10. COMPLEX NUMBERS

Using complex numbers easily in C is relatively new - it has been introduced only in the late nineties. Therefore, many old(er) books use structures when complex numbers are involved, and all necessary mathematical operations are hardwired. With the new (C99) standard, things are much easier nowadays.

10.1 Required libraries

All the functions required for complex calculus and the necessary modifications of data types are included in the library file `complex.h`.

10.2 Variable declaration

In C, a complex variable is declared by adding "`complex`" to the data type, such as

```
...
float complex x;
double complex y;
int complex d;
float complex *fptr;
...
```

Each of these variables will be a complex number consisting of a real and an imaginary part. If you want to assign a value to these variables, you must use a statement of the following form

```
x = 5.0 + I * 3.0;
```

the resulting value of x is (in mathematical notation) $x = 5.0 + 3.0i$.

You have to be careful that, when using the `complex.h` library, you never declare a variable named `I` in your main program or any function of your own; the symbol `I` is then reserved for the complex unit (I $= \sqrt{-1} = 0.0 + 1.0i$).

Code 10.1: Dynamic memory allocation for complex numbers.

```
01      ...
02      #include<complex.h>
03      ...
04      int main()
05      {
06          ...
07          double complex *C;
08          double real,imag;
09          ...
10          C = (double complex *) malloc(10*sizeof(double complex));
11          ...
12          C[6]=5.0+I*3.0;
13          real=creal(C[6]);
14          imag=cimag(C[6]);
15          ...
16          free(C)
17      }
```

Once you have declared some complex variables and assigned values to them, you can do the standard mathematical operations as demonstrated in the code of listing (10.2). You just have to be careful when using the mathematical function defined in math.h - these are not conform with complex algebra. Instead, you have to use the modified functions, some of which are listed in section 10.5.

10.3 Dynamic memory allocation

Dynamic memory allocation for complex numbers follows the rules for real numbers. First, you declare a pointer of the data type you need (e.g., `float complex *`, `double complex *`, ...). Then, you use the function `malloc` to allocate memory for as many elements as required. The last step of your program will be to release the memory using the function `free`. See code (10.1).

10.4 ... and for functions

The same rules apply for complex numbers and arrays as for real arrays; you will just have to use the proper complex data type in the parameter list of a function or the data type of a function, e.g.

```
double complex *funct_name(...,float complex *x,...)
{
    ...
}
```

is the frame of a function definition that returns an array of data type `double complex` and receives (among others) an array of data type `float complex`.

10.5 Build-in functions

A never complete(?) list:

creal to obtain the real part of a complex number (e.g., `r = creal(z);`)

cimag to isolate the imaginary part of a complex number (`c = cimag(z);`)

cabs to calculate the absolute value of a complex number (`a = sqrt(r*r + c*c);`)

cpow for the calculation of $z_1 = Z_2^{z_3}$

csqrt to determine the value of $z_1 = \sqrt{z_2}$

catan Some trigonometric functions for complex numbers

ccos

csin

clog to determine the natural logarithm of a complex number

cexp to calculate $z_1 = e^{z_2}$

clog10 in principle the same as for real numbers

clog2

With most of these function you must be careful. Atan, sqrt, pow, ... have periodic solutions; these functions above determine the solution for the first (or main) "branch", but other possible solutions are not considered. In doubt, consult your mathematical skills and knowledge (or your complex calculus books).

Listing (10.2) shows some typical complex calculations in a C-program.

Code 10.2: Sample program on the use of complex numbers in C.

```
01 #include<stdio.h>
02 #include<math.h>
03 #include<complex.h>
04 /*------------------------------------------------------
05    Example program of how to use the complex functions
06    in C. Note that the compiler must be compatible to
07    the C99 standard.
08    This version runs with the Intel icc compiler.
09    The gnu comiler (gcc) doesn't need the double specifier
10    before the complex statement.
11    Compiling command:
12       icc -lm -o test_complex test_complex.c
13 ------------------------------------------------------*/
14 int main()
15 {
16    double x1,y1;
17    int i;
18    double complex z;
19    double complex w;
20    double complex r;
21    double complex *D;
22
23    /*--- Write the complex numbers ---*/
24    /*--------------------------------------
25       I is the imaginary unit, defined in
26       complex.h
27    --------------------------------------*/
28    x1=10.0;
29    y1=15.0;
30    z=10.0-3.5*I;
31    w=x1+y1*I;
```

```
32
33    r=z+w;
34       printf("\n%lf+%fi",creal(r),cimag(r));
35    r=z*w;
36       printf("\n%lf+%fi",creal(r),cimag(r));
37    r=z/w;
38       printf("\n%lf+%fi",creal(r),cimag(r));
39    r=csqrt(z);
40       printf("\n%lf+i%f",creal(r),cimag(r));
41
42    D=(double complex *) malloc(sizeof(double complex)*5);
43    for(i=0;i<5;i++)
44       D[i]=i*x1+i*y1*I;
45    for(i=0;i<5;i++)
46       printf("\n%lf+%fi",creal(D[i]),cimag(conj(D[i])));
47    free(D);
48    printf("\n");
49 }
```