Complexvec1

Complex number extension for C++ vector class library

Agner Fog

Chapter 1

Introduction

The file complexvec1.h provides classes, operators, and functions for calculations with complex numbers and complex number vectors. This is an extension to the Vector Class Library.

The classes listed below are defined. Common operators and functions are defined for these classes:

Table 1.1: Comples number vector classes

Complex vector class	**Precision**	**Complex elements per vector**	**Corresponding real vector class**	**Total bits**	**Recommended minimum instruction set**
Complex1f	single	1	Vec4f	128	SSE2
Complex2f	single	2	Vec4f	128	SSE2
Complex4f	single	4	Vec8f	256	AVX
Complex8f	single	8	Vec16f	512	AVX512
Complex1d	double	1	Vec2d	128	SSE2
Complex2d	double	2	Vec4d	256	AVX
Complex4d	double	4	Vec8d	512	AVX512

1.1 Compiling

The complex vector class extension to the Vector Class Library is compiled in the same way as the Vector Class Library itself. All x86 and x86-64 platforms are supported, including Windows, Linux, and Mac OS. The following C++ compilers can be used: Gnu, Clang, Microsoft, and Intel. See the vector class library manual for further details.

This example shows how to use the complex number vectors:

Example 1.1.

```
// Example for complex number vectors
#include <stdio.h>
#include "vectorclass.h"  // vector class library
#include "complexvec1.h"  // complex number extension

// function to print complex number vector:
template <typename C>
void printcx (const char * text, C a) {
    auto aa = a.to_vector(); // get elements as real vector
    printf("\n%s", text);    // print text
```

```
    for (int n = 0; n < a.size(); n++) { // loop through elements
        printf("(%.3G,%.3G)  ", aa[2*n], aa[2*n+1]);
    }
}

int main() {
    // define vectors of two complex numbers
    Complex2d a( 1, 2, 3,  4); //  1+i*2, 3+i*4
    Complex2d b(-2, 1, 0, -3); // -2+i*1, 0-i*3
    Complex2d c = a + b;       // add complex numbers
    Complex2d d = a * b;       // multiply complex numbers

    // print results
    printcx("a = ", a);        // a = (1,2)   (3,4)
    printcx("b = ", b);        // b = (-2,1)  (0,-3)
    printcx("c = ", c);        // c = (-1,3)  (3,1)
    printcx("d = ", d);        // d = (-4,-3) (12,-9)
}
```

Chapter 2

Constructing vectors and loading data into vectors

There are many ways to create vectors and put data into vectors. These methods are listed here.

Method	default constructor
Defined for	all complex classes
Description	the vector is created but not initialized. The value is unpredictable
Efficiency	good

```
// Example:
Complex4f a;     // creates a vector of four complex numbers
```

Method	Construct from single real
Defined for	all complex classes
Description	The parameter defines the real part of all elements. The imaginary parts are zero.
Efficiency	good

```
// Example:
Complex4d a(3);  // a = (3,0) (3,0) (3,0) (3,0)
```

Method	Construct from single real/imaginary pair
Defined for	all complex classes
Description	All elements get the same real/imaginary values
Efficiency	good

```
// Example:
Complex4d a(3,1);  // a = (3,1) (3,1) (3,1) (3,1)
```

Method	Construct from multiple real/imaginary pairs
Defined for	all complex classes
Description	The parameters define all the real/imaginary pairs
Efficiency	good

```
// Example:
Complex4d a(1,0, 2,2, 3,-3, 0,4); // a = (1,0) (2,2) (3,-3) (0,4)
```

Method	Construct from single complex scalar
Defined for	all complex classes
Description	The complex number is broadcast into all elements
Efficiency	good

```
// Example:
Complex1d a(1,2)
Complex4d b(a);  // a = (1,2) (1,2) (1,2) (1,2)
```

Method	Construct from multiple complex scalars
Defined for	all complex classes
Description	Each parameter defines one complex pair
Efficiency	good

```
// Example:
Complex1d a(1,2)
Complex1d b(3,4)
Complex1d c(5,6)
Complex4d d(a,b,c,b);  // a = (1,2) (3,4) (5,6) (3,4)
```

Method	Construct from two complex vectors of half the size
Defined for	Complex2f, Complex4f, Complex8f, Complex2d, Complex4d
Description	The two vectors are concatenated into one bigger vector
Efficiency	good

```
// Example:
Complex2f a(1,2, 3,4)
Complex2f b(5,6, 7,8)
Complex4f c(a,b) // c = (1,2) (3,4) (5,6) (7,8)
```

Method	member function load(p)
Defined for	all complex classes
Description	Load data from array of same precision. Each real part must be followed by the corresponding imaginary part.
Efficiency	good

```
// Example:
double a[8] = {1,2,3,4,5,6,7,8};
Complex4d b;
b.load(a);  // b = (1,2) (3,4) (5,6) (7,8)
```

Method	member function store(p)
Defined for	all complex classes
Description	Save data into array of same precision. Each real part is followed by the corresponding imaginary part.
Efficiency	good

```
// Example:
float a[8];
Complex4f b(1,2,3,4,5,6,7,8);
```

```
b.store(a);  // a = {1,2,3,4,5,6,7,8}
```

Method	member function real()
Defined for	Complex1f, Complex1d
Description	Get real part of complex scalar
Efficiency	good

```
// Example:
Complex1d a(1,2);
double r = a.real();  // a = 1
```

Method	member function imag()
Defined for	Complex1f, Complex1d
Description	Get imaginary part of complex scalar
Efficiency	good

```
// Example:
Complex1d a(1,2);
double r = a.imag();  // a = 2
```

Method	member function extract(i)
Defined for	all complex classes
Description	Extract one complex number from vector at position i. The first element has index 0.
Efficiency	good with AVX512VL, medium otherwise

```
// Example:
Complex4d a(1,2 ,3,4 ,5,6, 7,8);
Complex1d b = a.extract(1);  // b = (3,4)
```

Method	member function get_low()
Defined for	Complex2f, Complex4f, Complex8f, Complex2d, Complex4d
Description	Get the lower half of a complex vector
Efficiency	good

```
// Example:
Complex4d a(1,2, 3,4, 5,6, 7,8);
Complex2d b = a.get_low();  // b = (1,2) (3,4)
```

Method	member function get_high()
Defined for	Complex2f, Complex4f, Complex8f, Complex2d, Complex4d
Description	Get the upper half of a complex vector
Efficiency	good

```
// Example:
Complex4d a(1,2, 3,4, 5,6, 7,8);
Complex2d b = a.get_high();  // b = (5,6) (7,8)
```

Method	member function size()
Defined for	all complex classes
Description	Get the number of complex pairs
Efficiency	good

```
// Example:
Complex4d a(0);
int b = a.size();  // b = 4
```

Method	member function elementtype()
Defined for	all complex classes
Description	Get the precision of the numbers. The return value is 0x110 for single precision and 0x111 for double precision complex number vectors. Non-complex vector classes return other values, defined in vcl_manual.pdf
Efficiency	good

```
// Example:
Complex4f a(0);
int b = a.elementtype();  // b = 0x110
```

Chapter 3

Operators

Operator	+
Defined for	all complex classes
Description	Add two complex number vectors, or one complex number vector and one real scalar of the same precision
Efficiency	good

```
// Example:
Complex2f a(1,2, 3,4);
Complex2f b(5,6, 7,8);
Complex2f c = a + b;      // c = (6,8)  (10,12)
Complex2f d = a + 10.0f; // d = (11,2) (13,4)
```

Operator	-
Defined for	all complex classes
Description	Subtract two complex number vectors, or one complex number vector and one real scalar of the same precision
Efficiency	good

```
// Example:
Complex2f a(5,6,  7,8));
Complex2f b(1,-1, 2,3);
Complex2f c = a - b;      // c = (4,7) (5,5)
Complex2f d = a - 2.0f;  // d = (3,6) (5,8)
Complex2f e = -a;         // e = (-5,-6) (-7,-8)
```

Operator	*
Defined for	all complex classes
Description	Multiply two complex number vectors, or one complex number vector and one real scalar of the same precision
Efficiency	medium
Accuracy	Complex multiplication involves the calculation of sums of products. Loss of precision may occur if the result is close to zero. It is possible that the results of `a * b` and `b * a` are slightly different in this case.

```
// Example:
Complex2f a(5,6,  0,2));
Complex2f b(2,-1, 4,3);
```

```
Complex2f c = a * b;      // c = (16,7)   (-6,8)
Complex2f d = a * 2.0f;   // d = (10,12)  (0,4)
```

Operator	/
Defined for	all complex classes
Description	Divide two complex number vectors, or one complex number vector and one real scalar of the same precision
Efficiency	medium
Accuracy	Complex division involves the same possible loss of precision as complex multiplication.

```
// Example:
Complex2f a(2,4, 8,4);
Complex2f b(1,2, 0,4);
Complex2f c = a / b;      // c = (2,0)       (1,-2)
Complex2f d = a / 2.0f;   // d = (1,2)       (4,2)
Complex2f e = 2.0f / a;   // e = (0.4,-0.8)  (0,-0.5)
```

Operator	$\sim$
Defined for	all complex classes
Description	Complex conjugate. The sign of the imaginary part is inverted
Efficiency	good

```
// Example:
Complex2f a(1,2, 3,4);
Complex2f b = ~ a;      // b = (1,-2)  (3,-4)
```

Operator	==
Defined for	all complex classes
Description	Compare for equality. The result is a boolean vector as defined by the vector class library. One complex number element corresponds to two boolean elements with the same value.
Efficiency	good

```
// Example:
Complex2f a(1, 2, 3,4);
Complex2f b(1,-2, 3,4);
Vec4fb    c = (a == b);  // c = (false,false,true,true)
```

Operator	!=
Defined for	all complex classes
Description	Compare for not equal. The result is a boolean vector as defined by the vector class library. One complex number element corresponds to two boolean elements with the same value.
Efficiency	good

```
// Example:
Complex2f a(1 ,2, 3,4);
Complex2f b(1,-2, 3,4);
Vec4fb    c = (a != b);  // c = (true,true,false,false)
```

Chapter 4

Mathematical functions

Function	abs
Defined for	all complex classes
Description	Gives the absolute value of each complex number element (also called modulus or Euclidean norm). The result is real numbers.
Efficiency	medium

```
// Example:
Complex2f a(0,2, 3,4);
Complex2f b = abs(a); // b = (2,0) (5,0)
```

Function	csqrt
Defined for	all complex classes
Description	Calculates the principal square root of complex numbers.
Efficiency	medium

```
// Example:
Complex4d a(-3,4, 3,-4, -16,0, -16,-0.)
Complex4d b = csqrt(a); // b = (1,2) (2,-1) (0,4) (0,-4);
```

Function	cexp
Defined for	all complex classes
Description	Calculates the complex exponential function
Implementation	The files vectormath_exp.h and vectormath_trig.h must be included before complexvec1.h. Must use version 2.00 or later of VCL.
Efficiency	poor

```
// Example:
const double pi = 3.14159265358979323846;
Complex2d a(0,pi/2, 1,pi);
Complex2d b = cexp(a); // b = (0,1) (-2.71828,0)
```

Function	clog
Defined for	all complex classes
Description	Calculates the principal complex logarithm
Implementation	The files vectormath_exp.h and vectormath_trig.h must be included before complexvec1.h. Must use version 2.00 or later of VCL.
Efficiency	poor

```
// Example:
const double e = 2.71828182845904523536;
Complex2d a(0,-1, e,0);
Complex2d b = clog(a); // b = (0,-1.5708) (1,0)
```

Function	chorizontal_add
Defined for	all complex classes
Description	Calculates the sum of the complex elements of a vector
Implementation	Must use version 2.00 or later of VCL.
Efficiency	medium

```
// Example:
Complex4d x(1,2, 3,4, 5,6, 7,8);
Complex1d y = chorizontal_add(x); // y = (16,20)
```

Chapter 5

Other functions

Function	to_float
Defined for	Complex1d, Complex2d, Complex4d
Description	Convert from double precision to single precision
Efficiency	good

```
// Example:
Complex2d a(1.0,2.0, 3.0,4.0);
Complex2f b = to_float(a); // b = (1.0f,2.0f) (3.0f,4.0f)
```

Function	to_double
Defined for	Complex1f, Complex2f, Complex4f
Description	Convert from single precision to double precision
Efficiency	good

```
// Example:
Complex2f a(1.0f,2.0f, 3.0f,4.0f);
Complex2d b = to_double(a); // b = (1.0,2.0) (3.0,4.0)
```

Function	to_vector
Defined for	all complex classes
Description	Convert to a vector of interleaved real and imaginary parts.
Efficiency	good

```
// Example:
Complex2f a(1,2, 3,4);
Vec4f     b = a.to_vector(); // b = (1,2,3,4)
```

Function	select
Defined for	All complex vectors
Description	Choose between the elements of two vectors. This is useful when there are branches in the algorithm. The selector is a boolean vector defined in the Vector Class Library. The number of elements in the boolean vector must be double the number of elements in the complex vector. Each pair of boolean elemens must have the same value, unless you want to separate real and imaginary parts.
Efficiency	good

```
// Example:
Complex2d a(1,2, 3,4);
Complex2d b(5,6, 7,8);
Vec4db s(true,true,false,false); // boolean vector
Complex2d c = select(s, a, b);   // c = (1,2) (7,8)
```