19.5 Checking the Installed Library

A. Purpose

The programs, `cdemo` and `cdemom`, simplify comparing MATH77 (or *mathc90*) demonstration driver results from different machines, or from different languages or precisions on the same machine. These programs may also prove useful for others wishing to compare results from running their codes on different machines.

B. Usage

B.1 Checking MATH77 and *mathc90*

We assume you have managed to get the source for MATH77 or *mathc90* into some directory on your machine, and have the corresponding demonstration drivers in another directory. First check the source for `amach`, see Chapter 19.1, and if necessary change it so that the machine constants are computed correctly for your environment. Then compile the source for the library, and make up a library so that it is easy to link in the required MATH77 or *mathc90* routines.

The `support` directory of the distribution contains a number of files that are meant to aid in getting the library checked out on your machine.

`cdcfg`	Configuration file – Described below.
`cdemo`	Stand alone program which compares results, described below – Compile and link this now.
`cdemom`	Stand alone program which generates command files, described below – Compile and link this now.
`xxx.t`	Sample template files used by `cdemom`, see below.
`xxx.res`	Results from some environments to use in comparisons

One uses `cdemom` together with `cdcfg` (or whatever name you put on the first line of the `cdjob` file) and a template file to generate a run stream for compiling, running, and collecting the results from the demonstration drivers. Given these results, and results from some other environment (several are provided in the `support` directory of the MATH77 & *mathc90* distribution), one can use the program `cdemo` together with `cdcfg` to compare results.

B.2 Comparing Results for Your Codes

If you are interested in comparing results you have computed in two different environments for your own codes, you should read the next section if you want some control over how the comparisons are made; otherwise you can run `cdemo` without having a file `cdcfg`. Each section of what you want to compare should contain a header line of the form '=name $\cdots$'. No comparisons are done until both result files have positioned themselves at such lines. Thus your other results should not contain an '=' in column 1. (There are ways around this, see the template file for the Univac in Section B.5 for details.) The only other section you need to read is Section B.6.

B.3 The Configuration File, `cdcfg`

The demonstration drivers distributed with MATH77 & *mathc90* generate output that is almost certain to be different for different machines, or even on the same machine when comparing MATH77 and *mathc90* results. The configuration file contains specifications that enable the program `cdemo` to distinguish which differences are of a nature that they should be called to ones attention. One can skip the rest of this section if one is just interested in using the configuration file as provided in the distribution for the checkout of the MATH77 & *mathc90* libraries.

In the descriptions below, items in brackets are optional, extra spaces between tokens are not significant, and upper and lower case letters are equivalent. For each demonstration driver this file contains an initial line of the form

=demonstration_driver_name [Fortran only]

If the '[Fortran only]' appears, this demonstration driver is not available in *mathc90*. ('[C only]' can be used for routines that are in C, but not in Fortran.)

This line is followed by 0 or more lines that indicate special conditions that apply to the demonstration driver named just above. The following statements are possible.

Set Tolerances on **on_expression**

TOL = **tol_expression** [on **on_expression**]

Ignore **ignore_list** [on **on_expression**]

Reset [on **on_expression**]

where

tol_expression a Fortran expression restricted to using

the operators '()*/'

the functions 'sqrt', and 'max'

the operands: integers, floating point numbers ('E' used for the exponent), and the variable 'TOL'.

on_expression is one of the following forms (n_1 and n_2 integers): 'columns n_1:[n_2]', 'lines n_1:[n_2]', or a Fortran character literal (i.e. `'string of characters'`). If the n_2 is missing it is assumed to be a very large number.

ignore_list is either 'extra lines', 'end failure' or a comma delimited list of one or more of the following: 'integer', 'floating', 'floating sign', or 'all'.

In addition a line with 'C', 'c' or '#' as a first character or a line starting with 8 blanks (or a blank line) is treated as a comment line.

Two floating point numbers, x, and y, are assumed to be sufficiently close to one another if

$$|x-y| \le \max(\max(1, M) \times \text{TOL},\ M \times 10^{1-d})$$

where $M = \max(|x|, |y|),\ d = \min(n_x, n_y)$, and n_x and n_y are the number of significant digits in x and y respectively.

The value used for TOL depends on four parameters: TOLSP1, TOLDP1, TOLSP2, and TOLDP2, and actions taken with 'TOL $= \cdots$' instructions. Initially the code sets TOLSP1 = TOLSP2 = 4×10^{-6}, and TOLDP1 = TOLDP2 = 7×10^{-15}. These values can be overridden by instructions in the file `cdjob` described below.

Finally, these values can be overridden with the 'Set Tolerances on $\cdots$' command. This instruction is meant to be used only with the demonstration driver DRMACH. As set up in the configuration file `cdcfg` this command sets these values to ($33 \times$ machine ε) for the two results being compared. There are separate values for single and double precision being defined, and the machine ε is defined as the smallest positive floating point number such that $1 + \varepsilon > 1$. Note that if this latter mechanism is used, the results for DRMACH must be the first ones given. In `cdcfg` as distributed this is the case, and the remaining demonstration drivers should be given in alphabetical order, since `cdemo` assumes an alphabetical order if it is trying to get a match on names, and there is a mismatch with the names in the result files.

When first encountering the start of results for a demonstration driver, it is assumed that the results are double precision if the third letter in the name is a 'D' or a 'd' and otherwise the results are for single precision. The permanent value for TOL assigned to comparing results for two demonstration drivers is then max(TOL?P1, TOL?P2), where the '?' is replaced by 'D' if the corresponding demonstration driver is in double precision, and else is replaced by 'S'.

A 'TOL $= \cdots$' instruction without an 'on $\cdots$' part sets the value of TOL to the value defined in the expression. When TOL appears in the expression, its value just prior to evaluating the expression is used. When an 'on $\cdots$' clause is present the value of TOL is not changed. Rather if two floating point numbers appear to be too large, a check is made to see if any 'on $\cdots$' clause provides a larger TOL, and if so that value is used to see if the numbers are close enough.

An 'on $\cdots$' expression means that a test is to be used only when a certain condition applies. In the case of 'on columns $\cdots$' the exceptional case applies when $n_1 \le i \le n_2$, where i is the column index in the first result file where the current token starts. The first result file is the first one given in `cdjob`, see the description of `cdemo` below.

In the case of 'on lines $\cdots$', the exceptional case applies when $n_1 \le \ell \le n_2$ where ℓ is the number of nonblank lines seen in the first result file since the line containing the header for the results for the demonstration driver, or if there has been a 'reset' instruction, the number of nonblank lines read since seeing this instruction. (Thus the line processed just after a reset instruction caused as a result of reading that line, will have an index of 0.) After reading an instruction that contains 'on lines $\cdots$', the configuration file is read no further until $\ell > n_2$, or a 'reset' instruction is executed as a result of seeing another line. When either of these events occurs, further lines in the configuration file are processed.

In the case of 'on `'string'`', the exceptional case applies when the line from the first result file contains `'string'`. The configuration file continues to be processed, except when the associated action is 'Reset', in which case processing of the configuration file stops until the 'Reset' action occurs or the end of output for this demonstration driver. If more than one of these is processed, only the last has any effect.

It is possible that all three of these mechanisms indicate that comparisons are to be done in an exceptional way. If any one of these mechanisms would excuse a difference that would otherwise be reported, no difference is reported.

For the 'ignore' instruction: 'integer' indicates that integers need not have the same value, 'floating' indicates that every floating point number is assumed to be equal to any other number, 'floating sign' indicates that floating point numbers should be compared with their signs stripped off, and 'all' indicates that any text matches any other text. The 'extra lines' means no diagnostic is given if the files match, except that one file has some lines embedded that are not in the other. The 'end failure' results in extra work if the lines match except for extra characters at the end of one line. When this occurs, an attempt is made to match the extra part of the longer line, with the start of another line at some point below.

Finally, the 'reset' instruction, sets everything to the values present just after processing the initial line for the demonstration driver. Thus there are no exceptional cases defined, TOL has its initial value, and the nonblank line counter is set to 0.

B.4 The Template File

You will probably need to make up your own template file that is used to generate the command file used to generate results. You will find various files with extensions '.tf' (for Fortran), and '.tc' (for C), that you can use as models. The template files use '#' as an escape character in the following ways.

#1 Substitute the name of the demonstration driver (default is lower case) for the #1.

#2 As for #1, except the name here and everywhere a #1 appears is in upper case.

#3 This is the end of the last line for output occurring one time at the very beginning. (#1 or #2 should not be used prior to a #3.)

#4 This is the end of the last line that is to be output for each of the demonstration drivers.

#5 This and the text following on the line are treated as a comment.

The nature of the template file is probably best understood by looking at an example. The following is for the Lahey Fortran compiler running under DOS on a PC.

```
del pcf77l.res /n#3
echo =#1 PC - Lahey F77L >>pcf77l.res
f77l k:\math77\demo\#1, #1.obj /NS
d:\progs\link #1,,,k:\math77\
fortran\obj\math77; #1 >>pcf77l.res
del #1.* /n#4
```

The output for running the demonstration drivers is being sent to the file pcf77l.res, and thus we start by deleting this file, since we don't want to add to an existing file. The #3 signals that this is all that is done one time only.

The echo statement writes a line starting with '=demonstration_driver_name' to the result file. The template file must arrange to write such a message ahead of the results for each driver. The '=' serves to signal the start of results for one demonstration driver, and the end of the results for the previous driver.

The next four lines compile, link, run the driver, and then delete the files that were created. Note that your file will almost certainly require different prefixes for the location of the source files, the location of library files, and the prefix used to find the location of the linker.

At least on one system, Exec 8 running on a Univac, one cannot avoid getting output from the compiling and linking into the output. To show how this is handled, we give the template we used for getting results on the Univac.

When the first line containing an "=" has text preceding the "=" this text is used as a sentinel that causes the current line and following lines to be ignored as far as making comparisons is concerned. The text after the "=" is used as a sentinel to cause comparisons to start being made again.

The way this file is setup, a '@@MSG,N =' at the start of a line in the output file signals that this line and any following lines not containing a header sentinel are to be ignored. The header sentinel is the text immediately following the '=' on the first line that contains an '='. In this case the start of output for a demonstration driver is flagged by a line of the form

@@XQT demonstration_driver_name.

The '~' output in the result file indicates a blank that occurs at the end of a line since `cdemo` has no way to recognize trailing blanks on a line. Thus `cdemo` will not use output in the result file that is not between a line of the form '@@XQT ⋯' and a line starting with '@@MSG,N =' for the purposes of making comparisons.

```
@@DELETE,C UNIRES
@@ASG,PU UNIRES
@@BRKPT PRINT$/UNIRES
@@MSG,N =@@XQT~#3
@@FTN #1
@@PREP
@@MAP,I ,#1
IN #1
LIB LIB*MATH77$
@@EOF
@@XQT #1
@@MSG,N = End of Fortran output on Univac for #1.
@@DELETE,AR #1#4
@@BRKPT PRINT$
```

A '@@MSG,N =' at the start of a line in the output file signals that this line and any following lines not containing a header sentinel are to be ignored. The header sentinel is the text immediately following the '=' on the first line that contains a '='. In this case the start of output for a demonstration driver is flagged by a line of the form

Finally for an HP workstation we used the following (the demonstration drivers were compressed).

```
rm hpfor.res#3
echo '='#1 ' HP Fortran ' `date` >>hpfor.res
zcat /math77/demo/#1.f.Z >#1.f
f77 -o #1 +OPP #1.f -lmath77
./#1 >>hpfor.res
rm #1*#4
```

Something similar should work on most Unix systems.

B.5 Obtaining the Result File Using `cdemom`

After getting the library installed and making up the template file for your environment, execute the program `cdemom`. You will be prompted for the name of your template file. The program reads the file `cdcfg` together with your template file to generate the run stream. Lines starting with anything other than a '=' in `cdcfg` are skipped. If the last character in the name of your template file is a 'C' or a 'c', lines with 'Fortran only' are also skipped. The name of the file containing the generated run stream is obtained as follows: If the file name contains a '.', the '.' and all characters to the right are deleted, and if the first characters of the template file are 'PC' or 'pc', the characters '.bat' are appended. If the file name does not contain a '.', an 'M' is appended to the name. The name used for the output file is printed before the program stops. Execute this file to get the result file.

B.6 Comparing Two Result Files Using `cdemo`

One needs to make up a file named `cdjob` that contains lines as follows.

Name of configuration file (blank line if none)
Name of first result file
Name of second result file
[F_1 F_2]

Where if F_1 and F_2 are given they must be floating point numbers. This replaces the default values of TOLSP1 …TOLDP2 with TOLSP1 = TOLSP2 = $\max(F_1,\ F_2)$, and TOLDP1 = TOLDP2 = $\min(F_1,\ F_2)$. These values may in turn be overwritten by the 'Set Tolerances on ⋯' instruction in the configuration file.

One then executes the program `cdemo`, which will print the names of demonstration drivers as their results are being compared, and outputs all results into the file `cdres`.

If as a result of doing the comparison you find differences there are five possibilities: there really is a problem in MATH77 or *mathc90*, the problem is in your compiler (check if the comparison is satisfactory when you turn off all compiler optimizations), the configuration file is not perfectly defined (we have not done an exhaustive analysis in picking the exceptions that are defined in this file), the result file you are comparing with is corrupted, or `cdemom` has a bug.

C. Examples and Remarks

Files ending with '.tf', and '.tc.' serve as examples of template files. The configuration file '`cdcfg`' can serve as an example of a configuration file should you want one for another purpose. The exceptions for `DRDSVA` illustrate the use of most of the features.

```
=drdsva
   Reset on '(Elements of V'
   Ignore FLOATING SIGN
   Reset on 'INDEX SING. VAL.'
   Ignore FLOATING SIGN
   TOL = TOL * 10 on lines 4:4
   TOL = TOL * 100 on lines 5:5
      ...
   Reset on '1 Earth'
   TOL = TOL * 10 on columns 25:38
   TOL = TOL * 100 on columns 40:53
   TOL = TOL * 1.E5 on columns 54:
   Reset on '1 Earth'
   TOL = TOL * 1.E7
```

D. Functional Description

When comparing DRMACH on the PC with that on the Hewlett Packard workstation, the following line is printed.

```
drmach PC Lahey F77L == drmach HP Fortran
FCOMP = 10
```

The 'FCOMP' refers to the number of comparisons that were made by computing and checking two floating point numbers. (Identical numbers are compared as text, and don't get counted in this comparison.) If the programs had differences, the '==' would have been replaced by '/=', and the FCOMP result would have been printed on a separate line after all the results for comparing these drivers had been printed. The result of the comparison in this case is a listing of the lines in the result files that were different. Those from the first file have a '1)' in the first two columns followed by the line number in the file, a ':' and then the line from the file. Those from the second file are similar except for starting with '2)'. If the difference was due to floating point errors a line something like the following is printed

```
F1=−3.3...E−13  F2=2.0...E−12  DIFF=−2.3...E−12
```

The F1 refers to a number in the first result file, the F2 to a number in the second result file, and DIFF to their difference. These numbers are printed from the value of the numbers and thus will typically not be formatted in the same way as they are in the result files. When numbers have extremely large exponents, the number printed may have a exponent that is less by 100. This allows results from a machine with an extremely large exponent range to be checked on a machine with a much smaller range. Also if one has an 'Ignore FLOATING SIGN' active, then the sign on one of the numbers may have been changed so that F1 and F2 have the same sign.

E. Error Procedures and Restrictions

If an error condition occurs, a Fortran 'STOP' is executed with a message describing the nature of the problem. Errors in the configuration file will also print the line causing the problem with an indication of where the problem was diagnosed.

F. Supporting Information

The source language is ANSI Fortran 77.

Program	**Required Files**
CDEMO	none
CDEMOM	none

Design and code due to F. T. Krogh, 1995.