Chapter 7

Installation and copyright

7.1 Installation

There are several different forms in which you might receive Meschach. Meschach is available over Internet/AARnet via netlib, or at the anonymous ftp site `des@thrain.anu.edu.au` in the directory `pub/meschach`. There are seven `.shar` files: `meschach1.shar`, `meschach2.shar`, `meschach3.shar` (which contain the library itself), `doc.shar` (which contains basic documentation), `include.shar` (which contains an include directory), `test.shar` (which contains some test routines), and `machines.shar` (which contains machines dependent files for a number of machines/operating systems/compilers). There is also a `readme` file that you should get. To get the library from netlib,

```
mail netlib@research.att.com
send all from c/meschach
```

To extract the files from the `.shar` files, put them all into a suitable directory and use

```
sh <file>.shar
```

to expand the files. Before you try compiling, check the `machines` directory for your machine, operating system or compiler. Save the machine dependent files `makefile`, `machine.c` and `machine.h`. Copy those files from the directory for you machine to the directory where the source code is. Then to get it up and running on a Unix machine with a standard Kernighan and Ritchie style compiler, type

```
make all
rm -i *.o
```

On a Sun 3 edit the line of `makefile` containing "`CFLAGS = ...`" to read

```
CFLAGS = -Dsun -O2
```

and for a Sun 4 or SPARC, edit it to read

```
CFLAGS = -Dsun -target sun4 -O2
```

For an ANSI C compiler you should add the switch `-DANSI_C`, both for compiling `meschach`, and for compiling programs using `meschach`.

On an IBM PC clone, the source code would be on a floppy disk. Use `xcopy a:* meschach` to copy it to the `meschach` directory. Then `cd meschach`, and then compile the source code. Different compilers on MSDOS machines will require different installation procedures. Check the directory
`meschach\machines` for the appropriate "makefile" for your compiler. If your compiler is not listed, then you should try compiling it "by hand", modifying the machine-dependent files as necessary.

To link into a program `prog.c`, compile it using

```
cc -o prog_name prog.c ....(source files).... meschach.a -lm
```

This code has been mostly developed on the University of Queensland, Australia's Pyramid 9810 running BSD4.3. Initial development was on a Zilog Zeus Z8000 machine running Zeus, a Unix workalike operating system. Versions have also been successfully used on various Unix machines including Sun 3's, IBM RT's, SPARC's and an IBM RS/6000 running AIX. It has also been compiled on an IBM AT clone using Quick C. It has been designed to compile under either Kernighan and Richie, Edition 1 C and under ANSI C. (And, indeed, it has been compiled in both ANSI C and non-ANSI C environments.)

You may very well want it to run on some other machine. The code has been written with this in mind. There are three files that you may wish to change in order to adapt the code to run (best) on your machine. They are `makefile`, `machine.h`, which contains most of the machine-specific definitions, and `machine.c` which contains the core routines and speed may be improved by re-writing these routines.

A library of these machine-dependent files is being developed for different machines, operating systems and compilers. If you have ported Meschach to a new machine/operating system/compiler that is not in this list, please let me know and I will include the machine dependency files for future releases.

7.1.1 makefile

The most likely change that you would want to make to this file is to change the line

```
CFLAGS = -O
```

to suit your particular compiler.

The code is intended to be compilable by both ANSI and non-ANSI compilers. To achieve this portability without sacrificing the ANSI function templates (which are very useful for avoiding problems with passing parameters) there is a token `ANSI_C` which must be `#define`'d in order to take full advantage of ANSI C. To do this you should do all compilations with

```
#define ANSI_C 1
```

This can also be done at the compilation stage with a `-DANSI_C` flag. Again, you will have to use the `-DANSI_C` flag or its equivalent whenever you compile, or insert the line

```
#define ANSI_C 1
```

in `machine.h`, to make full use of ANSI C with this matrix library.

7.1.2 machine.h

There are a few quantities in here that should be modified to suit your particular compiler. Firstly, the macros `mem_copy()` and `mem_zero()` need to be correctly defined here. The original library was compiled on BSD systems, and so it originally relied on `bcopy()` and `bzero()`.

In `machine.h` you will find the definitions for using the standard ANSI C library routines:

```
/*--------------------ANSI C--------------------*/
#include        <stddef.h>
#include        <string.h>
#define mem_copy(from,to,size)  memcopy((to),(from),(size))
#define mem_zero(where,size)    memset((where),'\0',(size))
```

Delete or comment out the alternative definitions and it should compile correctly.

There are two further machine-dependent quantities that should be set. These are *machine epsilon* or the *unit roundoff* for double precision arithmetic, and the maximum value produced by the `rand()` routine, which is used in `rand_vec()` and `rand_mat()`. The current definitions of these are

```
#define MACHEPS 2.2e-16
#define MAX_RAND 2.147483648e9
```

The value of MACHEPS should be correct for all IEEE standard double precision arithmetic.

However, ANSI C's <float.h> contains \#define'd quantities DBL_EPS and RAND_MAX, so if you have an ANSI C compiler and headers, replace the above two lines of machine.h with

```
#include <float.h>
#define MACHEPS DBL_EPSILON
#define MAX_RAND RAND_MAX
```

The default value given for MAX_RAND is 2^{31}, as the Pyramid 9810 and the SPARC 2's both have 32 bit words. There is a program macheps.c which is included in your source files which computes and prints out the value of MACHEPS for your machine.

Some other macros control some aspects of Meschach. One of these is SEGMENTED which should be #define'd if you are working with a machine or compiler that does not allow large arrays to be allocated. For example, the most common memory models for MS-DOS compilers do not allow more than 64Kbyte to be allocated in one block. This limits square matrices to be no more than 90×90. Inserting #define SEGMENTED 1 into machine.h will mean that matrices are allocated a row at a time.

Another macro (for version 1.1c) is MEM_THRESH which sets a threshold for retaining workspace vectors and matrices. If memory efficiency is your main requirement, then put

```
#define MEM_THRESH 0
```

in machine.h. If you want to use an intermediate value, use

```
#define MEM_THRESH 10000
```

for example, so that workspace objects with 10 000 or more entries (*not bytes*) are destroyed on exiting functions. If time efficiency is your main requirement then put #undef MEM_THRESH at the end of machine.h.

7.1.3 machine.c

The core routines in machine.c as they presently are, are adequate on scalar processors. However, they are not designed to make best use of the recent super–scalar processors, or of vector processors. If you wish to make best use of these features of your machine in using the matrix library, then you should rewrite these appropriately, possibly in assembly language. This has already been done to some extent, using "loop-unrolling":

```
sum0 = sum1 = sum2 = sum3 = 0.0;

len4 = len / 4;
len  = len % 4;

for ( i = 0; i < len4; i++ )
{
        sum0 += dp1[4*i]*dp2[4*i];
        sum1 += dp1[4*i+1]*dp2[4*i+1];
        sum2 += dp1[4*i+2]*dp2[4*i+2];
        sum3 += dp1[4*i+3]*dp2[4*i+3];
}
sum = sum0 + sum1 + sum2 + sum3;
dp1 += 4*len4;        dp2 += 4*len4;

for ( i = 0; i < len; i++ )
        sum += dp1[i]*dp2[i];
```

It may seem odd to use `dp1[i]*dp2[i]` instead `(*dp1++)*(*dp2++)` in the quest for speed, but optimising compilers being what they are they cannot be trusted to do what you intend. The expression `dp1[i]*dp2[i]` was recognised for what it is, but `(*dp1++)*(*dp2++)` was not, by the RS/6000 optimising compiler. This may be a matter of taste by the compiler writers, so check it out on your own system before making any terminal decisions about what is fastest on your machine.

Also note that the `__zero__()` routine is defined in `machine.c`. This uses the `mem_zero()` macro in `machine.h` in the standard release. However, if the double precision zero is **not** represented by a bitstring of zeros, the body of this routine would need to be replaced by

```
while ( len-- )
    *dp++ = 0.0;
```

These are the only routines that need be modified, as essentially all other routines rely on these routines and on the `mem_copy()` macro, to provide adequate speed.

Such a re-writing effort may be worthwhile on, say, the i860 processor, where the speed of computing inner products in assembly (using special pipeline instructions) is an order of magnitude faster than general arithmetic operations. (See "Personal supercomputing with the Intel i860" by Stephen S. Fried, *Byte*, **16**, no. 1, Jan 1991, pp. 347–358 for an indication of possible performance.) Better use of the IBM RS/6000 super-scalar architecture has been obtained by re-writing some of the routines in `machine.c`. The speed of the core inner product routine on a 20MHz RS/6000 320 went from near the LINPACK rating of 7Mflops to about 20Mflops, half the theoretical peak speed of 40Mflops for a multiply and add each clock cycle.

7.2 Copyright

The copyright provisions for Meschach are intended to follow the lead of the Free Software Foundation in ensuring that the rights of people using and modifying the library cannot take away rights from others, while still enabling commercial use of the library. In that sense Meschach is not entirely "in the public domain". Notice that there is no intention to restrict the possible uses to which Meschach and parts of it are put, or to impede the work of programmers. The intent is only to make sure that users of any derivatives or modified versions of Meschach can still obtain access to the original code, and also to protect the reputations of myself and other programmers who modify or use Meschach

Copyright subsists on the documentation and on the matrix library and source code for same and is held by David Edward Stewart. It may be used free of charge provided the following rules are followed:

For legal purposes, in this section "the matrix library" shall refer to the "Meschach matrix library" as copyrighted by David Edward Stewart.

1. Anyone to whom software is sold containing part or all of the matrix library in any form, whether modified or not, must have the matrix library source code made available to them in machine readable form at nominal cost.

2. Anyone distributing the library must ensure that copyright notices "Copyright (C) David E. Stewart, 1986–1992" are published prominently along with the distribution in whatever form.

3. Anyone making changes to the library must prominently display this fact on any documentation relating to any use of the library (whether the use involves source or compiled code). Also, any such modification must be reflected in the routine `m_version()`, which prints out the current list of modifications to `stdout`.

4. Any code sold in object code form must include `m_version()` so that if the user so desires, he/she can determine what modifications and/or extensions to the original library have been made and who by.

Item (4) is deemed to be satisfied if there is a "version" command which executes the `m_version()` routine.

Finally, there is the usual statement about legal rights if something goes wrong in using the software. Trying to frame conditions under which Meschach can be guaranteed to work is unlikely to be a rewarding

task for anyone to undertake, especially with the wide range of software and hardware systems it could work under. This is further complicated by the usual problems of numerical analysis where "proof of correctness" is not a realistic possibility and round-off errors are always present. Finally, due to the non-commercial nature of Meschach, there is unlikely to be any value to persons attempting to sue me for failure of the library in any situation.

Meschach IS PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, THE AUTHOR DOES NOT MAKE ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

Contents