

9. Import/export of data; Low-Level I/O; Code generation

In this lecture we go deeper into the capabilities of Maple to interact at a low level to retrieve or save data. We end this part with a demonstration how Maple can generate code.

9.1 Import/export of data

Raw numerical data in the form of matrices of floating-point numbers can be brought in and out of Maple with the commands `readdata` and `writedata`.

Let us first create a matrix:

```
[> indfun := (i,j) -> 1/(i+j-1);    # index function
[> data := matrix(3,5,indfun);      # 3-by-5 Hilbert matrix
[> Digits := 20:                    # write a 20 digit approximation to file
[> writedata(terminal,data,float);  # first write to terminal as test
[> writedata("C:\\MCS320\\mydata",data,float);
[> data := 'data';                  # release variable to check retrieval
```

Matrices in `readdata` are considered to be organized columnwise. Besides the file, we give in the number of columns we wish to read :

```
[> data := readdata("C:\\MCS320\\mydata",5);
```

9.2 Low-Level I/O

We can manipulate files very much like if we were programming in C. A sample session is below:

```
[> ourfile := "C:\\MCS320\\testfile":
[> fopen(ourfile,WRITE,TEXT);
[> n := writeline(ourfile,'this is some text');
[> fclose(ourfile);
[> fopen(ourfile,READ,TEXT);
[> readline(ourfile);
[> fclose(ourfile);
[> fopen(ourfile,APPEND,TEXT);
[> fprintf(ourfile,"number of characters on previous line :%d\n",n);
[> fclose(ourfile);
```

9.3 Code Generation

If we wish to export functions, for example to evaluate polynomials or other mathematical expressions, we can generate C or Fortran code.

```
[> with(codegen);
[> p := x^4 + 3*x + 2;
[> fortran(p,'optimized',precision=double,mode=double);
[> C(p);
```

To send the output to a file, we proceed as follows:

```
[> C(p, 'optimized', filename="C:\\MCS320\\prog.c");  
[> ssystem("type C:\\MCS320\\prog.c");
```

We can also convert Maple procedures into C:

```
[> fp := makeproc(p,x);           # make procedure from formula  
[> fp(3);                        # test the function  
[> C(fp, 'optimized');  
[> fortran(fp, 'optimized');
```

The difference with the previous code generation is that now we get a complete function, instead of just a translation of the code to evaluate the polynomial.

Since Maple 7, there is support for MathML (Mathematical Markup Language), an evolving standard for putting mathematics on the web. Here is how it looks like:

```
[> MathML(fp);
```

Note that the package `codegen` contains several other procedures like `GRAD`, `GRADIENT`, `HESSIAN`, and `JACOBIAN`. These procedures take on input a procedure and return derivatives. Taking derivatives of functions (or programs in general) is known as *automatic differentiation*. We will return to this topic in the third part of this course.

9.4 Assignments

1. Type `m := matrix([[1,2],[3,4],[5,6]])`; to create the matrix `m`.

Give the Maple commands to write `m` to file, and to read the first column of `m` from file, assigning this first column to `c1`.

2. Generate a C function to evaluate the “cubic formula” for the *first* root of a general polynomial of degree three: $ax^3 + bx^2 + cx + d$, where a, b, c , and d are parameters.

First generate a Maple procedure with arguments a, b, c , and d before converting to C.