \rfloor} n_k 2^k \quad \text{with} \quad n_k = 0 \text{ or } 1, \tag{6.19}$$

where $\lfloor\ \rfloor$ is the Gauss bracket, or "floor function", signifying the integer part.

The binary expansion coefficients $n_k$ can be found by any of a variety of "analog"-to-digital conversion algorithms.

Omitting all terms with $n_k = 0$ in the sum, we may write

$$n = \sum_{m=1}^{M} 2^{c_m}. \tag{6.20}$$

In other words we write, for example, $6 = 2+4$ or $13 = 1+4+8$, etc.

Using this decomposition into a sum of powers of 2, we write $b^n$ as follows:

$$b^n = \prod_{m=1}^{M} b^{c_m} = \underbrace{(\dots(b^2)\dots)^2}_{c_1 \text{ squarings}} \cdots \underbrace{(\dots(b^2)\dots)^2}_{c_M \text{ squarings}}. \tag{6.21}$$

In words: we calculate $b^n$ by squaring $b$ $c_1$ times in succession. Then we square $b$ $c_2$ times. (Of course, we make use of the previous result, i. e., we need square only $c_2 - c_1$ more times, etc.) Then we multiply the results of all these squarings together to obtain $b^n$.

Apart from the gain in computational efficiency (if $n$ is a power of 2, then only about $\log_2 n$ squarings are required, instead of $n$ multiplications), the main raison d'être for the repeated squaring algorithm is that if we want the result modulo $m$, then after each squaring we can reduce the intermediate result modulo $m$ without running the risk of calculator "overflow" (as long as $m^2$ is smaller than the largest number the machine can handle).

Here is another rule that makes working with powers and congruences easier:

$$(x+y)^n \equiv x^n + y^n \pmod{n}, \tag{6.22}$$

$n$ prime (or absolute pseudoprime), i. e., of the $n+1$ terms obtained upon expanding $(x+y)^n$ binomially only two remain, because all others, being multiplied by $\binom{n}{k}$, with $k \neq 0$ or $n$, are divisible by $n$ and therefore do not contribute to the end result modulo $n$.

Incidentally, the condition $k \neq 0$ or $n$ can be expressed with the following more widely applicable and succinct notation:

$$k \not\equiv 0 \pmod{n}.$$

Read: $k$ is *not* congruent to zero modulo $n$.