higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas
2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

MCS 507 Lecture 37
Mathematical, Statistical and Scientific Software
Jan Verschelde, 15 November 2023

higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas
2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

PARI/GP

PARI stems from Pascal ARIthmetic (although switched to C, now a pun on “pari de Pascal”), and GP means the Great Programmable calculator, distributed under the GNU General Public License.

PARI/GP is a widely used computer algebra system designed for fast computations in number theory (factorizations, algebraic number theory, elliptic curves...), but also contains a large number of other useful functions to compute with mathematical entities such as matrices, polynomials, power series, algebraic numbers, etc., and a lot of transcendental functions.

Originally (1985-1996) developed at Université Bordeaux I by a team led by Henri Cohen, PARI/GP is maintained since 1995 by Karim Belabas (Université Bordeaux I) with the help of many volunteer contributors.
The “Ateliers PARI/GP” are periodic workshops about PARI/GP or PARI/GP development, see `pari.math.u-bordeaux1.fr`.

PARI/GP in Sage

Although originally intended as a debugging device for the PARI system library, `gp` became a powerful user-friendly stand-alone programmable calculator.

GP is the name of `gp`'s scripting language.

The compiler gp2c translates GP code to C which makes it run 3 to 10 times faster (from the User's Guide to PARI/GP).

We run `gp` explicitly in Sage via

1. the class `Gp`, do `help(gp);` or
2. opening a Terminal Session with gp,
   type `gp_console();` in a Sage terminal.

At `https://pypi.org/project/cypari2`, there is a Python interface to PARI/GP, developed by SageMath.

using PARI/GP in SageMath

```
SageMath version 8.8, Release Date: 2019-06-26
Using Python 2.7.15. Type "help()" for help.
```

```
sage: gp_console()
                GP/PARI CALCULATOR Version 2.11.1 (released)
          i386 running darwin (x86-64/GMP-6.0.0 kernel) 64-bit version
       compiled: Sep 28 2019, Apple LLVM version 10.0.1 (clang-1001.0.46.4)
                             threading engine: single
                 (readline v6.3 enabled, extended help enabled)

                     Copyright (C) 2000-2018 The PARI Group

PARI/GP is free software, covered by the GNU General Public License, and comes
WITHOUT ANY WARRANTY WHATSOEVER.

Type ? for help, \q to quit.
Type ?17 for how to get moral (and possibly technical) support.

parisize = 8000000, primelimit = 500000
?
```

higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas
2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

approximating π

To approximate π we could use

$$\begin{aligned} \frac{\pi}{4} &= \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \\ &= 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots \end{aligned}$$

but this series converges very slowly.

Because of slow convergence, we could conclude that π cannot be computed as above, but `gp` proves us wrong.

checking the slow sum

```
? \p 10
   realprecision = 19 significant digits \
(10 digits displayed)
? sum4pi = 4*sum(k=0,1000,(-1.0)^k/(2*k+1))
%4 = 3.142591654
? sum4pi - Pi
%5 = 0.0009990007497
? sum4pi = 4*sum(k=0,100000,(-1.0)^k/(2*k+1))
%6 = 3.141602653
? sum4pi - Pi
%7 = 9.999900001 E-6
? sum4pi = 4*sum(k=0,10000000,(-1.0)^k/(2*k+1))
%8 = 3.141592754
? sum4pi - Pi
%9 = 9.999999000 E-8
```

the command `sumalt`

```
? \p 100
   realprecision = 115 significant digits \
(100 digits displayed)
? pi100 = 4*sumalt(k=0, (-1)^k/(2*k+1) )
%1 = 3.1415926535897932384626433832795028841971\
69399375105820974944592307816406286208998628034\
34825342117068
? Pi
%2 = 3.1415926535897932384626433832795028841971\
69399375105820974944592307816406286208998628034\
34825342117068
? pi100 - Pi
%3 = 6.09106040955755813 6 E-115
```

`sumalt` accelerates the convergence of alternating series with algorithms of Cohen, Villegas, and Zagier.

in a Sage session

```
sage: gp("default(realprecision,100)");
sage: pi100 = gp("4*sumalt(k=0, (-1)^k/(2*k+1))");
sage: pi100
3.1415926535897932384626433832795028841971693993\
75105820974944592307816406286208998628034825342 1\
17068
sage: type(pi100)
<class 'sage.interfaces.gp.GpElement'>
sage: RF = RealField(340);
sage: sagepi100 = RF(pi100)
sage: sagepi100
3.1415926535897932384626433832795028841971693993\
75105820974944592307816406286208998628034825342 1\
1706800
sage: type(sagepi100)
<type 'sage.rings.real_mpfr.RealNumber'>
```

higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas
2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

rational and real numbers

```
? \p 10
   realprecision = 19 significant digits \
(10 digits displayed)
? a = 3/4
%10 = 3/4
? type(a)
%11 = "t_FRAC"
? b = 3/4 + 0.
%12 = 0.7500000000
? type(b)
%13 = "t_REAL"
```

best rational approximation

```
? bestappr(Pi)
%14 = 17004845848539028973023/541281054662\
1363616752
? \p 100
   realprecision = 115 significant digits \
(100 digits displayed)
? bestappr(Pi)
%15 = 17382575228544640824232996315033678889\
46872125388528540003785218861108845053813327\
29836651372515448507755586292080366116 9/553\
30455425792237409825334741304197883700560 35\
63748351097693801716002217179952188259021 91\
38105306410292238782930808810
? %16 - Pi
%17 = 0.E-114
```

the function `bestappr`

The second optional argument of `bestappr`
is the upper bound on the size of the denominator.

```
? bestappr(sqrt(2),10)
%18 = 7/5
? bestappr(sqrt(2),10^4)
%19 = 8119/5741
? bestappr(sqrt(2),10^8)
%20 = 131836323/93222358
```

For approximations of increasing precision:

```
? apply(k->bestappr(sqrt(2),10^k),[1,2,3,4])
%21 = [7/5, 99/70, 1393/985, 8119/5741]
```

higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas

2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

functions and formulas

```
? f(x) = x*(x^2-1)
%1 = (x)->x*(x^2-1)
? f(2)
%2 = 6
? f(z)
%3 = z^3 - z
? dfz = deriv(f(z),z)
%4 = 3*z^2 - 1
? g(x) = substpol(dfz,z,x)
%5 = (x)->substpol(dfz,z,x)
? g(2)
%6 = 11
? g(z)
%7 = 3*z^2 - 1
```

the Cauchy integral formula

The number of roots of $f(z)$ in a disk $C_{a,r}$ in the complex plane centered at $z = a$ and with radius r is

$$\frac{1}{2\pi i} \oint_{C_{a,r}} \frac{f'(z)}{f(z)} dx$$

We compute circular integrals with `intcirc`.

```
? \p 5
? f(z)
%8 = z^3 - z
? g(z)
%9 = 3*z^2 - 1
? intcirc(z=0,0.5,g(z)/f(z))
%10 = 1.0000 + 0.E-21*I
? intcirc(z=0,1.5,g(z)/f(z))
%11 = 3.0000 + 0.E-20*I
? intcirc(z=0.9,0.3,g(z)/f(z))
%13 = 1.0000 + 0.E-21*I
```

higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas

2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

binary and other expansions

```
? b = binary(142)
%1 = [1, 0, 0, 0, 1, 1, 1, 0]
? c = 142 + O(4^6)
%2 = 2 + 3*4 + 2*4^3 + O(4^6)
? c = 142 + O(4^8)
%3 = 2 + 3*4 + 2*4^3 + O(4^8)
? d = truncate(c)
%4 = 142
? e = 142 + O(7^6)
%5 = 2 + 6*7 + 2*7^2 + O(7^6)
```

The same syntax applies for power series:

```
? exp(z)*sin(z) + O(z^6)
%6 = z + z^2 + 1/3*z^3 - 1/30*z^5 + O(z^6)
```

calculating with expansions

```
? a = 142 + O(7^6)
%1 = 2 + 6*7 + 2*7^2 + O(7^6)
? b = 300 + O(7^6)
%2 = 6 + 6*7^2 + O(7^6)
? c = a + b
%3 = 1 + 2*7^2 + 7^3 + O(7^6)
? lift(c)
%4 = 442
```

`lift(x)`: lifts an element `x` of $\mathbb{Z}_n$ to $\mathbb{Z}$.

calculating with series

```
? p = exp(z) * sin(z) + O(z^6)
%1 = z + z^2 + 1/3*z^3 - 1/30*z^5 + O(z^6)
? q = exp(z) * cos(z) + O(z^6)
%2 = 1 + z - 1/3*z^3 - 1/6*z^4 - 1/30*z^5 + O(z^6)
? r = p + q
%3 = 1 + 2*z + z^2 - 1/6*z^4 - 1/15*z^5 + O(z^6)
? truncate(r)
%4 = -1/15*z^5 - 1/6*z^4 + z^2 + 2*z + 1
```

`truncate(r)`: converts a series `r` to a polynomial by truncation.

continued fractions

```
? L = contfrac(sqrt(21),19)
%7 = [4, 1, 1, 2, 1, 1, 8, 1, 1, 2, 1, 1, 8, 1, 1, 2, 1, 1,
```

The continued fraction expansion of a rational number is computed by repeated greatest common divisor computations of numerator and denominator.

$$\sqrt{21} = 4 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{8 + \cdots}}}}}}$$

We denote the periodicity in $\sqrt{21}$ by $[4, \overline{1, 1, 2, 1, 1, 8}]$.

consecutive convergents

```
? A = contfrac(sqrt(21),9)
%8 = [4, 1, 1, 2, 1, 1, 8, 2]
? B = contfracpnqn(A)
%9 =
[999 472]

[218 103]

? 999^2 - 21*(218^2)
%10 = -3
? 472^2 - 21*(103^2)
%11 = -5
```

This implies $\frac{999^2}{218^2} - 21 = \frac{-3}{218^2}$ or $21 = \frac{999^2}{218^2} + \frac{3}{218^2}$,
so $\sqrt{21} = \frac{999}{218} + \frac{\sqrt{3}}{218}$.

higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas

2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

integer relation detection

We have $\log(15) = \log(5 \cdot 3) = \log(5) + \log(3)$.

```
? \p 10
   realprecision = 19 significant digits \
(10 digits displayed)
? u = [log(15.0), log(5.0), log(3.0)]
%12 = [2.708050201, 1.609437912, 1.098612289]
? lindep(u)
%13 = [-1, 1, 1]~
```

We find $(-1)\log(15) + (+1)\log(5) + (+1)\log(3)$.

Calling `lindep(u,-3)` uses the PSLQ Algorithm.

Note: the type of `%13` is `"t_COL"` (column vector).

lattice basis reduction

A lattice is similar to a vector space. Like a vector space, a lattice is spanned (or generated) by a finite number of basis vectors.
Unlike a vector space, the basis vectors have only integer entries and only integer linear combinations of the basis vectors are allowed.

Given any basis of a lattice, the LLL-algorithm finds a reduced basis, consisting of short vectors. A short vector has small length and its entries are small numbers, like the entries in the vectors of the subset sum problem are either zero or one.

Consider the following weights: $\mathbf{w} = (366, 385, 392, 401, 422, 437)$ and sum $s = 1215$. This leads to the lattice basis $(1, 0, 0, 0, 0, 0, -366)$, $(0, 1, 0, 0, 0, 0, -385)$, $(0, 0, 1, 0, 0, 0, -392)$, $(0, 0, 0, 1, 0, 0, -401)$, $(0, 0, 0, 0, 1, 0, -422)$, $(0, 0, 0, 0, 0, 1, -437)$, and $(0, 0, 0, 0, 0, 0, 1215)$.
The output of the LLL-algorithm gives a reduced basis, which may contain a solution to the subset sum problem $s = \mathbf{w} \cdot \mathbf{x}$.

solving a knapsack problem

```
? w = [366, 385, 392, 401, 422, 437]
%1 = [366, 385, 392, 401, 422, 437]
? type(w)
%2 = "t_VEC"
? L = List(w)
%3 = List([366, 385, 392, 401, 422, 437])
? s = 1215
%4 = 1215
? listinsert(L,-s,1)
%5 = -1215
? L
%6 = List([-1215, 366, 385, 392, 401, 422, 437])
? v = Vec(L)
%7 = [-1215, 366, 385, 392, 401, 422, 437]
? x = lindep(v)
%8 = [1, 0, 0, 1, 1, 1, 0]~
```

algebraic numbers

For a floating point number x, we can ask to compute the polynomial that has x as its root.

`algdep`(x, k) is a call to `lindep`$([1, x, \ldots, x^k])$

```
? \p 10
? x = 3.0^(1/6)
%14 = 1.200936955
? a = algdep(x,10)
%15 = x^8 - 3*x^2
? a = algdep(x,10,-3)
%16 = x^10 - x^7 - 3*x^4 + 3*x
```

Using PSLQ with the flag `-3` does not work well.

higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas

2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

linear algebra

```
? A = [5, -4; -1, 1]
%17 =
[5 -4]

[-1 1]

? x = [80; 100]
%18 =
[80]

[100]

? b = A*x
%19 =
[0]

[20]
```

the Hermite normal form

The Hermite normal form is computed by `mathnf`:

```
? L = mathnf(A,1)
%20 = [[1, 0; 0, 1], [1, 4; 1, 5]]
? U = L[2]
%21 =
[1 4]

[1 5]

? U*A
%22 =
[1 0]

[0 1]
```

solving linear systems

Transforming a right hand side vector:

```
? b = [0; 20]
%23 =
[0]

[20]

? U*b
%24 =
[80]

[100]
```

To solve the linear system directly:

```
? y = matsolve(A,b)
%25 =
[80]

[100]
```

INTMOD objects

```
? a = Mod(19,5)
%23 = Mod(4, 5)
? b = Mod(23,5)
%24 = Mod(3, 5)
? a + b
%25 = Mod(2, 5)
? (19 + 23) % 5
%26 = 2
? 19 % 5 + 23 % 5
%27 = 7
? type(a)
%28 = "t_INTMOD"
? c = lift(a)
%29 = 4
? type(c)
%30 = "t_INT"
```

modular arithmetic

```
? inva = 1/a
%31 = Mod(4, 5)
? a*inva
%32 = Mod(1, 5)
? cba = a^(1/2)j
%33 = Mod(3, 5)
? cba^2
%34 = Mod(4, 5)
```

higher arithmetic with PARI/GP

1. PARI/GP in SageMath
   - the computer algebra system PARI/GP
   - acceleration of alternating series
   - rational approximations
   - functions and formulas

2. Arithmetic Functions
   - binary expansions and continued fractions
   - integer relation detection
   - modular arithmetic
   - linear systems of congruences

linear systems of congruences

Consider the input data:

- $M = [m_{ij}]$ is any integral matrix;
- D a column vector of moduli; and
- B an integer column vector.

We look for **x**, a small integer solution to

$$\sum_j m_{ij} x_j \equiv b_i \mod d_i$$

for i running over all rows of M.

Chinese remaindering (Sun-tzu Suan-ching)

```
? A = [1,1; 1,2]
%1 =
[1 1]

[1 2]

? m = [3,7]~
%2 = [3, 7]~
? y = [0,5]~
%3 = [0, 5]~
? x = matsolvemod(A,m,y)
%4 = [1, 2]~
? A*x
%5 = [3, 5]~
?
```

Summary + Exercises

The *SIAM 100-digit challenge. A Study in High-Accuracy Numerical Computing* book by F. Bornemann, D. Laurie, S. Wagon, and J. Waldvogel (SIAM 2004) contains many advanced GP scripts.

Read *A Tutorial for PARI/GP* (available online).

1. Redo the experiments of lecture one with `lindep`.
2. Explore the computation of the Smith normal form of a matrix and compare with the implementation in GAP.