CERN-ACC-NOTE-2013-0005
August 14, 2013
revised April 15, 2014
laurent.deniau@cern.ch

Ndiff Reference Manual

Laurent Deniau
CERN – BE/ABP

Keywords: Numeric diff, numeric constraints, regression tests, data validation.

Summary

`ndiff` *is an efficient and flexible tool designed to compare unformatted text files with numerical content. It is well suited for regression testing, for validating data versus templates, and for filtering data following templates. This technical note is the reference manual of the* `ndiff` *tool initially developed for the MAD-X test system and subsequently extended for data validation.*

Contents

1 Motivation

The `ndiff` tool is a program developed for the test system of the MAD-X application [1]. The purpose of this tool is to compare line by line unformatted text files with numerical content in a portable way. The portability of the comparison is resulting from the acceptance of (user-defined) small numerical differences and varying number representation that occur across runs, compilers and platforms. For example, `ndiff` will consider to be equal the numbers `0.001`, `1e-3`, `100.00e-5` and `0.0001e+001`, while `diff`-like Unix tools will report differences.

To the knowledge of the author, the `ndiff` tool does not seem to have any equivalent freely available on the world wide web, despite the obvious interest and need for the scientific community. Other similar open source tools exist like `numdiff` [2] or `ndiff` [3] (same name but not same tool), but they do not provide the flexibility required by the test system of MAD-X. Many options and commands have been added to `ndiff` in order to solve problems related to testing MAD-X, hence examples will often refer to the MAD-X test system.

The `ndiff` tool is written entirely in the C programming language with no external dependencies to ensure maximum portability and performance. It is very efficient to deal with both large input of data and large number of rules (i.e. user-defined constraints). `ndiff` can process more than 100 MB of data per second (i.e. nearly at the rate of disk I/O) and compare few millions numbers per second on recent computers.

2 Principle

The main application of the `ndiff` tool is to compare line by line unformatted text files with numerical content in a portable way. Comparing two files is as simple as:

`ndiff` *fileA fileB*

During the comparison, `ndiff` recognizes and interprets numbers and compares their respective values. In its simplest form, the *accepted error* ϵ between two numbers x and y read from the two input files is computed using the formula $|x-y| \leq \epsilon$. For each rejected difference between *numbers*, `ndiff` reports the discrepancy on the console with a message starting with the tag `numbers:`. This indicates that the content of the input files has been effectively recognized and interpreted as numbers.

Everything not interpreted as numbers in the input files is compared as strings, that is compared character by character for strict equality. For each encountered difference between *strings*, `ndiff` reports the discrepancy on the

console with a message starting with the tag `strings:`. This indicates that the content of the input files has NOT been interpreted as numbers, even if digits are present in the sequence of characters.

Unlike the `diff`-like Unix tools, `ndiff` does not try to match shifted inputs, and immediately reports any difference between strings.

2.1 Configuration

The command used previously to compare two files relies on the default behavior of `ndiff` (i.e. rule `#0`). In the absence of user-defined configuration file, `ndiff` compares numbers within the machine precision. For IEEE 754 standard representation, the smallest positive value for double precision floating point numbers is $eps =$ `2.22507e-308`. This value is often too small to ensure good portability of the comparisons due to the variations inherent to floating point numbers: code generated by compilers, libraries provided by platforms, type of architectures, precision of conversions and representations of numbers, rounding errors in numerical expressions, non-associativity of associative operations on floating point numbers, etc... Hence the user will often need to provide extra settings to `ndiff` to relax on purpose the error tolerances applied to the encountered differences. Comparing two files with a custom configuration is as simple as:

`ndiff` *fileA fileB config*

The configuration file is optional, but generally needed to customize the comparison of the two files and tune it for portability. It contains the definition of the *rules* that configures each comparison of numbers encountered in the input files, and much more as we will see in the next section.

3 Rules

The `ndiff` configuration file is a sequence of rule definitions that apply to the two files under comparison. The configuration is not persistent, that is all the rules are loaded before the comparison starts and discarded afterward. This might be important in the case of chaining multiple comparisons into a single `ndiff` command (see 6, Running modes).

The remaining of this section describes the syntax and the meaning of the rules defined in the configuration file in order to customize the comparison of two input files. In the following of this document, *optional content is enclosed with matching pairs of square brackets.*

```
# Test config for the Jacobian knobs
# file test-jacobian-knobs.cfg

# rows    cols    constraints
1-7       *       skip         # head banner
149-$     *       skip         # tail banner

# first matching
37-38     1-2     rel=1e-12
39        2       abs=1e-21    # from job
41        1       rel=1e-12

# second matching
109-110   1-2     rel=1e-12
111       2       abs=1e-21    # from job
113       1       rel=1e-12
```

Figure 1: Example of `ndiff` configuration file used for the regression test of the MAD-X matching command using the Jacobian algorithm with knobs.

3.1 Format

The format of the configuration file is simple and line–oriented as shown in figure 1. Each line can define a **single rule** with three fields: the range of rows, the range of columns and the actions or constraints relevant over these ranges. For each numbers comparison, `ndiff` selects the active rule amongst these definitions. If some rules are defined with overlapping ranges, **rules defined first have lower priority than rules defined last** in the configuration file. This allows to `ndiff` to always uniquely determine the active rule.

3.2 Numbers and identifiers

The range of column is defined in term of the count of encountered numbers on the current line scanned. Because the text representation of numbers varies between compilers and platforms, using the count of characters column would be improper to specify the active rules. Therefore we need an unambiguous definition of what must be interpreted as a number to get the proper count of columns.

Numbers A number is defined by the following formatting:

- Signs are optional for mantissa and exponent; a single space is accepted as a replacement for the optional `+` sign in front of the mantissa.
- Decimal point is optional, but *at least one digit must appear before or after* the decimal point (if any).
- Exponent is optional; valid exponent separators are `d`, `e`, `D`, `E`.
- Prefix delimiters can be spaces (white space, `\t`, `\n`, `\r`), and any punctuation character that is not part of identifiers.
- Suffix delimiters are any non-digit characters following the decimal point or the exponent (if any).

The following regular expression [4] defines the input format of numbers:

```
sign     ::= [-+ ]
integer  ::= sign? [0-9]+
decimal  ::= ([0-9]+\.[0-9]*) | ([0-9]*\.[0-9]+)
exponent ::= [edED] [+-]? [0-9]+
float    ::= sign? decimal exponent?
number   ::= integer | float
```

Examples of numbers (enclosed in boxes):

3, .3, 3., 3.2, 1.5e5, 3.4d3, 1.e5, .2e−3

−15, +12, +−10, %05f, ?5, <5, =5, 0.15.6, .15.6, .15 .6

15MB, MB 15, MB 15., MB .15, 15.9e−2 $\mathrm{m.s^{-1}}$

Identifiers An identifier is any sequence of non-space characters that is not interpreted as a number. Identifiers can contain sequence of digits as long as that sequence cannot be interpreted as a number, generally because of the prefix delimiters. By default, `ndiff` accepts the punctuation characters `$` (dollar), `_` (underscore) and `.` (dot) as part of identifiers, i.e. valid MAD-X tokens. This can be configured at runtime through the command line option `--punct '`*chrs*`'` (see 6), or permanently at compile time (see 8). Note that allowing '`+-`' and spaces as valid identifier characters can lead to unexpected results.

The following regular expression [4] defines the input format of identifiers:

```
letter     ::= [A-Za-z]
digit      ::= [0-9]
punct      ::= [_.$]
identifier ::= (letter | punct)(letter | digit | punct)*
```

Examples of identifiers with sequence of digits not interpreted as numbers:

_15MB, MB15, MB15., MB.15, MB.15A, MB15.0e-3, MQ.A2.3, _15, _15.6d3

3.3 Comments

A comment starts with the characters `#` or `!` and lasts until the end of the line.

3.4 Ranges

A range selects the rows or the columns affected by an action or a constraint. The count of rows and columns starts from one and increases as the comparison of the two input files progresses. The *row number* corresponds to the count of lines read in the current file. The *column number* corresponds to the count of numbers read on the current line: the first number is at column one, the second number is at column two, etc... The column *zero* exists and is used to activate rules that apply while scanning for the first number of the current line. It is worth understanding what `ndiff` interprets as a number (see 3.2) to specify correctly the column ranges.

Row and column ranges can take the following forms:

- An integer n to specify the single n-th row or column. The special character `$` represents the last row or column.
- A slice *start* `:` *size* [`/` *stride*] to specify *size* rows or columns starting from *start*, optionally with steps of *stride*. For example, `5:2` specifies the range {`5`, `6`}, and `11:3/5` specifies the range {`11`, `16`, `21`}.
- A range *start* `-` *end* [`/` *stride*] to specify rows or columns starting from *start*, ending at *end* (included), optionally with steps of *stride*. For example, `5-6` specifies the range {`5`, `6`}, and `9-21/5` specifies the range {`9`, `14`, `19`}.
- An asterisk `*` to specify all rows or columns, this is a shortcut for `0-$`.

3.5 Constraints

The constraints are at the heart of `ndiff`, as they allow to release the accepted error difference ϵ between two input numbers x and y, i.e. $|x - y| \leq \epsilon$. Constraints can include any combination of the following commands and qualifiers:

- `abs=`ϵ specifies the absolute error ϵ in $-\epsilon \leq x - y \leq \epsilon$, with $\epsilon \in [0, 1]$.
- `rel=`ϵ specifies the relative error ϵ in $-\epsilon \leq \frac{x-y}{\min(|x|,|y|)} \leq \epsilon$, with $\epsilon \in [0, 1]$. If $\min(|x|, |y|) = 0$, the relative error becomes infinite so it is converted to an absolute error, i.e. $\min(|x|, |y|) = 1$. This constraint does not apply to comparison of integer numbers.
- `dig=`ϵ specifies the relative digital error ϵ in $-\varepsilon \leq \frac{x-y}{\min(|x|,|y|)} \leq \varepsilon$, with $\varepsilon = \epsilon \cdot 10^{-\max(\#x,\#y)}$ and $\epsilon \geq 1$, and where $\#a$ is the number of digits in the mantissa of a as read from the input file after discarding the leading zeros. Hence, ϵ is the amount of precision (e.g. number of trailing digits) that the user is ready to discard. If $\min(|x|, |y|) = 0$, the error becomes absolute, i.e. $\min(|x|, |y|) = 1$. This constraint does not apply to comparison of integer numbers.
- `-abs=`ϵ' specifies the negative absolute error in $\epsilon' \leq x - y \leq \epsilon$, with $\epsilon' \in [-1, 0]$, and where ϵ is defined by the previous `abs` in the rule.
- `-rel=`ϵ' specifies the negative relative error in $\epsilon' \leq \frac{x-y}{\min(|x|,|y|)} \leq \epsilon$, with $\epsilon' \in [-1, 0]$, and where ϵ is defined by the previous `rel` in the rule.
- `-dig=`ϵ' specifies the negative relative error in $\varepsilon' \leq \frac{x-y}{\min(|x|,|y|)} \leq \varepsilon$, with $\varepsilon' = \epsilon' \cdot 10^{-\max(\#x,\#y)}$ and $\epsilon' \leq -1$, and where ϵ (resp. ε) is defined by the previous `dig` in the rule.
- `large` (qualifier) allows for large error $\epsilon > 1$ for the *following* `abs`, `rel` and $\epsilon' < -1$ for the *following* `-abs`, `-rel`.
- `scl=`a specifies the error scale a in $a(x - y) + b$ for error calculation in `abs`, `rel` and `dig` (default $a = 1$).
- `off=`b specifies the error offset b in $a(x - y) + b$ for error calculation in `abs`, `rel` and `dig` (default $b = 0$).
- `lhs=`x specifies the *left hand side* value x to use for error calculation in `abs`, `rel` and `dig`. The corresponding number in the left input file is still read and saved but not used in error calculation.
- `rhs=`y specifies the *right hand side* value y to use for error calculation in `abs`, `rel` and `dig`. The corresponding number in the right input file is still read and saved but not used in error calculation.
- `equ` specifies strict equality, i.e. numbers must compare equal character by character. This is different from `abs=0` as it requires the exact same textual representation and has a higher precedence.
- `ign` specifies that differences between numbers should be ignored. This is equivalent to `abs=`$+\infty$ except that it has a higher precedence.

- `istr` specifies that differences between strings, including *sequence of digits*, should be ignored while scanning for *numbers*. Combining `ign` and `istr` allows for missing numbers in the input files. It makes sense to combine `istr` and `omit` as the latter also applies to *numbers*.

- `omit='tag'` specifies that differences between *sequence of digits* (i.e. inside strings) or *numbers* preceded by '*tag*' should be ignored. It can be used standalone or *in combination with other constraints with lower precedence*. The maximum length of '*tag*' is 64 characters.

- `any` (qualifier) specifies that constraints are disjunctive.

- `all` (qualifier) specifies that constraints are conjunctive (default).

- `alt` (qualifier) specifies that the rule is an alternate rule that remain hidden until the previous rule in the configuration file fails.

- `eval` (qualifier) specifies that the operations must be evaluated even if the rule fails.

- `nofail` (qualifier) specifies that failure of the rule will not be counted nor displayed.

In case of multiple constraints defined in the same rule, the precedence is the following:

`ign` > `omit` > `equ` > `abs` = `rel` = `dig`

Constraints with equal precedence can be combined into conjunctive constraints (default, optional `all`) or into disjunctive constraints using the qualifier `any` (last qualifier wins).

When **no constraint or action** is present in a rule, the constraint `abs=`*eps* becomes the default, where *eps* is the smallest representable positive floating point number (see 2.1).

When **no rule are active** during the comparison of two numbers, the special rule `#0` defined by:

`* * abs=`*eps*

becomes automatically active. Because of the rule `#0`, `ndiff` always interprets the numerical content of the compared files, unless a rule like:

```
* * equ
```

is defined last in the file, i.e. with higher priority, and no action are present.

3.6 Actions

Actions are line-based exclusive commands interpreted with higher priorities than any constraint, independently of their respective line position in the configuration file. Hence *any information present in the range of columns is ignored for activating actions*. Note however that line positions still matter for overlapping actions. The list of supported actions is:

- `skip` is the action to skip the lines covered by the specified range of rows. This action is useful to discard lines that contain information changing from run to run, such as timings or platform information.
- `goto='`*tag*`'` is the action to skip the lines starting from the range of rows, and until the string '*tag*' is encountered in both input files. Because the number of lines read to find '*tag*' can differ between the two files, `ndiff` always favors the **smallest** row count as the new count for further rule selection. The maximum length of '*tag*' is 64 characters. This action is useful to discard content with varying number of lines from run to run, such as intermediate output of iterative algorithms.
- `goto='`*num*`'` is the action to skip the lines starting from the range of rows, and until the number '*num*' is encountered in both input files. Because the number of lines read to find '*num*' can differ between the two files, `ndiff` always favors the **smallest** row count as the new count for further rule selection. *This action can be combined with range of columns and constraints that will be applied when searching for 'num'*, using x as the input numbers and y='*num*' for **both input files**. This action is useful to discard content with varying number of lines from run to run, and where the (re)synchronization of the input files is based on numbers. To treat '*num*' like a '*tag*' (i.e. previous action), the rule must have `*` (or `0-$`) for the range of columns and `equ` in its constraints.

In case of multiple actions (accidentally) defined in the same rule, the precedence is the following:

`skip` > `goto='`*tag*`'` > `goto='`*num*`'`

It is considered as good style to group the actions before the constraints at the beginning of the configuration file for quick identification, since anyway they have a higher priority than any constraint.

3.7 Operations

Operations are based on registers used in conjunction with constraints and actions. `ndiff` has 99 registers named `R`n for $1 \leq n \leq 99$, i.e. registers

`R1` (or `R01`) to `R99`, that allow to store and reuse numbers read from input files and build complex recursive constraints. Registers store numbers in their numerical form, which means that the exact representation of the saved numbers is lost. Hence, rules loading values from registers and involving the constraint `equ` are likely to fail.

Saving numbers to registers The basics of operations is to use numbers read from input files and stored into registers. `ndiff` uses the *read-only* registers `R1` to `R9` to save special values each time a number is encountered in the input files, and the *write-only* register `R0` to print values on the console:

- `R0` prints the values on the right hand side of the assignment.
- `R1` contains the number read from the *left hand side* input file.
- `R2` contains the number read from the *right hand side* input file.
- `R3` contains the computed difference $x - y$.
- `R4` contains the computed scaled error a`R3`.
- `R5` contains the computed absolute error `R4`$+b$.
- `R6` contains the computed relative error `R5/R8`.
- `R7` contains the computed digital relative error `R6/R9`.
- `R8` contains the computed relative normalization $\min(|x|, |y|)$.
- `R9` contains the computed input precision $10^{-\max(\#x,\#y)}$.

Note that registers `R1` and `R2` always contain the values read from the two input files, while the registers `R3`..`R9` may contain values not computed from `R1` and `R2` if the commands `lhs` or `rhs` are present in the rule. Remember that rule `#0` applies if no user-defined rule is active, and it will update the register `R1` to `R9` as any other rule.

Loading numbers from registers All the constraints and actions that expect a *number* can replace this number by a reference to a register, with the possibility to perform simple operations on the value when it is loaded. Loading values from registers has a higher precedence than loading constant numbers. Loading a value from an uninitialized register, i.e. where no value was saved before, leads to an undefined behavior.

- `abs=R`n, `-abs=R`n, `rel=R`n, `-rel=R`n, `dig=R`n, `-dig=R`n, `scl=R`n, `off=R`n, `lhs=R`n and `rhs=R`n are all equivalent to their homologous commands described in 3.5 except that they take their values from the specified registers.

- `goto=R`n is equivalent to its homologous commands `goto='`*num*`'` described in 3.6 except that it takes the value of '*num*' from the specified register.

Note that values are (re)loaded from registers *each time a rule becomes active*. Therefore, if a rule is activated periodically and it uses values loaded from registers, each activation can see different constraint specifications depending if the registers were updated between the activations or not.

Operations on registers It is possible to perform operations when loading values from registers or when moving values between registers. `ndiff` can record up to 10 operations per rule, not including the special loads described in the previous paragraph:

- `=R`n loads the value from register n.

- `=-R`n loads the negated value from register n.

- `=/R`n loads the inverted value from register n.

- `=\R`n loads the negated and inverted value from register n.

- `=^R`n loads the exponential value from register n.

- `=|R`n loads the absolute value from register n.

- `=[R`n loads the value from register n rounded toward zero.

- `=]R`n loads the value from register n rounded toward infinity.

- `R`n`=R`p`+R`q loads the sum of registers p and q to register n.

- `R`n`=R`p`-R`q loads the difference of registers p and q to register n.

- `R`n`=R`p`*R`q loads the product of registers p and q to register n.

- `R`n`=R`p`/R`q loads the ratio of registers p and q to register n.

- `R`n`=R`p`%R`q loads the reminder of registers p and q to register n.

- `R`n`=R`p`^R`q loads the power of registers p and q to register n.

- `R`n`=R`p`<R`q loads the min of registers p and q to register n.

- Rn=Rp>Rq loads the max of registers p and q to register n.
- Rn=Rp~Rq moves the registers from p to q, $p < q$, to registers from n to $n + q - p$. The sequence of registers specified on the right can overlap with the sequence specified on the left of the assignment.

The first nine operations with nothing specified on the left hand side of the assignment also apply to special loads of the previous paragraph. Note that moving a sequence of registers allows to save or print the read-only registers `R1`..`R9` in one command or to shift an 'array' of numbers before saving new values.

Precedence of operations Multiple operations can appear at different places in a rule definition. Hence, understanding how registers are updated during the execution of the rules is important:

1. The read-only registers `R1` and `R2` are updated.
2. The loads for `lhs`, `rhs`, `scl` and `off` are performed.
3. The errors are computed from `lhs`, `rhs`, `scl` and `off`.
4. The read-only registers `R3`..`R9` are updated.
5. The non-numerical constraints `ign`, `omit` and `equ` are checked.
6. The loads for [-]`abs`, [-]`rel` and [-]`dig` are performed.
7. The numerical constraints `abs`, `rel` and `dig` are checked.
8. The operations are performed in the order specified.

If *any* of the non-numerical constraints of step 5 fail, the subsequent steps are skipped. If *any* of the numerical constraints (or *all* if the `any` qualifier is present) of step 7 fail, the subsequent steps are skipped. If the `eval` qualifier is present in the rule, the last step is still performed even if step 5 or step 7 failed.

3.8 Alternates rules

An alternate rule is a rule that contains the qualifier `alt` (see 3.5). Such rule remains hidden until the rule preceding it in the configuration file fails. Hidden rules cannot become active and have to wait for being visible first before becoming selectable for activation, which may never happen. Once the rule become visible, its selection will follow the normal range-based activation process.

```
debug: (args.c:197): debug mode on
debug: (args.c:319): summary set to '../tests-summary.txt'
debug: (args.c:265): test name set to 'test-match'
processing test-match
debug: (utils.c:102): file test-match.out open for reading
debug: (utils.c:102): file test-match.ref open for reading
debug: (utils.c:102): file test-match.cfg open for reading
debug: (main.c:133): rules list:
[#0:0] * * abs=eps
[#1:3] 1-7 * skip
[#2:4] 234 * skip
[#3:7] 21 1 rel=1e-12
[#4:8] 23 * goto='Final difference norm:'
[#5:9] 24 1 rel=1e-12
[#6:10] 33-53/5 1 abs=1e-10
[#7:11] 57 1 abs=1e-16
[#8:12] 69 1 abs=1e-16
[#9:15] 77 1 rel=1e-12
[#10:16] 79 * goto='Final difference norm:'
[#11:17] 80 1 rel=1e-12
[#12:18] 89-109/5 1 abs=1e-10
[#13:19] 113 1 abs=1e-16
[#14:20] 125 1 abs=1e-16
[#15:23] 133 1 rel=1e-10
files test-match from test test-match do not differ
   236 lines have been diffed in files test-match
```

Figure 2: Example of `ndiff` output in debug mode for test-match suite.

3.9 Debugging rules

Internally, `ndiff` can use two different algorithms to select the active rules. The default linear algorithm always enabled is a stack-based automata that exhibits linear algorithmic complexity to select active rules. Another brute force algorithm with quadratic algorithmic complexity is also available for crosscheck and debugging purpose, and can be enabled by the command line option `--xcheck`.

The command line option `--debug` can be used to display debugging information during the run, including the list of rules known by `ndiff` after the loading of the configuration file, as shown in the figure 2. This option includes the option `--xcheck`.

The command line option `--trace` can be used to display even more debugging information during the run. This option includes the options

`--debug`.

Because the `--trace` option is very verbose, another mechanism has been implemented with the rule qualifier `trace` to trace only the active rules:

- `trace` (qualifier) specifies to trace only when the rule is active. It is useful for debugging purpose and provides finer control than the command line option `--trace` (global setup).
- `traceR` (qualifier) same as `trace` but trace also the registers modified by the rule.

Note that qualifying rules with `trace` does NOT enable the options `--debug` and `--xcheck`. Hence qualifying all rules with `trace` is not equivalent to the command line option `--trace`.

4 Examples

Many examples of configuration files can be found in the MAD-X tests repository [5], where the files can be browsed directly from the web. In the following of this section, we show only some simple and typical use cases of `ndiff` rules.

4.1 Basic constraints

The rule:

```
3-$ 5-$ ign
```

specifies that all numbers after line 3 and column 5 included must be ignored in both input files.

The rule (*conjunctive*):

```
3 * abs=1e-10 rel=1e-8
```

specifies that all numbers (i.e. all columns) on line 3 must be equal between the two input files within an absolute error of 10^{-10} AND a relative error of 10^{-8}; that is both constraints must be fulfilled.

The rule (*disjunctive*):

```
5-$ 2-5 any abs=1e-12 rel=1e-9 dig=100
```

specifies that all numbers in columns 2 to 5 after line 5 included must be equal between the two input files within an absolute error of 10^{-12} OR a relative error of 10^{-9} OR accepts that they differ on the tailing one or two digits; *that is one of the three constraints must be fulfilled.* It is common

to tighten the constraints when the `any` qualifier is present to avoid lazy acceptance.

The rule (*large error*):

```
* 3 rel=1e6
```

is invalid. The constraint should be either `rel=1e-6`, or preceded by the `large` qualifier. The explicit requirement of the `large` qualifier avoids to forget the minus sign of the exponent and accidentally validate all the differences.

4.2 Basic actions

The two rules:

```
1-7   * skip # banner
119-$ * skip # trailer
```

specify that the lines from 1 to 7 and from 119 to the end of both files are ignored. The specification of the range of columns is ignored (see actions).

4.3 Basic operations

The two rules:

```
1-$/2 3 ign      R10=R1  # save R1
2-$/2 2 abs=1e-4 scl=R10 # scale error
```

save the number read from the left input file at column 3 for all odd lines into register `R10`, whatever the difference is with the right input file. Then reuse the previous saved value to scale the absolute error computed for all even lines at column 2 between the two input files.

4.4 Advanced constraints

The rule:

```
3-$ 5-$ ign abs=1e-2
```

specifies that all numbers after line 3 and column 5 included must be ignored in both input files. The constraint `abs` is never considered because of its lower precedence than the command `ign`.

The rule:

```
* 3 abs=1e-6 -abs=0
```

specifies that all numbers in column 3 must be must be equal between the two

input files within an absolute *asymmetric error* of 10^{-6}, and where negative error are forbidden. This command is useful to validate data (e.g. measurements) versus some references (e.g. template).

The rule:

```
3-$ * abs=1e-9 large rel=1e2 scl=1e-2
```

specifies that all numbers after line 3 must be must be equal between the two input files within an absolute error of 10^{-9} AND a relative error of 10^{2}, both after scaling the errors by 10^{-2}. The `large` qualifier must be explicitly mentioned to avoid common mistakes, like typing `abs=1e9` instead of `abs=1e-9` in this example making the rule accidentally ineffective. The purpose of the `large` qualifier is to accept absolute errors between very large numbers or relative errors between very small numbers. The purpose of the `scl` command is to scale the error such that for example relative errors remain sensible.

The following four pairs of rules:

`*  3 abs=INF`	`*  3 equ`	`*  3 equ`	`*  3 ign`
`1-$/2 3 equ`	`2-$/2 3 abs=INF`	`2-$/2 3 ign`	`1-$/2 3 equ`

are all equivalent and specify that all numbers in column 3 must be strictly equal (i.e. same representation) for odd rows and ignored (i.e. infinite absolute error) for even rows during the comparison of the two input files.

The rule:

```
8-$ 0-3 abs=1e-9 rel=1e-7 omit='DRIFT_'
```

specifies that all numbers in columns 1 to 3 after line 8 included must be equal between the two input files within an absolute error of 10^{-9} AND a relative error of 10^{-7}. During *the scan for numbers* — starting from column zero — any *sequence of digits not interpreted as a number* and preceded by the string '`DRIFT_`' will be ignored. This command is useful to accept identifiers that contain sequence of digits that may vary across runs, compilers and platforms (e.g. shared implicit drift elements in MAD-X Twiss table).

The rule:

```
8-$ 2 any abs=1e-9 rel=1e-7 dig=1e2 omit='sec.s since start: '
```

specifies that all numbers in column 2 after line 8 included must be equal between the two input files within an absolute error of 10^{-9} OR a relative error of 10^{-7} OR accepts that they differ on the tailing one or two digits. During *the interpretation of numbers* in column 2, any *number* preceded by the string '`sec.s since start: `' including the trailing space will be ignored. This command is useful to accept varying numbers that occurs from time to time in the middle of large numerical content (e.g. tables, measurements).

4.5 Advanced actions

The rule:

```
13 * goto='penalty function: '
```

read lines starting at line 13, and until it encounters the string '`penalty function: `' including the trailing space in both input files. Assuming that the string was encountered at lines 57 and 119 respectively, the new row count of both files becomes 57 for further rule activation (see `goto` action). Hence, the `goto` command requires that one of the two input files is used as a *reference file*, which must always be shorter than its companion input file in order to get a deterministic row count during the comparison. This *reference row count* must be used to setup the range of rows for the rules selectable after the execution of the `goto` command. As a rule of thumb, the activation of `goto` command must take place *at least one line before the next supposedly active constraint*, otherwise the later will never be effective due to its lower priority.

The two rules:

```
13-120 * goto='penalty function: '
57     * abs=1e-10
```

read lines starting at line 13, and until it encounters the string '`penalty function: `' including the trailing space in both input files *right after the specified range of rows*. Assuming that the first string was encountered at lines 57 and 119 respectively (as before), the new row count of both files becomes 57 for further rule activation. Therefore this rule will remains the active rule due its highest priority (once activated), and until it encounters the string '`penalty function: `' outside its own range of rows using the row count of the shortest input file. Note that the rule containing the `abs` constraint is never activated despite that its definition appears after the `goto` action in the configuration file and that its range for rows corresponds to the line of the first occurrence of the string searched. If the `goto` action had `12-40` for its range of rows, the `abs` constraint would have been applied to the values on the line 57, e.g. the value of the penalty function.

The rule:

```
251 2 goto='11850' large abs=0.5 -abs=-10
```

read lines starting at line 251, and until it encounters a number $11840 \leq x \leq 11850.5$ in the second column for both input files. Note that number `2` in the range of columns is used for searching the number that fulfills the constraints, not for triggering the action that will be active as soon as the line 251 is reached. The rest of the behavior is similar to `goto` on strings. If you need to search for a number as if it were a tag, you can use the rule:

```
251 * goto='11850.20' equ
```

This will have the same effect as searching for the string '11850.20'. Note that the `*` (or `0-$`) in the range of columns matters to obtain the same behavior.

4.6 Advanced operations

The two rules (*delay*):

```
1-11  2  ign               R11=R10~R20 R10=R1 R0=R10~R20 # init
11-$  2  abs=1e-9 lhs=R20 R11=R10~R20 R10=R1             # delay
```

implement a delay of 11 rows in the column 2 of the left input file using 10 registers before checking for the absolute error. The first rule prints the content of the registers `R10..R20` on the console to check the shift initialization.

The four rules (*derivative*):

```
# Goal: check that X Y points (left file)
#       follow reference derivatives (right file)
# Compute:
# -e_k <= (y_k-y_{k-1})/(x_k-x_{k-1}) - (dy/dx)_k <= e_k
# Inputs: (left file)     (right file)
#           x    y         dy/dx   e

# initialization (k=0)
1    1  ign R10=R1   # R10 = x_k
1    2  ign R11=R1   # R11 = y_k

# row iterations (k>0)
# R12=x_k-x_{k-1}, R13=(dy/dx)_k, R10=x_k, R11=y_k
2-$  1  ign R12=R1-R10 R13=R2 R10=R1
2-$  2  lhs=R1 rhs=R11 scl=/R12 off=-R13 abs=R2 R11=R1 eval
```

compare the derivatives of the x and y coordinates read from the left input file with the reference derivatives dy/dx and the error tolerances e read from the right file, using the backward difference approximation. The `eval` qualifier is required to ensure that the value y_k is saved even if the rule fails. Otherwise `ndiff` would always report differences after encountering the first one because the backward difference would be broken.

5 Output

When a difference between two strings or two numbers is rejected by the active rule, `ndiff` tries to report useful information about the discrepancy as shown in figure 3.

```
warng: (*) files 'test-match'|'test-match.ref' from test
-->        'test-match' differ
warng: (1) files differ at line 21 column 1 between
-->        char-columns 30|30 and 52|40
warng: (1) numbers: '0.2761043129903641e+14'|'0.2761e+14'
warng: (1) relative error (rule #3, line 7: -1e-12<=rel<=1e-12)
-->        abs=4.3e+08, rel=1.6e-05, ndig=16
warng: (=)   236 lines have been diffed
warng: (=)     1 diff  has  been detected
```

Figure 3: Example of `ndiff` output for a single difference rejected during the comparison of the files `test-match` and `test-match.ref` in the test-match suite. The symbol `-->` indicates the continuation of the previous line.

The left marker `(*)` indicates the beginning of a list of discrepancies, followed by the name of the files under comparison. The left markers (*num*) group the multi-lines output reported for each difference encountered with its respective count number. Each group displays the line and column number where the difference has been rejected as well as the corresponding character columns; remember that the column number is defined by the count of numbers. The symbol '`|`' is used to separate the information belonging to the left (first) and the right (second) files under comparison. In the case shown in figure 3, the character columns are different after the parsing of the input numbers, while the column number remains the same.

The tag `numbers:` indicates that the difference occurred between numbers after numeric interpretation, and displays within quotes the numbers belonging to the left and right input files. The other possible tag `strings:` would indicate that the difference occurred between sequences of characters that do not (yet) represent numbers, and would display the first 25 characters of the difference.

The next line indicates respectively the kind of constraint that triggered the rejection of the discrepancy, here a relative error, the rule specification (`#3, line 7:` $-10^{-12} \leq$ `rel` $\leq 10^{-12}$), and the computed absolute and relative errors as well as the maximum number of significant digits of the mantissa read from the two input files.

5.1 Setup strategy

The strategy to setup quickly a configuration file for `ndiff` is to run it the first time without any configuration file while keeping a large number of errors (option `--keep`). Then the `ndiff` output will contain useful information for estimating efficiently the most appropriate kind of constraints and their

initial guess, and validate (i.e. accept) the error differences encountered. `ndiff` tries to align vertically the computed errors such that quick browsing of large amount of errors is possible. This strategy proved to be effective during the setup of the hundreds configuration files of the MAD-X test system.

Another possible strategy for less sensitive data is to release first the constraint for a large part of the input, like in the following rule (adapted from `test-dynap`), and tighten them on purpose for identified cases after few runs:

```
4-$    2  any abs=1e-9 rel=1e-7 dig=1e2 omit='timing: '
4-$/5  2  abs=1e-9 rel=1e-7    # special cases
```

The `any` qualifier should always be used with care when applied to large inputs, as it can silently 'disable' your comparison and always succeed.

However, it is considered as a better practice to setup global strict constraints first and release the error tolerances afterward, i.e. using constraints with higher priority defined later in the configuration and active only on small specific ranges. Hence, the example above should remain exceptional on sensitive data.

6 Running modes

The command line options of `ndiff` can be obtained by running the help (`ndiff --help`). This will also display some summary information about the rules; see the appendix A for the complete output of the help.

The `ndiff` options are recorded or executed as they are parsed on the command line and should not appear between the files to compare. Hence the same option can appear more than once to setup different parts of the command line. In the following of this section, each pair of square brackets means that its content is optional and can be omitted.

6.1 Single mode

The single mode is the simplest way to compare two files. A typical use would look like:

`ndiff` *fileA*[`.out`] *fileB*[`.ref`] [*fileC*[`.cfg`]]

If `ndiff` is not able to open a file with the name specified on the command line, it tries again after concatenating the *expected extension*. Expected extensions are positional, that is for the first file `ndiff` will try to open *fileA*

then *fileA*`.out`, for the second file it will try to open *fileB* then *fileB*`.ref`, and finally for the third file (if any), it will try to open *fileC* then *fileC*`.cfg`.

If all the base names are identical, the same effect can be achieved more concisely with the list mode (see 6.5):

`ndiff` -l *file* $\Longleftrightarrow$ `ndiff` *file*[`.out`] *file*`.ref` [*file*`.cfg`]

If the file name is '`-`' for an input file, `ndiff` reads the data from the standard input (i.e. `stdin`).

6.2 Filter mode

The filter mode use `ndiff` like a data filter, and allows to write *all valid lines* into result files. A valid line is a line that fulfills all the constraints of the rules active for that line. The file name of a result file is the same as its input file with the extension `.res` appended. A typical use would look like:

`ndiff --nowarn --lhsres` *fileA*[`.out`] *fileB*[`.ref`] [*fileC*[`.cfg`]]

The `--lhsres` and `--rhsres` options can be used together to filter both input files. These options are reset after each comparison, and they must be restated on the command line for each comparison where they should be enabled. The loop on file numbering, i.e. the serie mode (see 6.4), does not reset these options; hence all the *serie* will have result files.

6.3 Recycle mode

The recycle mode is useful when using `ndiff` with templates that can be recycled for the same input file, i.e. when they represent digested data or relative data like derivatives instead of values. A typical use would look like:

`ndiff --trunc --rhsrec` *fileA*[`.out`] *fileB*[`.ref`] [*fileC*[`.cfg`]]

The `--lhsrec` and `--rhsrec` options are exclusive and cannot be used together, otherwise this would result in an infinite loop. These options can be used only if the input file can be rewinded, that is not if the input file in coming from the standard input or through a decompression command.

These options are reset after each comparison, and they must be restated on the command line for each comparison where they should be enabled. The loop on file numbering, i.e. the serie mode (see 6.4), does not reset these options; hence the recycling will be perform on all the *serie* of files.

6.4 Serie mode

The serie mode extends the single mode by adding incremental numbering between the file name and the expected extension (if any) until a file opening fails:

`ndiff --serie` *fileA*[`.out`] *fileB*[`.ref`] [*fileC*[`.cfg`]]

is equivalent to try successively:

`ndiff` *fileA*[`.out`] *fileB*[`.ref`] [*fileC*[`.cfg`]]
`ndiff` *fileA*`1`[`.out`] *fileB*`1`[`.ref`] [*fileC*`1`[`.cfg`]]
`ndiff` *fileA*`2`[`.out`] *fileB*`2`[`.ref`] [*fileC*`2`[`.cfg`]]
`ndiff` *fileA*`3`[`.out`] *fileB*`3`[`.ref`] [*fileC*`3`[`.cfg`]]
etc ...

The first attempt may freely fail as non-numbered version of the files may or may not exist. This mode can be combined with the list mode (see 6.5).

The useful option for this mode is:

- `--seriefmt "`*fmt*`"` specifies the `printf` format of the numbering. For example `--seriefmt "%03d"` will generate the numbering `001`, `002`, ...

6.5 List mode

The list mode allows to compare a list of files where only the base name is specified and let `ndiff` build the full name using the expected extensions. A typical use would look like:

`ndiff --list` *fileA fileB* ...

For each base name in the list, `ndiff` will build the filenames *fileA*[`.out`], *fileA* `.ref` and [*fileA* `.cfg`], and will try to open them as if it were running in single mode (see 6.1). This mode can be combined with the serie mode (see 6.4).

6.6 Test mode

The test mode allows to execute multiple comparisons on the same command line and output some summary for each test:

`ndiff --test 'test-1'` *fileA*[`.out`] *fileB*[`.ref`] [*fileC*[`.cfg`]]
`--test 'test-2'` *fileD*[`.out`] *fileE*[`.ref`] *fileF*[`.cfg`]
fileG[`.out`] *fileH*[`.ref`] [*fileI*[`.cfg`]]

Note that either the configuration file $fileF$[`.cfg`], either the extension of the

file *fileG*`.out` is required to allow `ndiff` to group correctly the files under comparison.

The output summary of the commands aforementioned could look like:

```
 + test-1                                (0.00 s) -  1/ 1 : PASS
 + test-2                                (0.00 s) -  2/ 2 : PASS
```

The numbers given on the right represent the number of files successfully compared versus the total number of files to compare for a given test, which should be the same if no difference is rejected.

6.7 Suite mode

The suite mode allows to execute multiple times `ndiff`, i.e. as in a test suite, and accumulate digested information from each run. This collected information can be used to display a summary of the runs. A typical use would look like:

```
ndiff -q --accum 'Sum' --reset
ndiff -q --accum 'Sum' --suite 'My Tests' --test 'test-1' ...
ndiff -q --accum 'Sum' --test 'test-2' ...
```

The first command resets the file `Sum` (e.g. summary) used to store intermediate information between runs. The second command runs the first test but also displays the banner of the test suite `My Tests` and the name of the test `test-1`. The third command runs the second test and displays the name of the test `test-2` (only).

The useful option for this mode is:

- `--suitefmt "`*fmt*`"` specifies the `printf` format of the suite title. For example `--suitefmt "-- %s --"` displays the title `-- My Tests --`.

The output summary of the commands aforementioned could look like:

```
[ My Tests ]
 + test-1                                (0.08 s) -  8/ 8 : PASS
 + test-2                                (0.02 s) -  7/ 7 : PASS
```

Advanced uses of `ndiff` commonly combine the *suite*, *test*, *list* and *serie* modes, and group all these commands and arguments into a single `ndiff` command line to run full regression tests in one shot for example.

6.8 Compressed files

`ndiff` is able to process automatically compressed files on the fly by using available decompressors in a subprocess, where the communications are performed through pipes, that is no temporary files are created. The command used to uncompress a files is based on its extension, and if no extension is present (e.g. list mode), `ndiff` will try all supported extensions and will build possible extrapolated filenames. The command must uncompress the files to `stdout` such that `ndiff` can capture the uncompress stream and process the comparison of the input text files as if they were read directly from the hard disk.

The useful option for this mode are:

- `--unzip "`*cmd*`"` specifies the command to uncompress files with extension `.zip`. The default command is `unzip -cq`.
- `--gzip "`*cmd*`"` specifies the command to uncompress files with extensions `.gz`, `.z`, `.Z`, `.tgz`, `.taz`, or `.taZ`. The default command is `gzip -cdq`.
- `--bzip2 "`*cmd*`"` specifies the command to uncompress files with extensions `.bz`, `.bz2`, `.tbz`, or `.tbz2`. The default command is `bzip2 -cdq`.

Adding new file extensions and new commands for decompressing unsupported compression format to `ndiff` is very simple.

7 Applications

7.1 Data validation

The principle of using `ndiff` for validating data is to compare a set of unknown data, for example measurements, with a set of known and validated data. The `ndiff` rules should provide enough flexibility to use the reference data as a template with the required freedom to validate the unknown data. The validation process would simply consist of running `ndiff` on the unknown data versus one or few templates.

7.2 Regression tests

The principle of regression testing is shown in the figure 4 from left to right. Each regression test runs a single MAD-X job (file `.madx`) that generates output files (files `.out`), then each output file is compared with its corresponding

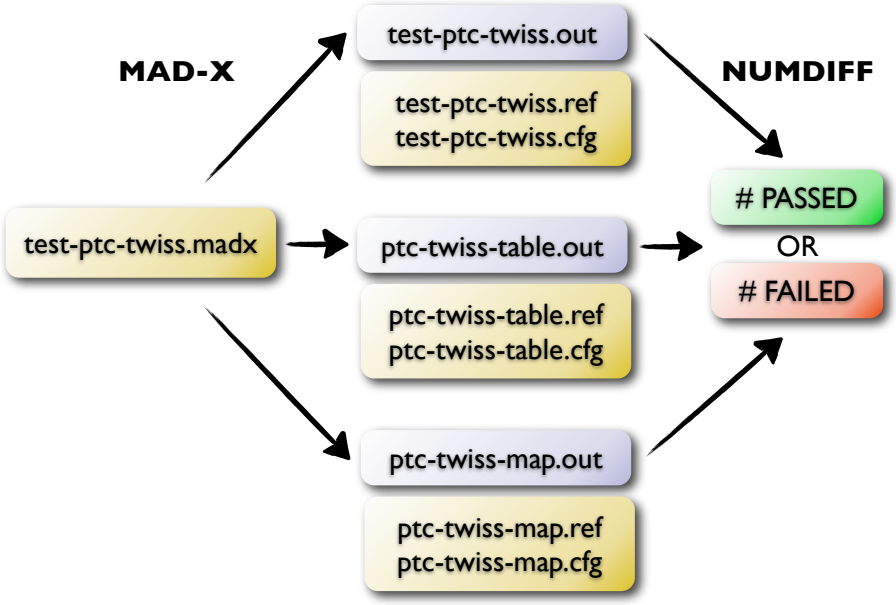

Figure 4: Schematic description of the process used to test MAD-X components. This PTC Twiss example runs a single `madx` job that generates three output (`.out`) files. The three reference (`.ref`) and configuration (`.cfg`) files are provided to `ndiff` by the MAD-X test system for regression testing, and are part of the test suites repository.

reference file (file `.ref`) under the rules defined in its corresponding configuration file (file `.cfg`) using `ndiff`. The latter reports on the console the passed and failed tests, plus some information on the discrepancies found in case of failure. In general, the job script as well as the reference and configuration files are distributed with the application as part of its test suites, while the output files are generated on the fly by the test system.

7.3 Test system

The `ndiff` command line used by the MAD-X test system is:

`ndiff -q [-d] -b -x -l -n -a` *sum-file* `-t` *test-name file-list ...*

The appendix A lists the complete set of options supported by `ndiff` and the appendix B shows the complete output of the MAD-X test system. Each test has its own directory named after the name of the test *test-name*, and the list of base names *file-list* is built from the list of all the reference files present in that directory, that is all the files with the extension `.ref` or `.ref.*`, where the asterisk can be any of the supported extensions for compressed files.

8 Installation

The last version of `ndiff` can be freely downloaded from the releases repository [6] where binaries for the Windows, Linux and Mac OS X platforms for 32-bit or 64-bit systems are available.

The source code (tarball) and this manual can be downloaded either from the same repository, or directly from the SVN web browser [7].

8.1 Compiling `ndiff`

Compiling `ndiff` with a C99 compiler offers the opportunity to permanently setup some default values by defining the following macros at compile time:

- `-DVERSION='`*str*`'` specifies the version of `ndiff`. This string cannot be changed at runtime.
- `-DMAXKEEP=`*num* specifies the default maximum number of warning displayed on the output, the original value is 25. This option does not affect the total count of encountered differences, only the output length.
- `-DMAXREGS=`*num* specifies the default number of registers, the original value is 99, the maximum value is 8192.
- `-DMAXREGOP=`*num* specifies the maximum number of operations per rule, the original value is 10. This number cannot be changed at runtime.
- `-DMAXTAGLEN=`*num* specifies the maximum length of tags, the original value is 64. This number cannot be changed at runtime.
- `-DCMTCHRS='`*chrs*`'` specifies the default set of characters that defined line of comments, the original set of characters is `""` (i.e. empty). If the first character is a white space, then white spaces are discarded before scanning for a comment character, otherwise comment character must sit on the first column.
- `-DPUNCTCHRS='`*chrs*`'` specifies the default set of punctuation characters accepted in identifiers, the original set of punctuation characters is `"_.$"` (i.e. MAD-X identifiers).
- `-DREGOFMT='`*fmt*`'` specifies the default format for printing the content of registers, the original format is `"%g "`.
- `-DSUITEFMT='`*fmt*`'` specifies the default format for suite title, the original format is `"[ %s ]"`.

- `-DSERIEFMT='`*fmt*`'` specifies the default format for serie numbering, the original format is `"%d"`.
- `-DOUTFILEEXT='`*str*`'` specifies the default extension for output files, the original extension is '`.out`'.
- `-DREFFILEEXT='`*str*`'` specifies the default extension for reference files, the original extension is '`.ref`'.
- `-DCFGFILEEXT='`*str*`'` specifies the default extension for configuration files, the original extension is '`.cfg`'.
- `-DRESFILEEXT='`*str*`'` specifies the default extension for result files, the original extension is '`.res`'.
- `-DUNZIPCMD='`*str*`'` specifies the default command to uncompress files with extension `.zip`, the original command is '`unzip -cq`'.
- `-DGZIPCMD='`*str*`'` specifies the default command to uncompress files with extensions `.gz`, `.z`, `.Z`, `.tgz`, `.taz`, or `.taZ`, the original command is '`gzip -cdq`'.
- `-DBZIP2CMD='`*str*`'` specifies the default command to uncompress files with extensions `.bz`, `.bz2`, `.tbz`, or `.tbz2`, the original command is '`bzip2 -cdq`'.

Note that unless specified, all these values can be overridden with command line options, see the `ndiff` options in appendix A.

A typical GNU `gcc` command line would be :

```
gcc -std=c99 -W -Wall -Wextra -pedantic -O3 src/*.c -o ndiff -lm
```

To redefine the default numbering of series using the macro aforementioned:

```
gcc -std=c99 -W ...  -O3 -DSERIEFMT='"%03d"' src/*.c -o ndiff -lm
```

Note that both single and double quotes are needed to form a string within the shell command that propagates correctly to the C files.

8.2 AFS clients

The `ndiff` tool can be run directly from AFS clients (e.g. `lxplus.cern.ch`) using the MAD repository:

```
~mad/bin/ndiff
~mad/bin/rel/last-dev/ndiff-linux32
~mad/bin/rel/last-dev/ndiff-linux64
~mad/bin/rel/last-dev/ndiff-macosx32
```

```
~mad/bin/rel/last-dev/ndiff-macosx64
~mad/bin/rel/last-dev/ndiff-win32.exe
~mad/bin/rel/last-dev/ndiff-win64.exe
```

The first binary is the default on `lxplus` servers and it is a link to the last development release for Linux (i.e. `ndiff-linux64`). The complete path to the MAD-X binaries, including `ndiff`, is:

```
/afs/cern.ch/user/m/mad/bin
```

9 Future work

The `ndiff` tool has been developed for the MAD-X test system and will continue to evolve according to the needs of this test system. New features might be implemented in the future if there is a significant demand (and possibly help) from the users of `ndiff`.

References

[1] MAD-X website,
`http://cern.ch/mad`

[2] `numdiff` tool,
`http://www.nongnu.org/numdiff`

[3] `ndiff` tool,
`http://www.math.utah.edu/~beebe/software/ndiff`

[4] POSIX Regular Expression,
`http://en.wikipedia.org/wiki/Regular_expression`

[5] MAD-X tests repository,
`http://cern.ch/mad/madX/tests`

[6] MAD-X releases repository,
`http://cern.ch/mad/releases`

[7] MAD-X SVN trunk for `ndiff`,
`http://svnweb.cern.ch/world/wsvn/madx/trunk/madX/tools/ndiff`

A `ndiff` help

```
$ ndiff --help

usage:
ndiff [options] fileA[.out] fileB[.ref] [fileC[.cfg]]
ndiff [options] --list fileA fileB ...
ndiff [options] --test '1st' fileA fileB --test '2nd' fileC ...

options:
  -a  --accum file    accumulate tests information in file
  -b  --blank         ignore blank spaces (space and tabs)
      --cfgext ext    specify the config file extension
  -c  --comment chrs  comment characters set
  -d  --debug         enable debug mode (include xcheck mode)
  -h  --help          display this help
  -i  --info          enable info mode (default)
  -k  --keep num      specify the number of diffs to display per file
      --lhsrec        recycle next lhs file (exclusive with --rhsrec)
      --lhsres        echo valid lines of next lhs file to its result file
  -l  --list          enable list mode (list of filenames)
      --long          disable short options
      --noloc         disable C file location during trace
      --nowarn        disable warnings
      --nregs num     specify the number of registers to allocate
      --outext ext    specify the output file extension
      --punct chrs    punctuation characters part of identifiers
  -q  --quiet         enable quiet mode (no output if no diff)
      --refext ext    specify the reference file extension
      --regfmt fmt    specify the (printf) format fmt for registers
  -r  --reset         reset accumulated information
      --resext ext    specify the result file extension
      --rhsrec        recycle next rhs file (exclusive with --lhsrec)
      --rhsres        echo valid lines of next rhs file to its result file
  -n  --serie         enable serie mode (indexed filenames)
      --seriefmt fmt  specify the (printf) format fmt for indexes
  -s  --suite name    set test suite name for output message (title)
      --suitefmt fmt  specify the (printf) format fmt for test suite
  -t  --test name     set test name for output message (item)
      --trace         enable trace mode (very verbose, include debug mode)
      --trunc         allow premature ending of one of the input file
      --utest         run the ndiff unit tests (still incomplete)
  -x  --xcheck        enable cross check mode (algorithms cross check)

decompression:
      --bzip2 cmd     command to uncompress .bz .bz2 .tbz .tbz2 files
      --gzip  cmd     command to uncompress .gz .z .Z .tgz .taz .taZ files
      --unzip cmd     command to uncompress .zip files

rules (.cfg):
  #rows      cols     commands
  1-5        *        skip                              # banner
  *          2-$      any abs=1e-15 rel=1e-12 dig=1.5 # global
```

```
  41          *         goto='penalty function'          # jump
  109:20/5    2-8/3     abs=1e-8                         # specific

ranges:
  num                   row or column number, num >= 0
  range                 start - end  [/ stride]
  slice                 start : size [/ stride]
  $, *                  last row or column, alias for 0-$

commands:
  abs=num or reg        absolute error (0 <= num <= 1)
  -abs=num or reg       negative absolute error (-1 <= num <= 0)
  all                   constraints are conjunctive (default, qualifier)
  alt                   declare the rule as an alternate rule (qualifier)
  any                   constraints are disjunctive (qualifier)
  dig=num or reg        input-defined relative error (num >= 1)
  -dig=num or reg       input-defined negative relative error (num <= 1)
  equ                   strict numbers equality (same representation)
  eval                  perform operations even if rule fails
  goto='tag'            skip lines until string 'tag' is found (action)
  goto='num' or reg     skip lines until number 'num' is found (action)
  ign                   ignore numbers, accept missing number if with istr
  istr                  ignore strings while scanning for numbers
  large                 allow num > 1 in abs and rel
                        and num < -1 in -abs and -rel (qualifier)
  lhs=num or reg        set left hand side 'x' in a(x-y)+b
  nofail                do not count nor display warning for failure
  off=num or reg        set error offset 'b' in a(x-y)+b
  omit='tag'            ignore strings or numbers if preceded by 'tag'
  rel=num or reg        relative error (0 <= num <= 1)
  -rel=num or reg       negative relative error (-1 <= num <= 0)
  rhs=num or reg        set right hand side 'x' in a(x-y)+b
  scl=num or reg        set error scaling factor 'a' in in a(x-y)+b
  skip                  skip lines (action)
  small                 forbid num > 1 in abs and rel
                        and num < -1 in -abs and -rel (default, qualifier)
  trace                 trace rule when active (debug, qualifier)
  traceR                trace rule and modified registers when active

registers:
  R1..R9                contain lhs, rhs, dif, err, abs, rel, dig, min, prec
  =Rn                   load value from register n
  =-Rn                  load negated value from register n
  =/Rn                  load inverted value from register n
  =\Rn                  load negated and inverted value from register n
  =^Rn                  load the exponential value from register n
  =|Rn                  load the absolute value from register n
  =[Rn                  load the value from register n rounded toward zero
  =]Rn                  load the value from register n rounded toward infty
  R0=                   print the value(s) on the console
  Rn=Rp+Rq              load the sum of registers p and q
  Rn=Rp-Rq              load the difference of registers p and q
  Rn=Rp*Rq              load the product of registers p and q
  Rn=Rp/Rq              load the ratio of registers p and q
```

```
  Rn=Rp%Rq            load the reminder of registers p and q
  Rn=Rp^Rq            load the power of registers p and q
  Rn=Rp<Rq            load the min of registers p and q
  Rn=Rp>Rq            load the max of registers p and q
  Rn=Rp~Rq            move registers p..q to registers n..n+q-p

info   :    http://cern.ch/mad/ndiff
author :    laurent.deniau@cern.ch
version:    2013.04.15
licence:    GPL v3
```

B MAD-X test system

The numbers given on the right represent the number of files successfully compared versus the total number of files to compare for a given test, which should be the same if no failure occurred. The timings given in parenthesis are the timings of `ndiff` itself; they do not include the timings of the MAD-X job run to generate the output files.

```
$ make tests-all

[ Special features ]
 + test-setvars_lin                          (0.00 s) -  1/ 1 : PASS
[ Makethin testsuite ]
 + test-makethin                             (0.00 s) -  2/ 2 : PASS
 + test-makethin-2                           (0.00 s) -  2/ 2 : PASS
[ Survey testsuite ]
 + test-survey                               (0.00 s) -  3/ 3 : PASS
 + test-survey-2                             (0.00 s) -  2/ 2 : PASS
[ Track testsuite ]
 + test-track                                (0.00 s) -  2/ 2 : PASS
 + test-track-2                              (0.00 s) -  3/ 3 : PASS
 + test-track-3                              (0.00 s) -  3/ 3 : PASS
 + test-track-4                              (0.00 s) -  2/ 2 : PASS
 + test-track-5                              (0.00 s) -  2/ 2 : PASS
 + test-track-6                              (0.00 s) -  8/ 8 : PASS
 + test-track-7                              (0.00 s) -  5/ 5 : PASS
 + test-track-8                              (0.00 s) -  2/ 2 : PASS
 + test-track-9                              (0.00 s) -  4/ 4 : PASS
 + test-track-10                             (0.00 s) -  4/ 4 : PASS
 + test-track-11                             (0.00 s) -  2/ 2 : PASS
[ Twiss testsuite ]
 + test-twiss                                (0.00 s) -  2/ 2 : PASS
 + test-twiss-2                              (0.00 s) -  3/ 3 : PASS
 + test-twiss-3                              (0.00 s) -  2/ 2 : PASS
 + test-twiss-4                              (0.00 s) -  3/ 3 : PASS
 + test-twiss-5                              (0.00 s) -  2/ 2 : PASS
 + test-twiss-6                              (0.00 s) -  3/ 3 : PASS
 + test-twiss-7                              (0.00 s) -  2/ 2 : PASS
[ Orbit Correction testsuite ]
 + test-cororbit                             (0.08 s) -  8/ 8 : PASS
 + test-cororbit-2                           (0.00 s) -  7/ 7 : PASS
 + test-cororbit-3                           (0.00 s) -  3/ 3 : PASS
[ Emit testsuite ]
 + test-emit                                 (0.00 s) -  1/ 1 : PASS
 + test-emit-2                               (0.00 s) -  1/ 1 : PASS
[ IBS testsuite ]
 + test-ibs                                  (0.03 s) -  3/ 3 : PASS
 + test-ibs-2                                (0.00 s) -  3/ 3 : PASS
 + test-ibs-3                                (0.03 s) -  3/ 3 : PASS
 + test-ibs-4                                (0.02 s) -  2/ 2 : PASS
[ Error testsuite ]
 + test-error                                (0.01 s) -  5/ 5 : PASS
```

```
 + test-error-2                                  (0.00 s) -  2/ 2 : PASS
[ Dynamic Aperture testsuite ]
 + test-dynap                                    (0.10 s) -  6/ 6 : PASS
[ SixTrack Conversion testsuite ]
 + test-c6t                                      (0.01 s) -  4/ 4 : PASS
 + test-c6t-2                                    (0.00 s) -  4/ 4 : PASS
[ Thick Quadrupole testsuite ]
 + test-thick-quad                               (0.00 s) -  3/ 3 : PASS
[ Jacobian testsuite ]
 + test-jacobian                                 (0.00 s) -  1/ 1 : PASS
 + test-jacobian-2                               (0.00 s) -  1/ 1 : PASS
 + test-jacobian-knobs                           (0.00 s) -  2/ 2 : PASS
[ Matching testsuite ]
 + test-match                                    (0.00 s) -  1/ 1 : PASS
 + test-match-2                                  (0.00 s) -  3/ 3 : PASS
 + test-match-3                                  (0.00 s) -  1/ 1 : PASS
 + test-match-4                                  (0.00 s) -  1/ 1 : PASS
 + test-match-5                                  (0.00 s) -  1/ 1 : PASS
 + test-match-6                                  (0.00 s) -  3/ 3 : PASS
 + test-match-7                                  (0.00 s) -  2/ 2 : PASS
 + test-match-8                                  (0.00 s) -  1/ 1 : PASS
[ Aperture testsuite ]
 + test-aperture                                 (0.00 s) -  2/ 2 : PASS
[ RF Multipole testsuite ]
 + test-rfmultipole                              (0.00 s) -  9/ 9 : PASS
 + test-rfmultipole-2                            (0.00 s) -  2/ 2 : PASS
 + test-rfmultipole-3                            (0.00 s) -  2/ 2 : PASS
 + test-rfmultipole-4                            (0.00 s) -  2/ 2 : PASS
[ PTC Twiss testsuite ]
 + test-ptc-twiss                                (0.00 s) -  5/ 5 : PASS
 + test-ptc-twiss-2                              (0.13 s) -  5/ 5 : PASS
 + test-ptc-twiss-old1                           (0.00 s) -  6/ 6 : PASS
 + test-ptc-twiss-old2                           (0.00 s) -  6/ 6 : PASS
 + test-ptc-twiss-old3                           (0.00 s) -  6/ 6 : PASS
 + test-ptc-twiss-old4                           (0.00 s) -  6/ 6 : PASS
 + test-ptc-twiss-old5                           (0.01 s) -  4/ 4 : PASS
 + test-ptc-twiss-old6                           (0.04 s) -  5/ 5 : PASS
 + test-ptc-twiss-old7                           (0.01 s) -  8/ 8 : PASS
 + test-ptc-twiss-5D                             (0.01 s) -  5/ 5 : PASS
 + test-ptc-twiss-56D                            (0.01 s) -  3/ 3 : PASS
 + test-ptc-twiss-56Dt                           (0.01 s) -  3/ 3 : PASS
 + test-ptc-twiss-56Dl                           (0.01 s) -  3/ 3 : PASS
 + test-ptc-twiss-56Dtl                          (0.01 s) -  3/ 3 : PASS
[ PTC Normal testsuite ]
 + test-ptc-normal                               (0.00 s) -  5/ 5 : PASS
[ PTC Trackline testsuite ]
 + test-ptc-trackline                            (0.00 s) -  2/ 2 : PASS
 + test-ptc-trackline-2                          (0.00 s) -  2/ 2 : PASS
[ Touschek testsuite ]
 + test-touschek                                 (0.00 s) -  2/ 2 : PASS
 + test-touschek-2                               (0.00 s) -  2/ 2 : PASS
 = tests summary (started at 2013.03.21 21:41:19)
   total diff time 0.52 s  -  total lines 219026  -  total numbers 1134078
   total run  time  380 s  -  total files    234  -  PASSED 234 - FAILED 0
```