A Brief Introduction to NZMATH

ICHIKI Shingo, OGURA Naoki, KOIZUMI Masahiro,
NISHIMOTO Keiichiro, TANAKA Satoru, MATSUI Tetsushi,
UCHIYAMA Shigenori, NAKAMULA Ken
Tokyo Metropolitan University

Abstract

NZMATH is a system for number theory which is being developed at Tokyo Metropolitan University. From the viewpoint of a NZMATH user, we show how you can use this system and what you can do with it. Moreover, we demonstrate how to install, execute commands and link NZMATH with other softwares.

1 Introduction

NZMATH[5] is a number theory oriented calculation system mainly developed by the Nakamula laboratory at Tokyo Metropolitan University. It is freely available and distributed under the BSD license at `http://tnt.math.metro-u.ac.jp/nzmath/`. The most distinctive feature of NZMATH is that it is written entirely using a scripting language called Python. Although NZMATH is at an early stage of development, it holds enormous potential.

This paper is organized as follows. In section 2, we will discuss NZMATH features and advantages against other similar systems. In section 3, we will show how to use NZMATH for beginners.

2 The Advantage of NZMATH

In this section, we will provide a detailed discussion on the advantages of NZMATH compared to other similar systems.

2.1 Open Source Software

Computational algebra systems, such as Maple[7], Mathematica[8], and Magma[9] are fare-paying systems. These non-free systems are not distributed with source codes. In this regard, users cannot modify such systems. It narrows these system's potentials for users not to take part in developing it. NZMATH, on the other hand, is an open-source software and the source code is openly available. Furthermore, NZMATH is distributed under the BSD license. BSD license claims as-is and redistribution or commercial use are permitted provided that these packages retain the copyright notice.

2.2 Speed of Development

We took over developing of SIMATH[6], which was developed under the leadership of Prof.Zimmer at Saarlandes University in Germany. However, it costs a lot of time and efforts to develop these system. Almost all systems including SIMATH are implemented in C or C++ for execution speed, but we have to take the time to work memory management, construction of an interactive interpreter, preparation for multiple precision package and so on. In this regard, we chose Python which is a modern programming language. Python provides automatic memory management, a sophisticated interpreter and many useful packages. We can concentrate on development of mathematical matters by using Python.

2.3 Bridging the Gap between Users and Developers

KANT/KASH[10] and PARI/GP[11] are similar systems to NZMATH. But these systems have different languages for users and developers. We think the gap between languages makes evolution of systems slow. NZMATH is being developed with using Python, we bridge this gap. Users are easy to understand Python grammar and read codes written by Python. And NZMATH which is one of Python libraries works on very wide platform including UNIX/Linux, Macintosh, Windows, and so forth. Users can modify the programs and feedback to developers. Developers can absorb their thinking. Then NZMATH will progress to more flexible user-friendly system.

2.4 Link with Other Softwares

NZMATH distributed as a Python library enables us to link other Python packages with it. For example, NZMATH can use with IPython[12] which is a comfortable interactive interpreter. And it can be linked with matplotlib[13] which is a powerful graphic software. There are many libraries and packages for softwares implemented in Python. Many of these packages are freely available. Users can use NZMATH with these packages and create an unthinkable powerful system.

3 How to Use

In this section, we will illustrate installation procedures and sample sessions of NZMATH.

3.1 Installation

There are three steps for installation of NZMATH.

First, check that Python is installed in the computer. Python 2.3 or a higher version is needed for NZMATH. If you do not have a copy of Python, please install it first. Python is available from `http://www.python.org/`.

Second, download NZMATH package and expand it. It is distributed at official web site:

`http://tnt.math.metro-u.ac.jp/nzmath/download`

or at sourceforge.net:

`http://sourceforge.net/project/showfiles.php?group_id=171032`

The latest version is 0.7.0. The package can be easily extracted, depending on the operating system. For systems with recent GNU tar, type a single command below:

```
% tar xf NZMATH-0.7.0.tar.gz
```

where, `%` is a command line prompt. Or with standard tar, type a following command:

```
% gzip -cd NZMATH-0.7.0.tar.gz | tar xf -
```

Then, a subdirectory named NZMATH-0.7.0 is created.

Finally, install NZMATH to the standard python path. Usually, this translates to writing files somewhere under `/usr/lib` or `/usr/local/lib`, and thus appropriate write permission is required. Typically, type commands below:

```
% cd NZMATH-0.7.0
% su
# python setup.py install
```

We also distribute the installation packages for specific platforms. Especially, we started distributing installer for Windows in 2007. This installer enables us to install NZMATH with only three clicks.

Figure 1: Windows installer for NZMATH 0.7.0

3.2 Sample Session

NZMATH is a Python library. So, it can be used in the same way as standard python packages. We will demonstrate how to use NZMATH and briefly discuss about some modules.

Start the Python interpreter:

```
% python
Python 2.3.4 (#1, Dec 11 2007, 05:28:55)
[GCC 3.4.6 20060404 (Red Hat 3.4.6-9)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Here, '>>>' is a Python prompt. Please type

```
>>> from nzmath import *
>>>
```

With this command, the whole NZMATH modules are imported.

```
>>> gcd.gcd(350, 525)
175
>>>
```

The `gcd` is a module for the greatest common divisors of integers. The example above means that 175 is the greatest common divisor of 350 and 525.

```
>>> factor.methods.factor(175)
[(5, 2), (7, 1)]
>>>
```

The `factor` module has functions for prime factorization. The sample command above shows that 175 is factorized $5^2 \times 7$ into prime factors.

```
>>> prime.nextPrime(175)
179
>>>
```

The `prime` module provides functions related with primality. For example, the command illustrated above gives us 179 which is the smallest prime larger than 175.

```
>>> arith1.legendre(3, 7)
-1
>>>
```

The `arith1` is module for miscellaneous arithmetic functions. One command in this module is `legendre` for legendre symbol. From the example above, $(\frac{3}{7}) = -1$. So 3 is quadratic non-residue modulo 7.

There is a trivial example of Python programming with NZMATH below.

```
>>> for i in range(10):
        [combinatorial.binomial(i, j) for j in range(i+1)]
...
...
[1]
[1, 1]
[1, 2L, 1]
[1, 3L, 3L, 1]
[1, 4L, 6L, 4L, 1]
[1, 5L, 10L, 10L, 5L, 1]
[1, 6L, 15L, 20L, 15L, 6L, 1]
[1, 7L, 21L, 35L, 35L, 21L, 7L, 1]
[1, 8L, 28L, 56L, 70L, 56L, 28L, 8L, 1]
[1, 9L, 36L, 84L, 126L, 126L, 84L, 36L, 9L, 1]
>>>
```

The `combinatorial` module includes combinatorial theoretical functions. This program outputs binomial coefficient ${}_n\mathrm{C}_m$ for $n = 0, \ldots, 9$. The `range` is one of a Python function. For example, we can express [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] by the `range(10)`. The "`L`" denotes long integer type.

NZMATH also has a module for elliptic curves called `elliptic`.

```
>>> E = elliptic.ECoverFp([1,2], 37)
>>>
```

We define the elliptic curve $E: \ y^2 = x^3 + x + 2$ over finite field F_{37}.

```
>>> P, Q = E.point(), E.point()
>>> print P, Q
[1, 35] [27, 19]
>>>
```

The `point` function returns a random point on elliptic curve. Now the point $P = (1, \ 35)$ and $Q = (27, \ 19)$ are taken from $E(F_{37})$.

```
>>> E.add(P, Q)
[5, 13]
>>>
```

This means that $P + Q = (5, \ 13)$ in $E(F_{37})$.

```
>>> E.mul(3, P)
[1, 2]
>>>
```

This shows that $3P = (1, \ 2)$ in $E(F_{37})$.

NZMATH may be used with other softwares. We show an example for using NZMATH with matplotlib which is one of powerful packages drawing graphs. It draws k-scalar($k <$ order of point) multiplication points for some point in curve $y^2 = x^3 + x + 2$ over F_{37}. We put the source code for it in Appendix.

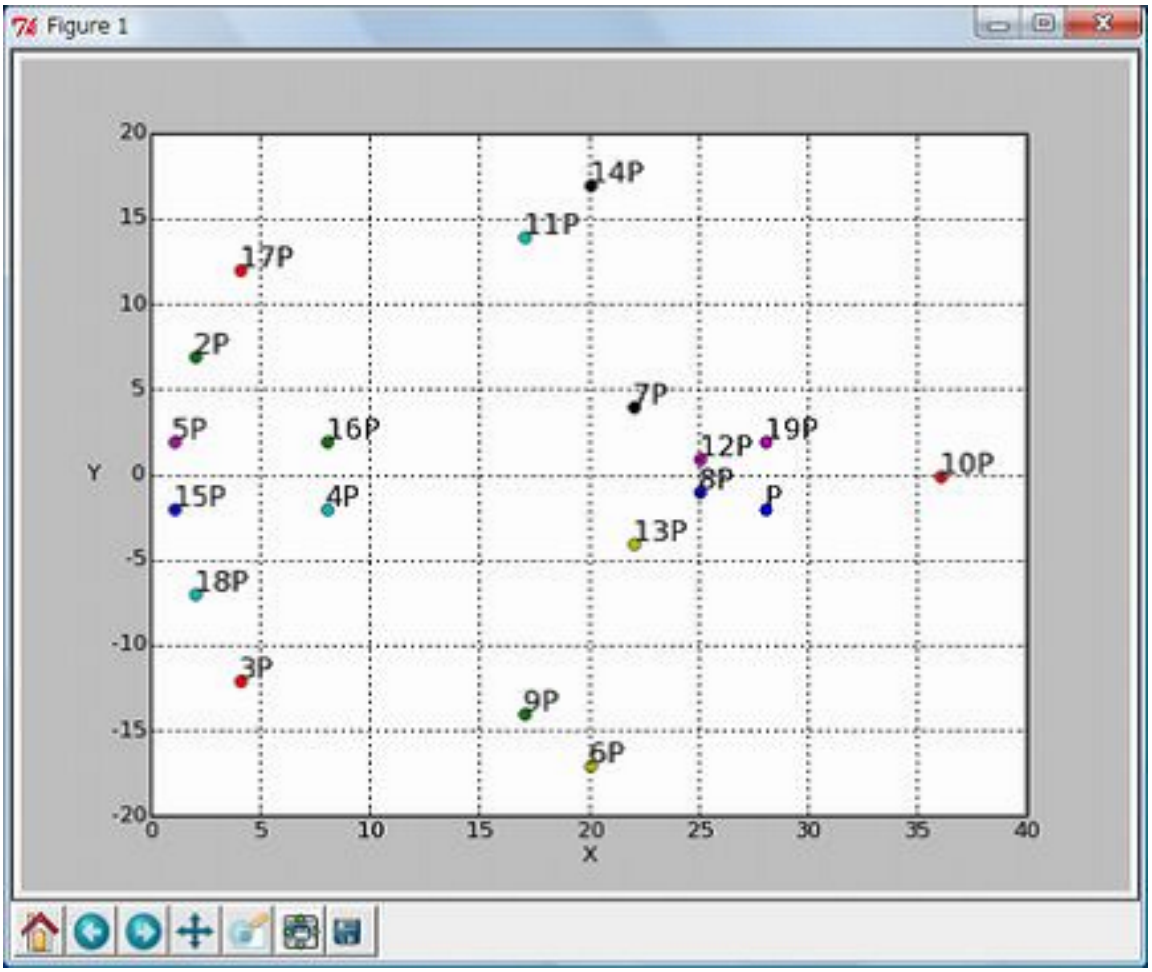

Figure 2: k-scalar multiplication in $y^2 = x^3 + x + 2$ over F_{37}

3.3 More Information

NZMATH has more than 25 modules. It has modules related with elementary number theory, combinatorial theory, solving equations, primality, factorization, multiplicative number theoretic functions, matrix, vector, polynomial, rational field, finite field, elliptic curve, and so on. NZMATH manual for users is at:

`http://tnt.math.metro-u.ac.jp/nzmath/manual/`

If you are interested in NZMATH, please visit the official website below to get more information about it.

`http://tnt.math.metro-u.ac.jp/nzmath/`

Note that NZMATH can be used even if user do not have experience writing programs in Python.

We have a mailing list called nzmath-user for NZMATH users. To join this mailing list, send a mail to

`nzmath-user-request@tnt.math.metro-u.ac.jp`

only with a line “subscribe” in the message. Then the address from which you send the mail will be added on the nzmath-user mailing list.

4 Conclusion

We have been developing NZMATH for four years. NZMATH is still at an early stage, but possibilities of NZMATH are pretty much unlimited. As the number of NZMATH users and developers increase, NZMATH becomes better. We will continue promoting NZMATH based on our philosophy and creating new NZMATH features.

References

[1] Tanaka S., Nishimoto K., MATSUI T., Uchiyama S., and NAKAMULA K. "Status and issues on development NZMATH." *AC2007 presentation* (2007)

[2] Matsui T. "NZMATH: Past and Future of the Development." *AC2005 proceedings*, `ftp://tnt.math.metro-u.ac.jp/pub/ac05/Matsui/nzmath.pdf` (2005)

[3] Matsui T. "Development of computational number theory system by a scripting language."(Japanese) *Computer Algebra . Design of Algorithms, Implementations and Applications*, RIMS Kokyuroku, No.1395 pp.144-149, Kyoto University (2004)

[4] Nakamula K. and Matsui T. "Developing a system for number theory by script language — Announcement of the release of NZMATH 0.1.1" *Algorithms and Number Theory*, Dagstuhl, `http://tnt.math.metro-u.ac.jp/~nakamula/talk/dag2004-s.pdf` (2004)

[5] NZMATH, `http://tnt.math.metro-u.ac.jp/nzmath/`

[6] SIMATH, `http://tnt.math.metro-u.ac.jp/simath/`

[7] Maple, `http://www.maplesoft.com/`

[8] Mathematica, `http://www.wolfram.com/products/mathematica/`

[9] Magma, `http://magma.maths.usyd.edu.au/magma/`

[10] KANT/KASH, `http://www.math.tu-berlin.de/~kant/kash.html`

[11] PARI/GP, `http://pari.math.u-bordeaux.fr/`

[12] IPython, `http://ipython.scipy.org/`

[13] matplotlib, `http://matplotlib.sourceforge.net/`

Appendix A Demo Code

The Python source code with nzmath and matplotlib for the demonstration which draws k-scalar multiplication points for some point in elliptic curve $y^2 = x^3 + x + 2$ over F_{37} is here.

```
import nzmath.elliptic
from pylab import *

p = 37
A = 1 # A>0
B = 2 # B>0

E = nzmath.elliptic.ECoverFp([A, B], p)
while True:
    P = E.point()
    if E.pointorder(P) > E.order()//3: # point whose order is large
        break

x_y = []
for i in range(E.pointorder(P)):
    pnt = E.mul(i, P)
    if len(pnt) > 1: # not point at infinity
        if pnt[1] > p//2:
            ele = [ [pnt[0]], [pnt[1] - p] ]
        else:
            ele = [ [pnt[0]], [pnt[1]] ]
        if i == 1:
            x_y.append(ele + ["P"])
        else:
            x_y.append(ele + [str(i) + "P"])

figure()
grid(True)
axis([-p//2, p//2 + 1, 0, p])
xlabel("X")
ylabel("Y", rotation=0)

for x, y, lab in x_y:
    plot(x, y, 'o')
    text(x[0], y[0], lab, size=15, weight='bold')
show()
```