Implementing p-adic numbers in Macaulay2 using its foreign function interface and FLINT

Douglas A. Torrance[1,2][0000−0003−3999−4973]

[1] Piedmont University, Demorest, GA 30535, USA
[2] Georgia Institute of Technology, Atlanta, GA 30332, USA dtorrance9@gatech.edu

Abstract. Macaulay2 is a computer algebra platform widely used by researchers in algebraic geometry and commutative algebra. Using the ForeignFunctions package, it is possible to make calls from Macaulay2 to dynamic libraries such as FLINT. We demonstrate this by introducing a new Macaulay2 package implementing p-adic numbers using FLINT via this interface. We discuss implementation details such as memory allocation, interaction with Macaulay2's garbage collector, and object-oriented design decisions that mirror the existing implementations of the real and complex number fields in Macaulay2.

Keywords: Macaulay2 · FLINT · foreign function interface · p-adic numbers

1 Introduction

Macaulay2 [3] is an important research tool for computational algebraic geometers and commutative algebraists. It supports computations in a wide variety of rings, including $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, $\mathbb{F}_{p^k}$, polynomial rings, quotient rings, and fraction fields. However, historically it has not supported computations in the field $\mathbb{Q}_p$ of p-adic numbers.

Enter FLINT [5], a popular C library for number theory, which does support p-adic numbers. Using the Macaulay2 foreign function interface [6], it is possible to make calls to p-adic number functions in FLINT from top-level Macaulay2. We introduce the *Padic* package for Macaulay2, available at https://github.com/d-torrance/macaulay2-padic, which does this to provide an interface for Macaulay2 users to perform computations in $\mathbb{Q}_p$. This package has been distributed with Macaulay2 since version 1.26.05 and provides complete documentation of its features.

This exists as a proof of concept, as potentially p-adic support (likely still using FLINT) might one day be added to the Macaulay2 engine and interpreter so that computations in not just $\mathbb{Q}_p$, but in polynomial rings over $\mathbb{Q}_p$ and using matrices with entries in $\mathbb{Q}_p$, are possible.

This paper is organized as follows. In Section 2, we provide a brief overview of the p-adic numbers, and in Section 3 we describe their support in FLINT. In Section 4, we describe how Macaulay2 interacts with FLINT at a low level, and in Section 5, we look at the interface from a higher level, describing in particular various design decisions. Finally, we close with a Macaulay2 computation demonstrating Hensel's lemma in Section 6.

2 A review of p-adic numbers

We open with a crash course on p-adic numbers. The reader is encouraged to reference a textbook such as [2] for more information.

Suppose p is a prime number. The field of *p-adic numbers* $\mathbb{Q}_p$ is a set whose elements may be expressed uniquely as formal Laurent series

$$\sum_{n=k}^{\infty} a_n p^n = a_k p^k + a_{k+1} p^{k+1} + \cdots, \tag{1}$$

where $k \in \mathbb{Z}$ and $a_n \in \{0, \ldots, p-1\}$, together with the usual operations of addition and multiplication in base p, but noting that carrying may continue indefinitely. An enlightening exercise is to verify that $-1 = \sum_{n=0}^{\infty} (p-1) \cdot p^n$.

For a given $x \in \mathbb{Q}_p$, its *p-adic valuation*, denoted $\nu_p(x)$, is the smallest exponent that appears with a nonzero coefficient in the expansion given in Equation (1), e.g., $\nu_3(1 \cdot 3^{-2} + 2 \cdot 3^{-1}) = -2$. In particular, $\nu_p(0) = \infty$.

These series are not convergent in general using the usual absolute value, but they do converge using the p-adic absolute value

$$|x|_p = p^{-\nu_p(x)}.$$

In fact, $\mathbb{Q}_p$ is the completion of $\mathbb{Q}$ under $|\cdot|_p$, just as $\mathbb{R}$ is the completion of $\mathbb{Q}$ under the usual absolute value.

The set of all $x \in \mathbb{Q}_p$ with $\nu_p(x) \geq 0$ forms a ring, known as the *p-adic integers* and denoted $\mathbb{Z}_p$. Furthermore, the multiplicative group $\mathbb{Z}_p^\times$ of the units of $\mathbb{Z}_p$ is precisely the $u \in \mathbb{Q}_p$ for which $\nu_p(u) = 0$. It follows that for any nonzero $x \in \mathbb{Q}_p$ with $\nu_p(x) = \nu$,

$$x = \sum_{n=\nu}^{\infty} a_n p^n = \left(\sum_{n=0}^{\infty} a_{n+\nu} p^n \right) p^\nu = u p^\nu, \tag{2}$$

where $u = \sum_{n=0}^{\infty} a_{n+\nu} p^n \in \mathbb{Z}_p^\times$.

The p-adic exponential and logarithmic functions are defined using their usual power series expansions provided that they converge, i.e., for $x \in \mathbb{Q}_p$ with $|x|_p < p^{-\frac{1}{p-1}}$,

$$\exp_p x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

and for $x \in \mathbb{Q}_p$ with $|x - 1|_p < 1$,

$$\log_p x = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}(x-1)^n}{n}.$$

Finally, for every $x \in \mathbb{Z}_p^\times$ there exists a unique $t \in \mathbb{Z}_p^\times$, known as the *Teichmüller lift* of x, for which $x \equiv t \pmod{p}$ and $t^p = t$. Note that t depends only on the congruence class of x modulo p.

3 *p*-adic numbers in FLINT

FLINT (Fast Library for Number Theory) [5] is a C library implementing a wide variety of number theoretical features. A number of these features, including ball arithmetic and certain computations over $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{F}_{p^k}$, are already used by the Macaulay2 engine and interpreter.

FLINT's p-adic number module was written by Sebastian Pancratz. To represent $x \in \mathbb{Q}_p$, the expansion from Equation (2) is used. However, since a given $u \in \mathbb{Z}_p^\times$ may require infinitely many digits in base p to represent, it is truncated up to a given precision N. In other words, the elements $x \in \mathbb{Q}_p$ that may be represented in FLINT are those of the form $x = up^\nu$ where

$$u = \sum_{n=0}^{N-1} a_{n+\nu} p^n. \tag{3}$$

Since u has finitely many nonzero digits, we certainly have $u \in \mathbb{Z}$, and so it is sufficient to represent x using the integers u, ν, N, and p.

FLINT uses two main types in its p-adic number module. A `padic_t` object contains u, ν, and N, and a `padic_ctx_t` object contains information about p. This information is separated so that the same `padic_ctx_t` object may be used for computations involving multiple `padic_t` objects.

A variety of functions exist for performing operations on p-adic numbers in FLINT. See Table 1 for a selection.

operation	FLINT function
$-x$	`padic_neg`
x^{-1}	`padic_inv`
$x + y$	`padic_add`
$x - y$	`padic_sub`
$x \times y$	`padic_mul`
$x \div y$	`padic_div`
x^n $(n \in \mathbb{Z})$	`padic_pow_si`
$\sqrt{x}$	`padic_sqrt`
$\exp_p x$	`padic_exp`
$\log_p x$	`padic_log`
Teichmüller lift	`padic_teichmuller`
$x = y$	`padic_equal`
$x = 0$	`padic_is_zero`

Table 1. FLINT functions for operations in $\mathbb{Q}_p$

Furthermore, functions exist for converting back and forth between `padic_t` objects and `fmpz_t` and `fmpq_t` objects, which are FLINT's implementations of integers and rationals, respectively. In particular, using the natural inclusion map $\mathbb{Z} \hookrightarrow \mathbb{Q}_p$ (resp. $\mathbb{Q} \hookrightarrow \mathbb{Q}_p$), we may promote an element of $\mathbb{Z}$ (resp. $\mathbb{Q}$) to $\mathbb{Q}_p$ using `padic_set_fmpz` (resp. `padic_set_fmpq`), and if possible, we may lift an element of $\mathbb{Q}_p$ to $\mathbb{Z}$ (resp. $\mathbb{Q}$) using `padic_get_fmpz` (resp. `padic_get_fmpq`).

4 Foreign function interface and memory allocation

The *ForeignFunctions* package in Macaulay2 [6] allows users to make calls to functions in shared libraries such as FLINT without writing any C code or re-compiling the Macaulay2 binary. In particular, we may use it to call the p-adic functions outlined in Section 3.

Macaulay2 uses the Boehm-Demers-Weiser conservative garbage collector [1] to allocate and, as needed, deallocate memory. However, each of the various FLINT types used in the p-adic number module has functions for both initializing and clearing the memory used by instances of that type. For example, `padic_init` (or `padic_init2` to specify the precision) is used to initialize a `padic_t` object and `padic_clear` is used to clear it.

Therefore, after allocating the memory for a FLINT object and calling the corresponding initialization function, we must register a finalizer so that when the garbage is collected and the memory is freed, the appropriate FLINT function is also called. See Listing 1. Note that when allocating memory for a `padic_t` object, we are allocating memory for a struct that fits three `long` integers – u, ν, and N from Equation (3).[3]

```
needsPackage "ForeignFunctions"
flint = openSharedLibrary "flint"
padicInit = foreignFunction(flint, "padic_init", void, voidstar)
padicClear = foreignFunction(flint, "padic_clear", void, voidstar)
y = getMemory(3 * size long)
padicInit y
registerFinalizer(y, padicClear)
```

Listing 1: Initializing a `padic_t` object

[3] Actually, u is a `fmpz_t` object, which is either a `long` integer or, if needed, a pointer to an arbitrary precision integer.

5 Object-oriented design

Just using the foreign function interface to call FLINT functions is not sufficient for doing computations over $\mathbb{Q}_p$ in Macaulay2. It is necessary to provide a top-level interface with which users will interact and that wraps the low-level calls to FLINT via the foreign function interface.

Because of the similarities with the real and complex fields in that we are dealing with numbers with potentially infinitely many digits, let us first review how $\mathbb{R}$ and $\mathbb{C}$ are implemented in Macaulay2. Real and complex numbers are instances of the classes `RR` and `CC`, respectively. These classes are both instances of `InexactFieldFamily` but subclasses of `InexactNumber`.[4]

To parallel this, for a given prime number `p`, we let the class `QQ_p` be an instance of `PadicFieldFamily` and a subclass of `PadicNumber`. See the class diagrams in Figure 1 and Figure 2.

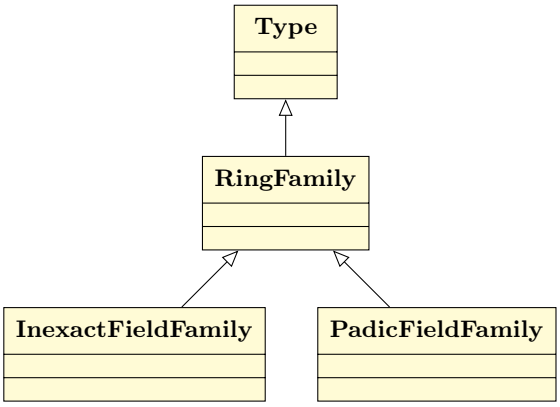

Fig. 1. Class diagram for number types

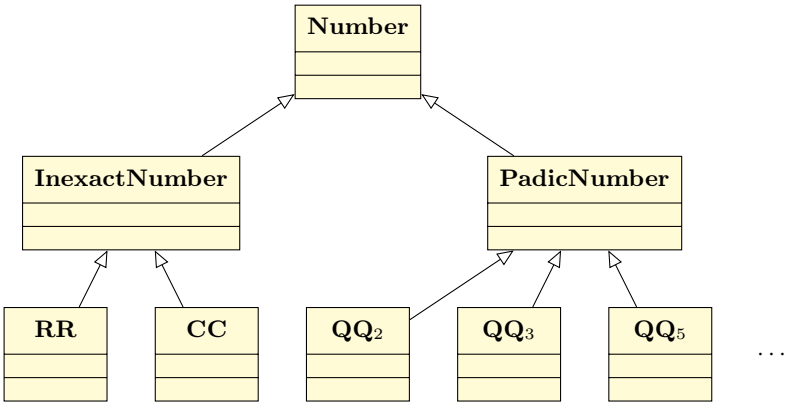

Fig. 2. Class diagram for number objects

Instances of `QQ_p` are hash tables containing pointers to the corresponding `padic_t` and `padic_ctx_t` objects. The latter are memoized to avoid duplication. Then methods are in-

[4] There also exists objects representing the fields $\mathbb{R}$ and $\mathbb{C}$ that depend on the floating-point precision used, e.g., `RR_53` and `CC_53` for double precision. We have not yet implemented analogous objects for $\mathbb{Q}_p$.

stalled on `PadicNumber` wrapping each of the foreign functions from FLINT, e.g., `PadicNumber + PadicNumber` calls `padic_add`.

The constructor methods for these classes either take a single Macaulay2 number, returning a p-adic number with the default precision of 20, or optionally an integer first argument to indicate the precision. See Listing 2.

```
i1 : needsPackage "Padic";

i2 : QQ_5(-1)

o2 = 4 + 4*5^1 + 4*5^2 + 4*5^3 + 4*5^4 + 4*5^5 + 4*5^6 + 4*5^7 + 4*5^8
     + 4*5^9 + 4*5^10 + 4*5^11 + 4*5^12 + 4*5^13 + 4*5^14 + 4*5^15 +
     4*5^16 + 4*5^17 + 4*5^18 + 4*5^19

o2 : QQ  (of precision 20)
       5

i3 : QQ_5(40, -1)

o3 = 4 + 4*5^1 + 4*5^2 + 4*5^3 + 4*5^4 + 4*5^5 + 4*5^6 + 4*5^7 + 4*5^8
     + 4*5^9 + 4*5^10 + 4*5^11 + 4*5^12 + 4*5^13 + 4*5^14 + 4*5^15 +
     4*5^16 + 4*5^17 + 4*5^18 + 4*5^19 + 4*5^20 + 4*5^21 + 4*5^22 +
     4*5^23 + 4*5^24 + 4*5^25 + 4*5^26 + 4*5^27 + 4*5^28 + 4*5^29 +
     4*5^30 + 4*5^31 + 4*5^32 + 4*5^33 + 4*5^34 + 4*5^35 + 4*5^36 +
     4*5^37 + 4*5^38 + 4*5^39

o3 : QQ  (of precision 40)
       5
```

Listing 2: Constructing $-1 \in \mathbb{Q}_5$ with different precisions

6 Example: Hensel's lemma

We close with an example, showing how to use our Macaulay2 package to find roots of polynomials over $\mathbb{Z}_p$. We first recall Hensel's lemma, a fundamental result in p-adic analysis.

Lemma 1 (Hensel [4]). *Suppose $f \in \mathbb{Z}_p[x]$. If there exists $\alpha_1 \in \mathbb{Z}_p$ for which $f(\alpha_1) \equiv 0 \pmod{p}$ and $f'(\alpha_1) \not\equiv 0 \pmod{p}$, then there exists $\alpha \in \mathbb{Z}_p$ satisfying $\alpha_1 \equiv \alpha \pmod{p}$ and $f(\alpha) = 0$.*

Furthermore, α may be constructed by defining the recursive sequence given by Newton's method,

$$\alpha_{n+1} = \alpha_n - \frac{f(\alpha_n)}{f'(\alpha_n)}.$$

Then $f(\alpha_n) \equiv 0 \pmod{p^n}$ for all n and $\alpha = \lim_{n\to\infty} \alpha_n$, where the limit is taken with respect to $|\cdot|_p$.

As with the Teichmüller lift, any choice of α_1 in the same congruence class modulo p will work, and in particular, it is natural to choose $\alpha_1 \in \{0, \ldots, p-1\}$ so that $\alpha = \alpha_1 \cdot p^0 + \cdots$ (see Equation (1)).

We would like to use Macaulay2 to compute the cube root of 2 in $\mathbb{Z}_5$ by finding a root of $x^3 - 2 \in \mathbb{Z}_5[x]$. Since $3^3 - 2 = 25 \equiv 0 \pmod{5}$ and $3 \cdot 3^2 = 27 \not\equiv 0 \pmod{5}$, we begin with $\alpha_1 = 3$. We then iterate Newton's method until it converges (up to the 5^{19} term, as FLINT's default precision for p-adics is $N = 20$). See Listing 3.

```
i1 : needsPackage "Padic";

i2 : newton = x -> x - (x^3 - 2)/(3*x^2);

i3 : alpha = QQ_5 3

o3 = 3

o3 : QQ  (of precision 20)
       5

i4 : while alpha != (alpha = newton alpha) do null

i5 : alpha

o5 = 3 + 2*5^2 + 2*5^3 + 3*5^4 + 1*5^5 + 4*5^6 + 2*5^8 + 3*5^9 +
     4*5^12 + 4*5^14 + 4*5^15 + 3*5^16 + 1*5^17 + 1*5^18 + 2*5^19

o5 : QQ  (of precision 20)
       5

i6 : alpha^3 -- check

o6 = 2

o6 : QQ  (of precision 20)
       5
```

Listing 3: Computing the cube root of 2 in $\mathbb{Z}_5$ using Macaulay2

Acknowledgments. The author thanks the anonymous referees for their helpful comments. This work was supported by a grant from the Simons Foundation International (SFI-MPS-SSRFA-00012695).

Disclosure of Interests. The author has no competing interests to declare that are relevant to the content of this article.

References

1. Boehm, H.J., Weiser, M.: Garbage collection in an uncooperative environment. Software: Practice and Experience **18**(9), 807–820 (1988)
2. Gouvêa, F.Q.: p-adic Numbers, An Introduction. Universitext, Springer-Verlag, Berlin, second edn. (1997), https://doi.org/10.1007/978-3-642-59058-0
3. Grayson, D.R., Stillman, M.E.: Macaulay2, a software system for research in algebraic geometry. Available at https://macaulay2.com/
4. Hensel, K.: Neue Grundlagen der Arithmetik. J. Reine Angew. Math. **127**, 51–84 (1904), https://doi.org/10.1515/crll.1904.127.51
5. The FLINT team: FLINT: Fast Library for Number Theory (2025), version 3.4.0, https://flintlib.org
6. Torrance, D.A.: ForeignFunctions package for Macaulay2. J. Softw. Algebra Geom. **15**(1), 1–9 (2025), https://doi.org/10.2140/jsag.2025.15.1