A Pari/GP Tutorial

by Robert B. Ash, Jan. 2007

The Pari/GP home page is located at http://pari.math.u-bordeaux.fr/

1. Pari Types

There are 15 types of elements (numbers, polynomials, matrices etc.) recognized by Pari/GP (henceforth abbreviated GP). They are listed below, along with instructions on how to create an element of the given type. After typing an instruction, hit return to execute. A semicolon at the end of an instruction suppresses the output. To activate the program, type gp, and to quit, type quit or \q

1. Integers

Type the relevant integer, with a plus or minus sign if desired.

2. Rational Numbers

Type a rational number as a fraction a/b, with a and b integers.

3. Real Numbers

Type the number with a decimal point to make sure that it is a real number rather than an integer or a rational number. Alternatively, use scientific notation, for example, 8.46E17 The default precision is 28 significant digits. To change the precision, for example to 32 digits, type \p 32.

4. Complex Numbers

Type a complex number as $a + b * I$ or $a + I * b$, where a and b are real numbers, rational numbers or integers.

5. Integers Modulo n

To enter $n \bmod m$, type $\mathrm{Mod}(n, m)$.

6. Polynomials

Type a polynomial in the usual way, but don't forget that $*$ must be used for multiplication. For example, type $x\hat{\ }3 + 3*x\hat{\ }2\ \text{-}1/7*x\ +8$.

7. Rational Functions

Type a ratio P/Q of polynomials P and Q.

8. Polynomials Modulo a Polynomial P

To enter $Q \bmod P$, type $\mathrm{Mod}(Q, P)$.

9. Vectors

Type a row vector as, for example, [1,2,3], and a column vector as [1,2,3]˜. The i-th component of a vector v can be recovered via $v[i]$.

10. Matrices

Separate components by commas, lines by semicolons. Thus $[1, 2, 3; 4, 5, 6]$ gives the matrix

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

To recover the ij element of a matrix A, type $A[i, j]$. The i-th row is $A[i,]$ and the j-th column is $A[, j]$.

11. Lists

L=listcreate(4) creates an empty list of (maximum) length 4. Then listput($L, a, 1$) sets the first entry of L to a. Similarly, after listput($L, b, 2$), listput($L, c, 3$), and listput($L, d, 4$), we have $L = [a, b, c, d]$. To recover the third entry, type $L[3]$, just as with a row vector.

12. Strings

A string is any sequence of symbols. It is identified by enclosing it in quotation marks. For example, $s =$ "The following sequence is exact".

13. Power Series and Laurent Series with Finitely Many Negative Powers

We can truncate the series at any point. For example,

$$\frac{4 * z^3 + 3 * z^2 + 2 * z + 1}{z^2} + O(z^2)$$

will produce the output

$$z^{-2} + 2 * z^{-1} + 3 + 4 * z + O(z^2)$$

GP will stop computing a Taylor or Laurent series at $O(x^{16})$. To change the series precision to something else, type, for example, \ps 19

14. p-adic Numbers

The syntax is similar to #13. For example, $905/7 + O(7^3)$ will produce output

$$2 * 7^{-1} + 3 + 4 * 7 + 2 * 7^2 + O(7^3)$$

We can find the expansion of any integer in any base. For example, to find the base 6 expansion of 86, type $86 + O(6^3)$ (or $86 + O(6^r), r > 3$), to get $2 + 2 * 6 + 2 * 6^2 + O(6^3)$. Replacing $O(6^3)$ by $O(6^2)$ will truncate the expansion.

15. Quadratic Numbers (Elements of a Quadratic Number Field)

If the number field is $\mathbb{Q}(\sqrt{m})$, then there is a canonical integral basis $(1, w)$, as follows. If the discriminant d is congruent to 0 mod 4, then $w = \sqrt{m} = \sqrt{d}/2$ and $d = 4m$. If d is congruent to 1 mod 4, then $w = (1 + \sqrt{m})/2 = (1 + \sqrt{d})/2$ and $d = m$. Therefore $m \equiv 1$ mod 4, so the first case corresponds to $m \not\equiv 1$ mod 4. This allows a quadratic number field to be identified by its discriminant d, and this is what GP does via the instruction $w =$ quadgen(d). You might as well use w in this instruction, because GP will change anything you write to w. The general element of the quadratic number field is written as $x + y * w$ with x and y integers.

Some general advice

It is very convenient to name objects. If you are working with the polynomial $x^2 + 3 * x + 4$, type f=x^2+3*x+4, and then you can use f in any computation.

Creating functions is easy. Type $f(x) = x^2 + 3 * x + 4$, and then $f(2)$ will return 4+6+4=14. Similarly for functions of several variables.

There are some reserved symbols that cannot be used as variable names. They include I (the square root of -1), Pi (3.14159...), and O (big oh). Also, if you create a function called g, you cannot use g as a variable name.

Several expressions can be combined on a single line by separating them with semi-colons.

Typing # activates a toggle switch that turns the timer on and off. Typing ## prints the time taken by the latest computation.

To redefine a variable s, first type **kill(s)**. To kill a function f, kill all of its variables and then type kill (f).

Outputs are numbered %1,%2,... It is quite legal in a computation to use %n as an abbreviation for the n-th output. Also, % with no number attached refers to the immediately preceding output.

To display an output, use the **print** command. For example, if $x = 3$, the command $\text{print}(2 * x + 1)$ will display the answer 7. If you want to call the output y, type $\text{print}("y = ", 2 * x + 1)$. Then $y = 7$ will be displayed.

The Boolean comparison operators are

<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	equal
!=	unequal

2. Transcendental and Other Nonrational Functions

Here is a list of many commonly used functions.

abs(x) is the usual absolute value on $\mathbb{R}$ or $\mathbb{C}$

acos(x) is arc cos x

acosh(x) is arc cosh x

agm(x,y) is the arithmetic-geometric mean of x and y. Start with $a_0 = x, b_0 = y$ and define a_n and b_n recursively by $a_{n+1} = (a_n + b_n)/2$ and $b_{n+1} = \sqrt{a_n b_n}$. Then a_n and b_n will converge to a common limit agm(x, y).

arg(x) is the argument of the complex number x, taken in $(-\pi, \pi]$.

asin(x) is arc sin x

asinh(x) is arc sinh x

atan(x) is arc tan x

atanh(x) is arc tanh x

bernfrac(n) gives the n-th Bernoulli number B_n (as a rational number)

bernvec(m) gives a vector whose components are the Bernoulli numbers $B_0, B_2, B_4, \dots, B_{2n}$

ceil(x) is the ceiling of x, the least integer greater than or equal to x.

conj(x) is the complex conjugate of x.

cos(x) is the cosine of x.

cosh(x) is the hyperbolic cosine of x.

cotan(x) is the cotangent of x

dilog(x) is the special case $m = 2$ of **polylog(m,x)**; see below.

eint1(x)   is $\int_x^\infty (e^{-t}/t)\, dt$

eint1(x,n)   is a vector whose n components are $\text{eint1}(mx), m = 1, \ldots, n$

erfc(x)   is the complementary error function $(2/\sqrt{\pi}) \int_x^\infty e^{-t^2}\, dt$

exp(x)   is the exponential function

floor(x)   is the floor of x, the greatest integer less than or equal to x

frac(x)   is the fractional part of x, that is, $x-\text{floor}(x)$

gamma(x)   is the gamma function $\Gamma(x)$

gammah(x)   is $\Gamma(x + \frac{1}{2})$. It is much faster than $\text{gamma}(x + \frac{1}{2})$

imag(x)   is the imaginary part of x

log(x)   is the natural logarithm

max(x,y)   is the maximum of x and y

min(x,y)   is the minimum of x and y

   To find the maximum or minimum of the components of a vector, use the commands **vecmax(v)** and **vecmin(v)**

norm(x)   is the product of x and its conjugates

polylog(m,x)   is the analytic continuation of $\text{Li}_m(x) = \sum_{n=1}^\infty x^n/n^m$

psi(x)   is the logarithmic derivative $\psi(x) = \Gamma'(x)/\Gamma(x)$ of the gamma function

random(N)   generates a random integer between 0 and $N - 1$

real(x)   is the real part of x. If x is a quadratic number, this gives the coefficient of 1 with respect to the canonical integral basis $\{1, w\}$. The coefficient of w is obtained via imag(x)

sign(x)   is 1 if $x > 0$, -1 if $x < 0$, and 0 if $x = 0$

sin(x)   is the sine of x

sinh(x)   is the hyperbolic sine of x

simplify(x)   will cancel common factors and change the type of an object when appropriate. For example, the complex number zero will be changed to the integer zero if the command expects an integer.

sqr(x)   is the same as x^2, the square of x

sqrt(x)   is the principal branch (angle in $(-\pi, \pi]$) of the square root of x

tan(x)   is the tangent of x

tanh(x)   is the hyperbolic tangent of x

truncate(x)   chops everything after the decimal point. For power series or p-adic numbers, it chops $O(x^k)$ or $O(p^k)$

zeta(s)   is the Riemann zeta function

   For n-th roots, a direct approach will work. For example, typing $2\hat{}(1/5)$will return the positive 5-th root of 2. For a primitive 5-th root of unity, type $\exp(2*Pi*I/5)$. There is a fancier method using **sqrtn(x,k)**, which gives the principal branch of the k-th root of x. For example, sqrtn(1.0,5) will return l.00000.... If you want a primitive fifth root of unity, type sqrtn(1.0,5,&z) [or sqrtn(1,5,&z)]. GP will still return 1.00000... [or 1], but now typing z will give a primitive root, and all the roots can be found via z^2, z^3 and z^4.

3. Arithmetic Functions

We now begin to do number theory. As an interesting example , we look at the problem of expressing an integer in binary and other bases. (Also see the discussion of p-adic numbers in Section 1.) Binary expansions are nice; if x is an integer, just type **binary(x)**. For example, $b = \text{binary}(142)$ yields $[1,0,0,0,1,1,1,0]$. But what if we want base 4? Here is a "workaround". Type $c = 142 + O(4^k)$ where any $k \geq 4$ will do (we must have $4^k \geq 142$). The output is $2 + 3 * 4 + 2 * 4^3 + O(4^k)$, which displays the desired base 4 expansion. Converting from base 4 back to decimal is accomplished via **d=truncate(c)**, which chops away the $O(4^k)$ and sums the series to get 142.

Let's do a very quick refresher on continued fractions. The Euclidean algorithm for dividing 41 by 18 looks like this:

$$41 = 2 \times 18 + 5$$
$$18 = 3 \times 5 + 3$$
$$5 = 1 \times 3 + 2$$
$$3 = 1 \times 2 + 1$$
$$2 = 2 \times 1 + 0$$

The corresponding continued fraction expansion is obtained from the column immediately to the right of the equals sign, traversed from top to bottom. We have

$$\frac{41}{18} = 2 + \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{2}}}}$$

which is abbreviated

$$2 + \frac{1}{3+} \; \frac{1}{1+} \; \frac{1}{1+} \; \frac{1}{2}$$

or $[2,3,1,1,2]$. Note that division of 18 by 41 begins with $18 = 0 \times 41 + 18$ and then proceeds as above. Thus $18/41 = [0, 2, 3, 1, 1, 2]$.

If $x = 41/18$, the command **contfrac(x)** gives the continued fraction expansion of x, namely $[2, 3, 1, 1, 2]$.

If $x = [a_0, a_1, \dots, a_n]$ is a finite continued fraction, the convergents $p_i/q_i, 0 \leq i \leq n$, of x are the successive continued fractions $[a_0], [a_0, a_1], \dots, [a_0, \dots, a_n]$. The p_i and q_i can be computed recursively by

$$p_m = a_m p_{m-1} + p_{m-2}, \quad q_m = a_m q_{m-1} + q_{m-2}$$

with initial conditions $p_0 = a_0, \quad q_0 = 1, \quad p_1 = a_1 a_0 + 1, \quad q_1 = a_1$. The command **contfracpnqn(x)** will return a 2 by 2 matrix whose first column is (p_n, q_n) and whose second column is (p_{n-1}, q_{n-1}). With $x = [2, 3, 1, 1, 2]$ we get $p_4 = 41, q_4 = 18, p_3 = 16, q_3 = 7$. Thus $p_4/q_4 = 41/18$ and $p_3/q_3 = 16/7 = [2, 3, 1, 1]$.

The instruction **contfrac(x,l)** will give an approximation to x based on l convergents. For example, contfrac(41/18,3) returns $[2,4]$, which is the same as $[2,3,1]$ (write out the expansion and check that the 3 and 1 can be added).

The continued fraction expansion of an irrational number is infinite, but when the irrational number is $\sqrt{N}$, there are very useful properties of periodicity and symmetry. We will illustrate with two examples. The continued fraction expansion of $\sqrt{21}$ is written as

$$\sqrt{21} = 4, \overline{1, 1, 2, 1, 1, 8}$$

The sequence under the overbar repeats periodically, so the complete expansion is

$$\sqrt{21} = [4, 1, 1, 2, 1, 1, 8, 1, 1, 2, 1, 1, 8, 1, 1, 2, 1, 1, 8, \dots]$$

as can be verified by the instruction contfrac(sqrt(21)). The periodic sequence always ends with twice the initial number. In this case, we have $8 = 4 \times 2$. The general pattern is

$$[a_0, a_1, \dots, a_n, 2a_0, a_1, \dots, a_n, 2a_0, a_1, \dots, a_n, 2a_0, \dots]$$

where the segment $a_1, \dots, a_n$ is symmetrical about its midpoint. If n is odd, then the n-th convergent gives a minimal solution of Pell's equation. In our case, $n = 5$ and the command contfracpnqn([4,1,1,2,1,1]) returns $p_n/q_n = 55/12$. And indeed we have

$$p_n^2 - Nq_n^2 = (55)^2 - 21(12)^2 = 3025 - 3024 = 1$$

Now consider

$$\sqrt{29} = 5; 2, 1, 1, 2, 10$$

which is an abbreviation of $5, \overline{2, 1, 1, 2, 10}$. Proceeding as in the first example, we apply the instruction contfracpnqn([5,2,1,1,2]) to get the fourth convergent $70/13$. We compute $(70)^2 - 29(13)^2 = -1$, not 1. In order to solve Pell's equation, we must go to the second period, specifically to position $2n + 1$. In this case $n = 4, 2n + 1 = 9$, so the appropriate command is contfracpnqn([5,2,1,1,2,10,2,1,1,2]). The program returns $9801/1820$, and $(9801)^2 - 29(1820)^2 = 1$.

A comment on what to do if the symmetrical segment is missing. For example, $\sqrt{2} = 1; 2 = [1, 2, 2, 2, \dots]$. We have $n = 0$, which is even, so for Pell's equation we go to the second period $[1,2]=1+1/2=3/2$. As expected, we get $3^2 - 2(2^2) = 1$. Going back to the first period, we have $[1]=1/1$, with $1^2 - 2(1^2) = -1$.

Here are some arithmetic commands. Unless otherwise specified, the arguments accepted are integers and polynomials (if the command makes sense for polynomials).

bestappr(x,k)   gives the best rational approximation to the real number x by a rational number with denominator at most k.

bezout(x,y)   returns u, v, d such that $u * x + v * y = d$, where d is the greatest common divisor of x and y, and u and v are, according to the manual, minimal in a natural sense. (This may mean the u and v found by the Euclidean algorithm.) For example bezout(123,54) returns $[-7, 16, 3]$, and we have $-7 * 123 + 16 * 54 = 3$.

bigomega(x)   returns the number of prime divisors of $|x|$, counting multiplicity. Thus bigomega(392) = 5 because $392 = 2^3 * 7^2$, so the sum of the exponents is 5.

binary(x) computes the binary expansion of x, as noted earlier.

binomial(x,y) is $\binom{x}{y}$, where y is an integer but x can be a real or even a complex number. The general definition is $\binom{x}{y} = x(x-1)\cdots(x-y+1)/y!$

chinese(x,y) gives the following output. If $x = \text{Mod}(a,m)$ and $y = \text{Mod}(b,n)$, then the program returns z such that $z \equiv a \mod m$ and $z \equiv b \mod n$. Thus if $x = 3 \mod 5$ and $y = 6 \mod 7$, then the output is $13 \mod 35$. Even if the moduli are not relatively prime, GP will attempt to find a solution, and will give an error message if no solution exists. If there are more than two simultaneous congruences, they can be handled by a workaround. If $z = \text{Mod}(c,p)$, we can let $w = \text{chinese}(x,y)$ and then compute $\text{chinese}(z,w)$. A more systematic and general method will be given in the section on linear algebra. See the command **matsolvemod**.

content(x) returns the greatest common divisor of the coefficients of a polynomial. If x is a rational function, the output is the ratio of the contents of numerator and denominator. If x is a vector or matrix, the result is the gcd of all the entries. If x is a scalar, the output is simply x.

core(x): If $n = cm^2$ where c is square-free, then $\text{core}(n) = c$. Thus c is the product of the distinct prime factors that appear an odd number of times in the prime factorization of n. In particular, if n is square-free, then $\text{core}(n) = n$.

divisors(x) returns the vector of divisors of x, in increasing order.

divrem(x,y) gives the Euclidean quotient and remainder. The remainder is always non-negative. Thus divrem(346,24) returns $[14,10]\tilde{\ }$, and $346 = 14*24+10$; divrem$(-346,24)$ returns $[-15,14]\tilde{\ }$, and $-346 = -15*24+14$.

eulerphi(n) computes the Euler phi function $\varphi(n)$.

factor(x) factors the integer or polynomial x. However, if x is an integer n, it is faster to use **factorint(n)**. The output gives all prime factors, with multiplicity. For example, factor$(x^3 + 13/2*x^2 + 12*x + 9/2)$ returns $[x+3 \quad 2]$ and $[2x+1 \quad 1]$. The factorization is $(x+3)^2*(2x+1)$. Thus GP will multiply by the least common denominator and then factor the resulting polynomial.

factorial(x) is $x!$, regarded as a real number. To treat the factorial as an integer, type **x!**.

factormod(f,p)) will factor a polynomial in $\mathbb{Z}[x]$ modulo the prime p. Possibly faster is **factorcantor(f,p)**. In either case, GP will divide by the leading coefficient and factor the resulting monic polynomial. For example, if $g = x^3 + 7*x^2 + 15*x + 9$, then factormod(g,11) will return the factorization $(x+1)(x+3)^2$. Factormod(g,2) will return $(x+1)^3$.

factormod(f,p,1) returns only the degrees of the irreducible polynomial factors. Thus if $p(x)$ is of degree 2 and $p(x)^5$ appears in the factorization, the contribution to the output will be $[2\ 5]$

fibonacci(n) gives the n-th fibonacci number.

ffinit(p,n,w) finds a monic irreducible polynomial of degree n over the finite field $\mathbb{F}_p$. For example, $F = \text{ffinit}(3,2,w)$ gives w^2+1. To express w^4 in terms of the basis elements 1 and w, type $\text{Mod}(\text{Mod}(1,3)*w^4, F)$, which returns

$$\text{Mod}(\text{Mod}(1,3), \text{Mod}(1,3)*w^2 + \text{Mod}(1,3)).$$

Thus $w^4 = 1$ in $\mathrm{GF}(3^2)$, so w is not primitive.

gcd(x,y) finds the greatest common divisor of x and y

isfundamental(x) returns 1 (true) if x is 1 or x is the discriminant of a quadratic field, and 0 (false) otherwise. Similarly for the Boolean operations **isprime(x)**, **issquare(x)**, and **issquarefree(x)**.

kronecker(x,y) returns the Legendre symbol (or its generalization, usually called the Jacobi symbol) $(\frac{x}{y})$. Thus kronecker(2,17)=1, kronecker(14,17)=-1, indicating that 2 is a quadratic residue mod 17, but 14 is not. Also, kronecker(2,21) returns -1 because $(\frac{2}{3}) = -1$ and $(\frac{2}{7}) = 1$.

lcm(x,y) is the least common multiple of x and y.

moebius(x) is the Möbius function $\mu(n) = 0$ if n contains a repeated prime factor, else $\mu(1) = 1$ and $\mu(n) = (-1)^k$, where k is the number of distinct prime factors of n.

nextprime(x) returns the smallest prime greater than or equal to x.

numdiv(x) is the number of divisors of $|x|$.

omega(x) is the number of distinct prime divisors of x.

precprime(x) is the largest prime less than or equal to x.

prime(n) is the n-th prime.

primes(n) returns a row vector whose components are the first n primes.

quaddisc(x) returns the discriminant of $\mathbb{Q}(\sqrt{x})$.

quadgen(d) creates the quadratic number field of discriminant d. The instruction $w = $ quadgen(d) will identify the canonical integral basis $\{1, w\}$. This has been discussed in Section 1, Pari type 15.

quadunit(d) returns the fundamental unit of the real quadratic field $\mathbb{Q}(\sqrt{d})$, where $d > 0$ is the discriminant. As an example of how to use this instruction, suppose we want the fundamental unit of $\mathbb{Q}(\sqrt{m})$, where $m = 11$. Since $m \not\equiv 1 \mod 4$, we have $w = \sqrt{11}$ and $d = 44$; also $\mathbb{Q}(\sqrt{11}) = \mathbb{Q}(\sqrt{44})$. The fundamental unit is $10 + 3 * w = 10 + 3\sqrt{11}$. On the other hand, if $m = 13 \equiv 1 \mod 4$, then $w = (1 + \sqrt{13})/2$ and $d = 13$. The fundamental unit is $1 + w = (3 + \sqrt{13})/2$.

sigma(x,k) is the sum of the k-th powers of the positive divisors of $|x|$. For example, sigma(12,3) $= 1^3 + 2^3 + 3^3 + 4^3 + 6^3 + 12^3 = 2044$.

sqrtint(x) returns the largest integer less than or equal to $\sqrt{x}$, where x is an integer.

znorder(x) computes the order of x in $\mathbb{Z}/n\mathbb{Z}$, where x is an integer mod n. There is an error message if x is not invertible. Thus typing $a =$Mod(36,49) followed by znorder(a) will give the output 7, indicating that 36 has order 7 mod 49.

znprimroot(x) calculates a primitive root of x, where x is a power of a prime. If p is prime and g is a primitive root mod p, then **znlog(x,g)** returns the discrete logarithm of x (to the base g) in $\mathbb{Z}/p\mathbb{Z}$. For example, 3 is a primitive root mod 17, and we have (mod 17) $3^0 = 1, 3^1 = 3, 3^2 = 9, 3^3 = 10, 3^4 = 13, 3^5 = 5, 3^6 = 15, 3^7 = 11, 3^8 = 16, 3^9 = 14, 3^{10} = 8, 3^{11} = 7, 3^{12} = 4, 3^{13} = 12, 3^{14} = 2, 3^{15} = 6, 3^{16} = 1$. Thus the discrete logs mod 17 (to the base 3) are

$$\log 1 = 0, \log 2 = 14, \log 3 = 1, \log 4 = 12, \log 5 = 5, \log 6 = 15, \log 7 = 11, \log 8 = 10,$$

$\log 9 = 2, \log 10 = 3, \log 11 = 7, \log 12 = 13, \log 13 = 4, \log 14 = 9, \log 15 = 6, \log 16 = 8.$

4. Polynomials and Power Series

algdep(x,k), where x is a real or complex number, finds a polynomial in $\mathbb{Z}[x]$ of degree at most k with x as a root.

concat(f,g) yields a vector whose first component is f and whose second component is g.

deriv(f,x) differentiates the polynomial or power series f with respect to x. If the variable is omitted, so the command is **deriv(f)**, then GP will choose the "main variable". One ranking sequence is $x > y > z > w > u > v$.

eval(f) will evaluate the polynomial f after an assignment of numbers to variables is given. Thus the sequence $f = x^2 + y^2;\quad x = 3;\quad y = 4;\quad \text{eval}(f)$ will return $f(3, 4) = 25$. Evaluation of power series is not yet implemented.

intformal(f,x) will antidifferentiate the polynomial or power series f with respect to x. If the variable is omitted, GP will choose the main variable as in deriv(f).

polcoeff(f,d,x) returns the coefficient of degree d in the variable x of the polynomial f. See above remarks for what happens if the variable is omitted.

poldegree(f,x) returns the x-degree of f. See above remarks for what happens if the variable is omitted.

poldisc(f,x) returns the discriminant of f, where f is regarded as a polynomial in x. See above remarks for what happens if x is omitted. If the roots of f are $x_1, \ldots, x_n$, then the discriminant is the product of $(x_i - x_j)^2$ taken over all $i < j$.

polinterpolate(x,y), where x is the vector $[x_1, \ldots, x_n]$ and $y = [y_1, \ldots, y_n]$, interpolates a polynomial through the n points (x_i, y_i). For example, let $x = [1, 2, 3], y = [4, 7, 0]$. The polynomial $-5x^2 + 18x - 9$ passes through $(1, 4), (2, 7)$ and $(3, 0)$.

polisirreducible(f) checks irreducibility of the polynomial f.

pollead(f,x) returns the leading coefficient of f, where f is regarded as a polynomial in the variable x. See above comments for what happens when the variable is omitted.

pollegendre(n,v) gives the n-th Legendre polynomial with variable v (default is x).

polrecip(f) returns the polynomial obtained from f by giving the coefficients in reverse order.

polresultant(f,g,x) computes the resultant of the polynomials f and g, regarded as functions of x. See above comments for what happens if the variable is omitted. For example, if $f(x) = a_0 x^2 + a_1 x + a_2$ and $g(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3$, then the resultant of f and g is

$$
\begin{vmatrix}
a_0 & 0 & 0 & b_0 & 0 \\
a_1 & a_0 & 0 & b_1 & b_0 \\
a_2 & a_1 & a_0 & b_2 & b_1 \\
0 & a_2 & a_1 & b_3 & b_2 \\
0 & 0 & a_2 & 0 & b_3
\end{vmatrix}
$$

The degree of f is 2 and the degree of g is 3, and there are 3 columns occupied by the coefficients of f and their shifts, and 2 columns occupied by the coefficients of g and their shifts.

polroots(f) returns a column vector whose components are the roots, each repeated according to multiplicity. **polroots(f,1)** uses the Newton-Raphson algorithm, which is faster but not guaranteed to converge.

polsym(f,s) returns a column vector giving the power sums of the roots $x_1, \dots, x_n$ of f, up to the power s. The power sum t_r is $\sum_{i=1}^n x_i^r$, with $t_0 = n$.

poltchebi(n,v) gives the n-th Chebyshev polynomial with variable v (default is x).

subst(f,y,z) replaces y by z in f, where f is a polynomial, rational function or power series.

taylor(f,x) computes the Taylor expansion of f, regarded as a function of x, around $x = 0$.

5. Linear Algebra

lindep(x), where x is a vector with real or complex coefficients, finds a linear dependence relation satisfied by the components of x. For example, lindep([1,2,3,4]) returns $[0, 1-2, 1]$.

matadjoint(A) computes the adjoint of the square matrix A (as in Cramer's rule).

matcompanion(f) returns the left companion matrix of the polynomial f. For example, if $f = 7y^3 + 2y^2 + 3y + 4$, then the output is

$$\begin{pmatrix} 0 & 0 & -4/7 \\ 1 & 0 & -3/7 \\ 0 & 1 & -2/7 \end{pmatrix}$$

matdet(A) computes the determinant of the square matrix A.

matdiagonal(v), where v is a vector, creates a diagonal matrix whose entries are the components of v.

mateigen(A) returns the eigenvectors of A, as columns of a matrix. The eigenvalues can be found from the eigenvectors. Here is an example:
$A = [0, 4; 2, -2]$ followed by $B =$ mateigen(A) returns $B = [-1, 2; 1, 1]$. Thus the eigenvectors are $[-1, 1]\tilde{\ }$ and $[2, 1]\tilde{\ }$. To get the eigenvalues, compute $C = A * B$ and $D = [C[1, 1]/B[1, 1], C[1, 2]/B[1, 2]] = [-4, 2]$. The eigenvalues are -4 and 2. If component-wise division of row 1 of C by row 1 of B runs into trouble because of division by zero, switch to a different row.

Eigenvalues can also be found directly. With the matrix A as above, Id=matid(2) creates the identity matrix of order 2. Then $f =$ matdet$(x * Id - A)$ returns $x^2 + 2 * x - 8$, and polroots(f) gives -4 and 2 as before. Alternatively, type $f =$ **charpoly(A)** to get the characteristic polynomial, and then apply polroots. [charpoly(A, y) returns the characteristic polynomial in the variable y; the default variable is x.] When the matrix is symmetric, eigenvalues and eigenvectors can be found with a single command, as we will see shortly.

mathilbert(n) returns an n by n matrix whose ij element is $1/(i + j - 1)$.

mathnf(A) computes the upper triangular Hermite normal form. (Allowable operations are interchange of columns and addition of a constant times one column to another column.)

matid(n) creates an n by n identity matrix.

matimage(A) finds a basis for the image (column space) of A.

Assume that matrices A and B have the same number of rows, and the columns of each matrix are linearly independent. Then

matintersect(A,B) finds a basis for the intersection of the column spaces (over $\mathbb{Q}$).

matinverseimage(A,b) computes $A^{-1}b$.

matisdiagonal(A) returns true iff A is a diagonal matrix.

matker(A) finds a matrix whose columns give a basis for the kernel (null space) of the matrix A. If A is known to have integer entries, use **matker(A,1)**.

matmuldiagonal(A,v) gives the same result as $A*\text{matdiagonal}(v)$, but is faster.

matmultodiagonal(A,B) gives the diagonal entries of $A * B$, with zeros elsewhere. It is faster than $A * B$ if the product is known to be diagonal.

matrank(A) returns the rank of A.

matsize(A) gives the number of rows and columns of A.

matsnf(A) computes the Smith normal form of A.

matsolvemod(A,m,y) generalizes the Chinese remainder theorem. A is a matrix with integer entries, m is a column vector of positive integer moduli, and y is a column vector with components in $\mathbb{Z}$. The output is a "small" integer solution to $\sum_j a_{ij}x_j \equiv y_i \mod m_i$.
matsolvemod(A,m,y,1) returns, in addition to the column vector x just found, a matrix whose columns are a basis for the homogeneous system $Ax \equiv 0$.

For example, consider the simultaneous equations $x + y \equiv 0 \mod 3, x + 2y \equiv 5 \mod 7$. Then $A = [1,1;1,2], m = [3,7]\tilde{}, y = [0,5]\tilde{}$. The program returns the column vector $[1,2]\tilde{}$, indicating the solution $x = 1, y = 2$. If matsolvemod$(A,m,y,1)$ is used (we say that the flag is set at 1), the output contains in addition the matrix $[-21,5;0,1]$. Thus $[-21,0]\tilde{}$ and $[5,1]\tilde{}$ are a basis for $x+y \equiv 0 \mod 3, x + 2y \equiv 0 \mod 7$.

matsupplement(A) extends the columns of the matrix A (assumed linearly independent, otherwise there will be an error message), to a basis for the entire space.

mattranspose(x) transposes a matrix or vector x.

qfgaussred(A) reduces the quadratic form associated with the symmetric matrix A to a sum of squares. For example, let $A = [1,2,3;2,4,5;3,5,6]$, with associated quadratic form $q = x^2 + 4xy + 6xz + 4y^2 + 10yz + 6z^2$. The output is the matrix

$$B = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 1/3 & 0 \\ 0 & 1/3 & -3 \end{pmatrix}$$

The quadratic form q can be expressed as

$$q = (x + 2y + 3z)^2 + (1/3)y^2 - 3(z + (1/3)y)^2.$$

The formula predicts what you might do in hand computation. Multiply out $(x+2y+3z)^2$; in order to get q, you need to add $-3z^2 - 2yz$. Complete the square to get $-3(z + y/3)^2$. You still need $y^2/3$, which is already in the desired form. The general formula is

$$q = \sum_i b_{ii}(x_i + \sum_{j \neq i} b_{ij}x_j)^2.$$

qfjacobi(A), where A is a real symmetric matrix, computes the eigenvalues and an orthogonal matrix of eigenvectors. For example, let $A = [4, 2, 2; 2, 4, 2; 2, 2, 4]$. The eigenvalues are 2 and 8, with 2 a double root. The orthogonal matrix of eigenvectors is

$$\begin{pmatrix} 1/\sqrt{3} & -1/\sqrt{2} & -1/\sqrt{6} \\ 1/\sqrt{3} & 1/\sqrt{2} & -1/\sqrt{6} \\ 1/\sqrt{3} & 0 & 2/\sqrt{6} \end{pmatrix}$$

The columns of this matrix are the eigenvectors, with column 1 associated with the eigenvalue 8 and columns 2 and 3 with 2.

qfsign(A) is the signature of the quadratic form associated with the symmetric matrix A, that is, the number of positive terms and the number of negative terms in the reduction to a sum of squares. If $A = 1, 2, 3; 2, 4, 5; 3, 5, 6]$, the matrix considered in connection with the instruction qfgaussred(A), there are two positive terms and one negative term, and the signature is $[2, 1]$.

trace(A) is the sum of the diagonal entries of the square matrix A.

GP regards a set as a row vector with strictly increasing entries, but you must instruct the program to convert a vector to a set. For example, if $v = [5, 4, 3, 2, 1]$, type **Set(v)**, and the result will be displayed as ["1", "2", "3", "4", "5"]. If v contains repeated entries, the repetition will be ignored. Several natural operations are available.

setisset(x) returns 1 (true) iff x is a set.

setintersect(x,y) is $x \cap y$.

setminus(x,y) is $x \setminus y$.

setsearch(x,y) returns j if y occupies position j of the set x. If $y \notin x$, the output is 0.

setunion(x,y) is $x \cup y$.

vecsort(v) sorts the vector v in ascending order.

6. Sums, Products and Integrals

intnum(x=a,b,f(x),j) numerically integrates $f(x)$ from $x = a$ to $x = b$. If the flag j is 0 or omitted, it is assumed that f is continuous and can be evaluated everywhere in $[a, b]$, with a and b "not too large". If $j = 1$, the program uses a relatively slow general numerical integration routine. The setting $j = 2$ is designed to handle infinite values of a or b. For practical purposes, infinity means 1.E4000, or much lower if the function decreases rapidly. We must have $ab > 0$, and if $b = \infty$, then a should be as large as possible, at least $a \geq 1$. If $j = 3$, then f is allowed to be continuous except for removable singularities, e.g., $(\sin x)/x$ at $x = 0$.

For example, to integrate $e^{1/x}$ from 1 to 2, type $\text{intnum}(x = 1, 2, \exp(1/x))$. The result is 2.0200586244...

prod(x=a,b,f(x),j) computes $\prod_{x=a}^{b} f(x)$. If the flag j is 1 or omitted, the computation is done exactly. If $j = 1.$, the computation uses real numbers with the default precision (28 significant figures). If $j = 1 + O(x^k)$, the computation uses power series with precision at most k.

For example, $\prod_{i=1}^{100}(1 - x^i)$ is a polynomial of degree $\sum_{i=1}^{100} i = 5050$. If we type the instruction $\text{prod}(i = 1, 100, 1 - x\hat{\ }i, 1 + O(x\hat{\ }101))$, the output will display the polynomial up to the term of degree 100, namely

$$1 - x - x^2 + x^5 + x^7 - x^{12} - x^{15} + x^{22} + x^{26} - x^{35} - x^{40} + x^{51} + x^{57} - x^{70} - x^{77} + x^{92}$$

$$+ x^{100} + O(x^{101}).$$

prodeuler(x=a,b,f(x)) computes the product of $f(x)$ as x ranges over all primes p such that $a \leq p \leq b$.

prodinf(x=a,f(x)) computes $\prod_{x=a}^{\infty} f(x)$.

prodinf(x=a,f(x),1) computes $\prod_{x=a}^{\infty}(1 + f(x))$.

solve(x=a,b,f(x)) finds a real root of $f(x)$ between a and b, assuming $f(a) * f(b) \leq 0$.

sum(x=a,b,f(x),j) computes $\sum_{x=a}^{b} f(x)$. See **prod** for syntax, with 1 replaced by 0 in the setting of the flag j. For example, the instruction $\text{sum}(i = 1, 5000, 1/i)$ yields a rational number with a huge numerator and denominator, but $\text{sum}(i = 1, 5000, 1/i, 0.)$ results in a much faster computation and the comprehensible result 9.094508852984.... A tricky point: $\text{sum}(i = 1, 5000, 1/i, 1.)$ will compute with real numbers, but will add 1 to the original sum to get 10.094508852984 .... Similarly, replacing $j = 1.$ in prod by $j = 2.$ will multiply the original product by 2.

sumalt(x=a,f(x)) computes $\sum_{x=a}^{\infty} f(x)$, where the series is alternating.

sumdiv(n,x,f(x)) sums $f(x)$ over all positive divisors of n. Note that $\text{sumdiv}(n, x, x^k)$ is the same as $\text{sigma}(x, k)$, but computation of $\text{sigma}(x, k)$ is much faster because it takes advantage of the multiplicativity of x^k.

suminf(x=a,f(x)) computes $\sum_{x=a}^{\infty} f(x)$.

7. Basic Programming

for(x=a,b,seq) evaluates seq from $x = a$ to $x = b$ in increments of 1, where a and b are real numbers. (If $a > b$, the program does nothing.) Note that the local variable x cannot be used to define a function inside the **for** loop. Functions should be defined before the loop is entered, as in the example below.

$f(x) = x + 1$
$g(x) = x^2$
$\text{for}(n = 1, 5, \text{print}(f(n) + g(n))$

The output is 3,7,13,21,31. $(f(4) + g(4) = 5 + 16 = 21$, etc.)

fordiv(n,x,seq) evaluates seq as x goes through all positive divisors of the integer n. With f and g as above,

fordiv$(12, x, \text{print}(f(x) + g(x)))$

yields the output 3,7,13,21,43,157, corresponding to the divisors 1,2,3,4,6 and 12.

forprime(x=a,b,seq) evaluates seq as x goes through all primes p such that $a \leq x \leq b$. At the end of each iteration, x is incremented to the next higher prime, so the last term evaluated is the largest prime less than or equal to b.

forstep(x=a,b,s,seq) evaluates seq as x goes from a to b in increments of s, where s is a nonzero real number, possibly negative.

forvec(w=v,seq) can be used to implement multiple for loops. Here is a sample program with annotations.

$w = [x, y]$ %w is a vector of variables
$v = [[1, 3], [1, 4]]$% v is a vector indicating the range of the variables, namely
$\qquad 1 \leq x \leq 3, 1 \leq y \leq 4$.
forvec$(w = v, \text{print}(w[1] + 2 * w[2]))$

% We are evaluating $x + 2y$ for x and y in the given range. Variables farther to the right move first, so y moves before x. Thus the output is $1 + 2 * 1 = 3, 1 + 2 * 2 = 5, 1 + 2 * 3 = 7, 1 + 2 * 4 = 9, 2 + 2 * 1 = 4, 2 + 2 * 2 = 6$, etc. The entire sequence is 3,5,7,9,4,6,8,10,5,7,9,11.

if(a,seq) evaluates seq if the Boolean variable a is not zero, otherwise does nothing.

if(a,,seq) (note the two commas) evaluates seq if $a = 0$, and otherwise does nothing. An equivalent command is **if(!a,seq)**. (The exclamation mark indicates "not".)

if(a,seq1,seq2) evaluates seq1 if $a \neq 0$, else evaluates seq2.

If we want to test whether a statement $a = b$ is true or false, we write the statement as **a==b**.

As an example, we will find integer solutions to $f(x, y) = x^3 - y^2 - 2 = 0$ for x and y between 0 and 100. Assuming that f has already been defined, a program is

$w = [x, y]$
$v = [[1, 100], [1, 100]]$
forvec$(w = v, \text{if}(f(w[1], w[2]) == 0, \text{print}("x = ", w[1], "y = ", w[2])))$

The output is $x = 3, y = 5$. With x and y arbitrary integers, it can be shown that the only solutions are $(3, \pm 5)$.

If you want to type the print instruction on a new line without triggering an error message, enclose the entire loop in **braces**, i.e. {forvec(...,print(...))}.

The same computation can be implemented via two for loops:

for$(x = 1, 100, \text{for}(y = 1, 100, \text{if}(f(x, y) == 0 , \text{print}("x = ", x, "y = ", y))))$

Here is a simple example illustrating how to name a procedure.
mult$(f, g) = f * g$
$f = x^2$
$g = x^3$
mult(f, g)
//The output is x^5

8. Algebraic Number Theory

We illustrate some basic calculations in number fields.

polcyclo(n) displays the n-th cyclotomic polynomial. If $n = 105$, you can verify that the polynomial does not have all its coefficients ± 1. It can be shown that any n with this property must be divisible by at least three distinct odd primes. The smallest number with this property is $3 * 5 * 7 = 105$. Thus for $n < 105$, all coefficients are ± 1.

Let $f = x^4 + 24 * x^2 + 585 * x + 1791$. The command polisirreducible(f) will check that f is irreducible.

F=nfinit(f) creates the number field defined by the monic irreducible polynomial f with integer coefficients.

nfbasis(f) finds an integral basis. The elements of the basis are

$$1, x, (1/3)x^2, (1/1029)x^3 + (33/343)x^2 - (155/343)x - (58/343)$$

where x is a root of f.

nfdisc(f) finds the field discriminant 18981.

The representation of ideals is tricky. Suppose $f = x^5 - 3$ and $F = \text{nfinit}(f)$ creates the number field defined by f. nfbasis(f) yields the integral basis $1, x, x^2, x^3, x^4$. Let $g = 1 + x + x^2$. Then g generates a principal ideal in the ring of algebraic integers. The command

idealhnf(F,g) gives a $\mathbb{Z}$-basis for this ideal, not a basis of algebraic integers (hnf stands for Hermite normal form). The result is the 5 by 5 matrix

$$\begin{pmatrix} 13 & 4 & 10 & 12 & 4 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The generators are $13w_1, 4w_1 + w_2, 10w_1 + w_3, 12w_1 + w_4$, and $4w_1 + w_5$, where the w_i are components of the integral basis just found. The instruction idealhnf(F,g,h) will give a $\mathbb{Z}$-basis for the ideal generated by g and h, but the program will not accept more than two generators. (An ideal in a Dedekind domain can be generated by at most two elements.)

As another example, let $f = x^2 - 13$, $F = \text{nfinit}(f)$; an integral basis is $w_1 = 1, w_2 = 1/2 + 1/2 * x$ where $x = \sqrt{13}$. The instruction idealhnf($F, 1/2 + 1/2 * x$) is equivalent to idealhnf($F, [0; 1]$). In each case, the output is $[3, 0; 0, 1]$, a diagonal matrix with entries 3 and 1. The generators are 3 and $1/2 + 1/2 * \sqrt{13}$. On the other hand, idealhnf(F, x) produces $[13, 6; 0, 1]$, with generators 13 and $6 + 1/2 + 1/2 * \sqrt{13} = 1/2 * (13 + \sqrt{13})$. (In the above paragraph, $1/2 * y$ means $(1/2)y$.)

To convert a polynomial g to an element of the number field F defined by f, type **Mod(g,f)**.

Another example: $g = x^4 - 7$, $G =$ nfinit(g); nfbasis(g) yields the integral basis $1, x, x^2, x^3$. Then the command idealhnf($G, 1 + x + x^2 + x^3$) returns the 4 by 4 matrix

$$\begin{pmatrix} 6 & 0 & 0 & 1 \\ 0 & 6 & 0 & 1 \\ 0 & 0 & 6 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A $\mathbb{Z}$-basis of the ideal consists of $6, 6x, 6x^2, 1 + x + x^2 + x^3$.

Here is the way to get maximum information about a number field with a minimum of instructions.

$f = x^4 - x^3 - 21 * x^2 + 17 * x + 133 // f$ is an irreducible polynomial.
$F =$**bnfinit(f)** creates the number field defined by f; bnf stands for "big number field".
$G =$**bnfclassunit(f)**. The program returns a column vector with 10 components and lots of good stuff. Type $G[1, 1]$ to recall the polynomial f. $G[2, 1]$ gives $[r_1, r_2]$ where r_1 is the number of real embeddings and $2r_2$ is the number of complex embeddings. In this case, $r_1 = 0, r_2 = 2$. $G[3, 1]$ is the field discriminant 18981 and the index 7. (The index of an algebraic integer α is $[\mathcal{O}_K : \mathbb{Z}[\alpha]]$.) $G[4, 1]$ gives an integral basis $[1, x, x^2, 1/7 * x^3 + 2/7 * x^2 - 1/7 * x]$. $G[5, 1]$ is a vector with three components. The first component is the class number, which is 4. The second component gives the structure of the ideal class group as a product of cyclic groups of various orders n_i. In this case, the ideal class group is cyclic of order 4. The third component of $G[5, 1]$ gives ideals in Hermite normal form that generate the cyclic components of the ideal class group. In our case there is only one component, so a single generator is given, represented by the matrix $[7, 5, 3, 6; 0, 1, 0, 0; 0, 0, 1, 0; 0, 0, 0, 1]$.

$G[6, 1]$ computes the regulator 3.794..., and $G[7, 1]$ is a measure of correctness of the regulator computation. (It should be close to 1.) In our case it is about .889; see the manual for details. $G[8, 1]$ gives the number m of roots of unity in the number field K, along with a primitive m-th root of unity. We have $m = 6$ with a primitive 6-th root of unity $w = 1/7 * x^3 - 5/7 * x^2 - 8/7 * x + 8$. To check that w^6 is 1 in K, type $u = \text{Mod}(w, f)$ and compute u^6. $G[9, 1]$ gives a system of fundamental units. In our case, $r_1 + r_2 - 1 = 1$, and there is only one fundamental unit, namely $x + 3$. $G[10, 1]$ is a measure of correctness of the calculation of fundamental units; it should be greater than 20 (we get 105). See the manual for details.

If you type $G =$bnfclassunit(f) and hit return, you will get all 10 components of G at once, which is likely to cause an information overload. To avoid this, put a semicolon after bnfclassunit(f) to suppress the output. Then you can view the components of G one at a time. The same comment applies to $F =$bnfinit(f) [or nfinit(f)].

We now illustrate many computations involving ideals by annotating a long example.
$f = x^3 - 2$
$F =$bnfinit(f);
//We adjoin a cube root of 2 to the rationals.
//The powers of x form an integral basis.
$i =$idealhnf($F, 1 + x$)
//The principal ideal generated by $(1+x)$ is represented by the matrix $[3, 1, 2; 0, 1, 0; 0, 0, 1]$,

//so the generators are $3, 1 + x$ and $2 + x^2$.

$j =$idealhnf$(F, 2 + 3 * x)$

//The principal ideal generated by $2+3*x$ is represented by the matrix $[62, 42, 34; 0, 1, 0; 0, 0, 1]$.

idealcoprime(F,i,j)

//This command produces an element α in the number field such that the ideals i and αj

//are relatively prime. In our case, the output $[1,0,0]\tilde{}$ says that i and j are already

//coprime. On the other hand, idealcoprime(F, i, i) gives $[1/3, -1/3, 1/3]\tilde{}$. In general,

// the coefficients of α are on an integral basis. Our case simplifies to

//$\alpha = 1/3 - 1/3 * x + 1/3 * x^2$.

idealaddtoone(F,i,j)

//Given relatively prime ideals i and j, we get elements $a \in i$ and $b \in j$ such that $a+b = 1$.

//The result is $a = [1, -2, -3]\tilde{}$, $b = [0, 2, 3]\tilde{}$. In general, the coefficients of a and b are

//on an integral basis.

idealadd(F,i,j)

//This adds the two ideals i and j. As expected, the sum is the unit ideal.

idealintersect(F,i,j)

//The intersection is represented by $[186, 166, 158; 0, 1, 0; 0, 0, 1]$.

$a1 = 1 - 2 * x - 3 * x^2$

$i1 =$idealhnf$(F, a1)$

//We create the principal ideal generated by $a1$, which is $[159, 88, 47; 0, 1, 0; 0, 0, 1]$.

idealmul(F,i1,i)

//This returns the product of the ideals $i1$ and i, which is $[159, 105, 64; 0, 3, 2; 0, 0, 1]$.

//But if we multiply $1 - 2 * x - 3 * x^2$ by $1 + x$ in the number field, we get, using $x^3 = 2$,

//$-5 - x - 5 * x^2$. The command idealhnf$(F, -5 - x - 5 * x^2)$ produces the same matrix

//$[159, 105, 64; 0, 3, 2; 0, 0, 1]$, as expected.

Commands with the prefix "nfelt" automatically convert a polynomial to an element in a number field, in other words, it is not necessary to type Mod(g, f). We continue the example with $f = x^3 - 2$.

$a = x^2 + 3 * x + 5$

$b = x + 1$

nfeltdiv(F,a,b) computes a/b in the number field, which is $x^2 + 3$.

$c = x^2 + 2$

nfeltdiveuc(F,b,c) finds an algebraic integer q such that $b - qc = r$ is "small". We get $q = x$ and $r = -x - 1$. The same result is obtained by applying the command **round( )** to the output of nfeltdiv(F, b, c).

nfeltmul(F,a,b) multiplies a and b in the number field. The result is $4 * x^2 + 8 * x + 7$.

nfeltdivmodpr(F,x,y,P) computes x/y modulo the prime ideal P.

nfeltmul(F,x,y,P) computes xy modulo the prime ideal P.

Let F be the number field defined by the monic irreducible polynomial f. If g is a polynomial with integer coefficients, we convert g to an element of the number field with the command **y=Mod(g,f)**.

trace(y), norm(y), charpoly(y) find the trace, norm and characteristic polynomial of y, respectively. The minimal polynomial is the unique irreducible factor of the characteristic polynomial.

We now consider factoring of prime ideals in extensions. Let F be the number field defined by the monic irreducible polynomial $f(x) = x^3 + 10 * x + 1$.
idealprimedec(F,2) factors (2) in the ring of algebraic integers of F. The output is $[[2,[1,1,0]\tilde{}\,,1,1,[1,1,1]\tilde{}\,,[2,[1,1,1]]\tilde{}\,,1,2],\,[1,1,0]\tilde{}\,]]]$. The first factor is $(2, 1 + x)$ (coefficients are on an integral basis, in this case $1, x, x^2$). The ramification index and residue class degree are $e = 1$ and $f = 1$. The term $[1,1,1]\tilde{}$ can be ignored. The second factor is $(2, 1 + x + x^2)$ with $e = 1$ and $f = 2$.

Similarly, the command idealprimedec$(F, 13)$ yields, ignoring the last term of the output, $[[13,[13,0,0]\tilde{}\,,1,3]]$. Thus (13) remains prime, with $e = 1$ and $f = 3$. The command factormod $(f, 13)$ confirms that f is irreducible over $\mathbb{F}_{13}$.

Here are some other useful commands.
polred(f) finds polynomials with "reasonably small" coefficients defining subfields of the number field defined by f.
idealinv(F,i) computes the inverse of the ideal i in the number field F.
ideallist(F,B) lists all ideals of norm $n, n = 1, 2, \ldots, B$.
idealnorm(F,i) computes the norm of the ideal i in the number field F.
idealpow(F,i,k) computes the k-th power of the ideal i in the number field F.
idealprincipal(F,a) creates the principal ideal generated by the algebraic number a in the number field F.
bnfisprincipal(F,i) tests whether the ideal i is principal (this holds iff the first component of the output is zero).
idealval(F,i,P) gives the valuation of the ideal i at the prime ideal P.
nfeltval(F,x,P) gives the valuation of the ideal (x) at the prime ideal P.
F.diff gives a $\mathbb{Z}$-basis for the different of the number field F (components with respect to an integral basis).
nfalgtobasis(F,a) converts the expression for the algebraic integer a in the number field F in terms of algebraic numbers to an expression in terms of an integral basis.
nfbasistoalg(F,a) is the inverse of nfalgtobasis. For example, in the number field $\mathbb{Q}(\sqrt{13})$, with $x = \sqrt{13}$, nfalgtobasis will convert x to $[-1,2]\tilde{}$. Since the integral basis consists of 1 and $1/2 + 1/2 * \sqrt{13}$, we get $-1 + 2 * (1/2 + 1/2 * \sqrt{13}) = \sqrt{13}$. The inverse operation will convert the vector $[-1,2]\tilde{}$ to x, represented as $\mathrm{Mod}(x, x^2 - 13)$.
nfrootsof1(F) will find the number t of roots of unity in the number field F, and a primitive t-th root of unity, expressed in terms of an integral basis.
nffactor(F,g) factors the polynomial g over the number field F. Use y for the variable of the defining polynomial f, and x as the variable of g (x has higher priority than y).
nfroots(F,g) finds the roots of g in the number field F, with the same variable-naming convention as in nffactor.
nffactormod(F,g,P) factors g mod the prime ideal P, with the same variable-naming convention as above.
nfsubfields(F,d) finds all subfields of F of degree d. If d is omitted,
nfsubfields(F) finds all subfields of F.

Additional Number Field Ideas (Thanks to Darrin Doud)
If an ideal i is principal, the second component of **bnfisprincipal(F,i)** will provide a generator. In general, **idealtwoelt(F,i)** will give two generators of i as $\mathcal{O}_F$-module. If one of the generators is 0, the ideal is principal. An example:

$f = x^3 + 10x + 1$

$F = \text{bnfinit}(f)$;

$i = \text{idealhnf}(F, 2, 1 + x)//[2,1,1;0,1,0;0,0,1]$

idealtwoelt$(F, i)//[2,[1,1,0]\tilde{\ }]$. The generators are 2 and $1 + x$, as expected.

polcompositum(f,g) produces a list of polynomials defining a number field $\mathbb{Q}(a, b)$, where a is a root of f and b is a root of g. The list may contain more than one member because of relations between the roots. For example, let $f = g = x^3 - 3$. There are two polynomials on the list, and the first defines $\mathbb{Q}(a, a) = \mathbb{Q}(a)$. The second defines $\mathbb{Q}(a, b)$, $\mathbb{Q}(a, c)$, or $\mathbb{Q}(b, c)$. (These three fields are isomorphic.) To (possibly) simplify one of these polynomials h, type **polredabs(h)**.

We have seen that in the finite field GF(3^2), the polynomial $x^2 + 1$ is irreducible but not primitive. Here is a general script that will find a primitive polynomial. The algorithm successively eliminates elements whose order properly divides $p^n - 1$.

primpoly$(p, n, w, s, f) = f = w^{p^n - 1)} - 1; s = \text{divisors}(p^n - 1); \text{for}(k = 1, \text{length}(s) - 1, f = f/\gcd(f, w^{s[k]} - 1));\text{factormod}(f, p)[1, 1]$

To get a primitive polynomial for $\mathbb{F}_9$, type primpoly(3,2,w,s,f). The output is $w^2 + w + 1$.

To clarify the general situation, look at the specific case $p = 3, n = 2$ and consider a script for that case.

$f = x^8 - 1$

$s = \text{divisors}(8)$

$\text{for}(k = 1, \text{length}(s) - 1, f = f/\gcd(f, x^{s[k]} - 1))$

$\text{factormod}(f, 3)[1, 1]$

//The output is $x^2 + x + 1$. Note that $[1,1]$ can be omitted in the last step. In that case, the output will be a vector whose first component is the primitive polynomial, and whose second component is the multiplicity, namely 1.

9. Elliptic Curves

An elliptic curve is represented in general Weierstrass form: $y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$.

E=ellinit([a1,a2,a3,a4,a6],1) creates an elliptic curve E with the given coefficients. The short Weierstrass form has $a_1 = a_2 = a_3 = 0$. If the flag (the "1" at the end of the sequence) is 0 or omitted, the output of E will contain 19 quantities instead of 13. In particular, if the curve is defined over $\mathbb{R}$, then $E[14]$ will return the roots of the Weierstrass equation. In both cases, the first 13 quantities are $a_1, a_2, a_3, a_4, a_6, b_2, b_4, b_6, b_8, c_4, c_6, \Delta$ (the discriminant) and j (the j-invariant). The discriminant can be recovered by either $E[12]$ or E.disc, and the j-invariant by $E[13]$ or $E.j$

elladd(E,z1,z2) adds the points $z1$ and $z2$ on the elliptic curve E, with addition according to the group law. Points on the curve are represented by $[x, y]$, with the point at infinity represented by $[0]$.

ellak(E,n) computes the coefficient a_n of the L-function of the elliptic curve E. If n is not prime to the conductor, E must be in minimal Weierstrass form. If it isn't, then first use **ellglobalred(E)**.

ellan(E,n) outputs a vector containing the first n coefficients of the L-function.

ellap(E,p), with p prime, computes a_p, where the number of finite points in $E(\mathbb{F}_p)$ is $p - a_p$. (Counting the point at infinity, we have $p + 1 - a_p$.) For example, $E = \text{ellinit}([0,0,0,-4,0])$ creates the elliptic curve $y^2 = x^3 - 4x$. Then $\text{ellap}(E, 11)$ yields output 0, so the number of finite points of E over $\mathbb{F}_{11}$ is 11. This can be checked by the following program:

$w = [x, y]$
$v = [[0, 10], [0, 10]]$
$\text{forvec}(w = v, \text{if}(\text{Mod}(w[1]^3 - 4 * w[1] - w[2]^2, 11) == 0, \text{print}(w)))$

The 11 points are $(0,0), (2,0), (3,\pm 2), (4,\pm 2), (6,\pm 4), (9,0), (10,\pm 5)$. To add the points (3,2) and (10,5), type $\text{elladd}(E,\text{Mod}([3,2],11),\text{Mod}([10,5],11))$. The result is the point (2,0).

A change of coordinates that preserves Weierstrass form is given by $x = u^2 x' + r, y = u^3 y' + s u^2 x' + t$.

ellchangecurve(E,v), where v is the vector $[u, r, s, t]$, gives the coefficients of the Wererstrass equation in the new coordinate system.

ellchangepoint(x,v) gives the coordinates of the point x in the new system.

ellisoncurve(E,z) is true iff z is on E.

elllseries(E,s) returns the L-series of E at the real number s. Note that the command has three consecutive l's.

ellorder(E,z) gives the order of z if z is a torsion element, otherwise retuns 0.

ellordinate(E,x) finds y such that (x, y) is on E. If there is no such y, then the output is [ ]. If there is exactly one y, the output is $[y]$, and if there are two or more y's, the output produces two of them, in the form $[y1, y2]$.

ellpow(E,z,n) computes nz, where z is on the curve E.

ellsub(E,z1,z2) finds $z1 - z2$, where $z1$ and $z2$ are points on E. To get the inverse of z, take $z1 = [0]$ and $z2 = z$.

elltors(E), where E is defined over $\mathbb{Q}$, returns $[t, v1, v2]$, where t is the order of the torsion subgroup, $v1$ gives the structure of the torsion group as a product of cyclic groups, sorted in increasing order, and $v2$ gives generators of the cyclic groups. For example, $E = \text{ellinit}([0,0,0,-1,0])$ creates the elliptic curve $y^2 = x^3 - x$. Then $\text{elltors}(E)$ outputs $[4, [2, 2], [[0, 0], [1, 0]]]$. The torsion subgroup has order 4 and is isomorphic to the product of two cyclic groups of order 2. Generators for the cyclic groups are [0,0] and [1,0].