Indian Journal of Cryptography and Network Security (IJCNS)
ISSN: 2582-9238 (Online), Volume-3 Issue-2, November 2023

Software Support for Arbitrary Precision Arithmetic in Programming Languages

Kannan Balasubramanian

Abstract: ***Arbitrary precision arithmetic, also known as bignum arithmetic, is a computational technique that allows programmers to perform arithmetic operations on numbers with significantly higher precision and magnitude than what is typically supported by the built-in numerical data types in programming languages. This technique is especially useful when working with extremely large or extremely precise numbers, such as in cryptography or scientific computations. Arbitrary precision arithmetic has many applications in the areas of Cryptography, Numerical Computation, Statistical Analysis, and High Precision measurements. For example, the calculation of the modulus in the RSA algorithm involves numbers with 1024-bit numbers and higher. An arithmetic calculation involving Multiplication and exponentiation of such numbers using Modulo arithmetic cannot be easily carried out in the existing programming Languages unless special software is provided. We can calculate the mathematical constant Pi to many thousand decimal places using the support provided in Programming Languages. Many programming languages provide built-in support for libraries for arbitrary precision arithmetic. We discuss the support provided in C/C++, Java and Python Languages with examples. Besides Programming Languages, Toolkits like Matlab and Sagemath also are used for scientific computation and special software support provided in these toolkits can enable arbitrary precision arithmetic. Most Programming Languages have support for floating-point arithmetic. We also discuss how arbitrary precision floating point arithmetic. can be supported in C/C++, Java, and Python. In addition, we discuss support for arbitrary precision integer arithmetic in Ruby, Javascript and Matlab and support for arbitrary precision floating point arithmetic in the Perl Language. Finally, we provide an example of computing the constant Pi to many decimal places using the sagemath tool.***

Keywords: Numerical Computation, RSA Algorithm, Arbitrary Precision Arithmetic, Floating-Point Arithmetic, Programming Languages, Programming Language Libraries, Pi Calculation

I. INTRODUCTION

Arbitrary precision arithmetic, also known as *bignum* arithmetic, refers to performing calculations with numbers of arbitrary precision, rather than relying on fixed-size numeric types. This approach allows handling extremely large or very small numbers that may exceed the limits of standard numeric types.

Manuscript received on 01 November 2023 | Revised Manuscript received on 09 November 2023 | Manuscript Accepted on 15 November 2023 | Manuscript published on 30 November 2023.

* Correspondence Author(s)
Dr. Kannan Balasubramanian*, Professor, School of Computing, SASTRA University, Thanjavur (TamilNadu), India. E-mail: kannanb@cse.sastra.edu, ORCID ID: 0000-0003-1134-0345

There are numerous applications for arbitrary precision arithmetic in scientific computations, including cryptography, numerical analysis, symbolic computation, mathematical research, financial modeling, statistical analysis, plasma physics, astrophysics, and high-precision measurements. Arbitrary precision arithmetic plays a vital role in scientific computations, enabling accurate and reliable calculations with numbers of arbitrary precision.

1. Cryptography: Asymmetric encryption algorithms such as RSA rely heavily on large prime numbers. Arbitrary precision arithmetic enables the generation and manipulation of massive integers used in cryptographic key generation, encryption, and decryption.
2. Numerical analysis: In scientific simulations and mathematical models, computations often involve high-precision calculations [1]. Arbitrary precision arithmetic allows accurate and reliable analysis of numerical algorithms, minimizes rounding errors, and ensures precise results.
3. Symbolic computation: Arbitrary precision arithmetic is fundamental in symbolic computation systems like computer algebra systems (CAS). These systems manipulate expressions symbolically rather than numerically, requiring exact arithmetic for simplification, factorization, equation solving, and differentiation.
4. Mathematical research: Researchers deal with complex calculations involving large numbers, factorization, number theory, and algebraic operations. Arbitrary precision arithmetic enables precise calculations and helps uncover patterns, relationships, and properties in mathematical structures.
5. Financial modeling: Financial calculations frequently involve precise decimal calculations, with precision requirements beyond the capabilities of fixed-size numeric types. Arbitrary precision arithmetic allows accurate computations in areas such as interest rate calculations, financial derivatives pricing, risk management, and simulation-based forecasting.
6. Statistical analysis: Statistical computations often require high precision when working with very small probabilities or extreme values[2][6][7][8]. Computing logarithms, exponentials, and other statistical functions with arbitrary precision ensures accurate analysis and avoids the potential loss of significant digits.
7. Plasma physics and astrophysics: Research in plasma physics and astrophysics involves complex calculations of celestial bodies, event horizon calculations, particle simulations, and gravitational interactions. Arbitrary precision arithmetic is essential for obtaining accurate results and understanding these intricate phenomena.

8. High-precision measurements: Precision measurements in various scientific fields, including physics, engineering, and chemistry, require precise calculations and accurate representation of measurement uncertainties. Arbitrary precision arithmetic allows reliable propagation of uncertainties and calculation of uncertainty budgets[3].
Programming Languages like C/C++, Java and Python do not directly support arbitrary precision except by using special libraries. This article reviews the software support provided by special Libraries in Programming Languages.
Many programming languages provide support for arbitrary precision arithmetic. Some of the programming languages that provide software support for bignum arithmetic are:

A. Python:

Python, a widely used and beginner-friendly language, offers a native library called `decimal` for arbitrary precision arithmetic. This library allows programmers to work with decimal numbers with customizable precision and rounding modes. Additionally, Python's `math` module provides several functions for performing mathematical operations with arbitrary precision.

B. Java:

Java provides the `BigInteger` and `BigDecimal` classes in its standard library, which support arbitrary precision arithmetic for integers and decimals, respectively. These classes offer methods for performing arithmetic operations, comparisons, and conversions on numbers with high precision.

C. C/C++:

Though C and C++ do not have built-in support for arbitrary precision arithmetic, several well-known libraries are available to handle this. One popular library is GNU MP (GMP), which provides a wide range of functions for performing bignum arithmetic, including efficient algorithms for common operations like addition, subtraction, multiplication, and division.

D. Ruby:

Ruby includes the `BigDecimal` class in its standard library to handle arbitrary precision decimal arithmetic. This class allows programmers to perform calculations with high precision and control over rounding behavior. Ruby also supports the `bigdecimal` gem, which provides additional features and flexibility.

E. JavaScript:

JavaScript, primarily used for web development, doesn't have built-in support for arbitrary precision arithmetic. However, there are various third-party libraries available, such as `bignumber.js` and `decimal.js`, that provide support for performing bignum operations in JavaScript.

F. MATLAB:

MATLAB, widely used in scientific and engineering applications, offers the Symbolic Math Toolbox, which provides the `vpa` (Variable-precision arithmetic) function. This function allows users to perform computations with arbitrary precision numbers, including floating-point and symbolic calculations.

These are just a few examples of popular programming languages and their support for arbitrary precision arithmetic. Other languages, such as Perl, Haskell, and many more, also provide libraries or built-in features for bignum arithmetic. When working with large or precise numbers, knowing and utilizing these capabilities can significantly enhance the accuracy and efficiency of your code.

II. ARBITRARY PRECISION ARITHMETIC

Arbitrary precision arithmetic, also known as bignum arithmetic, is a programming technique that enables the manipulation and computation of numbers with an arbitrary number of digits or precision. In contrast to fixed-size numeric types, like integers or floating-point numbers, which have a limited range and precision, arbitrary precision arithmetic allows for computations involving extremely large or small numbers without loss of accuracy. In traditional programming languages, such as C or Java, fixed-size numeric types have limitations on the number of bits allocated for representing numbers. For example, a 32-bit integer can represent values from -2,147,483,648 to 2,147,483,647, while a double-precision floating-point number has a limited number of significant digits. When calculations involve numbers that exceed these limits, the result may overflow, leading to incorrect or unpredictable results. Arbitrary precision arithmetic libraries or data types address these limitations by dynamically allocating memory to accommodate numbers with any number of digits. These libraries typically provide functions or methods for performing arithmetic operations like addition, subtraction, multiplication, and division, as well as comparison and conversion functions.

The precision of arbitrary precision arithmetic is limited only by the amount of available memory in the computer system. Therefore, computations involving extremely large numbers or numbers with many decimal places can be accurately performed. However, it's important to note that arbitrary precision arithmetic operations can be slower compared to fixed-size types due to the extra overhead involved in dynamically allocating and manipulating memory. Arbitrary precision arithmetic is particularly useful in various domains, including cryptography, financial calculations, and scientific simulations, where high precision and accuracy are required. It enables programmers to handle numbers of arbitrary size and precision, ensuring accurate results even in complex calculations. In this section, we describe the arbitrary precision support in programming languages.

A. Software Support in Python for Arbitrary Precision Arithmetic.

Arbitrary precision arithmetic in Python is often implemented using libraries like "gmpy2," "mpmath," or "decimal." These libraries allow you to perform arithmetic operations with numbers of arbitrary precision, which means you can work with very large or very small numbers without losing precision.

Here's an overview of how to use these libraries: Gmpy2 is a Python interface to the GMP (GNU Multiple Precision) library, which is a high-performance arbitrary precision arithmetic library. You need to install the gmpy2 library, which can be done using the pip. Command. Figure1 provides an example of using gmpy2 to perform arbitrary precision arithmetic:

```
import gmpy2
a = gmpy2.mpz(12345678901234567890123456789012345678901234567890)
b = gmpy2.mpz(98765432109876543210987654321098765432109876543210)
result = gmpy2.add(a, b)
print(result)
```

Figure 1. Python Program using Gmpy2 to Perform Arbitrary Precision Arithmetic

Mpmath is a Python library for arbitrary-precision floating-point arithmetic which can be installed using the pip command Figure 2. Provides an example of using mpmath to perform arbitrary precision arithmetic.

```
from mpmath import mp
mp.dps = 50  # Set the number of decimal places
a = mp.mpf("12345678901234567890123456789012345678901234567890")
b = mp.mpf("98765432109876543210987654321098765432109876543210")
result = a + b
print(result)
```

Figure 2. Python Program using Mpmath to Perform Arbitrary Precision Arithmetic

The decimal module is part of Python's standard library, and it provides arbitrary precision decimal arithmetic. It can be used without installing any additional libraries: Figure 3. provides an example of how to use the Decimal for arbitrary precision arithmetic.

```
from decimal import Decimal
a = Decimal("12345678901234567890123456789012345678901234567890")
b = Decimal("98765432109876543210987654321098765432109876543210")
result = a + b
print(result)
```

Figure 3. Python Program using Decimal to Perform Arbitrary Precision Arithmetic

Each of these libraries has its unique features and use cases. The choice of which one to use depends on the specific requirements and preferences.

B. Software support in Java for Arbitrary Precision Arithmetic.

In Java, if you need to perform arbitrary precision arithmetic, you can use libraries that provide support for working with arbitrary precision integers and floating-point numbers. BigInteger and BigDecimal (Java Standard Library): Java's standard library provides the BigInteger and BigDecimal classes for arbitrary precision integer and floating-point arithmetic, respectively. You can perform arithmetic operations using these classes without the need for external libraries. Figure 4 provides an example using BigInteger to perform arbitrary precision arithmetic. Figure 5 provides an example using BigDecimal to perform arbitrary precision arithmetic

```
import java.math.BigDecimal;
BigDecimal a = new BigDecimal("0.1");
BigDecimal b = new BigDecimal("0.2");
BigDecimal result = a.add(b);
System.out.println(result);
```

Figure 4. Java Program using Biginteger to Perform Arbitrary Precision Arithmetic

```
import java.math.BigInteger;
BigInteger a = new BigInteger("12345678901234567890123456789O");
BigInteger b = new BigInteger("98765432109876543210987654321O");
BigInteger result = a.add(b);
System.out.println(result);
```

Figure 5. Java Program using Bigdecimal to Perform Arbitrary Precision Arithmetic

C. Software support in C/C++ for Arbitrary Precision Arithmetic.

In the C programming language, performing arbitrary precision arithmetic often requires the use of external libraries, as C's built-in data types have fixed sizes and cannot handle arbitrarily large numbers. There are several libraries available for arbitrary precision arithmetic in C, and one of the most popular and widely used libraries is the GNU Multiple Precision Arithmetic Library (GMP). The following steps are required to use the GMP library:

1. Install GMP: For example on a Linux system, you can use the following command
   sudo apt-get install libgmp-dev
2. Include the GMP header.
   #include<gmp.h>
3. When you compile the C program, you need to link it with the gmp library.
   gcc my_program.c -o my_program -lgmp

Remember that GMP offers a wide range of functions for arithmetic operations, comparisons, conversions, and more. You can refer to the GMP documentation for more information on using the library: https://gmplib.org/manual/. The GMP library can also used in the C Besides GMP, there are other libraries like MPFR (Multiple Precision Floating-Point Reliable) for arbitrary-precision floating-point arithmetic and FLINT (Fast Library for Number Theory) for number theory operations in C. The choice of library depends on your specific needs and preferences. Figure 6 provides an example of a C program that uses the GMP library.

```
#include <stdio.h>
#include <gmp.h>
int main() {
    mpz_t a, b, result;
    mpz_init(a);
    mpz_init(b);
    mpz_init(result);
    mpz_set_str(a, "12345678901234567890123456789O1234567890", 10);
    mpz_set_str(b, "98765432109876543210987654321O9876543210", 10);
    mpz_add(result, a, b);
    gmp_printf("Result: %Zd\n", result);
    mpz_clear(a);
    mpz_clear(b);
    mpz_clear(result);
    return 0;
}
```

Figure 6. C Program using GMP Program to Perform Arbitrary Precision Arithmetic

III. ARBITRARY PRECISION FLOATING POINT ARITHMETIC

Arbitrary precision floating-point arithmetic, also known as arbitrary precision arithmetic or arbitrary-precision numbers, is a way of performing mathematical operations with numbers of extremely high precision. This is particularly useful in scientific and engineering applications where standard floating-point types, such as float or double, do not provide enough precision. Here are some programming languages and libraries that provide support for arbitrary precision floating-point arithmetic:

1. Python: Python's decimal module is a standard library module that provides arbitrary precision decimal arithmetic. You can use the Decimal class to perform calculations with a high degree of precision. Figure 7 provides an example of how arbitrary precision floating point arithmetic can be used in Python.

```
from decimal import Decimal, getcontext
getcontext().prec = 50  # Set the precision to 50 decimal places
x = Decimal('0.1')
y = Decimal('0.2')
result = x + y
```

Figure 7. An Example Program in Python Showing Arbitrary Precision Floating Point Arithmetic

2. Java: The Java platform provides the BigDecimal class in its standard library. It allows you to perform arbitrary precision decimal arithmetic. Figure 5 provides an example of using BigDecimal for Floating Point arithmetic
3. C/C++:The GNU Multiple Precision Arithmetic Library (GMP) is a popular library for performing arbitrary precision arithmetic, including arbitrary-precision floating-point arithmetic. It provides data types like mpf_t for arbitrary precision floating-point numbers.
4. R: R is a programming language and environment designed for statistical computing and graphics. It has a package called "gmp" that provides support for arbitrary precision arithmetic, including arbitrary precision floating-point numbers.
5. Perl: Perl has the bignum module that allows you to perform arbitrary precision arithmetic. Figure 8. shows an example of using the bignum module in Perl.

```
use bignum;
my $x = 0.1;
my $y = 0.2;
my $result = $x + $y;
```

Figure 8. An Example in Perl for Arbitrary Precision Floating Point Arithmetic

IV. AN EXAMPLE OF ARBITRARY FLOATING-POINT ARITHMETIC

SageMath[1] is a powerful open-source mathematics software system that can perform arbitrary precision arithmetic using its built-in support for symbolic computation and numerical computations. Figure 9 shows an example of performing arbitrary precision arithmetic in SageMath:

```
# Create arbitrary precision real numbers
a = RealNumber('1.234567890123456789', precision=100)
b = RealNumber('2.345678901234567890', precision=100)
# Perform arithmetic operations with arbitrary precision
sum_ab = a + b
product_ab = a * b
# Display the results
print("Sum: ", sum_ab)
print("Product: ", product_ab)
```

Figure 9. An Example Program for Sagemath for Arbitrary Precision Floating Point Arithmetic

You can adjust the precision by changing the precision argument when creating the Real Number objects. Sage Math [4] can handle a wide range of mathematical operations using arbitrary precision arithmetic, making it a useful tool for high-precision calculations. In this example, we'll calculate the value of π (pi) to a large number of decimal places. SageMath is well-suited for arbitrary precision arithmetic, and it can calculate π to a high degree of accuracy.[5]:

```
from mpmath import mp
# Set the desired precision (number of decimal places)
mp.dps = 1000  # You can increase this number for even more precision
# Calculate pi
pi = mp.pi
# Print the calculated value of pi
print(pi)
```

Figure 10. A Program to Calculate pi to 1000 Decimal Places in Sagemath

V. RESULTS AND DISCUSSION

The above C/C++, Java and Python programs could be run using any compiler. The results are not provided here for consideration of brevity. However the output from the program to calculate Pi is shown in Figure 11.

VI. CONCLUSION

Many Scientific applications require integers and real numbers of arbitrary precision. This paper investigated the software support provided in the form of libraries in many Programming Languages and in mathematical tools like Matlab and Sagemath.

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620
8998628034825342117067982148086513282306647093844609550582231725359408128481117
4502841027019385211055596446229489549303819644288109756659334461284756482337867
8316527120190914564856692346034861045432664821339360726024914127372458700660631
5588174881520920962829254091715364367892590360011330530548820466521384146951941
5116094330572703657595919530921861173819326117931051185480744623799627495673518
8575272489122793818301194912983367336244065664308602139494639522473719070217986
0943702770539217176293176752384674818467669405132000568127145263560827785771342
7577896091736371787214684409012249534301465495853710507922796892589235420199561
1212902196086403441815981362977477130996051870721134999999837297804995105973173
2816096318595024459455346908302642522308253344685035261931188171010003137838752
8865875332083814206171776691473035982534904287554687311595628638823537875937519
5778185778053217122680661300192787661119590921642019 9
```

Figure 11. The Value of Pi is Generated to 1000 Decimal Places from Sagemath

DECLARATION STATEMENT

Funding	No, I did not receive.
Conflicts of Interest	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consent to participate with evidence.
Availability of Data and Material	Not relevant.
Authors Contributions	I am only the sole author of the article.

REFERENCES

1. Krougly, Z.L., Jeffrey D.J. & Tsarapkina, D., Software Implementation of Numerical Algorithms in Arbitrary Precision, 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 10.1109/SYNASC.2013.25, 2013.
2. Cheng, Y & Cheng, G, A Unified Root Algorithm in Bignum Arithmetic, IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference, 10.1109/IAEAC.2018.857778, 2018.
3. Samuel, Vivien, Parallel Integer Multiplication, 30th Euromicro International Conference On Parallel, Distributed and Network-based Processing(PDP), 10.1109/PDP55904.2022.00024, 2022.
4. www.sagemath.org, Free Open-source Mathematics Software System
5. www.i4cy.com Calculating Pi to a high precision.
6. Aliff, M., Hamdi, A. H., Yusof, I., & Samsiah, N. (2019). Development of Control System for Dual NC Machine with Single 6-Axis Robot. In International Journal of Innovative Technology and Exploring Engineering (Vol. 9, Issue 2, pp. 4505–4511). https://doi.org/10.35940/ijitee.b9017.129219
7. Novel Space Efficient Indices for Kannada Text: VKTPY Trie Family. (2019). In International Journal of Engineering and Advanced Technology (Vol. 9, Issue 1, pp. 6312–6320). https://doi.org/10.35940/ijeat.a1985.109119
8. An Efficient Exchanged Hyper Cube for Parallel and Distributed Network. (2019). In International Journal of Recent Technology and Engineering (Vol. 8, Issue 2S10, pp. 821–829).. https://doi.org/10.35940/ijrte.b1150.0982s1019
9. Muthukrishnan, Dr. R., & Prakash, N. U. (2023). Validate Model Endorsed for Support Vector Machine Alignment with Kernel Function and Depth Concept to Get Superlative Accurateness. In International Journal of Basic Sciences and Applied Computing (Vol. 9, Issue 7, pp. 1–5). https://doi.org/10.35940/ijbsac.g0486.039723

AUTHOR PROFILE

Dr. Kannan Balasubramanian, is currently working as Professor in the School of Computing, Sastra University, Thanjavur. He received his M. Sc. (Tech) degree in Computer Science from BITS Pilani in 1989 and M. Tech degree in Computer Science and Engineering from IIT Bombay in 1991 and Ph. D degree in Computer Science from UCLA in 1999. He has worked on the areas of multiple access protocols for optical WDM networks and scheduling algorithms for input queued switches focussing on simulation of networks and network switches. He has published two books with IGI-Global and has published in many International Journals and Conferences. His areas of Interest are Computer Networks, Cryptography and Network Security, Cyber Security and Machine Learning for Cryptography.