EZMath Reference Manual

Yi Wang
yw2580@columbia.edu

Piaoyang Cui
pc2618@columbia.edu

Shangjin Zhang
sz2425@columbia.edu

Zhejiao Chen
zc2291@columbia.edu

Contents

1. Preface

Complex mathematical operations and representations are always highly demanded for scientific programming. When creating topnotch academic papers, LaTeX, a markup language to typeset document, is often used to prettify mathematical expressions and the overall layout. By adopting syntax from LaTeX, user can easily type complicated mathematical equations for calculation purpose. Thus, we propose a new programming language called EZMath written completely in LaTeX syntax.

The targeted usage of this language can be described in the following scenario which is also illustrated in Figure 1: A topnotch mathematic paper involving a substantial amount of complicated math expressions and functions along with text is written in LaTeX syntax(paper.tex in Figure 1), thus it can be compiled by LaTeX compiler to a beautifully and smoothly typed pdf file(paper.pdf). Additionally, taking in the same source file paper.tex, compile it through the EZMath compiler, the output file is a C++ source file(paper.cpp) which can further be compiled be a C++ compiler and generates an executable file. The C++ source file translates every valid formula defination into a function while the main() function directs a complete report of the EZMath compiling process to an output file report.tex which is in LaTeX syntax. The report.tex can be compiled by LaTeX compiler to report.pdf.

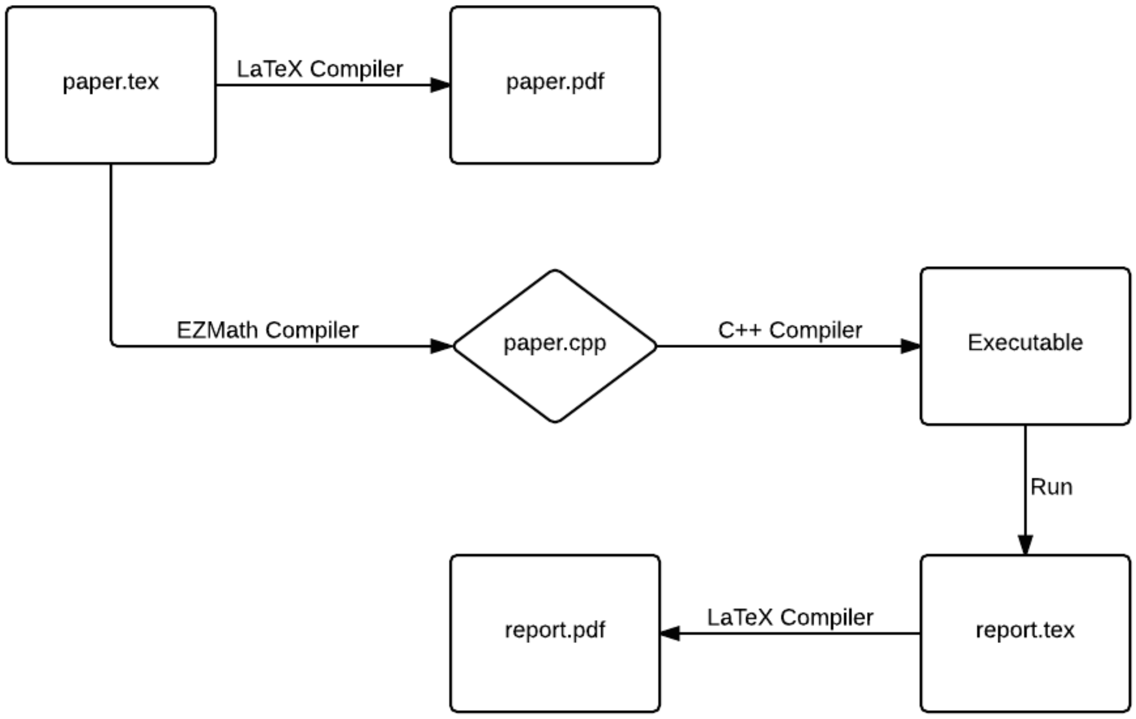

Figure 1. Relationships

2. Lexical Elements

This chapter describes the lexical elements that make up EZMath source code after preprocessing. These elements are called tokens. There are five types of tokens: identifiers, keywords, constants, operators, and separators. White space, sometimes required to separate tokens, is also described in this chapter.

2.1. Identifiers

An identifier is a sequence of letters and digits; the first character must be alphabetic. Upper and lower case letters are considered different. Characters besides letters and digits are not allowed in identifiers, including underscore (_) and hyphen (-).

Declaring two identifiers with the same literal or changing the definition of a previously declared identifiers is not allowed. Compiler should report error on such attempts.

2.2. Keywords

The following identifiers are reserved for use as keywords, and may not be used otherwise:

e	The natural logarithm approximately equals to 2.718281828.
PI	π approximately equal to 3.1415926536.

Please follow other keywords in LaTeX reference manual.

2.3. Constants

There're several kinds of constants, as fllows:

2.3.1. Integer Constants. An integer constant is a sequence of digits. An integer is taken to be octal if it begins with 0, decimal otherwise. The digits 8 and 9 have octal value 10 and 11 respectively.

2.3.2. Real Number Constants. Real Number constants are numerical literals in the code. The type of the constants is double floating-point number. A number constant consists of an integer part, a decimal part, a fraction part, an e or E, an optionally signed integer exponent. The integer and fraction parts both consist of a sequence of digits. Either the integer part, or the fraction part (not both) may be missing; either the decimal point or the e and the exponent (not both) may be missing. Some examples:

15   15.   0.5e-15   .3e+2   .2   1e5

2.3.3. Matrix Constants. Matrix constants are matrix literals in the code. The type of the elements of the matrix is number constant (defined in Section 3.3). Matrix constant is initialized by LaTeX matrix grammer.

```
\begin{bmatrix}...\end{bmatrix}
```

Array is a special kind of matrix. Some examples:

$$\begin{bmatrix} 10.5 & 20.2 & 30.5 \end{bmatrix} \quad \begin{bmatrix} 5.2 \\ 6.1 \\ 7.3 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

2.4. Operators

2.4.1. Basic Arithmetic Operators. EZMath supports plus, minus, multiply and divide (+, -, *, /) as basic arithmetic operators. These operations follow the conventions in languages like C and Java. Basic arithmetic operators can be applied between double-typed variables or number constants. The result will be a double-type value.

2.4.2. Matrix Operators. Details see in section 3.3.2.

2.4.3. Formula Operators. EZMath supports plus, minus, multiply and divide (+, -, *, /) as formula operators. These operations follow the conventions in languages like C and Java. Formula operators can be applied between two formulas. The result will be a formula-type value.

2.4.4. Logical Operators. EZMath supports the following logical operator: larger, less than, equal, larger or equal, less or equall, inequal ($>$, $<$, $=$, $\geq$, $\leq$, $\neq$). Logical operators can be used in conditions of piecewise formulas and logical expressions.

2.4.5. Built-in Operations. EZMath supports a subset of LaTeX's original math symbols, including:

arccos	cos	csc	exp	min
sinh	arcsin	cosh	lg	ln
arctan	cot	log	sec	tan
coth	max	sin	tanh	

2.5. Separators

A separator separates tokens. White space (see next section) is a separator, but it is not a token. The other separators are all single-character tokens themselves:

( ) , & \\

2.6. WhiteSpaces

White space is the collective term used for several characters: the space character, the tab character, the newline character, the vertical tab character, and the form-feed character. White space is ignored (outside of string and character constants), and is therefore optional, except when it is used to separate tokens.

3. Data Types

3.1. Double

The Double type represents real numbers, such as 3.14 or 0.01. The precision is 64-bit floating-point.

3.2. Formula

A formula is a procedure of computations upon given parameters. It's essentially the same as the function in C or Java language. Since EZMath compiler only operates on text between two $$, whenever defining or calling a formula, embed the whole statement inside two $$.

3.2.1. Formula Definitions. A formula definition contains a name (identifier), a pair of parentheses, and an optional list of parameters in the parentheses, an equal sign, and an expression, from left to right order. The expression cannot contain any identifiers other than the paramters. Some examples:

```
$$g() = 3$$
$$f(x) = \sin {x}$$
$$f(a, b) = \sin {a + b} $$
```

3.2.2. Formula Parameters. A formula can either has no parameters or has any number of parameters of double type. The naming rules of parameters is the same as identifers, except we can have parameters have the same name, as long as they appear in different definitions.

3.2.3. Formula Return Value. A formula always returns a double type value.

3.2.4. Calling Formula. A formula can be called as follows:

```
$$g()$$
$$f(x())$$
$$f(8.0, 3)$$
```

In the second example above, $x()$ is a formula which returns a double value.

3.2.5. Formula Evaluation. In the main() function inside the output code of the EZMath compiler, if the user explicitly calls a formula with valid argument, e.g, $f(5)$, which means inexplicitly the number of arguments is correct and each argument is of type double, then the main() function will print out the returned value of type double. Otherwise, a specialized function will handle the error appropriately.

3.2.6. Recursive Formula. A formula represents a function, thus it's intuitive to support recursive formula. The usage of a recursive formula is illustrated as follows:

```
$$
f(x) = 2 * f(x/2);
$$
```

However, we should also use piecewise formula to define the termination conditions for recursive formulas.

3.2.7. Piecewise Formula. EZMath also supports piecewise formulas. The usage of a nested formula is illustrated as follows:

```
$$
f(x) = \begin{cases} f(x-1)*x & x>0 \\
                       1        & x=0
         \end{cases}
$$
```

As shown above, we use \\to seperate cases. And in a particular case, we use & to seperate expression and condition.

3.2.8. Nested Formula. Nested formula means referencing another formula in the definition of a formula. The formula being referenced must be defined earlier. The usage of a nested formula is illustrated as follows:

```
$$
g(x) = x + 1
f(x) = g(x) * 2
$$
```

3.3. Matrix

Matrix is supported following the LaTeX convetion.

```
\begin{bmatrix}...\end{bmatrix}
```

3.3.1. Matrix Definition. A matrix definition consists of information regarding the matrix's name, value if possible.

matrix m;

or

$$\textbf{matrix}\ \mathrm{m} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

3.3.2. Matrix Operations. EZMath supports plus, minus, multiply and transpose (+, -, *, ', .*) as matrix operators. Operands can be matrix-type variables or matrix constants. Plus, minus and multiply can be applied between two matrices, but the sizes of the operands must agree on the requirements of matrix's operation. Transpose can be applied on a single matrix. The result of matrix operators will be a matrix-type value.

Operator	Definition	Example
+	matrix add	A + B
-	matrix minus	A - B
*	matrix multiply	A * B
.*	multiply the corresponding elements	A .* B
'	matrix transpose	A'

4. Variables

Variables in EZMath should be defined in

```
 $$...$$,
```

e.g.

```
$$ a = 2 $$
```

By default, any variable defined is of type Double for the ease of arithmetic calucations.

5. Logical Expressions

Logical expressions are supported in LaTeX as following:

$$3^2 + 4^2 = 5^2$$

EZMath compiler will report true for this logical expression.

$$a^2 + b^2 = c^2$$

EZMath compiler will check if a, b, c are defined and assigned values. If not, it will report error in report.tex. Otherwise, it will check if the equality satisfies. If it does, it will report true in report.tex, false otherwise.

$$a < b$$

EZMath compiler will check if a, b are defined and assigned values. If not, it will report error in report.tex. Otherwise, it will check if the inequality satisfies. If it does, it will report true in report.tex, false otherwise.

If the elements of an logical expression are all constants, EZMath will remember this logical expression and validate its correctness in report.tex.

If any of the elements is an identifier, it must be defined and assigned earlier. The compared values from left and right hand side of the logical operator are of type double only. Matrix is not allowed in any kind of logical expression. If an invalid logical expression is encountered, the EZMath compiler will report the error in report.tex.

6. Comments

In accordance with the LaTeX syntax for comments, everything after the % character until the end of the line are comments and are ignored by EZMath compiler.

7. Input Program Structure and Scope

7.1. Program Structure

The input file of EZMath should be a LaTeX file. The compiler of EZMath only operates on statements between the first $$ encountered and the second $$ encountered. It ignores all other text outside the scope of what mentioned just now.

Users who want to display only LaTeX math statements can use \[and \] instead. Statements between \[and \] will be compiled by LaTeX compiler but will be ignored by EZMath compiler.

To sum up, EZMath Compiler will detect following statements:

1) Logical Expressions
2) Definition of Formulas
3) Calling of Formulas
4) Definition of Variables
5) Matrix Operations

7.2. Scope

All symbols defined in input program are global values.

8. Output Program Structure and Scope

8.1. Program Structure

Basically, EZMath will output a .cpp file corresponding to the input LaTeX file. The C++ program contains several math functions and one main functions. Math functions correspond to each formula defined in the LaTeX program, the main function is used to calculate and express the expression in the LaTeX program.

Compile the .cpp ouput file will further generate a LaTeX file. The LaTeX file is a report of the declarations and computations for the original input file. It will contain:

1) The basic information about the original input (title, arthors, etc)
2) All user defined variables and matrices
3) The validations of logical expressions
4) The declarations of formulas and
5) The literals and results of the computations, following the corresponding index (line number) in original input.

See the sample output for reference.

8.2. Scope

By default, the output program's math functions are globally visible.

9. Sample Program and Report

9.1. Sample.tex and output PDF

```
\documentclass[twocolumn]{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}

\title{Pythagorean Triples}
\author{EZMath Team}
\date{October 2014}

\setlength{\parindent}{0pt}

\begin{document}

\maketitle
\thispagestyle{empty}
\section{Introduction}
A "Pythagorean Triple"  is a set of positive integers, x, y and z
that fits the rule:
\[x^2 + y^2 = z^2\]
Example: The smallest Pythagorean Triple is 3, 4 and 5.

Let's check it:
$$x = 3, y = 4, z = 5$$
And
$$x^2 + y^2 = z^2$$
is true.
\section{Properties}
An interesting fact: a Pythagorean Triple always consists of:
\begin{enumerate}
\item all even numbers, or
\item two odd numbers and an even number.
\end{enumerate}
A Pythagorean Triple can never be made up of all odd numbers or two
even numbers and one odd number. This is true because:
\begin{enumerate}
\item The square of an odd number is an odd number and the square
of an even number is an even number.
\item The sum of two even numbers is an even number and the sum
of an odd number and an even number is in odd number.
\end{enumerate}
So, when both a and b are even, c is even too. Similarly when one of
a and b is odd and the other is even, c has to be odd!
\section{Constructing Pythagorean Triples}
It is easy to construct sets of Pythagorean Triples.
```

```
When m and n are any two positive integers:

$$a(m,n) = \begin{cases} n^2 - m^2 & n>=m \\
                         m^2 - n^2 & n<m
            \end{cases}
$$

$$b(m,n) = 2*n*m$$
$$c(m,n) = n^2 + m^2$$


Then a, b, and c form a Pythagorean Triple.

\section{Example}
Here is a example how we construct Pythagorean Triples:

$$a(1,2)$$
$$b(1,2)$$
$$c(1,2)$$

\section{Extra}
Factorial is defined as:
$$f(x) = \begin{cases} f(x-1)*x & x>0 \\
                       1 & x=0
        \end{cases}
$$
We want to evaluate the value of $$f(10)$$
\end{document}
```

Pythagorean Triples

EZMath Team

October 2014

1 Introduction

A "Pythagorean Triple" is a set of positive integers, x, y and z that fits the rule:

$$x^2 + y^2 = z^2$$

Example: The smallest Pythagorean Triple is 3, 4 and 5.
Let's check it:

$$x = 3, y = 4, z = 5$$

And

$$x^2 + y^2 = z^2$$

is true.

2 Properties

An interesting fact: a Pythagorean Triple always consists of:

1. all even numbers, or
2. two odd numbers and an even number.

A Pythagorean Triple can never be made up of all odd numbers or two even numbers and one odd number. This is true because:

1. The square of an odd number is an odd number and the square of an even number is an even number.
2. The sum of two even numbers is an even number and the sum of an odd number and an even number is in odd number.

So, when both a and b are even, c is even too. Similarly when one of a and b is odd and the other is even, c has to be odd!

3 Constructing Pythagorean Triples

It is easy to construct sets of Pythagorean Triples. When m and n are any two positive integers:

$$a(m,n) = \begin{cases} n^2 - m^2 & n >= m \\ m^2 - n^2 & n < m \end{cases}$$

$$b(m,n) = 2 * n * m$$

$$c(m,n) = n^2 + m^2$$

Then a, b, and c form a Pythagorean Triple.

4 Example

Here is a example how we construct Pythagorean Triples:

$$a(1,2)$$

$$b(1,2)$$

$$c(1,2)$$

5 Extra

Factorial is defined as:

$$f(x) = \begin{cases} f(x-1) * x & x > 0 \\ 1 & x = 0 \end{cases}$$

We want to evaluate the value of

$$f(10)$$

9.2. result.cpp

```cpp
/**
*  EZMath
*  Fast documentation and compu-related text
*  Created by EZMath Compiler v0.1
**/
#include <iostream>
#include <fstream>
#include <ctime>

using namespace std;

#define NUM_OF_EXPS 4

#define NUM_OF_VARIABLES 3

#define NUM_OF_LOGICAL_EXPRESSIONS 1

#define NUM_OF_FORMULA_DEFINITION 4

enum class ErrorType {IllegalParameter};

void Error(ErrorType errorInstance) {
    if(errorInstance == ErrorType::IllegalParameter)
      cerr << "Error: " << "IllegalParameter" << endl;
}

double a(double m, double n)
{
    if(n >= m)
        return n * n - m * m;
    else
        return m * m - n * n;
}

double b(double m, double n)
{
    return 2 * n * m;
}

double c(double m, double n)
{
    return n * n + m * m;
}

double f(double x)
```

```
{
    if(x > 0)
        return f(x-1) * x;
    if(x == 0)
        return 1;
    else
        Error(ErrorType::IllegalParameter);

    return -1.0;
}


const string Title = "Pythagorean Triples";

const string Author = "EZMath Team";

const string Date = "October 2014";

const string variables[NUM_OF_VARIABLES] = {
  "x", "y", "z"
};

const double variablesValue[NUM_OF_VARIABLES] = {
  3, 4, 5
};

const string logicalExpressions[NUM_OF_LOGICAL_EXPRESSIONS] = {
  "x^2 + y^2 = z^2"
};

const bool logicalExpressionsValue[NUM_OF_LOGICAL_EXPRESSIONS] = {
  true
};

const string formulaDefinition[NUM_OF_FORMULA_DEFINITION] = {
      "a(m,n) = \\begin{cases} n^2 - m^2 & n>=m \\\\ m^2 - n^2 & n<m\\end{cases}",
      "b(m,n) = 2*n*m",
      "c(m,n) = n^2 + m^2",
      "f(x) = \n\\begin{cases} f(x-1)*x & x>0 \\\\1 & x=0 \\end{cases}"
};


const string expressions[NUM_OF_EXPS] = {
    "a(1,2)",
    "b(1,2)",
    "c(1,2)",
    "f(10)"
};
```

```
/**
* Main Function
**/
int main(int argc, char ** argv) {
    time_t t = time(0);
    struct tm * now = localtime( & t );
    ofstream file("result.tex");

    //Begin
    file << "\\documentclass{article}\n"
         << "\\usepackage[utf8]{inputenc}\n"
         << "\\usepackage{amsmath}\n"
         << "\\title{Summary of " << Title << "}\n"
         << "\\author{EZMath Compiler}\n"
         << "\\date{"<< (now->tm_year + 1900) << '-'
                     << (now->tm_mon + 1) << '-'
                     <<  now->tm_mday
                     <<"}\n"
         << "\\begin{document}\n"
         << "\\maketitle\n";
    //Content
    file << "\\section*{Variables Definition}\n";
    for (int i = 0; i < NUM_OF_VARIABLES; i++) {
        file << "$$" << variables[i] << " = " << variablesValue[i] << "$$\n";
    }

    file << "\\section*{Formula Definition}\n";

    for (int i = 0; i < NUM_OF_FORMULA_DEFINITION; i++) {
      file << "$$" << formulaDefinition[i] << "$$\n";
    }

    file << "\\section*{Logical Validation}\n";
    file << "\\begin{center}\\begin{tabular}{c c}";
    for (int i = 0; i < NUM_OF_LOGICAL_EXPRESSIONS - 1; i++) {
        file << "$" << logicalExpressions[i] << "$&" << "is " <<
        (logicalExpressionsValue[i]?"true":"false") << "\\\\";
    }
    file << "$" << logicalExpressions[NUM_OF_LOGICAL_EXPRESSIONS-1] << "$&" <<
    "is " << (logicalExpressionsValue[NUM_OF_LOGICAL_EXPRESSIONS-1]?"true":"false"

    file << "\\section*{Formula Evaluation}\n";

    double result[NUM_OF_EXPS];
    //double[] result = new double[NUM_OF_EXPS];
    result[0] = a(1,2);
    result[1] = b(1,2);
    result[2] = c(1,2);
    result[3] = f(10);

    for (int i = 0; i < NUM_OF_EXPS; i++) {
        file << "$$" << expressions[i] << " = " << result[i] << "$$\n";
    }

    //End
    file << "\\end{document}\n";

    file.close();
    return 0;
}
```

9.3. report.tex

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{amsmath}
\title{Summary of Pythagorean Triples}
\author{EZMath Compiler}
\date{2014-10-26}
\begin{document}
\maketitle
\section*{Variables Definition}
$$x = 3$$
$$y = 4$$
$$z = 5$$
\section*{Formula Definition}
$$a(m,n) = \begin{cases} n^2 - m^2 & n>=m \\ m^2 - n^2 & n<m \end{cases}$$
$$b(m,n) = 2*n*m$$
$$c(m,n) = n^2 + m^2$$
$$f(x) =
\begin{cases} f(x-1)*x & x>0 \\1 & x=0 \end{cases}$$
\section*{Logical Validation}
\begin{center}\begin{tabular}{c c}$x^2 + y^2 = z^2$&is true\end{tabular}\end{cente
\section*{Formula Evaluation}
$$a(1,2) = 3$$
$$b(1,2) = 4$$
$$c(1,2) = 5$$
$$f(10) = 3.6288e+06$$
\end{document}
```

9.4. report.pdf

Summary of Pythagorean Triples

EZMath Compiler

2014-10-26

Variables Definition

$$x = 3$$

$$y = 4$$

$$z = 5$$

Formula Definition

$$a(m,n) = \begin{cases} n^2 - m^2 & n >= m \\ m^2 - n^2 & n < m \end{cases}$$

$$b(m,n) = 2 * n * m$$

$$c(m,n) = n^2 + m^2$$

$$f(x) = \begin{cases} f(x-1) * x & x > 0 \\ 1 & x = 0 \end{cases}$$

Logical Validation

$$x^2 + y^2 = z^2 \quad \text{is true}$$

Formula Evaluation

$$a(1,2) = 3$$

$$b(1,2) = 4$$

$$c(1,2) = 5$$

$$f(10) = 3.6288e + 06$$