Numdiff User Manual, version 5.8

"...und der eignen Kraft vertrauend steigt ein frei Geschlecht empor!"

This manual describes how to install and use Numdiff, a program which compares putatively similar files line by line and field by field, ignoring small numeric differences or/and different numeric formats.

Table of Contents

1 Copying

2 Acknowledgments

I want to thank Mr. Norman Clerman norm(dot)opcon(at)fuse(dot)net for several suggestions he gave me to improve the readability and the effectiveness of the output produced by Numdiff. He also pointed out the need to implement a filter to resynchronize the lines between two files in case of addition or deletion of one or more lines. I have to give him credit for the urge to prepare the versions 4.x and 5.x of Numdiff.

Moreover, I want to thank my friends Mariapia Palombaro, since she removed some errors while reviewing the first version of this document, and Paolo Caramanica, who suggested me to add more information to the output of the option '`-S`'.

3 Overview

Computer users often find occasion to ask how two files differ. Perhaps one file is a newer version of the other file. Or maybe the two files started out as identical copies but were changed by different people.

There are several ways to think about the differences between two files. One way to think of the differences is as a series of lines that were deleted from, inserted in, or changed in one file to produce the other file. The well-known `diff` program compares two files line by line, finds groups of lines that differ, and reports each group of differing lines. Without particular options, the `diff` program considers any change in the amount or in the type of the characters as a relevant difference. However, trough some command line options it also provides ways to suppress certain kinds of differences that are not important to the user. For instance, `diff` provides ways to ignore differences in the amount of white space between words or lines, or differences in alphabetic case.

Another way to think of the differences is as a series of words that were deleted from, inserted in, or changed in one file to produce the other file. Here "word" refers to a sequence of non white-space characters delimited by a couple of white-spaces, one before and the other one after the word.

The less known `wdiff` program by Franc,ois Pinard `pinard(at)iro(dot)umontreal(dot)ca` compares words in two files and reports the differences.

At last, one can think of the differences between two files as a sequence of pairs of bytes that can be either identical or different. The `cmp` program reports the differences between two files byte by byte, instead of line by line or word by word. As a result, it is often more useful than `diff` or `wdiff` for comparing binary files.

However, none of these approaches turns out to be good when you want to compare a couple of text files composed partially or entirely by numerical fields. Indeed, when you compare a couple of such files, what you want to obtain usually is a list of the numerical fields in the second file which **numerically** differ from the corresponding fields in the first file. But, as you probably knows, a number can be written using different notations and programs like `diff` or `wdiff` can not recognize whether a difference between two numeric fields is only due to the notation or is actually a difference of numerical values.

For instance, 11.23 and 11.2300000 are the same number but represented in different ways. While, if you are interested in the numerical values, it is obvious that the difference in the representation is not meaningful and then it should be ignored, however `diff` and `wdiff` consider the previous one as a relevant difference and there is no way for you to tell these programs to ignore it!

Another example of this type is given by 98765.4321 and 9.87654321E04 where the difference is only due to the use of the scientific notation in place of the ordinary decimal notation.

Moreover, depending on your country you could stick to different conventions in writing numbers. For instance, the amount "three hundred millions and fifty-two thousands of dollars and forty-six cents" is usually written by an Italian accountant as 300.052.000,46$ while an American accountant would write 300,052,000.46$. Of course, 300.052.000,46$ and 300,052,000.46$ represent the same amount of money but `diff` and `wdiff` would report a difference, which probably is not what you want in a similar case.

At last, sometimes you could want to ignore even differences in numerical values as long as they do not overcome a certain threshold. In other words, you could desire to suppress all "small" numerical differences too.

For instance, it could happen that you want to ignore all numerical differences whose absolute value is not greater than 0.0001. If this is the case, then the numerical fields 33 and 33.00009 must be considered equal, while 33 and 33.00011 must be reported as different.

However, `diff` and `wdiff` can not be used to ignore "small" numerical differences, since they do not even know what a numerical difference is.

What I have been saying till now explains why I decided to implement a new program with the capability to "appropriately" compare files containing numerical fields. In writing this program I was inspired by `ndiff`, a GPL'ed software by Nelson H. Baabe of the Salt Lake City University. The author of `ndiff` had the same good reasons as me to write `ndiff`. `ndiff` is actually a good tool and I used it for a while. But I did not completely like the way it works and so `numdiff` was born. Although `ndiff` inspired `numdiff`, they are completely different from the viewpoint of the source code: `numdiff` has been entirely written from scratch with the addition of code coming from GNU bc, GNU diff and GNUlib. In addition, the last versions of Numdiff offer much more features than `ndiff` does.

`numdiff` can be used to compare putatively similar files line by line and field by field, ignoring small numeric differences or/and different numeric formats. `numdiff` takes two mandatory arguments, the paths of the two files to compare, and, after splitting them into lines and the lines into fields according to a given list of field delimiters, it compares every field of every line of the first file with the corresponding field of the second file. What *corresponding* here exactly means depends on the options passed to the program on the command line. With no options, corresponding means the field of the second file at the same position, where position refers both to the line number and to the location within the line. If the compared fields are both legal numerical values, then `numdiff` performs a numerical comparison between them, else it performs a literal comparison, i.e. the usual byte-by-byte comparison. In case of literal comparison, two fields are regarded as equal if they are formed by the same sequence of characters. In case of numerical comparison and without specific command line options, two fields are regarded as equal if their numerical difference is zero. Mind that, if you do not explicitly specify a list of field delimiters by means of the option '`-s`' or '`-D`', `numdiff` takes as field delimiters the characters newline ('`\n`', ASCII code 0x0A), horizontal tabulation ('`\t`', ASCII code 0x09), and blank (' ', ASCII code 0x20).

For example, if the file '`list1`' contains the data

```
accident        123     23Joshua          34.55           +3+4i          water
dog       -3455.321     cat             2.345678e-9     .0005-6.23e2i
```

and file '`list2`' contains the data

```
Accident        123     23456          34.5500         +3.0001+4i
dog       -3455.320098  Cat     +2.345678e-9            -6.23e2i    $$$
A new line
```

then the output of the command '`numdiff list1 list2`' will be:

```
----------------
##1       #:1   <== accident
##1       #:1   ==> Accident
@                                                    @@
##1       #:3   <== 23Joshua
##1       #:3   ==> 23456
@                                                    @@
##1       #:5   <== +3+4i
##1       #:5   ==> +3.0001+4i
@ Absolute error = 1.0000000000e-4, Relative error = 2.0000000000e-5
##1       #>6   <== water
##1             ==>
@ Line 1 in file "list2" is shorter than expected!
```

```
----------------
##2       #:2   <== -3455.321
##2       #:2   ==> -3455.320098
@ Absolute error = 9.0200000000e-4, Relative error = 2.6104672633e-7
##2       #:3   <== cat
##2       #:3   ==> Cat
@                                                       @@
##2       #:5   <== .0005-6.23e2i
##2       #:5   ==> -6.23e2i
@ Absolute error = 5.0000000000e-4, Relative error = 8.0256821830e-7
##2             <==
##2       #>6   ==> $$$
@ Line 2 in file "list1" is shorter than expected!
----------------
##3             <==
##3       #>1   ==> A new line
@ Line 3 in file "list1" is shorter than expected!
----------------
          <==
##4       ==>


+++  File "list1" differs from file "list2"
```

At the same time `numdiff` will print the following error message on stderr:

```
***  End of file "list1" reached while trying to read line 4.
     File "list2" has more lines than file "list1",
     line 4 is the last one read from file "list2"
```

It is worth remarking that `numdiff` can recognize complex numbers, provided that they are written in the form $a + bi$ or $a - bi$ with no extra characters between the values a, b and the sign $+$ or $-$ (the symbol i, used to represent the imaginary unit, can be changed by a suitable command line option, see Chapter 5 [Invoking numdiff], page 21). If you do not know what complex numbers are, do not worry! In this case probably you will never manage files containing complex numbers and so you can happily continue to ignore them. :)

We consider now an example which shows how Numdiff can resynchronize the lines between two files in case of addition or deletion of one or more lines. The versions of Numdiff prior to 5 did not work well if one of the two files to compare contains in the middle some lines more or less than the other one. For instance, if you have one file that is 1000 lines long that you are comparing to a second file 1001 lines long, and except for that one extra line, located, let us say, at line 500, the files are identical, then `numdiff` version 4.x does **not** show only the one line difference: once the files are out of synchronization `numdiff` 4.x reports every line as different. Since version 5 it is possible in such cases to activate a filter which handles additions and deletions of lines. There are several options ruling how the filter actually works and I will give later a detailed explanation on how to use them to obtain each time the wished result. The simplest way to activate the filter consists in using the option '`-z @`'. If '`bill1`' and '`bill2`' are given by

```
 Month          Expenses
------------------------
 Jan09          $  233.56
 Feb09          $  850.77
 Mar09          $   12.55
 Apr09          $  524.00
```

```
 May09          $   78.25
 Jun09          $  230.00
 Jul09          $  443.10
 Aug09          $   67.65
 Sep09          $   10.00
 Oct09          $  201.45
 Nov09          $  110.00
 Dec09          $  200.27
-------------------------
Total           $ 2961.60
```

and

```
 Month          Expenses
 Jan09          $  234.00
 Mar09          $   13.00
 May09          $   78.25
 Jul09          $  443.10
 Sep09          $   10.00
 Nov09          $  110.00
 Jan10          $  200.00
-------------------------

Total           $ 1088.35
```

respectively, then the differences between the two files are:

- the insertion of the separator `-------------------------` in '`bill1`' before the list of the months,
- the deletion in '`bill2`' of the lines related to the expenses for the months February, April, June, August, October, December,
- small changes in '`bill2`' to the expenses of the months January 2009 and March,
- the presence in '`bill2`' of an entry for January 2010 just before the separator `-------------------------`,
- the addition of an empty line to '`bill2`' after the separator `-------------------------`,
- and the different values for the total sum of the expenses.

The output of the command '`numdiff -z @ -V bill1 bill2`' (I have added here the option '`-V`' to let Numdiff show which couples of lines it is comparing each time) is exactly then what you expect:

```
     ----------------
     ##2       <== -------------------------
               ==>

     ----------------
     ##3       <==  Jan09          $  233.56
     ##2       ==>  Jan09          $  234.00

     ##3       #:3   <== 233.56
     ##2       #:3   ==> 234.00
     @ Absolute error = 4.4000000000e-1, Relative error = 1.8838842268e-3
     ----------------
     ##4       <==  Feb09          $  850.77
```

```
          ==>

----------------
##5       <==  Mar09          $   12.55
##3       ==>  Mar09          $   13.00

##5       #:3   <== 12.55
##3       #:3   ==> 13.00
@ Absolute error = 4.5000000000e-1, Relative error = 3.5856573705e-2
----------------
##6       <==  Apr09          $  524.00
          ==>

----------------
##8       <==  Jun09          $  230.00
          ==>

----------------
##10      <==  Aug09          $   67.65
          ==>

----------------
##12      <==  Oct09          $  201.45
          ==>

----------------
##14      <==  Dec09          $  200.27
##8       ==>  Jan10          $  200.00

##14      #:1   <== Dec09
##8       #:1   ==> Jan10
@                                                    @@
##14      #:3   <== 200.27
##8       #:3   ==> 200.00
@ Absolute error = 2.7000000000e-1, Relative error = 1.3500000000e-3
----------------
          <==
##10      ==>

----------------
##16      <== Total           $ 2961.60
##11      ==> Total           $ 1088.35

##16      #:3   <== 2961.60
##11      #:3   ==> 1088.35
@ Absolute error = 1.8732500000e+3, Relative error = 1.7211834428e+0

+++  File "bill1" differs from file "bill2"
```

Numdiff has reported correctly the following differences:

- the second line of file '`bill1`', i.e. the one containing the separator, has no correspondance,

or, if you prefer, has been deleted from file '`bill2`'.

- The lines related to the months January and March 2009 have been slightly modified in '`bill2`', namely the values of the expenses are slightly different. Notice that the line with the expenses for January 2009 is the third one in file '`bill1`' and the second one in file '`bill2`'. This information is printed by Numdiff in the form

```
##3       <==  Jan09          $  233.56
##2       ==>  Jan09          $  234.00
```

  Analogously

```
##5       <==  Mar09          $   12.55
##3       ==>  Mar09          $   13.00
```

  says that the line for March is the fifth one in '`bill1`' and the third one in '`bill2`'.
- The line related to the total amount of the expenses appears also differently in the two files, since the amount of the expenses is different. Notice that this line is the 16th one in file '`bill1`' and the 11th one in file '`bill2`'.
- The lines related to the months February, April, June, August and October, i.e. the lines no. 4, 6, 8, 10 and 12 of '`bill1`', are not present in '`bill2`'.
- The line of '`bill1`' with the expenses for December 2009 is replaced in '`bill2`' by the line containing the value of the expenses for January 2010.
- The tenth line of '`bill2`', i.e. the empty line after the separator, is not present in '`bill1`'. With respect to '`bill1`' this line represents then an addition.

If you compare '`bill1`' and '`bill2`' without using the option '`-z @`', the result is completely misleading. This is the output of '`numdiff -V bill1 bill2`':

```
----------------
##2       <== -------------------------
##2       ==>  Jan09          $  234.00

##2       #:1   <== -------------------------
##2       #:1   ==> Jan09
@                                                  @@
##2             <==
##2       #>2   ==> $  234.00
@ Line 2 in file "bill1" is shorter than expected!
----------------
##3       <==  Jan09          $  233.56
##3       ==>  Mar09          $   13.00

##3       #:1   <== Jan09
##3       #:1   ==> Mar09
@                                                  @@
##3       #:3   <== 233.56
##3       #:3   ==> 13.00
@ Absolute error = 2.2056000000e+2, Relative error = 1.6966153846e+1
----------------
##4       <==  Feb09          $  850.77
##4       ==>  May09          $   78.25

##4       #:1   <== Feb09
##4       #:1   ==> May09
```

```
@                                                          @@
##4       #:3   <== 850.77
##4       #:3   ==> 78.25
@ Absolute error = 7.7252000000e+2, Relative error = 9.8724600639e+0
----------------
##5       <==  Mar09          $   12.55
##5       ==>  Jul09          $  443.10

##5       #:1   <== Mar09
##5       #:1   ==> Jul09
@                                                          @@
##5       #:3   <== 12.55
##5       #:3   ==> 443.10
@ Absolute error = 4.3055000000e+2, Relative error = 3.4306772908e+1
----------------
##6       <==  Apr09          $  524.00
##6       ==>  Sep09          $   10.00

##6       #:1   <== Apr09
##6       #:1   ==> Sep09
@                                                          @@
##6       #:3   <== 524.00
##6       #:3   ==> 10.00
@ Absolute error = 5.1400000000e+2, Relative error = 5.1400000000e+1
----------------
##7       <==  May09          $   78.25
##7       ==>  Nov09          $  110.00

##7       #:1   <== May09
##7       #:1   ==> Nov09
@                                                          @@
##7       #:3   <== 78.25
##7       #:3   ==> 110.00
@ Absolute error = 3.1750000000e+1, Relative error = 4.0575079872e-1
----------------
##8       <==  Jun09          $  230.00
##8       ==>  Jan10          $  200.00

##8       #:1   <== Jun09
##8       #:1   ==> Jan10
@                                                          @@
##8       #:3   <== 230.00
##8       #:3   ==> 200.00
@ Absolute error = 3.0000000000e+1, Relative error = 1.5000000000e-1
----------------
##9       <==  Jul09          $  443.10
##9       ==> -------------------------

##9       #:1   <== Jul09
##9       #:1   ==> -------------------------
@                                                          @@
##9       #>2   <== $  443.10
```

```
##9             ==>
@ Line 9 in file "bill2" is shorter than expected!
----------------
##10      <==  Aug09          $   67.65
##10      ==>

##10      #>1   <== Aug09          $   67.65
##10            ==>
@ Line 10 in file "bill2" is shorter than expected!
----------------
##11      <==  Sep09          $   10.00
##11      ==> Total           $ 1088.35

##11      #:1   <== Sep09
##11      #:1   ==> Total
@                                                    @@
##11      #:3   <== 10.00
##11      #:3   ==> 1088.35
@ Absolute error = 1.0783500000e+3, Relative error = 1.0783500000e+2
----------------
##12      <==  Oct09          $  201.45
          ==>


***  End of file "bill2" reached while trying to read line 12.
     File "bill1" has more lines than file "bill2",
     line 12 is the last one read from file "bill1"


+++  File "bill1" differs from file "bill2"
```

Numdiff compares now the first, second, third line of '`bill1`' with the first, second, third line of '`bill2`' and so on. But probably this is not what you want in such a case: what is reasonable here is to compare entries related to the same month and not lines having the same location, i.e. the same line number.

Numdiff offers also an option to run just the filter and see how it resynchronizes the two given files without doing any comparison between corresponding lines. The output of '`numdiff -z @ -f bill1 bill2`' is

```
 Month          Expenses          Month          Expenses
-------------------------     <
 Jan09          $  233.56          Jan09          $  234.00
 Feb09          $  850.77     <
 Mar09          $   12.55          Mar09          $   13.00
 Apr09          $  524.00     <
 May09          $   78.25          May09          $   78.25
 Jun09          $  230.00     <
 Jul09          $  443.10          Jul09          $  443.10
 Aug09          $   67.65     <
 Sep09          $   10.00          Sep09          $   10.00
 Oct09          $  201.45     <
 Nov09          $  110.00          Nov09          $  110.00
 Dec09          $  200.27     |    Jan10          $  200.00
```

```
-------------------------        -------------------------
                                >
Total           $ 2961.60        Total           $ 1088.35


+++  File "bill1" differs from file "bill2"
```

and shows that the filter is doing its job in the right way, associating the lines according to the month and not to the line number. Running just the filter is extremely useful in all situations when you are not sure if the filter is working as you wish. You have indeed to instruct the filter in the right way to let it work correctly, and this requires the use of different options depending on the structure of the files to compare. Since to guess the right options can be sometime tricky, running just the filter and see the result is the best way to be certain that you are setting up everything correctly. Later, see Chapter 8 [Filtering], page 50, I will explain in detail

- what the filter does behind the scenes to understand if and how it has to resynchronize the files to compare,
- and how the related options affect the action of the filter.

By the way, it is even possible to use '`-f`' without any other additional option for the filter, like in '`numdiff -f bill1 bill2`', but the result is more or less the same you would obtain by performing a byte-by-byte comparison with removal of the field delimiters.

The option '`-f`' can be followed by an argument in the form of an integer number whose meaning will be explained later, see [Use of the option -f], page 61.

Even if the output of `numdiff` is self-explanatory, in the next section I will explain in details all you have to know about it.

3.1 Output format

Let us go back to our first example. If the files '`list1`' and '`list2`' contain the data

```
accident        123     23Joshua          34.55           +3+4i          water
dog       -3455.321     cat             2.345678e-9     .0005-6.23e2i
```

and

```
Accident        123     23456          34.5500         +3.0001+4i
dog       -3455.320098  Cat     +2.345678e-9            -6.23e2i    $$$
A new line
```

respectively, then the output of the command '`numdiff list1 list2`' will be:

```
----------------
##1       #:1   <== accident
##1       #:1   ==> Accident
@                                                    @@
##1       #:3   <== 23Joshua
##1       #:3   ==> 23456
@                                                    @@
##1       #:5   <== +3+4i
##1       #:5   ==> +3.0001+4i
@ Absolute error = 1.0000000000e-4, Relative error = 2.0000000000e-5
##1       #>6   <== water
##1             ==>
@ Line 1 in file "list2" is shorter than expected!
----------------
```

```
##2       #:2   <== -3455.321
##2       #:2   ==> -3455.320098
@ Absolute error = 9.0200000000e-4, Relative error = 2.6104672633e-7
##2       #:3   <== cat
##2       #:3   ==> Cat
@                                                       @@
##2       #:5   <== .0005-6.23e2i
##2       #:5   ==> -6.23e2i
@ Absolute error = 5.0000000000e-4, Relative error = 8.0256821830e-7
##2             <==
##2       #>6   ==> $$$
@ Line 2 in file "list1" is shorter than expected!
----------------
##3             <==
##3       #>1   ==> A new line
@ Line 3 in file "list1" is shorter than expected!
----------------
          <==
##4       ==>


+++  File "list1" differs from file "list2"
```

`numdiff` prints a report on the standard output for every field of the first file which differs from the corresponding field of the second file.

First this report indicates the locations of the fields, namely the numbers of the lines where the fields appear and their positions within the line. The position in the line is "1" for the first field of a line, "2" for the second field, "3" for the third one and so on: fields are numerated starting from the left hand of the line and proceeding towards the right hand. For each report the line number is introduced by the symbol "##", while the field number by "#:". Then `numdiff` shows in what the difference consists. For instance,

```
##1       #:1   <== accident
##1       #:1   ==> Accident
@                                                     @@
```

means that the first field of the first line is "accident" in the first file, while in the second file it appears as "Accident". This difference could be then canceled by removing "accident" from the first file and inserting "Accident" in place of it. The arrows "<==" and "==>" try to visualize this idea. Analogously,

```
##2       #:2   <== -3455.321
##2       #:2   ==> -3455.320098
@ Absolute error = 9.0200000000e-4, Relative error = 2.6104672633e-7
```

means that the second field of the second line is "-3455.321" in the first file and "-3455.320098" in the second one. Since the contents of the field are numerical in both files, `numdiff` also prints the absolute and relative errors.

The absolute error (or absolute difference) is given by the absolute value of the difference between the values appearing in the two files.

The relative error (or relative difference) is actually defined in a more complicated way. If *n1* is the value appearing in the first file and *n2* is the value in the second file, then the absolute error is given by the formula $A = |n1 - n2|$, while the relative error R is given by:

- $R = 0$ if *n1* and *n2* are equal,
- *Inf* (infinity) if *n2* differs from *n1* and at least one of them is zero,

- *R = A/ min(|n1|, |n2|)* if *n1* and *n2* are both non zero and *n2* differs from *n1*. *min(|n1|, |n2|)* denotes the minimum between the absolute value of *n1* and the absolute value of *n2*.

With these definitions of absolute and relative error it turns out that *A(n2, n1) = A(n1, n2)* and *R(n2, n1) = R(n1, n2)*. In other words, the absolute/relative error does not change if you only change the order of the compared values. Since version 5 it is actually possible to let Numdiff compute the relative error always with respect to the value from the first file or always with respect to the value from the second file, instead of using the preceding formula. This can be done through the option '`-F`', see [Alternative formulas for the computation of the relative difference], page 32.

If at least one of the compared fields is not numerical, then the output line reporting absolute and relative errors is replaced by the separator:

```
@                                                    @@
```

It can happen that a line in one of the two files to compare contains more fields than the corresponding line of the other file. When this is the case, `numdiff` reports this difference by telling that a certain line (identified by its line number) appears to be shorter than expected, just as in

```
##1       #>6   <== water
##1             ==>
@ Line 1 in file "list2" is shorter than expected!
```

or in

```
##3             <==
##3       #>1   ==> A new line
@ Line 3 in file "list1" is shorter than expected!
```

In addition, `numdiff` shows the *tail* of the longer line, using the notation "#>*n*" to indicate the number *n* of the first field of the longer line for which there is no corresponding field in the shorter line. For example,

```
##1       #>6   <== water
##1             ==>
@ Line 1 in file "list2" is shorter than expected!
```

means that none of the fields of the first line starting from the sixth one has a corresponding field in the second file ('`list2`'). In this context, the symbol `<<*>>` (when it appears) is used to denote the End-Of-File, i.e. a line or the tail of a line which is located at the end of the corresponding file and does not have a terminating *newline* character.

It can also happen that one of the two files to compare has less lines than the other one. In this case, if no special option is passed to the program, `numdiff` prints the number of the first line which appears in only one of the two files and a message on the standard error telling in which of the two files the end has been prematurely reached:

```
***  End of file "list1" reached while trying to read line 4.
     File "list2" has more lines than file "list1",
     line 4 is the last one read from file "list2"
```

Unless the option '`-q`' is used (see Chapter 5 [Invoking numdiff], page 21), `numdiff` prints on standard output a message reporting the final status of the comparison. This message says either the two files are equal or they are different, just as in the example we are considering:

```
+++  File "list1" differs from file "list2"
```

3.2 Overview mode

Since version 5.6 an alternative way to display the differences between two files is available, which can be activated through the option '`-O`'. If this option is present on the command line, `numdiff` prints a side-by-side report instead of the usual one.

For example, if '`sheet1`' contains the text

```
A    1    1
B    2    4
C    3    9
D    4   16
E    5   25
F    6   36
G    7   49
H    8   64
I    9   81
J   10  100
```

and '`sheet2`' the following lines

```
A    1      1
B    2      4
C    3.3    9.03
D    4     16
E    5.5   25.05
F    6.6   36
G    7.7   49.49
H    8     64
I    9.9   81.09
```

then '`numdiff -O sheet1 sheet2`' prints this report

```
     A    1    1                              A    1      1
     B    2    4                              B    2      4
     C    3    9                            :!:C    3.3    9.03
     D    4   16                              D    4     16
     E    5   25                            :!:E    5.5   25.05
     F    6   36                            :!:F    6.6   36
     G    7   49                            :!:G    7.7   49.49
     H    8   64                              H    8     64
     I    9   81                            :!:I    9.9   81.09
     J   10  100                            :<:

     ***  End of file "sheet2" reached while trying to read line 10.
          File "sheet1" has more lines than file "sheet2",
          line 10 is the last one read from file "sheet1"


     +++  File "sheet1" differs from file "sheet2"
```

On the left side you can see the lines coming from the file specified as first on the command line, i.e. '`sheet1`', on the right side the lines from the second file of the command line, in this case '`sheet2`'. In the middle there is a *gutter* which contains one of these markers:

'`white spaces`'
: The corresponding lines are in common. That is, either the lines are identical, or the difference is ignored because of one of the options '`-s`', '`-D`', '`-I`', '`-X`', '`-a`', '`-r`', '`-P`' and '`-N`'.

‘`:!:`’ The corresponding lines have at least one field which differs.

‘`:<:`’ The files differ and only the first file contains the line.

‘`:>:`’ The files differ and only the second file contains the line.

In the case of ‘`sheet1`’ and ‘`sheet2`’ a message is printed after the report saying that the end of the second file has been prematurely reached. The two files do not have indeed the same number of lines and the filter has not been activated.

The option ‘`-O`’ can take an optional argument, which allows to set the width of the output and eventually to suppress common lines, see Chapter 5 [Invoking numdiff], page 21. The default value for the width of the side-by-side report is 130. No wonder then that the command ‘`numdiff -O40 sheet1 sheet2`’ displays a report with shorter lines:

```
A    1    1          A    1      1
B    2    4          B    2      4
C    3    9      :!: C    3.3    9.03
D    4   16          D    4     16
E    5   25      :!: E    5.5   25.05
F    6   36      :!: F    6.6   36
G    7   49      :!: G    7.7   49.49
H    8   64          H    8     64
I    9   81      :!: I    9.9   81.09
J   10  100      :<:

***  End of file "sheet2" reached while trying to read line 10.
     File "sheet1" has more lines than file "sheet2",
     line 10 is the last one read from file "sheet1"


+++  File "sheet1" differs from file "sheet2"
```

A negative argument makes that only the differences are listed in the side-by-side report, as shown by the output of the command ‘`numdiff -O40 sheet1 sheet2`’:

```
C    3    9      :!: C    3.3    9.03
E    5   25      :!: E    5.5   25.05
F    6   36      :!: F    6.6   36
G    7   49      :!: G    7.7   49.49
I    9   81      :!: I    9.9   81.09
J   10  100      :<:

***  End of file "sheet2" reached while trying to read line 10.
     File "sheet1" has more lines than file "sheet2",
     line 10 is the last one read from file "sheet1"


+++  File "sheet1" differs from file "sheet2"
```

If you set the width of the report to a too small value, it can happen that some or even all lines from the compared files appear truncated as in the output of ‘`numdiff -O24 sheet1 sheet2`’:

```
A    1        A    1
B    2        B    2
C    3   :!:  C    3.3
D    4        D    4
```

```
E    5    :!:  E    5.5
F    6    :!:  F    6.6
G    7    :!:  G    7.7
H    8         H    8
I    9    :!:  I    9.9
J   10    :<:

***  End of file "sheet2" reached while trying to read line 10.
     File "sheet1" has more lines than file "sheet2",
     line 10 is the last one read from file "sheet1"


+++  File "sheet1" differs from file "sheet2"
```

If you set the width of the report to a very small value, Numdiff ignores it and uses the default value, i.e. 130.

Notice that the numeric argument must immediately follow the option '`-O`', intermediate spaces are not allowed. This is also the case for the optional argument of '`-f`', while the options of Numdiff which require a mandatory argument permit the presence of intermediate spaces between them and the argument.

The option '`-O`' can be used together with any other option of Numdiff except for '`-f`', '`-q`', '`-U`', '`-E`', '`-V`' and '`-b`'. When '`-O`' is in use, '`-U`', '`-E`', '`-V`' and '`-b`' are ignored. If '`-q`' is present on the command line together with '`-O`', then '`-O`' is ignored. Finally, if both '`-f`' and '`-O`' are present, then the behavior depends on the order: the option which appears first on the command line is the one which matters.

Therefore, the command '`numdiff -O40 -f sheet1 sheet2`' displays the same report as '`numdiff -O40 sheet1 sheet2`', while the output of '`numdiff -f -O40 sheet1 sheet2`' is given by

```
A    1    1                        A    1      1
B    2    4                        B    2      4
C    3    9                      | C    3.3    9.03
D    4   16                        D    4     16
E    5   25                      | E    5.5   25.05
F    6   36                      | F    6.6   36
G    7   49                      | G    7.7   49.49
H    8   64                        H    8     64
I    9   81                      | I    9.9   81.09
J   10  100                      <


+++  File "sheet1" differs from file "sheet2"
```

and coincides then with the one of '`numdiff -f sheet1 sheet2`'.

The option '`-O`' can be used together with the filter to cope with the addition/deletion of lines. If the file '`sheet3`' contains the text

```
A    1      1
C    3.3    9.03
E    5.5   25.05
G    7.7   49.49
I    9.9   81.09
J    10   100.00
K    0      0.02
```

then '`numdiff -O40 sheet1 sheet3`' prints a wrong report, as in the example with files '`bill1`' and '`bill2`':

```
A    1    1           A    1      1
B    2    4     :!:   C    3.3    9.03
C    3    9     :!:   E    5.5   25.05
D    4   16     :!:   G    7.7   49.49
E    5   25     :!:   I    9.9   81.09
F    6   36     :!:   J    10   100.00
G    7   49     :!:   K    0      0.02
H    8   64     :<:

***  End of file "sheet3" reached while trying to read line 8.
     File "sheet1" has more lines than file "sheet3",
     line 8 is the last one read from file "sheet1"


+++  File "sheet1" differs from file "sheet3"
```

On the other hand, the presence of '`-z @`' makes Numdiff always compare fields of corresponding lines, as shown by the output of the command '`numdiff -O40 -z @ sheet1 sheet3`':

```
A    1    1           A    1      1
B    2    4     :<:
C    3    9     :!:   C    3.3    9.03
D    4   16     :<:
E    5   25     :!:   E    5.5   25.05
F    6   36     :<:
G    7   49     :!:   G    7.7   49.49
H    8   64     :<:
I    9   81     :!:   I    9.9   81.09
J   10  100           J    10   100.00
                :>:   K    0      0.02

+++  File "sheet1" differs from file "sheet3"
```

Side-by-side format is easy to read, but it has limitations. It generates much wider output than usual, and truncates lines that are too long to fit. Also, it relies on lining up output quite heavily, so its output looks particularly bad if you use varying width fonts, nonstandard tab stops, or nonprinting characters.

3.3 Output of the filter

The output produced just by running the filter (option '`-f`') is a side-by-side difference listing of the compared files like the one displayed by GNU sdiff. The files are listed in two columns with a gutter between them. The gutter contains one of the following markers:

'`white space`'
: The corresponding lines are in common. That is, either the lines are identical, or the difference is ignored because of one of the options '`-s`', '`-D`', '`-I`', '`-X`', '`-z`' and '`-Z`'.

'`|`'
: The corresponding lines differ, and they are either both complete or both incomplete.

'`<`'
'`(`'
: The files differ and only the first file contains the line.

‘>’
‘)’ The files differ and only the second file contains the line.

‘\’ The corresponding lines differ, and only the first line is incomplete.

‘/’ The corresponding lines differ, and only the second line is incomplete.

An input line is incomplete if its last character is not a newline. This can happen only if the line is the last one of its file. When an output line of the side by side difference listing represents two differing lines, one might be incomplete while the other is not. In this case the gutter is marked ‘\’ if the line from the first file is incomplete, ‘/’ if the line from the second file is.

Like ‘`-O`’, the option ‘`-f`’ can take an optional argument which allows to set the width of the output and eventually to suppress common lines, see Chapter 5 [Invoking numdiff], page 21 and [Use of the option -f], page 61.

More generally, the user can always make `numdiff` avoid to print, partially or totally, the messages that it would otherwise send to standard output. This can be achieved by some suitable command line options, see Chapter 5 [Invoking numdiff], page 21.

4 Installing

To successfully compile, build and install Numdiff some tools are required. The first one is an ANSI C compiler. This compiler should at least accept the option '`-o`' to write its output to a specified file, the option '`-D`' for macros predefinition, the option '`-l`' to search for a specified library, and the options '`-I`' and '`-L`' to add a given directory to the search path for include and library files respectively.

In addition, you need a POSIX implementation of the `make` utility (I used both GNU make and smake by Joerg Schilling to compile Numdiff) and a POSIX implementation of the commands `rm` and `find`. At last, you need a proper installation of GNU Texinfo (in order to install the info documentation) and a shell sh-compatible.

Numdiff has been successfully compiled and tested on:

- Slackware® GNU/Linux 10.2 with the version 3.3.6 of the GNU C Compiler (GCC),
- Slackware GNU/Linux 11 with GCC 3.4.6,
- Slackware GNU/Linux 12.2 with GCC 4.2.4,
- Slackware GNU/Linux 13 with GCC 4.3.3,
- Debian® GNU/Linux 4.0 with GCC 4.1.2 20061115 (prerelease) (Debian 4.1.1-21),
- Debian GNU/Linux 6.0.3 with GCC 4.4.5 (Debian 4.4.5-8),
- Debian GNU/Linux 7.1 with GCC 4.7.2 (Debian 4.7.2-5),
- SunOS® 5.8 with GCC 2.95.3, and
- SunOS 5.10 (i386) with the version 5.9 of the Sun C compiler.

Configuration, building and installation of Numdiff can be performed through the standard three steps:

```
./configure
make
make install
```

If you leave enabled the Natural Language Support and you also want to install the localization files (at the moment only the Italian localization is supplied), then, after '`make`', you will have to type and run

```
make install-nls
```

By default, '`make install`' will install all the files in '`/usr/local/bin`', '`/usr/local/info`' etc. You can specify an installation prefix other than '`/usr/local`' using the option '`--prefix`' in the `configure` step, for instance '`--prefix=$HOME`':

```
./configure --prefix=$HOME
```

For better control, you can use the options '`--bindir`', '`--infodir`', and so on. Type '`./configure --help`' to obtain the complete list of all the available options.

Anyway, the documentation files, including a full User Manual available in several formats (HTML, PDF and plain ASCII text), will always be put in '*`DOCDIR`*`/numdiff`', where *DOCDIR* is the path specified by the option '`--docdir`' or, if this option has not been given to `configure`, '*`PREFIX`*`/local/doc`'. Here *PREFIX* is the installation prefix specified by the option '`--prefix`' or the default '`/usr/local`'.

Once Numdiff has been installed you can remove all the files previously installed by a simple '`make uninstall`'. If you have also installed the localization files trough '`make install-nls`', then, in order to remove also these ones, use '`make uninstall-nls`' in place of '`make uninstall`'.

Between the options accepted by `configure` there are '`--enable-debug`', '`--enable-optimization`', '`--enable-nls`' and '`--enable-gmp`'.

The option '`--enable-debug`' turns on debugging when compiling the source code. This is obtained by passing to the compiler the '`-g`' option, but you can change this default debugging flag (which could not even be recognized by your compiler) by setting the environment variable `DBGFLAGS` before calling `configure`.

The option '`--enable-optimization`' turns on basic optimization when compiling the source code. This is obtained by passing to the compiler the '`-O`' option, but you can change this default flag (which could not even be recognized by your compiler) by setting the environment variable `OPTFLAGS` before calling `configure`.

The option '`--enable-nls`' turns on Natural Language Support. But you do not need to use it explicitly, since Natural Language Support is enabled by default. However, you can disable it by using '`--disable-nls`'. Disabling Natural Language Support is suggested whenever you want to install Numdiff on a system where the GNU gettext library is not present. In this case the installation of Numdiff can be accomplished, for example, through

```
./configure --disable-nls
make
make install
```

Since version 5.2.0 Numdiff uses to perform all computations the GNU Multiple Precision Arithmetic Library (also called GNU MP or GMP), if this library is available at build-time. The old internal support for multiple precision arithmetic is a fall-back in case GNU MP is absent. However, it is possible to use the internal support for multiple precision arithmetic even when GNU MP is available: it is sufficient to pass the option '`--enable-gmp=no`' or '`--disable-gmp`' to the configure script before building the program, like in

```
./configure --disable-gmp
make
make install
```

Enabling the old internal support for multiple precision arithmetic is deprecated, see [with GNU MP is better], page 63. The latest version of GNU MP is available at `ftp://ftp.gnu.org/gnu/gmp/`. See the GNU MP web page at `http://gmplib.org/` for up-to-date information on GNU MP.

5 Invoking numdiff

SYNOPSIS

```
numdiff -h|--help|-v|--version
```

or

```
numdiff [-s IFS][-D DELIMS][-a THRVAL[:RANGE|:RANGE1:RANGE2]]
[-r THRVAL[:RANGE|:RANGE1:RANGE2]][-2][-F NUM][-# NUM][-P][-N][-I]
[-c CURRNAME][-d C1C2][-t C1C2][-g N1N2][-p C1C2][-n C1C2][-e C1C2]
[-i C1C2][-X 1:RANGE][-X 2:RANGE][-E][-U][-b][-V][-O[NUM]][-q][-S]
[-z 1:RANGE][-z 2:RANGE][-Z 1:RANGE][-Z 2:RANGE][-m][-H][-f[NUM]]
[-T][-B][-l PATH][-o PATH] FILE1 FILE2
```

where *FILE1* and *FILE2* are the names of the two files to compare and *RANGE*, *RANGE1* and *RANGE2* stay for a positive integer value or for a range of integer values, like '`1-`', '`3-5`' or '`-7`'.

In the first case `numdiff` prints a short help (not so short actually :)) or/and version number, Copyright, License notice, NO-Warranty disclaimer and some information about the way it was built. In the second case `numdiff` compares the files specified by the two mandatory arguments which follow the list of the options. The complete paths of the files should be given, a directory name is not accepted. In addition, the two arguments cannot refer to the same file but one of them can be `-`, which refers to stdin.

OPTIONS

'`-s, --separators=`*IFS*'
: Specify the set of characters to use as delimiters while splitting the input lines into fields (The default set of characters is space, tab and newline). If *IFS* is prefixed with '`1:`' or '`2:`' then use the given delimiter set only for the lines from the first or the second file respectively

'`-D, --delimiters=`*DELIMS*'
: Specify the set of strings to use as delimiters while splitting the input lines into fields (The default set of characters is space, tab and newline). If *DELIMS* is prefixed with '`1:`' or '`2:`' then use the given delimiter set only for the lines from the first or the second file respectively

'`-a, --absolute-tolerance=`*THRVAL*`[:`*RANGE*`|:`*RANGE1*`:`*RANGE2*`]`'
: Set to *THRVAL* the maximum absolute difference permitted before that two numeric fields are regarded as different (The default value is zero). If a *RANGE* is given, use the specified threshold only when comparing fields whose indexes lie in *RANGE*. If both *RANGE1* and *RANGE2* are given and have the same length, then use the specified threshold when comparing a field of *FILE1* lying in *RANGE1* with the corresponding field of *FILE2* in *RANGE2*

'`-r, --relative-tolerance=`*THRVAL*`[:`*RANGE*`|:`*RANGE1*`:`*RANGE2*`]`'
: Set to *THRVAL* the maximum relative difference permitted before that two numeric fields are regarded as different (The default value is zero). If a *RANGE* is given, use the specified threshold only when comparing fields whose indexes lie in *RANGE*. If both *RANGE1* and *RANGE2* are given and have the same length, then use the specified threshold when comparing a field of *FILE1* lying in *RANGE1* with the corresponding field of *FILE2* in *RANGE2*

'`-2, --strict`'
: Consider two numerical values as equal only if both absolute and relative difference do not exceed the corresponding tolerance threshold

‘`-F, --formula=`*NUM*’
: Use the formula indicated by *NUM* to compute the relative errors. If *NUM* is 0 use the classic formula. If *NUM* is 1 compute the relative errors by considering the values in *FILE1* as sample values. If *NUM* is 2 compute the relative errors by considering the values in *FILE2* as sample values.

‘`-#, --digits=`*NUM*’
: Set to *NUM* the number of digits in the significands used in multiple precision arithmetic

‘`-P, --positive-differences`’
: Ignore all differences due to numeric fields of the second file that are less than the corresponding numeric fields in the first file

‘`-N, --negative-differences`’
: Ignore all differences due to numeric fields of the second file that are greater than the corresponding numeric fields in the first file

‘`-I, --ignore-case`’
: Ignore changes in case while doing literal comparisons

‘`-c, --currency=`*CURRNAME*’
: Set to *CURRNAME* the currency name for the two files to compare. *CURRNAME* must be prefixed with ‘`1:`’ or ‘`2:`’ to specify the currency name only for the first or the second file

‘`-d, --decimal-point=`*C1C2*’
: Specify the characters representing the decimal point in the two files to compare

‘`-t, --thousands-separator=`*C1C2*’
: Specify the characters representing the thousands separator in the two files to compare

‘`-g, --group-length=`*N1N2*’
: Specify the number of digits forming each group of thousands in the two files to compare

‘`-p, --plus-prefix=`*C1C2*’
: Specify the (optional) prefixes for positive values used in the two files to compare

‘`-n, --minus-prefix=`*C1C2*’
: Specify the prefixes for negative values used in the two files to compare

‘`-e, --exponent-letter=`*C1C2*’
: Specify the exponent letters used in the two files to compare

‘`-i, --imaginary-unit=`*C1C2*’
: Specify the characters representing the imaginary unit in the two files to compare

‘`-X, --exclude=1:`*RANGE*’
: Select the fields of the first file that have to be ignored

‘`-X, --exclude=2:`*RANGE*’
: Select the fields of the second file that have to be ignored

‘`-E, --essential`’
: While printing the differences between the two compared files show only the numerical ones

‘`-U, --dummy`’
: While printing the differences between the two compared files neglect all the numerical ones (dummy mode)

‘`-b, --brief`’
: Suppress all messages concerning the differences discovered in the structures of the two files

‘`-V, --verbose`’
: For every couple of lines which differ in at least one field print an header to show how these lines appear in the two compared files

‘`-O, --overview[=`*NUM*`]`’
: Display a side by side difference listing of the two files showing which lines are present only in one file, which lines are present in both files but with one or more differing fields, and which lines are identical. If *NUM* is zero or is not specified, output at most 130 columns per line. If *NUM* is a positive number, output at most *NUM* columns per line. If *NUM* is a negative number, do not output common lines and display at most -*NUM* columns per line.

‘`-q, --quiet, --silent`’
: Suppress all the standard output

‘`-S, --statistics`’
: Add some statistics to the standard output

‘`-z, --blur-if-numerical=1:`*RANGE*’
: Select the fields of the first file that have to be blurred during the synchronization procedure only if they turn out to be numeric

‘`-z, --blur-if-numerical=2:`*RANGE*’
: Select the fields of the second file that have to be blurred during the synchronization procedure only if they turn out to be numeric

‘`-Z, --blur-unconditionally=1:`*RANGE*’
: Select the fields of the first file that have to be unconditionally blurred during the synchronization procedure

‘`-Z, --blur-unconditionally=2:`*RANGE*’
: Select the fields of the second file that have to be unconditionally blurred during the synchronization procedure

‘`-m, --minimal`’
: During synchronization try hard to find a smaller set of changes

‘`-H, --speed-large-files`’
: During synchronization assume large files and many scattered small changes

‘`-f, --test-filter[=`*NUM*`]`’
: Run only the filter and then show the results of its attempt to synchronize the two files. If *NUM* is zero or is not specified, output at most 130 columns per line. If *NUM* is a positive number, output at most *NUM* columns per line. If *NUM* is a negative number, do not output common lines and display at most -*NUM* columns per line.

‘`-T, --expand-tabs`’
: Expand tabs to spaces in output while displaying the results of the synchronization procedure (meaningful only together with option ‘`-O`’ or ‘`-f`’)

‘`-B, --binary`’
: Treat both files as binary files (only meaningful under Doz/Windoz)

‘`-l, --warnings-to=`*PATH*’
: Redirect warning and error messages from stderr to the indicated file

‘`-o, --output=`*PATH*’
: Redirect output from stdout to the indicated file

‘`-h, --help`’
: Show help message and predefined settings

‘`-v, --version`’
: Show version number, Copyright, Distribution Terms and NO-Warranty

DIAGNOSTICS

The exit status is 1 if the two given files differ, 0 if they are equal, -1 (255) in case of error.

DEFAULT NUMERIC FORMAT (for both files to compare):

Currency name = ""

Decimal point = ‘.’

Thousands separator = ‘,’

Number of digits in each thousands group = 3

Leading positive sign = ‘+’

Leading negative sign = ‘-’

Prefix for decimal exponent = ‘e’

Symbol used to denote the imaginary unit = ‘i’

SOME EXPLANATIONS

The options ‘`-U`’, ‘`-E`’, ‘`-b`’ and ‘`-q`’ are used to hide part of the standard output of the program according to some rule.

The option ‘`-U`’ triggers the *dummy mode*. In this mode `numdiff` does not print the numerical differences. A numerical difference occurs whenever the compared fields turn out to be both of numerical type, but the field from the second file has a value which differs from the one of the field from the first file. The *dummy mode* is so called since in this mode `numdiff` does not perform the job for which I created it.

The option ‘`-E`’ triggers the *essential mode*. In this mode `numdiff` only prints the numerical differences between the files and, if there are some, the differences in the structure. The latter ones occur either when one of the files contains a line for which there is no corresponding line in the other file, or when, comparing two lines, it turns out that one of them contains a field for which there exists no corresponding field in the other line. If you are not running any filter or cutting out any fields through the option ‘`-X`’, then the differences in the structure simply consist either in a different number of lines in the two files, or in a different number of fields on a line.

The option ‘`-b`’ triggers the *brief mode*. In this mode `numdiff` does not print the differences in the structure of the two files (see above for an explanation about what they are).

The option ‘`-q`’ triggers the *quiet mode*. In this mode `numdiff` does not print anything on the standard output. The *quiet mode* is useful if you only want to know whether a couple of files are equal or not. This information can be obtained by looking at the exit status of the program.

The option ‘`-O`’ activates the *overview mode*, which makes `numdiff` print a side-by-side report in the form described in section Section 3.2 [Overview mode], page 14. The optional numeric

argument after '`-O`' must immediately follow, intermediate spaces are not allowed. The option '`-O`' can be used together with any other option of Numdiff except for '`-f`', '`-q`', '`-U`', '`-E`', '`-V`' and '`-b`'. When '`-O`' is in use, '`-U`', '`-E`', '`-V`' and '`-b`' are ignored. If '`-q`' is present on the command line together with '`-O`', then '`-O`' is ignored. Finally, if both '`-f`' and '`-O`' are present, then the behavior depends on the order: the option which appears first on the command line is the one which matters.

The option '`-V`' triggers the *verbose mode*. In this mode `numdiff` produces a richer report by printing an header whenever the compared lines differ. The header shows how and where these lines appear in the compared files. For instance, if the files '`data1`' and '`data2`' contain the data

```
12      33
22      44.5
0.008   1.002
221.12  -34.56  water
2101.21 boats
```

and

```
12      33
22.3      44.5
0.008   1.202
221.12  -34.56
2101.21 boats   dogs
```

respectively, then the command '`numdiff -V data1 data2`' will print the following output:

```
----------------
##2       <== 22      44.5
##2       ==> 22.3      44.5

##2       #:1   <== 22
##2       #:1   ==> 22.3
@ Absolute error = 3.0000000000e-1, Relative error = 1.3636363636e-2
----------------
##3       <== 0.008   1.002
##3       ==> 0.008   1.202

##3       #:2   <== 1.002
##3       #:2   ==> 1.202
@ Absolute error = 2.0000000000e-1, Relative error = 1.9960079840e-1
----------------
##4       <== 221.12  -34.56  water
##4       ==> 221.12  -34.56

##4       #>3   <== water
##4             ==>
@ Line 4 in file "data2" is shorter than expected!
----------------
##5       <== 2101.21 boats
##5       ==> 2101.21 boats   dogs

##5             <==
##5       #>3   ==> dogs
@ Line 5 in file "data1" is shorter than expected!
```

```
+++  File "data1" differs from file "data2"
```

You must care that the options '`-b`' and '`-V`' will be overridden if '`-q`' is also set.

The amount of additional information printed by '`-V`' is trivially influenced by the options that alter the way `numdiff` performs the comparisons between fields (for instance '`-a`', '`-r`', '`-2`', '`-N`', '`-P`', '`-U`', '`-E`', '`-I`', '`-X`').

In the headers printed by `numdiff` when in "verbose mode" can also appear the symbol `<<*>>`. This symbol, if present, is always located at the end of a line to mean that the line is at the end of the corresponding file and does not have a terminating **newline** character.

The option '`-S`' adds to the standard output of `numdiff` a statistical report with the following information:

- the number of numeric comparisons the program has done (this quantity, like the successive ones, is influenced by the options '`-P`' and '`-N`') and the number of those comparisons whose outcome is a relevant numerical difference.
- the largest absolute error in the set of relevant numerical differences and the corresponding relative error,
- the largest relative error in the set of relevant numerical differences together with the corresponding absolute error,
- the sum and the arithmetic mean of all absolute errors,
- the sum and the arithmetic mean of the relevant absolute errors,
- the square root of the sum of the squares of all absolute errors,
- the square root of the sum of the squares of the relevant absolute errors,
- the quadratic mean of all absolute errors, and
- the quadratic mean of the relevant absolute errors.

By relevant numerical differences and relevant absolute errors I mean those ones appearing in the output of `numdiff` when the options '`-U`', '`-f`', '`-O`' and '`-q`' are not used. The information printed by '`-S`' is not removed when this option is used together with '`-q`'.

The options '`-a`', '`-r`', '`-2`', '`-P`' and '`-N`' affect the way `numdiff` performs the comparisons between numerical values. Without any of these options, `numdiff` considers two numerical fields as equal when their difference is zero.

The option '`-a`' can be used to order that two numerical fields must be considered equal as long as their absolute difference does not exceed a certain threshold, which is specified by the argument that follows the '`-a`' option. This argument can take several forms. The basic form consists of a single numerical value, the extended form adds the specification of one or two ranges of integer values.

Independently of the form of the argument, if the absolute difference between two fields does not exceed the given threshold value, the fields are considered equal; otherwise, `numdiff` prints the difference in its report, unless some other option, for example '`-P`' or '`-N`', makes the difference unimportant. If nothing else follows the threshold value, what has been just explained applies to all comparisons between numerical fields. To see this in practice, if the file '`many_columns1`' contains the text

```
A       1        1.2      1      0.1      11.0         1.0e-1
B       2        2.4      4      0.4      24.0         1.0e-2
C       3        3.6      9      0.9      39.0         1.0e-3
D       4        4.8     16      1.6     416.0         1.0e-4
```

and the contents of the file '`many_columns2`' are given by

```
A       1.1     1.08     1.01    0.1     11.011        -1.0e-1
B       2.2     2.16     4.04    0.4     24.024        -1.0e-2
```

```
C       3.3     3.24     9.09    0.9     39.039        -1.0e-3
D       4.4     4.32    16.16    1.6    416.039        -1.0e-4
```

then the output of the command '`numdiff -a 0.5 many_columns1 many_columns2`' is

```
+++  Files "many_columns1" and "many_columns2" are equal
```

The highest absolute difference between a field from '`many_columns1`' and the corresponding field from '`many_columns2`' is given indeed by |4.32 - 4.8| = |-0.48| = 0.48, and then all numeric differences between the two files remain below the threshold value 0.5.

On the other hand, the command '`numdiff -a 0.35 many_columns1 many_columns2`' prints the report

```
----------------
##3       #:3   <== 3.6
##3       #:3   ==> 3.24
 Absolute error = 3.6000000000e-1, Relative error = 1.1111111111e-1
----------------
##4       #:2   <== 4
##4       #:2   ==> 4.4
 Absolute error = 4.0000000000e-1, Relative error = 1.0000000000e-1
##4       #:3   <== 4.8
##4       #:3   ==> 4.32
 Absolute error = 4.8000000000e-1, Relative error = 1.1111111111e-1

+++  File "many_columns1" differs from file "many_columns2"
```

since the absolute differences |3.24-3.6| = |-0.36| = 0.36, |4.4-4| = |0.4| = 0.4 and |4.32 - 4.8| = 0.48 exceed the value 0.35, and the other differences are below this threshold.

If you want that the specified threshold value applies only when comparing some particular fields, you have to use the extended form for the argument of '`-a`'. This means that after the threshold value one or two ranges of integer numbers must follow, each preceded by a colon (':'). If you specify only one range of numbers after the threshold value, `numdiff` uses the given threshold only when comparing fields whose positions lie in the specified range. Remember that the positions of the fields on a line are numbered starting from the left hand of the line and proceeding towards the right hand. For example, '`-a 0.01:2-5`' sets the threshold value to 0.01 only for the comparisons between numerical fields which occupy on their lines a position between the second and the fifth one inclusive. For the other comparisons the threshold value is left unchanged and is in particular equal to zero if it has not been explicitly set. If the files '`many_columns1`' and '`many_columns2`' are the same as before, then the command '`numdiff -a 0.5:3-6 many_columns1 many_columns2`' displays the following report

```
----------------
##1       #:2   <== 1
##1       #:2   ==> 1.1
 Absolute error = 1.0000000000e-1, Relative error = 1.0000000000e-1
##1       #:7   <== 1.0e-1
##1       #:7   ==> -1.0e-1
 Absolute error = 2.0000000000e-1, Relative error = 2.0000000000e+0
----------------
##2       #:2   <== 2
##2       #:2   ==> 2.2
 Absolute error = 2.0000000000e-1, Relative error = 1.0000000000e-1
##2       #:7   <== 1.0e-2
##2       #:7   ==> -1.0e-2
```

```
 Absolute error = 2.0000000000e-2, Relative error = 2.0000000000e+0
----------------
##3       #:2   <== 3
##3       #:2   ==> 3.3
 Absolute error = 3.0000000000e-1, Relative error = 1.0000000000e-1
##3       #:7   <== 1.0e-3
##3       #:7   ==> -1.0e-3
 Absolute error = 2.0000000000e-3, Relative error = 2.0000000000e+0
----------------
##4       #:2   <== 4
##4       #:2   ==> 4.4
 Absolute error = 4.0000000000e-1, Relative error = 1.0000000000e-1
##4       #:7   <== 1.0e-4
##4       #:7   ==> -1.0e-4
 Absolute error = 2.0000000000e-4, Relative error = 2.0000000000e+0

+++  File "many_columns1" differs from file "many_columns2"
```

since the threshold value 0.5 applies now only when comparing fields in third, fourth, fifth and sixth position, while for the other comparisons the threshold value is the default one, i.e. zero. If you want to specify a non null threshold also for the fields in second and seventh position, you can do it by using the option '`-a`' more times. The command '`numdiff -a 0.5:3-6 -a 0.25:2 -a 4e-3:7 many_columns1 many_columns2`' sets the threshold value to 0.25 for the comparisons between the fields in second position, and to 4e-3 for the comparisons of the fields in seventh position. No wonder then, that the command prints exactly this report:

```
----------------
##1       #:7   <== 1.0e-1
##1       #:7   ==> -1.0e-1
 Absolute error = 2.0000000000e-1, Relative error = 2.0000000000e+0
----------------
##2       #:7   <== 1.0e-2
##2       #:7   ==> -1.0e-2
 Absolute error = 2.0000000000e-2, Relative error = 2.0000000000e+0
----------------
##3       #:2   <== 3
##3       #:2   ==> 3.3
 Absolute error = 3.0000000000e-1, Relative error = 1.0000000000e-1
----------------
##4       #:2   <== 4
##4       #:2   ==> 4.4
 Absolute error = 4.0000000000e-1, Relative error = 1.0000000000e-1

+++  File "many_columns1" differs from file "many_columns2"
```

Observe that `:2` and `:7` are abbreviations of `:2-2` and `:7-7`, respectively. It is even possible to use range expressions like '*m*`-`' or '`-`*n*'. The first expression corresponds to all fields starting from the *m*th one (inclusive) till to the end of line, the second selects all fields from the first one till to the *n*th one, both inclusive.

If you specify two ranges of numbers after the threshold value and they have the same length (the length of a range is the difference between its maximum and its minimum), `numdiff` uses the given threshold when comparing a field of the first file lying in the first range with the corresponding field of the second file from the second range. For example, '`-a 1e-4:3-5:4-6`' sets the threshold value to 0.0001 only for the numerical comparisons of the third, fourth, and

fifth field of each line from the first file with the fourth, fifth and sixth field respectively of the corresponding line from the second file. For the other comparisons the threshold value is left unchanged and is in particular equal to zero if it has not been explicitly set. This way to restrict the application of a threshold value is useful in conjunction with the option '`-X`', which makes `numdiff` ignore one or more fields from one of the compared files.

The file '`many_columns3`':

```
A    I  1.1     1.08     1.01    0.1     11.011        -1.0e-1
B   II  2.2     2.16     4.04    0.4     24.024        -1.0e-2
C  III  3.3     3.24     9.09    0.9     39.039        -1.0e-3
D   IV  4.4     4.32    16.16    1.6    416.039        -1.0e-4
```

has one column more than the file '`many_columns1`', namely the second one. When comparing '`many_columns1`' with '`many_columns3`' it is natural then to ignore the second column of the second file. This can be achieved by passing the argument 2:2 to the option '`-X`' (for a full description of the use of this option, see [Restriction of the comparison to particular fields], page 44). Ignoring the second field of each line of '`many_columns3`' implies that the fields in the third column of this file are compared with the corresponding fields of the second column of '`many_columns1`', the fields in the fourth column of '`many_columns3`' are compared with the ones in the third column of '`many_columns1`', and so on. Therefore, if you want to set a threshold value only for the comparisons between some particular fields you have to consider that '`-X 2:2`' makes `numdiff` compare the first, second, third, fourth, fifth, sixth, and seventh field of each line of '`many_columns1`' with the first, third, fourth, fifth, sixth, seventh, and eighth field respectively of the corresponding line of '`many_columns3`'. Therefore, the command '`numdiff -X 2:2 -a 0.5:3-6 many_columns1 many_columns3`' will use 0.5 as threshold value only when comparing the third, fourth, and fifth field of a line from '`many_columns1`' with the fourth, fifth, and sixth field respectively of the corresponding line of '`many_columns3`'. This explains why the report of '`numdiff -X 2:2 -a 0.5:3-6 many_columns1 many_columns3`'

```
----------------
##1       #:2   <== 1
##1       #:3   ==> 1.1
 Absolute error = 1.0000000000e-1, Relative error = 1.0000000000e-1
##1       #:6   <== 11.0
##1       #:7   ==> 11.011
 Absolute error = 1.1000000000e-2, Relative error = 1.0000000000e-3
##1       #:7   <== 1.0e-1
##1       #:8   ==> -1.0e-1
 Absolute error = 2.0000000000e-1, Relative error = 2.0000000000e+0
----------------
##2       #:2   <== 2
##2       #:3   ==> 2.2
 Absolute error = 2.0000000000e-1, Relative error = 1.0000000000e-1
##2       #:6   <== 24.0
##2       #:7   ==> 24.024
 Absolute error = 2.4000000000e-2, Relative error = 1.0000000000e-3
##2       #:7   <== 1.0e-2
##2       #:8   ==> -1.0e-2
 Absolute error = 2.0000000000e-2, Relative error = 2.0000000000e+0
----------------
##3       #:2   <== 3
##3       #:3   ==> 3.3
 Absolute error = 3.0000000000e-1, Relative error = 1.0000000000e-1
##3       #:6   <== 39.0
```

```
##3       #:7   ==> 39.039
 Absolute error = 3.9000000000e-2, Relative error = 1.0000000000e-3
##3       #:7   <== 1.0e-3
##3       #:8   ==> -1.0e-3
 Absolute error = 2.0000000000e-3, Relative error = 2.0000000000e+0
----------------
##4       #:2   <== 4
##4       #:3   ==> 4.4
 Absolute error = 4.0000000000e-1, Relative error = 1.0000000000e-1
##4       #:6   <== 416.0
##4       #:7   ==> 416.039
 Absolute error = 3.9000000000e-2, Relative error = 9.3750000000e-5
##4       #:7   <== 1.0e-4
##4       #:8   ==> -1.0e-4
 Absolute error = 2.0000000000e-4, Relative error = 2.0000000000e+0

+++  File "many_columns1" differs from file "many_columns3"
```

does not show the same difference listing of the command '`numdiff -a 0.5:3-6 many_columns1 many_columns2`'.

If what you want is to obtain the same difference listing of '`numdiff -a 0.5:3-6 many_columns1 many_columns2`', then the right command is '`numdiff -X 2:2 -a 0.5:3-6:4-7 many_columns1 many_columns3`'. The report printed by this last command is indeed

```
----------------
##1       #:2   <== 1
##1       #:3   ==> 1.1
 Absolute error = 1.0000000000e-1, Relative error = 1.0000000000e-1
##1       #:7   <== 1.0e-1
##1       #:8   ==> -1.0e-1
 Absolute error = 2.0000000000e-1, Relative error = 2.0000000000e+0
----------------
##2       #:2   <== 2
##2       #:3   ==> 2.2
 Absolute error = 2.0000000000e-1, Relative error = 1.0000000000e-1
##2       #:7   <== 1.0e-2
##2       #:8   ==> -1.0e-2
 Absolute error = 2.0000000000e-2, Relative error = 2.0000000000e+0
----------------
##3       #:2   <== 3
##3       #:3   ==> 3.3
 Absolute error = 3.0000000000e-1, Relative error = 1.0000000000e-1
##3       #:7   <== 1.0e-3
##3       #:8   ==> -1.0e-3
 Absolute error = 2.0000000000e-3, Relative error = 2.0000000000e+0
----------------
##4       #:2   <== 4
##4       #:3   ==> 4.4
 Absolute error = 4.0000000000e-1, Relative error = 1.0000000000e-1
##4       #:7   <== 1.0e-4
##4       #:8   ==> -1.0e-4
 Absolute error = 2.0000000000e-4, Relative error = 2.0000000000e+0
```

```
+++  File "many_columns1" differs from file "many_columns3"
```

and up to the positions of the fields from ‘`many_columns3`’ coincides with the one of ‘`numdiff -a 0.5:3-6 many_columns1 many_columns2`’.

The option ‘`-a`’ can appear more times on the command line. In case of conflicts, the last setting is the one which matters. If you look at the report of the command ‘`numdiff -a 0.5:3-6 -a 0.08:4 many_columns1 many_columns2`’

```
----------------
##1       #:2   <== 1
##1       #:2   ==> 1.1
 Absolute error = 1.0000000000e-1, Relative error = 1.0000000000e-1
##1       #:7   <== 1.0e-1
##1       #:7   ==> -1.0e-1
 Absolute error = 2.0000000000e-1, Relative error = 2.0000000000e+0
----------------
##2       #:2   <== 2
##2       #:2   ==> 2.2
 Absolute error = 2.0000000000e-1, Relative error = 1.0000000000e-1
##2       #:7   <== 1.0e-2
##2       #:7   ==> -1.0e-2
 Absolute error = 2.0000000000e-2, Relative error = 2.0000000000e+0
----------------
##3       #:2   <== 3
##3       #:2   ==> 3.3
 Absolute error = 3.0000000000e-1, Relative error = 1.0000000000e-1
##3       #:4   <== 9
##3       #:4   ==> 9.09
 Absolute error = 9.0000000000e-2, Relative error = 1.0000000000e-2
##3       #:7   <== 1.0e-3
##3       #:7   ==> -1.0e-3
 Absolute error = 2.0000000000e-3, Relative error = 2.0000000000e+0
----------------
##4       #:2   <== 4
##4       #:2   ==> 4.4
 Absolute error = 4.0000000000e-1, Relative error = 1.0000000000e-1
##4       #:4   <== 16
##4       #:4   ==> 16.16
 Absolute error = 1.6000000000e-1, Relative error = 1.0000000000e-2
##4       #:7   <== 1.0e-4
##4       #:7   ==> -1.0e-4
 Absolute error = 2.0000000000e-4, Relative error = 2.0000000000e+0

+++  File "many_columns1" differs from file "many_columns2"
```

you see that 0.08 and not 0.5 is taken as threshold value for the comparison of the fields in fourth position.

Finally, if ‘`-a`’ is not present on the command line, then the default threshold value of zero applies to all comparisons of numerical fields and any non null absolute difference is considered as significant, unless some other option, for example ‘`-P`’ or ‘`-N`’, makes `numdiff` ignore it.

The option ‘`-r`’ can be used to order that two numerical fields must be considered equal as long as their relative difference does not exceed a certain threshold, which is specified by the argument that follows the ‘`-r`’ option. As for the option ‘`-a`’, the argument of ‘`-r`’ can have

several forms. These forms are the same accepted by '`-a`' and have the same meanings, but the threshold value applies to the relative difference, not to the absolute one.

The relative difference is normally defined in this way. If *n1* is a value from the file specified as first on the command line and *n2* is the corresponding value from the second file, then the absolute difference is given by the formula $A=|n1\text{-}n2|$. The relative difference R is given by:

- $R = 0$ if *n1* and *n2* are equal,
- *Inf* (infinity) if *n2* differs from *n1* and at least one of them is zero,
- $R = A / min(|n1|, |n2|)$ if *n1* and *n2* are both non zero and *n2* differs from *n1*. $min(|n1|, |n2|)$ denotes the minimum between the absolute value of *n1* and the absolute value of *n2*.

With this definition of relative difference it turns out that *R(n2, n1) = R(n1, n2)*: the relative difference does not change if you only change the ordering of the compared files on the command line.

However there are cases when this default definition of relative error makes no sense. This can happen for instance when one of the files is a sample file containing a list of expected data, which could have been computed theoretically or come from experiments in a laboratory. In this case it is more natural to define the relative difference as the ratio between the absolute difference and the absolute value of the number coming from the sample file. If you use the option '`-F`' together with the argument 1 (or 2), then Numdiff always compute the relative difference as the ratio between the absolute difference and the absolute value of the number from the first file (the second file, respectively). More precisely, with '`-F 1`' the relative difference R is computed according to these rules:

- $R = 0$ if *n1* and *n2* are equal,
- *Inf* (infinity) if *n2* differs from *n1* and *n1* is zero,
- $R = |n1\text{-}n2| / |n1|$ if *n1* is not zero and *n2* differs from *n1*.

With '`-F 2`' the rules become:

- $R = 0$ if *n1* and *n2* are equal,
- *Inf* (infinity) if *n2* differs from *n1* and *n2* is zero,
- $R = |n1\text{-}n2| / |n2|$ if *n2* is not zero and *n2* differs from *n1*.

With the last two sets of rules it is not anymore true that *R(n2, n1) = R(n1, n2)*: the relative difference changes, together with the ordering of the files on the command line. As a simple example, suppose that *file1* and *file2* contain

```
1    9.9  0.5  440
```

and

```
1.2  8    0.51 400
```

respectively. Then '`numdiff file1 file2`' displays

```
----------------
##1       #:1   <== 1
##1       #:1   ==> 1.2
 Absolute error = 2.0000000000e-1, Relative error = 2.0000000000e-1
##1       #:2   <== 9.9
##1       #:2   ==> 8
 Absolute error = 1.9000000000e+0, Relative error = 2.3750000000e-1
##1       #:3   <== 0.5
##1       #:3   ==> 0.51
 Absolute error = 1.0000000000e-2, Relative error = 2.0000000000e-2
##1       #:4   <== 440
##1       #:4   ==> 400
```

```
 Absolute error = 4.0000000000e+1, Relative error = 1.0000000000e-1

+++  File "file1" differs from file "file2"
```

'`numdiff -F 1 file1 file2`' prints

```
----------------
##1       #:1   <== 1
##1       #:1   ==> 1.2
 Absolute error = 2.0000000000e-1, Relative error = 2.0000000000e-1
##1       #:2   <== 9.9
##1       #:2   ==> 8
 Absolute error = 1.9000000000e+0, Relative error = 1.9191919192e-1
##1       #:3   <== 0.5
##1       #:3   ==> 0.51
 Absolute error = 1.0000000000e-2, Relative error = 2.0000000000e-2
##1       #:4   <== 440
##1       #:4   ==> 400
 Absolute error = 4.0000000000e+1, Relative error = 9.0909090909e-2

+++  File "file1" differs from file "file2"
```

the output of '`numdiff -F 2 file1 file2`' is

```
----------------
##1       #:1   <== 1
##1       #:1   ==> 1.2
 Absolute error = 2.0000000000e-1, Relative error = 1.6666666667e-1
##1       #:2   <== 9.9
##1       #:2   ==> 8
 Absolute error = 1.9000000000e+0, Relative error = 2.3750000000e-1
##1       #:3   <== 0.5
##1       #:3   ==> 0.51
 Absolute error = 1.0000000000e-2, Relative error = 1.9607843137e-2
##1       #:4   <== 440
##1       #:4   ==> 400
 Absolute error = 4.0000000000e+1, Relative error = 1.0000000000e-1

+++  File "file1" differs from file "file2"
```

'`numdiff -F 1 -r 0.195 file1 file2`' displays

```
----------------
##1       #:1   <== 1
##1       #:1   ==> 1.2
 Absolute error = 2.0000000000e-1, Relative error = 2.0000000000e-1

+++  File "file1" differs from file "file2"
```

and, finally, '`numdiff -F 2 -r 0.195 file1 file2`' displays

```
----------------
##1       #:2   <== 9.9
##1       #:2   ==> 8
 Absolute error = 1.9000000000e+0, Relative error = 2.3750000000e-1

+++  File "file1" differs from file "file2"
```

The option '`-2`' is only meaningful when the user specifies a non-zero tolerance threshold for both absolute and relative difference. Without this option `numdiff` considers two numerical fields equal as long as at least one between absolute and relative difference does not exceed the corresponding threshold. With the option '`-2`' `numdiff` regards two numerical fields as equal only if both absolute and relative difference do not exceed the thresholds of tolerance specified for those fields. For example, if *file1* contains the unique line

```
100
```

and *file2* the line

```
100.00012
```

then the output of the command '`numdiff` *`file1 file2`*' will be

```
----------------
##1       #:1   <== 100
                ==> 100.00012
@ Absolute error = 1.2000000000e-4, Relative error = 1.2000000000e-6

+++  File "file1" differs from file "file2"
```

The output of the commands '`numdiff -a 1.0e-4` *`file1 file2`*' and '`numdiff -r 1.0e-6` *`file1 file2`*' will be the same as above, but '`numdiff -a 1.0e-4 -r 1.3e-6` *`file1 file2`*' and '`numdiff -a 1.3e-4 -r 1.0e-6` *`file1 file2`*' will print the message

```
+++  Files "file1" and "file2" are equal
```

since the relative difference is 1.2e-6 < 1.3e-6, the absolute difference is 1.2e-4 < 1.3e-4, and it is sufficient that one of them does not exceed its tolerance threshold.

On the other hand, the commands '`numdiff -a 1.0e-4 -r 1.3e-6 -2` *`file1 file2`*' and '`numdiff -a 1.3e-4 -r 1.0e-6 -2` *`file1 file2`*' will print the message

```
----------------
##1       #:1   <== 100
                ==> 100.00012
@ Absolute error = 1.2000000000e-4, Relative error = 1.2000000000e-6

+++  File "file1" differs from file "file2"
```

since the option '`-2`' makes `numdiff` regard two values as equal only if both absolute and relative difference do not exceed the corresponding threshold of tolerance.

The option '`-P`' makes `numdiff` consider two values equal whenever the second one, i.e. the value coming from the file specified as last on the command line, is less or equal than the first one, which is the value coming from the file specified as first on the command line. If the values to compare are complex numbers, saying that the second one is less or equal than the first one means that both real and imaginary part of the second value are not greater than the real part and, respectively, the imaginary part of the first value.

Finally, the option '`-N`' makes `numdiff` consider two values equal whenever the second one, i.e. the value coming from the file specified as last on the command line, is greater or equal than the first one, which is the value coming from the file specified as first on the command line. If the values to compare are complex numbers, saying that the second one is greater or equal than the first one means that both real and imaginary part of the second value are not less than the real part and, respectively, the imaginary part of the first value.

The options '`-B`', '`-I`', '`-l`', '`-o`', '`-h`' and '`-v`' do not require further explanations. The options '`-l`' and '`-o`' are only supplied for the users of some poorly designed operating systems (like MSDog or MSWindoze), whose default shell does not allow the redirection of standard error

and standard output. The option '`-I`' has no effect on the outcome of numerical comparisons but affects the action of the filter, see Chapter 8 [Filtering], page 50.

The option '`-s`' requires as argument a set of characters, which will be taken as field delimiters. It is better to quote the set of the delimiters, just as in the next examples:

```
numdiff -s ' \t\n,;:.' file1 file2
numdiff -s ' \t\n\r\f\v"\:;' file1 file2
numdiff -s `` \t\n'' file1 file2
```

If you want to include in the set of delimiters also some special characters, e.g the **blank**, then you must quote it. I recommend you to always use the single quote character (') to enclose the list of the delimiters, since in this way you will prevent any substitution or handling of characters by the shell.

`numdiff` recognizes and interprets the following sequences of characters within the argument passed to the option '`-s`':

- '`\a`' alert (bell),
- '`\b`' backspace,
- '`\f`' form feed,
- '`\n`' newline,
- '`\r`' carriage return,
- '`\s`' blank,
- '`\t`' horizontal tab,
- '`\v`' vertical tab,
- '`\\`' backslash,
- '`\`*nnn*' the eight-bit character whose value is the octal value *nnn* (one to three digits),
- '`\x`*HH*' the eight-bit character whose value is the hexadecimal value *HH* (one or two digits).

By passing the string ' `\t\n,;:.`' as argument for the option '`-s`', one tells `numdiff` to use as field delimiters the characters **blank**, **horizontal tab**, **newline**, **comma**, **semicolon**, **colon** and **dot**. Passing ' `\t\n`' as argument to the option '`-s`' is the same as not using at all the option '`-s`', since **blank**, **horizontal tab** and **newline** are the default field delimiters.

In the list of field delimiters the character **backslash** ('\') is always treated in a special way. If it forms, combined with the subsequent character(s), one of the backslash escape sequences listed above, then it is considered to be an escape character and the whole escape sequence is decoded as shown above. Otherwise, the **backslash** is just ignored.

Therefore, the delimiters specified by the command line

```
numdiff -s' \t\n\\\"' file1 file2
```

are **blank**, **horizontal tab**, **newline**, **backslash** and **double quote**, since '`\\`' and '`\"`' are interpreted by `numdiff` as '\' and '"'.

Even if I have recommended to enclose the set of delimiters in single quotes, there are cases in which you will be constrained to use the double quote character ('`"`') to enclose the set of field delimiters, e.g. if the single quote character is used as field delimiter, like in one of the precedent examples. However you must take into account that in this case the shell could make some substitutions on the command line before executing `numdiff`. For instance, if your shell is GNU bash, then (citing the man page of GNU bash)

> Enclosing characters in double quotes preserves the literal value of all characters within the quotes, with the exception of '`$`', '`` ` ``', and '`\`'. The characters '`$`' and '`` ` ``' retain their special meaning within double quotes. The backslash retains its special meaning only when followed by one of the following characters: '`$`', '`` ` ``', '`"`', '`\`', or **<newline>**. A double quote may be quoted within double quotes by preceding it

> with a backslash ... The special parameters * and @ have special meaning when in double quotes ...

Therefore, if the set of delimiters is formed by ' ', '\t', '\n', '\' and '"', and you decide to enclose them in double quotes, the `numdiff` command line should be

```
numdiff -s'' \t\n\\\\\"'' file1 file2
```

and not

```
numdiff -s'' \t\n\\\"'' file1 file2
```

In this last case the shell would indeed replace the string

' \t\n\\\"'

by

' \t\n\"'

and then `numdiff` would take ' ', '\t', '\n' and '"' as field delimiters.

`numdiff` requires the presence of the **newline** in the set of characters passed to '`-s`'. The absence of the **newline** in the set of delimiters causes the issue of a suitable warning message and the termination of the program.

If you run Numdiff with the option '`-B`' ('`--binary`') on files created under MS-Dog/MSWindoze, then you should put the **carriage return** in the set of field delimiters. Otherwise, this character would be included in all the fields which stay at the end of a line and this would cause some undesirable effects. For instance, a number put at the end of a line would not be regarded as a numerical field by `numdiff`, since `numdiff` would consider the final **carriage return** as part of the field and this one would be then qualified as non-numerical.

You can specify different delimiters for the two files to compare by putting the prefix '`1:`' or '`2:`' in front of the set of characters passed to '`-s`'. If the argument of '`-s`' begins with '`1:`', the characters after this prefix are used as field delimiters only for the file passed as first on the command line. Analogously, if the prefix is '`2:`', then the characters after it are used as field delimiters only for the file specified as second on the command line. You can also provide an explicit set of delimiters for just one of the files to compare, in which case `numdiff` uses the default field delimiters **blank**, **tab** and **newline** for the other file. Therefore, with '`numdiff -s '1:: \n' file1 file2`' the program will take **colon**, **blank** and **newline** as delimiters for *file1*, and **blank**, **tab** and **newline** as delimiters for *file2*. The recommendations about quoting the set of delimiters are valid also in presence of a prefix.

Starting from version 5.8 `numdiff` allows to specify whole strings as field delimiters instead of single characters. To this purpose the option '`-D`' is provided. Assume that file '`register1`' and file '`register2`' contain

```
--A:    +1.0---
--B:    -2.0---
--C:    +3.0---
--D:    -4.0---
--E:    +5.0---
--F:    -6.0---
```

and

```
--a:    +1.1---
--b:    -2.2---
--c:    +3.3---
--d:    -4.4---
--e:    +5.5---
--f:    -6.6---
```

respectively. Assume in addition, you would like that the dashes at the begin and at the end of every line are treated as delimiters and then neglected during the line by line comparison. To obtain this you cannot just specify the character `-` (minus) as delimiter via the option '`-s`': if you do it, then the negative numbers appearing in the two files will be treated as positive, since the minus sign will be regarded as a delimiter. By means of the option '`-D`' you can tell `numdiff` to consider the strings `--` and `---` as field delimiters but not the single character `-`. To see this in practice, look at the output of the command '`numdiff -D ': -- --- \s \n' register1 register2`':

```
----------------
##1       #:1   <== A
##1       #:1   ==> a
                                                        @
##1       #:2   <== +1.0
##1       #:2   ==> +1.1
 Absolute error = 1.0000000000e-1, Relative error = 1.0000000000e-1
----------------
##2       #:1   <== B
##2       #:1   ==> b
                                                        @
##2       #:2   <== -2.0
##2       #:2   ==> -2.2
 Absolute error = 2.0000000000e-1, Relative error = 1.0000000000e-1
----------------
##3       #:1   <== C
##3       #:1   ==> c
                                                        @
##3       #:2   <== +3.0
##3       #:2   ==> +3.3
 Absolute error = 3.0000000000e-1, Relative error = 1.0000000000e-1
----------------
##4       #:1   <== D
##4       #:1   ==> d
                                                        @
##4       #:2   <== -4.0
##4       #:2   ==> -4.4
 Absolute error = 4.0000000000e-1, Relative error = 1.0000000000e-1
----------------
##5       #:1   <== E
##5       #:1   ==> e
                                                        @
##5       #:2   <== +5.0
##5       #:2   ==> +5.5
 Absolute error = 5.0000000000e-1, Relative error = 1.0000000000e-1
----------------
##6       #:1   <== F
##6       #:1   ==> f
                                                        @
##6       #:2   <== -6.0
##6       #:2   ==> -6.6
 Absolute error = 6.0000000000e-1, Relative error = 1.0000000000e-1
```

```
+++  File "register1" differs from file "register2"
```

The argument '`-D ': -- --- \s \n''` instructs `numdiff` to regard every occurrence of a colon (`:`), of a blank (`\s`), of a newline (`\n`), as well as every occurrence of the strings `--` and `---` as field delimiters. The minus sign in front of the negative numbers is then handled as it should be.

In general the argument to the option '`-D`' is a blank separated sequence of one or more strings each of which contains no blank. Thus, the general form of the argument to the option '`-D`' is

string1 string2 ... stringN

where *string1*, *string2*, and so on are sequences of one or more characters (strings) containing no blank.

Mind that at least one of these strings must be '`\n`'. In addition, if a string contains the newline character, this must be the only one: strings like '`#\n`', '`%%\n`', or '`\s\n`' are not allowed (entering such a string makes the program terminate after issuing a suitable warning message).

Since the blank character has a special meaning for the shell, if the argument of '`-D`' is formed by two or more strings it should be quoted either with a single ('`''`') or with a double quote ('`"`'). Quoting is also adviced if one of the strings passed to '`-D`' contains a character (or a sequence of characters) having a special meaning for the shell.

For the usage of single and double quoting to delimit the argument of '`-D`' the same warnings and recommendations apply as for the argument of '`-s`'.

If you want to set as delimiter a string which contains one or more blanks, then you have to make use of the escape sequence '`\s`', like in the example above: within the argument of '`-D`' the blank character is always interpreted as a separator of adjacent delimiters.

More generally, when writing the argument of '`-D`' the same escape sequences are allowed as for the argument of '`-s`'. This turns out to be particularly useful whenever a multibyte character is used as delimiter in (one of) the files to compare. As example consider the comparison between '`ledger1`':

```
                In                          Out

Jan       1200.00€                  1000.00€
Feb        800.40€                   650.00€
Mar       1620.50€                  1500.00€
Apr        760.00€                   900.00€

Total     4380.90€                  4050.00€

Difference: +330.90€
```

and '`ledger2`':

```
                In                          Out

Jan       1100.00€                  1000.00€
Feb        800.40€                   750.00€
Mar       1620.50€                  1700.00€
Apr        750.00€                   900.00€

Total     4270.90€                  4350.00€

Difference: -79.10€
```

Since the Euro symbol is attached to all values, `numdiff` cannot compare them in the proper way if it is run with the default field delimiters, as the output of the command '`numdiff ledger1 ledger2`' shows:

```
----------------
##3       #:2   <== 1200.00€
##3       #:2   ==> 1100.00€
                                                          @
----------------
##4       #:3   <== 650.00€
##4       #:3   ==> 750.00€
                                                          @
----------------
##5       #:3   <== 1500.00€
##5       #:3   ==> 1700.00€
                                                          @
----------------
##6       #:2   <== 760.00€
##6       #:2   ==> 750.00€
                                                          @
----------------
##8       #:2   <== 4380.90€
##8       #:2   ==> 4270.90€
                                                          @
##8       #:3   <== 4050.00€
##8       #:3   ==> 4350.00€
                                                          @
----------------
##10      #:2   <== +330.90€
##10      #:2   ==> -79.10€
                                                          @

+++  File "ledger1" differs from file "ledger2"
```

The trick to perform the comparison in the proper way consists in specifying the € symbol as field delimiter, in addition to blank, horizontal tabulation and newline. If '`ledger1`' and '`ledger2`' are encoded in UTF-8, this can be done by using the option '`-D`' with the argument '`\xE2\x82\xAC \s \t \n`', since the hexadecimal representation of € in UTF8 is given by the byte sequence 0xE2 0x82 0xAC. On my PC the output of the command '`numdiff -D '\xE2\x82\xAC \s \t \n' ledger1 ledger2`' shows that in this case `numdiff` performs indeed a numerical comparison of the values contained in the two files:

```
----------------
##3       #:2   <== 1200.00
##3       #:2   ==> 1100.00
 Absolute error = 1.0000000000e+2, Relative error = 9.0909090909e-2
----------------
##4       #:3   <== 650.00
##4       #:3   ==> 750.00
 Absolute error = 1.0000000000e+2, Relative error = 1.5384615385e-1
----------------
##5       #:3   <== 1500.00
##5       #:3   ==> 1700.00
 Absolute error = 2.0000000000e+2, Relative error = 1.3333333333e-1
```

```
----------------
##6       #:2   <== 760.00
##6       #:2   ==> 750.00
 Absolute error = 1.0000000000e+1, Relative error = 1.3333333333e-2
----------------
##8       #:2   <== 4380.90
##8       #:2   ==> 4270.90
 Absolute error = 1.1000000000e+2, Relative error = 2.5755695521e-2
##8       #:3   <== 4050.00
##8       #:3   ==> 4350.00
 Absolute error = 3.0000000000e+2, Relative error = 7.4074074074e-2
----------------
##10      #:2   <== +330.90
##10      #:2   ==> -79.10
 Absolute error = 4.1000000000e+2, Relative error = 5.1833122630e+0

+++  File "ledger1" differs from file "ledger2"
```

If '`ledger1`' and '`ledger2`' had been saved using a multi-byte encoding different from UTF-8, then the sequence of bytes which corresponds to € in this other encoding should have been passed to '`-D`'.

As for '`-s`', with '`-D`' you can specify different delimiters for the two files to compare by means of the prefixes '`1:`' and '`2:`', like in '`numdiff -D '1:\t \n' -D '2: -- \s \n' first_file second_file`'. The recommendations about quoting the set of delimiters are valid also in presence of a prefix. Mind that, if you provide an explicit set of delimiters for just one of the files to compare, `numdiff` uses the default field delimiters **blank**, **tab** and **newline** for the other file.

If you run Numdiff with the option '`-B`' ('`--binary`') on files created under MSDog/MSWindoze, you should always include the character '`\r`' in the set of field delimiters.

The option '`-s`' and '`-D`' can appear more than once on the command line. In case of conflicts, `numdiff` assumes as set of delimiters for a given file the one specified last on the command line.

By means of the option '`-#`' the user can set the number of digits in the significands used in multiple precision arithmetic. The default value is 35, the largest admissible value is 180. If `numdiff` has been linked against the GNU Multiple Precision Arithmetic Library (also called GNU MP), then the precision it uses is typically higher than the specified one. On my machine the actual value of the precision is 20 if the user gives a value between 0 and 20, 30 if the user specifies a precision between 21 and 30, 40 for a user-specified value between 31 and 40, and so on. Anyway, the actual precision is never less than the one required by the user.

Take into account that an higher precision makes the execution of `numdiff` slower. This is particularly true if `numdiff` is not using the computational routines from the GNU MP library and the files to compare contain a lot of numerical fields. In addition, you have to care that `numdiff` truncates the value of a numerical field if it has *too much* digits with respect to the current precision. To be precise, denoted by P the current value of the precision, the following rules apply.

- If `numdiff` has been built with its own internal support for multiple precision arithmetic, then
  - if a number is written in ordinary decimal notation, `numdiff` will consider, in addition to all digits of the integer part, only the first P digits of the fractional part;
  - if a value is written in scientific notation, then `numdiff` will only consider the first P digits of the fractional part of the mantissa.

- If `numdiff` uses the routines from the GNU MP library to perform its computations, the value of a numerical field is first translated into scientific notation and then only the first P digits of the fractional part of the mantissa are considered.

You can find out whether your local version of `numdiff` is relying on GNU MP or not by executing the command '`numdiff -v`'. If `numdiff` uses GNU MP, then this command will display the following message or similar (possibly translated into your mother language) among other information:

```
The software has been linked against
the GNU Multiple Precision Arithmetic Library,
version number 4.2.4.
```

If `numdiff` does not rely on GNU MP, then the displayed message will be (up to translation into your mother language)

```
The software has been built with
its own internal support for multiple precision arithmetic.
```

By means of the option '`-c`' the user can qualify a string as a symbol or name for a currency. The string passed as argument to this option is ignored by `numdiff` whenever it appears immediately before the first digit of a number. In particular, the presence of this string does not prevent a field from being considered of numeric type. By prefixing the argument of '`-c`' with '`1:`' or '`2:`' it is possible to set the currency name/symbol only for one of the compared files, or to specify different currency names for the two files. As example we consider the files '`money1`':

```
        Profits                 Expenses

        +$430.10                -$300.50
        +$750.20                -$550.02
        +$876.24                -$720.00

Totals  $2056.54               -$1570.52
```

and '`money2`':

```
        Profits                 Expenses

        USD430.10               -USD300.50
        USD750.20               -USD550.02
        USD876.24               -USD720.15

Totals    2056.54                 -1570.67
```

To properly compare them we have to tell `numdiff` that '`$`' and '`USD`' are the currency symbols for '`money1`' and '`money2`' respectively. This can be achieved by '`-c 1:$`' and '`-c 2:USD`'. The output of the command '`numdiff -c 1:$ -c 2:USD money1 money2`' is

```
    ----------------
    ##5       #:2   <== -$720.00
    ##5       #:2   ==> -USD720.15
     Absolute error = 1.5000000000e-1, Relative error = 2.0833333333e-4
    ----------------
    ##7       #:3   <== -$1570.52
    ##7       #:3   ==> -1570.67
     Absolute error = 1.5000000000e-1, Relative error = 9.5509767466e-5

    +++  File "money1" differs from file "money2"
```

as it should be.

The argument of '`-c`' may also be a multi-byte string, in particular a multi-byte string encoded in UTF-8. If your locale uses UTF-8 as encoding, you can write the argument directly in this form. For instance, you can write '`-c €`' to specify as currency name the Euro symbol. If your locale does not use UTF-8 as encoding, or UTF-8 is not supported by your terminal, you may still write an UTF-8 encoded string as a multi-byte string by specifying each single byte of every (multi-byte) character. To this purpose you can use the same octal and hexadecimal escape sequences recognized by the options '`-s`' and '`-D`'.

For example, if the files to compare are encoded in UTF-8, you can set € as currency name by adding '`-c '\xE2\x82\xAC'`' to the command line of `numdiff`, since the hexadecimal representation of € in UTF-8 is given by the sequence of bytes 0xE2 0x82 0xAC. Mind that in this case the argument of '`-c`' has to be quoted to avoid the interpretation of the hexadecimal escape sequences by the shell.

To see this in practice, if '`euro1`' contains the text

```
Profits        Expenses

+€430.10       -€300.50
+€750.20       -€550.02
+€876.24       -€720.00
```

and '`euro2`' the text

```
Profits        Expenses

+€430.10       -€300.00
+€750.20       -€550.02
+€876.00       -€720.00
```

then the report of '`numdiff -c '\xE2\x82\xAC' euro1 euro2`' is

```
----------------
##3       #:2   <== -€300.50
##3       #:2   ==> -€300.00
 Absolute error = 5.0000000000e-1, Relative error = 1.6666666667e-3
----------------
##5       #:1   <== +€876.24
##5       #:1   ==> +€876.00
 Absolute error = 2.4000000000e-1, Relative error = 2.7397260274e-4

+++  File "euro1" differs from file "euro2"
```

Please consider that '`-c`' is only provided to let `numdiff` regard a field as numeric also in presence of a currency name immediately before its first digit: `numdiff` does not know anything about currencies and can not perform any kind of conversion between them. In addition, mind that the number after the currency name can be written in any format, not only in financial notation. `numdiff` can even cope with the currency name when it appears in a complex number. For example, with '`-c EUR`' `numdiff` considers `+EUR12-EUR0.24i` and `+12-0.24i` as equal.

The options '`-d`', '`-t`', '`-g`', '`-p`', '`-n`', '`-e`' and '`-i`' can be used to instruct `numdiff` about the numeric formats used in the files which it is going to compare. The two files to compare do not have to adopt the same numeric format and then `numdiff` allows to specify different numeric formats for them. Each of the options '`-d`', '`-t`', '`-g`', '`-p`', '`-n`', '`-e`', and '`-i`' can have as argument one or two (single-byte) characters, in particular one or two digits if the option is '`-g`'. In the first case the argument refers to both files to compare, in the second case the first character is for the file specified first on the command line, the second character for the file specified last. For instance, the option '`-d`' can be used to tell `numdiff` which character(s) is(are) used to indicate the decimal point in the two files to compare. If you give the command

‘`numdiff -d_ file1 file2`’, then `numdiff` will understand that both in *file1* and in *file2* the character **underscore** (‘`_`’) is used in place of the default one (‘`.`’) to indicate the position of the decimal point in the numerical values. But if the command is ‘`numdiff -d_: file1 file2`’, then `numdiff` will understand that the decimal point is indicated by the character **underscore** in *file1*, and by **colon** (‘`:`’) in *file2*.

If you omit to use one of the options ‘`-d`’, ‘`-t`’, ‘`-g`’, ‘`-p`’, ‘`-n`’, ‘`-e`’, and ‘`-i`’, then the corresponding attribute will take its default value, see [Default Numeric Format], page 24.

You should be careful whenever you use one or more of these options. First, not all characters can be passed to them as arguments. The arguments of the option ‘`-g`’ must be digits, the arguments of the options ‘`-d`’ and ‘`-t`’ must be punctuation marks (punctuation marks are all the characters of the ASCII set for which the standard C function `ispunct` returns a non zero value), those ones of the options ‘`-p`’, ‘`-n`’, ‘`-e`’ and ‘`-i`’ must be graphical characters but digits (graphical characters are all the characters of the ASCII set for which the standard C function `isgraph` returns a non zero value).

It is not possible to set the decimal point, the thousands separator, the positive sign, the negative sign, the prefix for decimal exponent or the symbol of the imaginary unit in such a way that, for a same file, two or more of these characters come out to be equal. This rule also applies if you miss/omit to explicitly select a symbol through the appropriate option. For instance, the command ‘`numdiff -d,. file1 file2`’ will make `numdiff` abnormally terminate after printing the error message:

```
The numeric format specified for the first file is illegal,
the following symbols should be all different
while two or more of them are actually equal:

Decimal point = ‘,’
Thousands separator = ‘,’
Leading positive sign = ‘+’
Leading negative sign = ‘-’
Prefix for decimal exponent = ‘e’
Symbol used to denote the imaginary unit = ‘i’
```

With the option ‘`-d`’ we have told `numdiff` that in the first file the decimal point is indicated by the character **comma**, but at the same time we have not modified the character in use to separate the groups of thousands, which has remained the default one, i.e. **comma**, for both files to compare. In this way we have implicitly told that in *file1* the character **comma** represents both decimal point and thousands separator. Since this is not reasonable, `numdiff` refuses to work. To avoid this problem it would be sufficient to set explicitly the thousands separator by means of the option ‘`-t`’: ‘`numdiff -d,. -t., file1 file2`’. Of course, we assume here that the decimal point and the thousands separator are represented in *file1* by **comma** and **dot** respectively, in *file2* by **dot** and **comma**.

I strongly suggest you, whenever you write a file, to avoid using the same symbol to mean two different things (like would be using **comma** for both decimal point and thousands separator), it is nonsense.

At last, it is possible (but stupid) to use as argument for the options ‘`-d`’, ‘`-t`’, ‘`-g`’, ‘`-p`’, ‘`-n`’, ‘`-e`’, and ‘`-i`’ one of the characters used as delimiters in the files to compare. In such a case `numdiff` does not complain, but you have to consider that it first uses the set of field delimiters to split the files into fields and then, when it has to distinguish between numerical and non-numerical fields, it takes into account the numeric formats specified for the two files. However, it should never happen to specify as argument for one of the options ‘`-d`’, ‘`-t`’, ‘`-g`’, ‘`-p`’, ‘`-n`’, ‘`-e`’, and ‘`-i`’ a character which is also used as field delimiter: in writing a file you should avoid (and people usually avoid it) to use the same symbol to mean two different things.

What we have said also explains why the argument of the option '`-c`' should never contain one or more field delimiters.

The option '`-X`' can be used to restrict the comparison between files to a certain group of fields. This option requires as argument a range of positive integer values or eventually just one positive integer number. The argument specifies the position(s) of the fields that `numdiff` has to ignore. Remember that the fields of a line are numerated starting from the left hand of the line and proceeding towards the right hand.

The argument passed to '`-X`' can start with a prefix, which must be either '`1:`' or '`2:`'. '`1:`' refers to the file passed as first on the command line, '`2:`' to the file specified as second. With the prefix '`1:`' only the fields of the first file corresponding to the given position(s) are ignored. Similarly, if you want to ignore only fields from the second file you have to use the prefix '`2:`'.

The option '`-X`' can appear more times on the command line, in which case `numdiff` will ignore all fields located in the positions so specified. Some examples can clarify the use of ranges and prefixes. If the file '`List1`' contains the data

```
* a  1    1     1     1
* b  2    2     2     2
* c  3    3     3     3
* d  4    4     4     4
* e  5    5     5     5
```

and '`List2`' the data

```
1    1.1  1.01   A  1.001  1.0001
2    2.2  2.02   B  2.002  2.0002
3    3.3  3.03   C  3.003  3.0003
4    4.4  4.04   D  4.004  4.0004
5    5.5  5.05   E  5.005  5.0005
```

then the output of '`numdiff -X 1:1-2 -X 2:4 -X 1:6 -X 2:5-6 List1 List2`' is

```
----------------
##1       #:4   <== 1
##1       #:2   ==> 1.1
@ Absolute error = 1.0000000000e-1, Relative error = 1.0000000000e-1
##1       #:5   <== 1
##1       #:3   ==> 1.01
@ Absolute error = 1.0000000000e-2, Relative error = 1.0000000000e-2
----------------
##2       #:4   <== 2
##2       #:2   ==> 2.2
@ Absolute error = 2.0000000000e-1, Relative error = 1.0000000000e-1
##2       #:5   <== 2
##2       #:3   ==> 2.02
@ Absolute error = 2.0000000000e-2, Relative error = 1.0000000000e-2
----------------
##3       #:4   <== 3
##3       #:2   ==> 3.3
@ Absolute error = 3.0000000000e-1, Relative error = 1.0000000000e-1
##3       #:5   <== 3
##3       #:3   ==> 3.03
@ Absolute error = 3.0000000000e-2, Relative error = 1.0000000000e-2
----------------
##4       #:4   <== 4
##4       #:2   ==> 4.4
```

```
@ Absolute error = 4.0000000000e-1, Relative error = 1.0000000000e-1
##4       #:5   <== 4
##4       #:3   ==> 4.04
@ Absolute error = 4.0000000000e-2, Relative error = 1.0000000000e-2
----------------
##5       #:4   <== 5
##5       #:2   ==> 5.5
@ Absolute error = 5.0000000000e-1, Relative error = 1.0000000000e-1
##5       #:5   <== 5
##5       #:3   ==> 5.05
@ Absolute error = 5.0000000000e-2, Relative error = 1.0000000000e-2

+++  File "List1" differs from file "List2"
```

Numdiff cuts off from '`List1`' the fields in the positions 1, 2 and 6 and from '`List2`' the fields in the positions 4, 5 and 6. In this way it compares the third, fourth and fifth field of every line of '`List1`' with the first, second and third field respectively of the corresponding line of '`List2`'. An equivalent form of the command '`numdiff -X 1:1-2 -X 2:4 -X 1:6 -X 2:5-6 List1 List2`' is given by '`numdiff -X 1:1-2 -X 2:4-5 -X 6 List1 List2`' : since the sixth field is cut off from both files we can refer to it without a prefix.

As you can see, you can specify a range of fields by using the notation '*m*`-`*n*', where *m* and *n* are the field numbers of the first and of the last field in the range. It is even possible to use range expressions like '*m*`-`' or '`-`*n*'. The first expression corresponds to all fields starting from the *m*th one (inclusive) till to the end of line, the second selects all fields from the first one till to the *n*th one (inclusive). Therefore the command '`numdiff -X 1:1-2 -X 2:4 -X 1:6 -X 2:5-6 List1 List2`' is equivalent to '`numdiff -X 1:-2 -X 2:4 -X 1:6 -X 2:5- List1 List2`' or to '`numdiff -X 1:-2 -X 1:6 -X 2:4- List1 List2`'.

Mind that the largest field number you can use while writing a specification for the option '`-X`' is 32768.

If you use the option '`-X`' the exit status of `numdiff` reflects the outcome of the restricted comparison. For instance, the exit status of '`numdiff -X 8-` *file1 file2*' is 1 only if `numdiff` has found a difference in the first seven fields of *file1* and *file2*. If the two files differ only in the fields after the seventh one, then `numdiff` ends with a zero exit status.

Going back to the example with '`List1`' and '`List2`', the output of '`numdiff -X 1:1-2 -X 1:4- -X 2:2- List1 List2`' is

```
+++  Files "List1" and "List2" are equal
```

since every field of '`List1`' at position 3 is equal to the first field in the corresponding line of '`List2`'. The exit code returned by the program to the shell is zero.

The options '`-z`', '`-Z`', '`-m`', '`-H`', '`-f`', and '`-T`' influence the action of the filter and their use is then described later, Chapter 8 [Filtering], page 50. Care that '`-z`' and '`-Z`' need both an argument in the same form required by '`-X`'.

Since version 5 Numdiff accepts also long options to conform to the GNU standards. Now it is then possible, e.g., to use '`--separators='\n\t %''` or '`--separators '\n\t %''` instead of using '`-s '\n\t %''`.

The long options, which start all with two dashes, are listed at the beginning of this chapter, each one near to the corresponding short option.

The argument of a long option may or may not preceded by the `=` sign. The only exceptions are the options '`--test-filter`' and '`--overview`', for which the presence of the `=` before the argument is mandatory. Then '`--test-filter=60`' is correct while '`--test-filter 60`' is not accepted.

6 Selecting lines and fields for the comparison

Together with the version 5.x of Numdiff is shipped the program `ndselect`. Originally, I decided to create this utility in order to deal with a situation that comes out often in Numerical Analysis. Here I present a very simple example of such a situation. Let us suppose that file '`list1`' contains the values of the square root, rounded to the 20th decimal digit, for all integer numbers between 12 and 24:

```
12      3.46410161513775458705
13      3.60555127546398929312
14      3.74165738677394138558
15      3.87298334620741688518
16      4
17      4.12310562561766054982
18      4.24264068711928514641
19      4.35889894354067355224
20      4.47213595499957939282
21      4.58257569495584000659
22      4.69041575982342955457
23      4.79583152331271954160
24      4.89897948556635619639
```

and *list2* contains *suitable* approximations of the square root only for the numbers between 12 and 21 which are multiple of 3:

```
12      3.46410162002945508100
15      3.87298387096774193548
18      4.24264705882352941176
21      4.58260869565217391304
```

These approximations could have been obtained by using the famous Heron's algorithm, which, given an approximation `a` for the square root of a number `x`, computes a better approximation by the formula `a := 0.5 * (x/a + a)`. What we want now is to understand by using `numdiff` how good the approximations contained in file *list2* are. Unfortunately, we cannot execute directly the command '`numdiff list1 list2`', since in this way we would compare the approximations provided for the square roots of 15, 18, and 21 with the square roots of 13, 14, and 15 respectively. To make the comparison in the right way, one could open '`list1`' in a text editor and remove from this file all lines but the ones related to the numbers 12, 15, 18, and 21. This approach is practicable since we have to remove only a few lines: one can easily figure out how boring and inefficient would be to manually remove hundreds or thousands of lines from a file.

An expert GNU user would suggest that it is possible to automate this removal by using the well known utilities `head` and `sed`, in this particular case '`head -n 10 list1 | sed -n -e '1~3 p' > List1`'. A quick explanation for the ones who do not know how to use `head` and `sed`: the previous command extracts from '`list1`' the first 10 lines, namely the lines containing the square roots of the numbers from 12 to 21, then picks every third line starting from the first one to select only the lines related to 12, 15, 18, and 21. Finally, these lines are printed on the file '`List1`', which then looks like:

```
12      3.46410161513775458705
15      3.87298334620741688518
18      4.24264068711928514641
21      4.58257569495584000659
```

Once obtained '`List1`', we can perform the comparison between the values we are interested in by means of '`numdiff List1 list2`' . Unfortunately, this trick only works if you have installed the GNU version of `sed`, which, as far as I know, is the only one to provide the extension *first~step*

to specify line addresses. That is way I decided to implement `ndselect`, which allows to obtain the same result as above with the simpler command '`ndselect -b 1 -e 10 -s 3 list1 > List1`'

The meaning of the arguments passed to the options '`-b`', '`-e`', and '`-s`' is the following: we tell `ndselect` to print every third line of file '`list1`' (the option '`-s`' specifies the step) starting from the first one (the option '`-b`' specifies the beginning) and ending within the tenth one possibly inclusive (the option '`-e`' specifies the end). Because of the presence of the redirection operator `>`, the previous command sends to the file '`List1`' what `ndselect` would print on the screen (standard output).

Since version 5.6 `ndselect` can also be used to select particular fields of a file. Instead of printing all fields of every line, you may want to print indeed only the fields at particular positions. To do this you can employ the option '`-F`' to indicate the position of the first field to print, the option '`-L`' to indicate the position of the last field that can be printed, the option '`-I`' to set the increment when selecting the fields. In addition, the option '`-S`' can be used to specify a set of field delimiters different from the default one (which consists of **blank**, **tab** and **newline**). As for `numdiff`, the field delimiters are used to split the input lines into fields.

The option '`-S`' of `ndselect` recognizes and accepts the same escape sequences of `numdiff` options '`-s`', '`-D`', and '`-c`'.

As example consider the selection of the even fields between the second and the sixth one inclusive from the file '`many_many_columns`', whose contents are shown here:

```
A |   I | 1.1 |   1.08  |   1.01  |  0.1  |   11.011    |    -1.0e-1
B |  II | 2.2 |   2.16  |   4.04  |  0.4  |   24.024    |    -1.0e-2
C | III | 3.3 |   3.24  |   9.09  |  0.9  |   39.039    |    -1.0e-3
D |  IV | 4.4 |   4.32  |  16.16  |  1.6  |  416.039    |    -1.0e-4
E |   V | 5.5 |   5.40  |  25.25  |  2.5  |  525.416    |    -1.0e-5
F |   # |  #  |    #    |    #    |   #   |     #       |      #
```

This selection can be accomplished by means of the command '`ndselect -S '| \t\n' -F 2 -L 6 -I 2 many_many_columns`', whose output shows only the selected fields:

```
I | 1.08  |   0.1
II | 2.16  |   0.4
III | 3.24  |   0.9
IV | 4.32  |  1.6
V | 5.40  |  2.5
# |  #    |    #
```

Of course, you can also select particular fields of particular lines, as shown by the output of the command '`ndselect -S '| \t\n' -b 1 -e 5 -s 3 -F 2 -L 6 -I 2 many_many_columns`':

```
I | 1.08  |   0.1
IV | 4.32  |  1.6
```

By default, `ndselect` reuses the delimiters found in the input lines while writing the selected fields to the standard output. You can specify a custom separator by means of the option '`-O`'. This one recognizes and accepts the same escape sequences of `numdiff` options '`-s`', '`-D`', and '`-c`'. For example, '`ndselect -S '| \t\n' -b 1 -e 5 -s 3 -F 2 -L 6 -I 2 -O '\t\t' many_many_columns`' puts two horizontal tabulations after every printed field:

```
I               1.08            0.1
IV              4.32            1.6
```

Even if the implementation of a filter in `numdiff` and the addition of the option '`-X`' have made `ndselect` much less useful than in the past, this tool can still be used to handle some special cases. In addition, it can be used as a filter for other programs than `numdiff`. The complete synopsis of `ndselect` can be found in the next chapter.

7 Invoking ndselect

SYNOPSIS

```
ndselect -h|--help|-v|--version
```

or

```
ndselect [-b N][-e N][-s N][-F N][-L N][-I N][-S IFS][-D DELIMS]
[-O OSEP][-x][-l PATH][-o PATH][FILE]
```

where *FILE* is the name of the file to read from.

In the first case `ndselect` prints a short help or/and version number, Copyright, License notice and NO-Warranty disclaimer. In the second case `ndselect` prints on the standard output a subset of lines and fields from *FILE*. The complete path of *FILE* should be given, a directory name is not accepted. If no input file is specified, the program reads from the standard input.

OPTIONS

‘`-b, --beginning, --start=N`’
: Set to *N* the number of the first line to print (The default behavior is to start with line number 1)

‘`-e, --end=N`’
: Set to *N* the number of the last line that can be printed (The default behavior is to arrive till to the end of the file)

‘`-s, --step=N`’
: Set to *N* the increment to use when selecting the lines to print (The default value for the increment is 1)

‘`-F, --first-field=N`’
: Set to *N* the number of the first field to print (The default behavior is to start with field number 1)

‘`-L, --last-field=N`’
: Set to *N* the number of the last field that can be printed (The default behavior is to arrive till to the end of every line)

‘`-I, --increment=N`’
: Set to *N* the increment to use when selecting the fields to print (The default value for the increment is 1)

‘`-S, --separators=IFS`’
: Specify the set of characters to use as delimiters while splitting the input lines into fields (The default set of characters is space, tab and newline)

‘`-D, --delimiters=DELIMS`’
: Specify the set of strings to use as delimiters while splitting the input lines into fields (The default set of delimiters is space, tab and newline)

‘`-O, --output-separator=OSEP`’
: Specify the string to use as separator while writing the selected fields to the standard output (The default behavior consists in reusing the delimiters found in the input lines)

‘`-x, --omit-empty-lines`’
: Do not print empty lines

‘`-l, --warnings-to=PATH`’
: Redirect warning and error messages from stderr to the indicated file

‘`-o, --output=`*PATH*’
: Redirect output from stdout to the indicated file

‘`-h, --help`’
: Show this help message

‘`-v, --version`’
: Show version number, Copyright, Distribution Terms and NO-Warranty

Passing 0 as argument to the option ‘`-L`’ or to ‘`-e`’ is equivalent to omit this option and leave enabled the default behavior (which consists in scanning till to the end of the line and of the file, respectively).

DIAGNOSTICS

The exit status is 0 in case of normal termination, -1 (255) in case of error.

As `numdiff` does, since version 5 also `ndselect` accepts long options. Thus, instead of ‘`ndselect -b 1 -e 10 -s 3 list1 > List1`’ you can write ‘`ndselect --start=1 --end=10 --step=3 list1 > List1`’.

The usage of the option ‘`-D`’ is the same as for `numdiff`. The option ‘`-S`’ corresponds to the option ‘`-s`’ of `numdiff`.

8 Using the filter of numdiff

Since version 5 it is possible to activate a filter when calling `numdiff`, so that the program performs automatically the comparison in the desired way. Recalling the example of chapter 6, if you run the command `numdiff -z 2- -V list1 list2` you obtain the following result:

```
----------------
##1       <== 12       3.46410161513775458705
##1       ==> 12       3.46410162002945508100

##1       #:2   <== 3.46410161513775458705
##1       #:2   ==> 3.46410162002945508100
 Absolute error = 4.8917004940e-9, Relative error = 1.4121122985e-9
----------------
##2       <== 13       3.60555127546398929312
          ==>

----------------
##3       <== 14       3.74165738677394138558
          ==>

----------------
##4       <== 15       3.87298334620741688518
##2       ==> 15       3.87298387096774193548

##4       #:2   <== 3.87298334620741688518
##2       #:2   ==> 3.87298387096774193548
 Absolute error = 5.2476032505e-7, Relative error = 1.3549253331e-7
----------------
##5       <== 16       4
          ==>

----------------
##6       <== 17       4.12310562561766054982
          ==>

----------------
##7       <== 18       4.24264068711928514641
##3       ==> 18       4.24264705882352941176

##7       #:2   <== 4.24264068711928514641
##3       #:2   ==> 4.24264705882352941176
 Absolute error = 6.3717042443e-6, Relative error = 1.5018250929e-6
----------------
##8       <== 19       4.35889894354067355224
          ==>

----------------
##9       <== 20       4.47213595499957939282
          ==>

----------------
```

```
##10      <== 21       4.58257569495584000659
##4       ==> 21       4.58260869565217391304

##10      #:2   <== 4.58257569495584000659
##4       #:2   ==> 4.58260869565217391304
 Absolute error = 3.3000696334e-5, Relative error = 7.2013423303e-6
----------------
##11      <== 22       4.69041575982342955457
          ==>

----------------
##12      <== 23       4.7958315233127195416
          ==>

----------------
##13      <== 24       4.89897948556635619639
          ==>


+++  File "list1" differs from file "list2"
```

Numdiff has recognized that the lines of ‘`list1`’ with the square roots for the numbers 13, 14, 16, 17, 19, 20, 22, 23 and 24 have been deleted from ‘`list2`’. The numerical comparison has been done by likening each line of ‘`list2`’ to the line of ‘`list1`’ which displays the square root for the same integer value. The output obtained running the filter of Numdiff by `numdiff -f -z 2- list1 list2` confirms this:

```
12       3.46410161513775458705          12       3.46410162002945508100
13       3.60555127546398929312      <
14       3.74165738677394138558      <
15       3.87298334620741688518          15       3.87298387096774193548
16       4                           <
17       4.12310562561766054982      <
18       4.24264068711928514641          18       4.24264705882352941176
19       4.35889894354067355224      <
20       4.47213595499957939282      <
21       4.58257569495584000659          21       4.58260869565217391304
22       4.69041575982342955457      <
23       4.7958315233127195416       <
24       4.89897948556635619639      <


+++  File "list1" differs from file "list2"
```

If you compare the command `numdiff -z 2- -V list1 list2` with the one used for the files ‘`bill1`’ and ‘`bill2`’, see [command], page 6, you surely notice that the filter has been invoked in different ways, first with ‘`-z @`’ and then with ‘`-z 2-`’.

The *synchronization* procedure used by the filter is based on *blurring* and byte-by-byte comparison. The options ‘`-z`’ and ‘`-Z`’ are used to select which fields from which file have to be blurred. They take both an argument in the same form requested by ‘`-X`’, see [Use of the option -X], page 44, but accept additionally the special value ‘`@`’ as abbreviation for the range of fields ‘`1-`’. Then the specifications ‘`1:@`’, ‘`2:@`’ and ‘`@`’ are used to mean all fields of the first file, of the second one or of both, respectively.

Employing '`-z`' and '`-Z`' in the right way is extremely important to let the filter work as desired. For instance, `numdiff -f -z @ list1 list2` matches the lines of '`list1`' and '`list2`' in the same wrong way

```
12      3.46410161513775458705          12      3.46410162002945508100
13      3.60555127546398929312          15      3.87298387096774193548
14      3.74165738677394138558          18      4.24264705882352941176
15      3.87298334620741688518          21      4.58260869565217391304
16      4                          <
17      4.12310562561766054982     <
18      4.24264068711928514641     <
19      4.35889894354067355224     <
20      4.47213595499957939282     <
21      4.58257569495584000659     <
22      4.69041575982342955457     <
23      4.79583152331271954160     <
24      4.89897948556635619639     <


+++  File "list1" differs from file "list2"
```

as Numdiff would do without employing the filter.

It is essential then to understand what *blurring a field* means and how the filter uses blurring to match the lines of the files to compare.

After reading the files the filter removes from each of them (from their images in the memory, actually) all the fields selected by the option '`-X`', then it replaces each of the fields that have to be blurred by a special character. This special character is the same for both files and it is so chosen that it cannot appear in the text. Blurring a field means to replace it by this sort of *place card*.

After doing this, the filter converts all remaining numerical fields to a standard format and compares the files byte by byte neglecting the field delimiters. This comparison is just used to establish which lines of the first file are not present in the second, which lines of the second file are missing in the first one and how to match the remaining lines to create a one-to-one correspondence.

Only at this point `numdiff` inspects each couple of corresponding lines, splits the two lines into the constituent fields, and neglecting those ones eventually specified through the option '`-X`' compares corresponding fields as it is supposed to do, performing a numerical comparison whenever the fields are both legal numerical values.

Blurring the right fields is essential to match the lines from the two files appropriately before doing any numerical comparison. Without blurring, the numerical fields could prevent `numdiff` from an appropriate matching of the lines, in case some of these are present in only one file, by creating confusion with their (maybe small) numeric differences.

Blurring can be of two types, conditional or unconditional. The blurring is conditional if it has to be performed only for fields which turn out to be legal numerical values. The arguments of the option '`-z`' indicate which fields of which file have to be blurred **under the condition that they are recognized as numeric fields**. Non-numeric fields are left by '`-z`' untouched (no blurring occurs for them). Then '`-z 1:5-7`' makes the filter blur the 5th, 6th and 7th field of each line of the first file whenever they are recognized as numeric.

By the option '`-Z`' you can specify which fields have to be unconditionally blurred, i.e. independently of their type, numeric or not. For example, '`-Z 2:3-4`' activates the blurring of the 3th and 4th field of each line of the second file.

Going back to the comparison of the files '`list1`' and '`list2`', the option '`-z 2-`' of the command '`numdiff -z 2- -V list1 list2`' makes the filter transform the (memory copies of the) two files as

```
12      •
13      •
14      •
15      •
16      •
17      •
18      •
19      •
20      •
21      •
22      •
23      •
24      •
```

and

```
12      •
15      •
18      •
21      •
```

respectively. Here • denotes the special symbol used by the filter in the blurring procedure, even if this symbol is not actually a bullet. Since in this example space, tab and newline are used as field delimiters, the byte-by-byte comparison between the transformed files produces the same result displayed by the command '`sdiff -W`' when applied to them:

```
12      •                       12      •
13      •                     <
14      •                     <
15      •                       15      •
16      •                     <
17      •                     <
18      •                       18      •
19      •                     <
20      •                     <
21      •                       21      •
22      •                     <
23      •                     <
24      •                     <
```

If you put the blurred fields back you obtain exactly the output of '`numdiff -f -z 2- list1 list2`':

```
12      3.46410161513775458705          12      3.46410162002945508100
13      3.60555127546398929312        <
14      3.74165738677394138558        <
15      3.87298334620741688518          15      3.87298387096774193548
16      4                             <
17      4.12310562561766054982        <
18      4.24264068711928514641          18      4.24264705882352941176
19      4.35889894354067355224        <
20      4.47213595499957939282        <
21      4.58257569495584000659          21      4.58260869565217391304
```

```
22       4.69041575982342955457        <
23       4.7958315233127195416         <
24       4.89897948556635619639        <


+++  File "list1" differs from file "list2"
```

Since the second field is a numerical value in all the lines of '`list1`' and '`list2`', to use the option '`-Z`' instead of '`-z`' makes no difference in this case. The output of '`numdiff -f -Z 2- list1 list2`' is then the same of '`numdiff -f -z 2- list1 list2`'.

After this explanation you can also understand why '`numdiff -f -z @ list1 list2`' gives a wrong result. Since also the first field is always a numerical value, the option '`-z @`' makes the filter transform the two given files as

```
•       •
•       •
•       •
•       •
•       •
•       •
•       •
•       •
•       •
•       •
•       •
•       •
•       •
```

and

```
•       •
•       •
•       •
•       •
```

respectively, so that it is not anymore possible to match the lines in a reasonable way.

We consider now a typical situation where it is better to use '`-Z`' in place of '`-z`'. If the file '`Table1`' contains

```
-6       2.449490
-5       2.236068
-4       2.000000
-3       1.732051
-2       1.414214
-1       1.000000
0        0
- - - - - - - - -
1        1.000000
2        1.414214
3        1.732051
4        2.000000
- - - - - - - - -
5        2.236068
6        2.449490
7        2.645751
```

```
- - - - - - - - -
8        2.828427
9        3.000000
10       3.162278
11       3.316625
12       3.464102
- - - - - - - - -
- - - - - - - - -
13       3.605551
14       3.741657
```

and ‘`Table2`’ contains

```
-6       Not_defined
-4       Not_defined
-2       Not_defined
 0       0.000000
 2       1.414216
 4       2.000000
 6       2.449494
 8       2.828469
 10      3.162278
 12      3.464102
 14      3.741658
**********END
```

then the output of ‘`numdiff -z 1:2 -Z 2:2 -f Table1 Table2`’ is

```
     -6       2.449490              -6       Not_defined
     -5       2.236068           <
     -4       2.000000              -4       Not_defined
     -3       1.732051           <
     -2       1.414214              -2       Not_defined
     -1       1.000000           <
     0        0                      0       0.000000
     - - - - - - - - -           <
     1        1.000000           <
     2        1.414214              2       1.414216
     3        1.732051           <
     4        2.000000              4       2.000000
     - - - - - - - - -           <
     5        2.236068           <
     6        2.449490              6       2.449494
     7        2.645751           <
     - - - - - - - - -           <
     8        2.828427              8       2.828469
     9        3.000000           <
     10       3.162278              10      3.162278
     11       3.316625           <
     12       3.464102              12      3.464102
     - - - - - - - - -           <
     - - - - - - - - -           <
     13       3.605551           <
     14       3.741657              14      3.741658
                                 >  **********END
```

```
+++  File "Table1" differs from file "Table2"
```

which is exactly what is expected. On the other hand the command '`numdiff -z 2 -f Table1 Table2`' displays

```
-6       2.449490          |  -6       Not_defined
-5       2.236068          |  -4       Not_defined
-4       2.000000          |  -2       Not_defined
-3       1.732051          <
-2       1.414214          <
-1       1.000000          <
0        0                    0        0.000000
- - - - - - - - - -        <
1        1.000000          <
2        1.414214             2        1.414216
3        1.732051          <
4        2.000000             4        2.000000
- - - - - - - - - -        <
5        2.236068          <
6        2.449490             6        2.449494
7        2.645751          <
- - - - - - - - - -        <
8        2.828427             8        2.828469
9        3.000000          <
10       3.162278             10       3.162278
11       3.316625          <
12       3.464102             12       3.464102
- - - - - - - - - -        <
- - - - - - - - - -        <
13       3.605551          <
14       3.741657             14       3.741658
                           >  ********************END
```

```
+++  File "Table1" differs from file "Table2"
```

which is partially wrong.

Notice that in '`Table1`' you find (truncated to the sixth decimal digit) the square roots of the absolute values of the integer numbers between -6 and 20, plus some randomly added lines. The file '`Table2`' contains some approximations (obtained by Newton's method) for the square roots of the even numbers between -6 and 20. Since the (real) square root is not defined for negative numbers, the values corresponding to -6, -4 and -2 are replaced by `Not_defined`.

Since `Not_defined` is not a numeric value, during the execution of the last command the filter transforms '`Table1`' in this way

```
-6       •
-5       •
-4       •
-3       •
-2       •
-1       •
0        •
```

```
- - - - - - - - -
1       •
2       •
3       •
4       •
- - - - - - - - -
5       •
6       •
7       •
- - - - - - - - -
8       •
9       •
10      •
11      •
12      •
- - - - - - - - -
- - - - - - - - -
13      •
14      •
```

and ‘`Table2`’ in this other way

```
-6      Not_defined
-4      Not_defined
-2      Not_defined
 0      •
 2      •
 4      •
 6      •
 8      •
 10     •
 12     •
 14     •
**********END
```

Unfortunately the first three lines are enough to confuse the synchronization procedure, which is based on a byte-by-byte comparison with exclusion of the field delimiters, as we explained before.

You can check that this is definitely the case by looking at the output of the command ‘`sdiff -W`’ on the transformed files, which is

```
-6      •          |  -6      Not_defined
-5      •          |  -4      Not_defined
-4      •          |  -2      Not_defined
-3      •          <
-2      •          <
-1      •          <
0       •             0      •
- - - - - - - - -   <
1       •          <
2       •             2      •
3       •          <
4       •             4      •
- - - - - - - - -   <
5       •          <
```

```
6       •              6       •
7       •          <
- - - - - - - - -      <
8       •              8       •
9       •          <
10      •              10      •
11      •          <
12      •              12      •
- - - - - - - - -      <
- - - - - - - - -      <
13      •          <
14      •              14      •
                   >  ********************END
```

If we give the command '`numdiff -z 1:2 -Z 2:2 -f Table1 Table2`' instead of '`numdiff -z 2 -f Table1 Table2`', the second field of the lines of '`Table2`' is always blurred. The filter transforms then '`Table2`' into

```
-6      •
-4      •
-2      •
 0      •
 2      •
 4      •
 6      •
 8      •
 10     •
 12     •
 14     •
********************END
```

and *re-synchronizes* the files '`Table1`' and '`Table2`' in the right way.

Using the unconditional blurring is suggested in all cases when a certain field, which you want to include in the comparison (use '`-X`' to completely neglect one or more fields), is of numeric type in almost all lines of (one of) the given files. This can be the case e.g. when in some lines the content of the field is given by a special numeric value, like `Infinity`, `Inf`, `+Inf` or `-Inf`, or by `NaN`, abbreviation for *Not a Number*.

Concerning the numeric fields which are not blurred, one has to remark that the filter is not confused by differences in the numeric format. Before the byte-by-byte comparison, numeric values are converted indeed to a standard format. To offer an example of this, let us suppose that '`short1`' contains a list of numbers with their logarithms

```
0.001          -3
0.01           -2
0.1            -1
1               0
1000            3
1000000         6
1000000000      9
```

and '`short2`' the same list of numbers and logarithms, but with differences in the numeric format:

```
******************
0.0010000      -3
.0100          -2
```

```
0000.10         -1
1.               0
1,000.000        3
1,000,000.       6
1,000,000,000    9
```

Then '`numdiff -f -z 2- short1 short2`' displays

```
                                  >  ******************
     0.001          -3                0.0010000       -3
     0.01           -2                .0100           -2
     0.1            -1                0000.10         -1
     1               0                1.               0
     1000            3                1,000.000        3
     1000000         6                1,000,000.       6
     1000000000      9                1,000,000,000    9


     +++  File "short1" differs from file "short2"
```

showing that the filter has matched the lines in the right way.

The filter can even handle the case when the same numerical value is written in decimal notation in one file and in scientific notation in the other one. If the files '`decimal`' and '`scientific`' contain

```
.001           -3
.01            -2
.1             -1
* * * * * * * * *
1               0
1000            3
1000000         6
1000000000      9
```

and

```
******************
1.0e-3         -3
1.0e-2         -2
1.0e-1         -1
1.0e0           0
1.0e3           3
1.0e6           6
1.0e9           9
******************
```

respectively, then '`numdiff -f -z 2- decimal scientific`' shows

```
                                  >  ******************
     .001           -3                1.0e-3         -3
     .01            -2                1.0e-2         -2
     .1             -1                1.0e-1         -1
     * * * * * * * * *           <
     1               0                1.0e0           0
     1000            3                1.0e3           3
     1000000         6                1.0e6           6
     1000000000      9                1.0e9           9
                                  >  ******************
```

```
+++  File "decimal" differs from file "scientific"
```

proving that the filter does not get confused.

No problems come out also in the case when for the same not blurred field the scientific notation is used in both files. If the files '`sc1`' and '`sc2`' contain

```
  1.E-3          -3
  1.00E-2        -2
  1.0E-1         -1
  1.0000E0        0
001.0E3           3
+01.000E6         6
  1.0E+09         9
  1.0E+10        10
* * * * * * * * * *
```

and

```
*****************
 1.0e-003      -3
 1.0e-2        -2
 1.0e-1        -1
 1.0e0          0
+1.0e3          3
 1.0e+6         6
 1.0e9          9
```

respectively, then '`numdiff -f -z 2- sc1 sc2`' correctly displays

```
                                >  *****************
      1.E-3          -3             1.0e-003      -3
      1.00E-2        -2             1.0e-2        -2
      1.0E-1         -1             1.0e-1        -1
      1.0000E0        0             1.0e0          0
    001.0E3           3            +1.0e3          3
    +01.000E6         6             1.0e+6         6
      1.0E+09         9             1.0e9          9
      1.0E+10        10         <
    * * * * * * * * * *          <
```

```
+++  File "sc1" differs from file "sc2"
```

The filter can even handle an improper use of the scientific notation, meaning for example that it can recognize '`123.456E+2`' and '`1.23456E+4`' as equal.

We can see this in the case of the files '`Scnot1`':

```
-------------------------
1.2E0             *     1
2.45E-1           *     2
-3.678E-2         *     3
```

and '`Scnot2`':

```
12E-1             *     1
245E-3            *     2
-0.003678E+1      *     3
```

'`numdiff -f -z 3- Scnot1 Scnot2`' displays the report:

```
-------------------------    <
1.2E0              *     1       12E-1              *     1
2.45E-1            *     2       245E-3             *     2
-3.678E-2          *     3       -0.003678E+1       *     3


+++  File "Scnot1" differs from file "Scnot2"
```

which is exactly what you would expect in such a case. Also pretty hard cases do not confuse the filter. If '`Scnot1`' is given by

```
 1.2000e0          *     1
02.4500e-1         *     2
-003.678E-2        *     3
```

and '`Scnot2`' is the same file as before, the output of the command '`numdiff -f -z 3- Scnot1 Scnot2`' is still right:

```
 1.2000e0          *     1       12E-1              *     1
02.4500e-1         *     2       245E-3             *     2
-003.678E-2        *     3       -0.003678E+1       *     3


+++  Files "Scnot1" and "Scnot2" have the same structure
```

Till now we have always used the option '`-f`' with no argument.

But '`-f`' accepts an optional argument, which can be used to control how '`-f`' displays its output. If you provide an argument, care not to leave any space between the option and the argument: '`-f60`' is correct while '`-f 60`' makes Numdiff terminate after printing an error message.

If the argument is a positive number *NUM*, then the side-by-side output produced by '`-f`' will be *NUM* columns wide. The default value for the width of the output is 130, which can fit onto a traditional printer line, and is the one used when '`-f`' has no argument, or the supplied argument is zero. In other words, '`-f`' and '`-f0`' are just easier to remind versions of '`-f130`'.

A negative argument has the same effect as the positive number with the same absolute value, but it causes in addition the removal of common lines from the output. For example, the command '`numdiff -z 1:2 -Z 2:2 -f-130 Table1 Table2`' displays the following text

```
-5       2.236068          <
-3       1.732051          <
-1       1.000000          <
- - - - - - - - - -       <
1        1.000000          <
3        1.732051          <
- - - - - - - - - -       <
5        2.236068          <
7        2.645751          <
- - - - - - - - - -       <
9        3.000000          <
11       3.316625          <
- - - - - - - - - -       <
- - - - - - - - - -       <
13       3.605551          <
                           >  ********************END
```

```
+++  File "Table1" differs from file "Table2"
```

In conjunction with the option '`-f`' or '`-O`' you can use '`-T`' to expand tabs to spaces in the output produced by '`-f`' / '`-O`'. This is useful to preserve the alignment of tabs in the input files, if it is thrown off by the presence of the gutter.

The options '`-H`' and '`-m`' affect the performance of the filter of Numdiff. But performance has more than one dimension and these options improve one aspect of performance at the cost of another, or they improve performance in some cases while hurting it in others.

The way that the filter re-synchronizes two files to compare always comes up with a near-minimal set of deletions/insertions of lines. Usually it is good enough for practical purposes. If the filter displays a large set of line deletions/insertions, you might want it to use a modified algorithm that sometimes produces a smaller set of differences. The '`-m`' option does this; however, it can also cause the filter to run more slowly than usual, so it is not the default behavior.

If the files you are comparing are large and have small groups of changes scattered throughout them, you can use the '`-H`' option to make a different modification to the algorithm that the filter uses. If the input files have a constant small density of changes, where change means deletion/insertion of lines, this option speeds up the comparisons without changing the output or in the worst case introducing minor modifications.

9 Warnings

- Bug reports have to be sent to the address ivprimi(at)libero(dot)it . Please, put Numdiff in the subject and indicate the version of the operating system you are running (in particular, do not forget to specify if it is a 32- or a 64-bit system), and, if you know it, the version of the compiler used to build Numdiff. Please write also whether your version of Numdiff uses the GNU MP library or not. Before writing an email be sure to run the latest stable version of Numdiff, I do not provide support for older versions.
- Numdiff does not accept numbers in scientific notation whose exponents lie outside the range -1073741824, ..., +1073741824. If such a number is found in any of the files to compare, the execution of the program is stopped after printing a suitable error message on stderr. Under the assumption that the numeric format in use is the default one, with "1.0001e-2147483640" the displayed error messages is

  ```
  numdiff: A number with a too small exponent has been found,
  namely "1.0001e-2147483640".
  Exponents smaller than -1073741824 are not accepted,
  the execution of the program ends now
  ```

- If Numdiff has been built with its own internal support for multiple precision arithmetic instead of being linked against the GNU MP Library, then performance degradation and memory exhaustion can already make impossible to handle exponents of magnitude $10^6 = 1000000$. This is what I obtained on my laptop, equipped with a dual core processor @1.50 GHz and with 1GB of RAM, when I tried to compare the numbers 1.101e1000000000 and 1.0e1000000000:

  ```
  numdiff: Insufficient memory for new allocation,
  the execution of the program ends now
  ```

  In addition, you can overload the processor with numbers whose exponents lie outside the range -1000000, ..., 1000000. But at least on my machine, everything works fine and quick enough as long as exponent and size of the mantissa of the numbers are in the range -1000, ..., 1000. Be careful and remember that Numdiff is distributed WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

  Consider also that, if you have numeric data with exponents outside the range -300, ..., 300, probably there is something wrong with your data: either you are using the wrong scale, or you should replace very small numbers, like 1e-100, by zero, or it is quite likely that the machine/program/algorithm which produced these data is not working right.
- If Numdiff has been linked against the GNU Multiple Precision Arithmetic Library (also called GNU MP), then the precision it uses is typically higher than the specified one. On my machine the actual value of the precision is 20 if the user gives a value between 0 and 20, 30 if the user specifies a precision between 21 and 30, 40 for a user-specified value between 31 and 40, and so on. Anyway, the actual precision is never less than the one required by the user.
- After reading a numeric field, Numdiff truncates its value if this number has *too much* digits with respect to the current precision. To be precise, denoted by P the current value of the precision, the following rules apply.

  If `numdiff` has been built with its own internal support for multiple precision arithmetic, then

  - if the number is written in ordinary decimal notation, `numdiff` will consider, in addition to all digits of the integer part, only the first P digits of the fractional part;
  - if the value is written in scientific notation, then `numdiff` will only consider the first P digits of the fractional part of the mantissa.

If `numdiff` uses the GNU MP library to perform its computations, the value of a numeric field is first translated into scientific notation and then only the first P digits of the fractional part of the mantissa are considered.

By current value of the precision I mean the integer value specified by the option '`-#`', or the default one (35) when this option is not in use.

- You can find out whether your local version of `numdiff` is relying on GNU MP or not by executing the command '`numdiff -v`'. If `numdiff` uses GNU MP, then this command will display the following message or similar (possibly translated into your mother language) among other information:

```
The software has been linked against
the GNU Multiple Precision Arithmetic Library,
version number 4.2.4.
```

  If `numdiff` does not rely on GNU MP, then the displayed message will be (up to translation into your mother language)

```
The software has been built with
its own internal support for multiple precision arithmetic.
```

- Numdiff can only be used on text files: the program terminates after printing a suitable error message if one of the files to compare turns out to be a binary file. To detect if a file is binary or not, `numdiff` checks for the presence of null bytes (0x00) in the file.
- If you are not including the so called white-space characters (usually ' ', '`\t`', '`\f`', '`\v`' and '`\r`') in the set of field delimiters, then a real and an imaginary number which are separated just by white-spaces can be coupled together and considered as a whole complex number. For example, if you are using only **colon** ('`:`') and **newline** as field delimiters and Numdiff finds a line like that

```
::::3.0-5.6e-356i::::-12.9   +4.34i::::-12.9    4.34i::::New York::::
```

  then it will consider this line as formed by four fields, the first two are numeric and given by the complex numbers *3.0-5.6e-356i* and *-12.9+4.34i*, the last two ones are the strings `New York` and `-12.9  4.34i`. I still do not know if I will modify this in the next version of Numdiff, so that the program recognizes only `3.0-5.6e-356i` as numeric field and treats `-12.9  +4.34i` as non-numeric due to the presence of spaces in the middle. `-12.9  4.34i` is already considered as non-numeric due to the absence of a leading sign in the imaginary value.
- We have seen that one of the two files passed to `numdiff` can be `-`, which refers to stdin (standard input). In this way one of the two files to compare can be the output produced by another command, like in '`cat file2 | numdiff -a 1.0e-3 file1 -`'. However, if you activate the filter by means of the options '`-z`' or/and '`-Z`', Numdiff can not work with the standard input unless you use also the option '`-f`'. Therefore, the command '`cat file2 | numdiff -a 1.0e-3 -z @ file1 -`' displays only the error message

```
numdiff: -: Illegal seek
```

  (or maybe the translation of this message in the language you are using on your computer) but '`cat file2 | numdiff -a 1.0e-3 -z @ -f file1 -`' works as expected.
- This manual describes the version 5.8 of Numdiff. Prior 5.x versions did not recognize all the options that are currently accepted, versions 4.0.0 and 3.x used even a different format for the output.

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index