

# CHANGE LOG

xint 1.2o

2017/08/29

1.2o (2017/08/29)	p. 1	1.09kb (2014/02/13)	p. 14
1.2n (2017/08/06)	p. 2	1.09k (2014/01/21)	p. 14
1.2m (2017/07/31)	p. 2	1.09j (2014/01/09)	p. 14
1.2l (2017/07/26)	p. 3	1.09i (2013/12/18)	p. 15
1.2k (2017/01/06)	p. 4	1.09h (2013/11/28)	p. 16
1.2j (2016/12/22)	p. 5	1.09g (2013/11/22)	p. 16
1.2i (2016/12/13)	p. 5	1.09f (2013/11/04)	p. 16
1.2h (2016/11/20)	p. 6	1.09e (2013/10/29)	p. 16
1.2g (2016/03/19)	p. 7	1.09d (2013/10/22)	p. 17
1.2f (2016/03/12)	p. 7	1.09c (2013/10/09)	p. 17
1.2e (2015/11/22)	p. 8	1.09b (2013/10/03)	p. 17
1.2d (2015/11/18)	p. 9	1.09a (2013/09/24)	p. 18
1.2c (2015/11/16)	p. 9	1.08b (2013/06/14)	p. 18
1.2b (2015/10/29)	p. 9	1.08a (2013/06/11)	p. 18
1.2a (2015/10/19)	p. 9	1.08 (2013/06/07)	p. 19
1.2 (2015/10/10)	p. 10	1.07 (2013/05/25)	p. 19
1.1c (2015/09/12)	p. 10	1.06b (2013/05/14)	p. 19
1.1b (2015/08/31)	p. 11	1.06 (2013/05/07)	p. 19
1.1a (2014/11/07)	p. 11	1.05 (2013/05/01)	p. 20
1.1 (2014/10/28)	p. 11	1.04 (2013/04/25)	p. 20
1.09n (2014/04/01)	p. 14	1.03 (2013/04/14)	p. 20
1.09m (2014/02/26)	p. 14	1.0 (2013/03/28)	p. 20

Source: xint.dtx 1.2o 2017/08/29 (doc 2017/08/29)

Author: Jean-Francois Burnol

Info: Expandable operations on big integers, decimals, fractions

License: LPPL 1.3c

## 1.2o (2017/08/29)

### Incompatible changes

- **xint**: `\xintAND`, `\xintOR`, ... and similar Boolean logic macros do not apply anymore `\xintNum` (or `\xintRaw` if **xintfrac** is loaded), to their arguments (often, from internal usage of `\xintSgn`), but only f-expand them (using e.g. `\xintiiSgn`). This is kept un-modified even if loading **xintfrac**.

## Deprecated

Deprecated macros raise an error but, generally, then expand as in former releases. They will all get removed at some future release.

- **xintcore**: `\xintiOpp`, `\xintiAbs`, `\xintiAdd`, `\xintiSub`, `\xintiMul`, `\xintiDivision`, `\xintiQuo`, `\xintiRem`, `\xintiDivRound`, `\xintiDivTrunc`, `\xintiMod`, `\xintiSqr`, `\xintiPow`, and `\xintiFac` are deprecated. Only the *ii*-named variants get defined.
- **xintcore**: `\xintCmp` and `\xintSgn` are deprecated from **xintcore** (which only defines `\xintiiCmp` and `\xintiiSgn`) as they actually belong to **xintfrac**.
- **xintcore**: `\xintiiFDg`, resp. `\xintiiLDg`, are renamed `\xintFDg`, resp. `\xintLDg`. Former denominations are deprecated.
- **xint**: `\xintMON`, `\xintMMON`, `\xintiMax`, `\xintiMin`, `\xintiMaxof`, `\xintiMinof`, `\xintiSquareRoot`, `\xintiSqrt`, `\xintiSqrtR`, `\xintiBinomial`, and `\xintiPFactorial` are deprecated. Only *ii*-named variants get defined.
- **xint**: `\xintEq`, `\xintGeq`, `\xintGt`, `\xintLt`, `\xintGtorEq`, `\xintLtorEq`, `\xintIsZero`, `\xintIsNotZero`, `\xintIsOne`, `\xintOdd`, `\xintEven`, `\xintifSgn`, `\xintifCmp`, `\xintifEq`, `\xintifGt`, `\xintifLt`, `\xintifZero`, `\xintifNotZero`, `\xintifOne`, `\xintifOdd`, are deprecated. These macros belong to **xintfrac**. Package **xint** defines only the *ii*-named variants.
- **xint**: `\xintNeq` was renamed to `\xintNotEq` which however is only provided by **xintfrac**. Package **xint** defines `\xintiiNotEq`, and `\xintNeq` is deprecated.
- **xint**: `\xintNot` was renamed to `\xintNOT`, former denomination is deprecated. See also item about Boolean logic macros in the *Incompatible Changes* section.

## 1.2n (2017/08/06)

### Incompatible changes

- **xintbinhex** does not load package **xintcore** anymore, but only **xintkernel**.

### Improvements and new features

- **xintbinhex** has only **xintkernel** as dependency.
- Macros of **xintbinhex** have been improved for speed and increased maximal sizes of allowable inputs.

## 1.2m (2017/07/31)

### Incompatible changes

- **xintbinhex**: the length of the input is now limited. The maximum size depends on the macro and ranges from about 4000 to about 19900 digits.
- **xintbinhex**: `\xintCHexToBin` is now the variant of `\xintHexToBin` which does not remove leading binary zeroes: *N* hex-digits give on output exactly *4N* binary digits.

## Improvements and new features

- **xintbinhex**: all macros have been rewritten using techniques from the 1.2 release (they had remained unmodified since 1.08 of 2013/06/07.) The new macros are faster but limited to a few thousand digits. The 1.08 routines could handle tens of thousands of digits, but not in a reasonable time.

## Bug fixes

- user manual: the **Changes** section wrongly stated at 1.21 that the macros of **xintbinhex** had been made robust against non terminated input such as `\number\mathcode`\-`. Unfortunately the author fell into the trap of believing his own documentation and he forgot to actually implement the change. Now done.
- user manual: the PDF bookmarks were messed up.
- **xint**, **xintfrac**: `\xintGeq`, `\xintMax`, `\xintMin`, suffered from some extra overhead. This was caused by use of some auxiliaries from the very early days which got redefined at some stage. This is fixed here with some additional efficiency improvements and pruning of old code.

## 1.21 (2017/07/26)

### Removed

- `\xintiiSumExpr`, `\xintiiPrdExpr` (**xint**) and `\xintSumExpr`, `\xintPrdExpr` (**xintfrac**). They had not been formally deprecated, but had been left un-documented since 1.09d (2013/10/22).

## Improvements and new features

- the underscore character `_` is accepted by the **xintexpr** parsers as a digit separator (the space character already could be used for improved readability of big numbers). It is not allowed as *first* character of a number, as it would then be mis-interpreted as the start of a possible variable name.
- some refactoring in **xintcore** auxiliary routines and in `\xintiiSub` and `\xintiiCmp` for some small efficiency gains.
- code comments in **xintcore** are better formatted, but remain sparse.
- **xintcore**, **xint**, **xintfrac**, ... : some macros were not robust against arguments whose expansion looks forward for some termination (e.g. `\number\mathcode`\-`), and particularly, most were fragile against inputs using non-terminated `\numexpr` (such as `\xintiiAdd{\the\numexpr1}{2}` or `\xintRaw{\numexpr1}`). This was not a bug per se, as the user manual did not claim such inputs were legal, but it was slightly inconvenient. Most macros (particularly those of **xintfrac**) have now been made robust against such inputs. Some macros from **xintcore** primarily destined to internal usage still accept only properly terminated arguments such as `\the\mathcode`\-<space>` or `\the\numexpr1\relax`.

The situation with expressions is unchanged: syntax such as `\xintexpr \numexpr1+2\relax` is illegal as the ending `\relax` token will get swallowed by the `\numexpr`; but it is needed by the `xintexpr`-ession parser, hence the parser will expand forward and presumably end with in an “illegal token” error, or provoke some low-level TeX error (N.B.: a closing brace `}` for example can not terminate an `xintexpr`-ession, the parser must find a `\relax` token at some point). Thus there must be in this example a second `\relax`.

- experimental code for error conditions; there is no complete user interface yet, it is done in preparation for next major release and is completely unstable and undocumented.

## Bug fixes

- **xintbinhex**: since 1.2 (2015/10/10), `\xintHexToDec` was broken due to an undefined macro (it was in `xint.sty`, but the module by itself is supposedly dependent only upon `xintcore.sty`).
- **xintgcd**: macro `\xintBezout` produced partially wrong output if one of its two arguments was zero.
- **xintfrac**: the manual said one could use directly `\numexpr` compatible expressions in arithmetic macros (without even a `\numexpr` encapsulation) if they were expressed with up to 8 tokens. There was a bug if these 8 tokens evaluated to zero. The bug has been fixed, and up to 9 tokens are now accepted. But it is simpler to use `\the\numexpr` prefix and not to worry about the token count. . . The ending `\relax` is now un-needed.

## 1.2k (2017/01/06)

### Incompatible changes

- macro `\xintFloat` which rounds its input to a floating point number does *not* print anymore `10.0...0eN` to signal an upwards rounding to the next power of ten. The mantissa has in all cases except the zero input exactly one digit before the decimal mark.
- some floating point computations may differ in the least significant digits, due to a change in the rounding algorithm applied to macro arguments expressed as fractions and to an improvement in precision regarding half-integer powers in expressions. See next.

### Improvements and new features

- the initial rounding to the target precision  $P$  which is applied by the floating point macros from **xintfrac** to their arguments achieves the *exact (aka correct) rounding* even for inputs which are fractions with more than  $P+2$  digits in their numerators and denominators ( $>1.$ ) Hence the computed values depend only on the arguments as rational numbers and not upon their representatives. This is not relevant to *expressions (xintexpr)*, because the `\xintfloatexpr` parser sees there `/` as an operator and does not (apart from special constructs) get to manipulate fractions as such.
- `\xintnewdummy` is public interface to a `1.2e` macro which serves to declare any given catcode 11 character as a dummy variable for expressions (**xintexpr**). This is useful for Unicode engines (the Latin letters being already all pre-declared as dummy variables.)
- added `\xintiSqrtR`, there was only `\xintiiSqrtR` alongside `\xintiSqrt` and `\xintiiSqrt` (**xint**).
- added non public `\xintLastItem:f:csv` to **xinttools** for faster `last()` function, and improved `\xintNewExpr` compatibility. Also `\xintFirstItem:f:csv`.

## Bug fixes

- the `1.2f` half-integer powers computed within `\xintfloatexpr` had a silly rounding to the target precision just *before* the final square-root extraction, thus possibly losing some precision. The `1.2k` implementation keeps guard digits for this final square root extraction. As for integer exponents, it

is guaranteed that the computed value differs from the exact one by less than  $0.52 \text{ ulp}$  (for inputs having at most `\xinttheDigits` digits.)

- more regressions from 1.2i were fixed: `\xintLen (xint, xintfrac)` and `\xintDouble (xintcore)` had forgotten that their argument was allowed to be negative. A regression test suite is now in place and is being slowly expanded to cover more macros.
- `\xintiisSquareRoot{0}` now produces `{1}{1}`, which fits better the general documented behaviour of this macro than 11.

## 1.2j (2016/12/22)

### Improvements and new features

- **xinttools** and **xintexpr**:
  1. slightly improves the speed of `\xintTrim`.
  2. speed gains for the handlers of comma separated lists implementing Python-like slicing and item extraction. Relevant non (user) documented macros better documented in `sourcexint.pdf`.
- significant documentations tweaks (inclusive of suppressing things!), and among them two beautiful hyperlinked tables with both horizontal and vertical rules which bring the documentation of the **xintexpr** syntax to a kind of awe-inspiring perfection. . . except that implementation of some math functions is still lacking.

### Bug fixes

- fix two 1.2i regressions caused by undefined macros (`\xintNthElt` in certain branches and `[list][N]` item extraction in certain cases.) The test files existed but were not executed prior to release. Automation in progress.

## 1.2i (2016/12/13)

### Incompatible changes

- `\xintDecSplit` second argument must have no sign (former code replaced it with its absolute value, a sign now may cause an error.)

### Removed

- deprecated macros `\xintifTrue`, `\xintifTrueFalse`, `\xintQuo`, `\xintRem`, `\xintquo`, `\xintrem`.

### Improvements and new features

- **xintkernel**: `\xintLength` is faster. New macros:
  - `\xintLastItem` to fetch the last item from its argument,
  - `\romannumeral\xintgobble` for gobbling many (up to 531440) upstream braced items or tokens.

- `\romannumeral\xintreplicate` which is copied over from the `expl3 \prg_replicate:nn` with some minor changes.
- **xinttools**: general token list handling routines `\xintKeep`, `\xintTrim` and `\xintNthElt` are faster; but the novel `\xintTrim` can only remove up to a maximum of 531440 items.  
Also, `\xintFor` partially improves on some issues which are reported upon in the documentation.
- some old macros have been rewritten entirely or partially using techniques which **xint** started using in release 1.2:
  - **xintcore**: `\xintDouble`, `\xintHalf`, `\xintInc`, `\xintDec`, `\xintiLDg`, `\xintDSR` (originally from **xint**), a novel `\xintDSRr`.
  - **xint**: `\xintDSH`, `\xintDSx`, `\xintDecSplit`, `\xintiiE`.
  - **xintfrac**: as a result of the above `\xintTrunc`, `\xintRound` and `\xintXTrunc` got faster. But the main improvement for them is with decimal inputs which formerly had not been treated separately from the general fraction case. Also, `\xintXTrunc` does not anymore create a dependency of **xintfrac** on **xinttools**.
- the documentation has again been (slightly) re-organized; it has a new sub-section on the Miller-Rabin primality test, to illustrate some use of `\xintNewFunction` for recursive definitions.
- the documentation has dropped the LaTeX “command” terminology (which had been used initially in 2013 for some forgotten reasons and should have been removed long ago) and uses only the more apt “macro”, as after all, all of **xint** is about expansion of macros (plus the use of `\numexpr`).

### Bug fixes

- `\xintDecSplitL` and `\xintDecSplitR` from **xint** produced their output in a spurious brace pair (bug introduced in 1.2f).

## 1.2h (2016/11/20)

### Improvements and new features

- new macro `\xintNewFunction` in **xintexpr** which allows to extend the parser syntax with functions in situations where `\xintdeffunc` is not usable (typically, because dummy variables are used over a not yet determined range of values because it depends on the variables).
- after three years of strict obedience to **xint** prefix, now `\thexintexpr`, `\thexintiexpr`, `\thexintfloatexpr`, and `\thexintiiexpr` are provided as synonyms to `\xinttheexpr`, etc. . .

### Bug fixes

- the `(cond)?{foo}{bar}` operator from **xintexpr** mis-behaved in certain circumstances (such as an empty `foo`).
- the **xintexpr** 1.2f binomial function (which uses `\xintiiBinomial` from **xint.sty** or `\xintFloatBinomial` from **xintfrac.sty**) deliberately raised an error for `binomial(x,y)` with  $y < 0$  or  $x < y$ . This was unfortunate, and it now simply evaluates to zero in such cases.
- similarly the `pfactorial` function was very strict and `pfactorial(x,y)` deliberately raised an out-of-range error if not used with non-negative integers with  $x$  less than  $y$ . It now avoids doing that and allows negative arguments.

- the `add` and `mul` from **xintexpr**, which work with dummy variables since 1.1, raised an error since 1.2c 2015/11/16 when the dummy variable was given an empty range (or list) of values, rather than producing respectively 0 and 1 as formerly.

## 1.2g (2016/03/19)

### Incompatible changes

- inside expressions, list item selector `[L][n]` counts starting at zero, not at one. This is more coherent with `[L][a:b]` which was already exactly like in Python since its introduction. A function `len(L)` replaces earlier `[L][0]`.
- former `iter` keyword now called `iterr`. Indeed it matched with `rrseq`, the new `iter` (which was somehow missing from 1.1) is the one matching `rseq`. Allows to iterate more easily with a “list” variable.

### Improvements and new features

- in **xintexpr.sty**: list selectors `[L][n]` and `[L][a:b]` are more efficient: the earlier 1.1 routines did back and forth conversions from comma separated values to braced tokens, the 1.2g routines use macros from **xinttools.sty** handling directly the encountered lists of comma separated values.
- in **xinttools.sty**: slight improvements in the efficiency of the `\xintNthElt`, `\xintKeep`, `\xintTrim` routines and new routines handling directly comma separated values. The latter are not included in the user manual (they are not `\long`, they don't make efforts to preserve some braces, do not worry about spaces, all those worries being irrelevant to the use in expressions for list selectors).
- a slight speed improvement to `\xintFloatSqrt` in its quest of correct rounding.
- float multiplication and division handle more swiftly operands (non-fractional) with few digits, when the float precision is large.
- the syntax of expressions is described in a devoted chapter of the documentation; an example shows how to implement (expandably) the Brent-Salamin algorithm for computation of Pi using `iter` in a float expression.

## 1.2f (2016/03/12)

### Incompatible changes

- no more `\xintFac` macro but `\xintiFac/\xintiiFac/\xintFloatFac`.

### Improvements and new features

- functions `binomial`, `pfactorial` and `factorial` in both integer and float versions.
- macros `\xintiiBinomial`, `\xintiiPFactorial` (**xint.sty**) and `\xintFloatBinomial`, `\xintFloatPFactorial` (**xintfrac.sty**). Improvements to `\xintFloatFac`.
- faster implementation and increased accuracy of float power macros. Half-integer exponents are now accepted inside float expressions.
- faster implementation of both integral and float square root macros.

- the float square root achieves *correct* (aka *exact*) rounding in arbitrary precision.
- modified behaviour for the `\xintPFloat` macro, used by `\xintthefloatexpr` to prettify its output. It now opts for decimal notation if and only if scientific notation would use an exponent between  $-5$  and  $5$  inclusive. The zero value is printed `0.` with a dot.
- the float macros for addition, subtraction, multiplication, division now first round their two operands to  $P$ , not  $P+2$ , significant places before doing the actual computation ( $P$  being the target precision). The same applies to the power macros and to the square root macro.
- the documentation offers a more precise (and accurate) discussion of floating point issues.
- various under-the-hood code improvements; the floatexpr operations are chained in a faster way, from skipping some unneeded parsing on results of earlier computations. The absence of a real inner data structure for floats (incorporating their precisions, for one) is however still a bit hair raising: currently the lengths of the mantissas of the operands are computed again by each float macro or expression operation.
- (TeXperts only) the macros defined (internally) from `\xintdeffunc` et al. constructs do not incorporate an initial `\romannumeral` anymore.
- renewed desperate efforts at improving the documentation by random shuffling of sections and well thought additions; cuts were considered and even performed.

### Bug fixes

- squaring macro `\xintSqr` from **xintfrac.sty** was broken due to a misspelled sub-macro name. Dates back to 1.1 release of 2014/10/28 :-((.
- 1.2c's fix to the subtraction bug from 1.2 introduced another bug, which in some cases could create leading zeroes in the output, or even worse. This could invalidate other routines using subtractions, like `\xintiisquareRoot`.
- the comparison operators were not recognized by `\xintNewIIExpr` and `\xintdefiifunc` constructs.

### 1.2e (2015/11/22)

#### Improvements and new features

- macro `\xintunassignvar`.
- slight modifications of the logged messages in case of `\xintverbosetrue`.
- a space in `\xintdeffunc f(x)<space>:= expression ;` is now accepted.
- documentation enhancements: the *Quick Sort* section with its included code samples has been entirely re-written; the *Commands of the xintexpr package* section has been extended and reviewed entirely.

### Bug fixes

- in **xintfrac**: the `\xintFloatFac` from release 1.2 parsed its argument only through `\numexpr` but it should have used `\xintNum`.
- in **xintexpr**: release 1.2d had broken the recognition of sub-expressions immediately after variable names (with tacit multiplication).
- in **xintexpr**: contrarily to what 1.2d documentation said, tacit multiplication was not yet always done with enhanced precedence. Now yes.



## 1.2d (2015/11/18)

### Improvements and new features

- the function definitions done by `\xintdeffunc` et al., as well as the macro declarations by `\xintNewExpr` et al. now have only local scope.
- tacit multiplication applies to more cases, for example  $(x+y)z$ , and always ties more than standard `*` infix operator, e.g.  $x/2y$  is like  $x/(2*y)$ .
- some documentation enhancements, particularly in the chapter on `xintexpr.sty`, and also in the code source comments.

### Bug fixes

- in **xintcore**: release 1.2c had inadvertently broken the `\xintiiDivRound` macro.

## 1.2c (2015/11/16)

### Improvements and new features

- macros `\xintdeffunc`, `\xintdefiifunc`, `\xintdeffloatfunc` and boolean `\ifxintverbose`.
- on-going code improvements and documentation enhancements, but stopped in order to issue this bugfix release.

### Bug fixes

- in **xintcore**: recent release 1.2 introduced a bug in the subtraction (happened when 00000001 was found under certain circumstances at certain mod 8 locations).

## 1.2b (2015/10/29)

### Bug fixes

- in **xintcore**: recent release 1.2 introduced a bug in the division macros, causing a crash when the divisor started with 99999999 (it was attempted to use with `1+99999999` a subroutine expecting only 8-digits numbers).

## 1.2a (2015/10/19)

### Improvements and new features

- added `\xintKeepUnbraced`, `\xintTrimUnbraced` (**xinttools**) and fixed documentation of `\xintKeep` and `\xintTrim` regarding brace stripping.
- added `\xintiiMaxof`/`\xintiiMinof` (**xint**).
- TeX hackers only: replaced all code uses of `\romannumeral-`0` by the quicker `\romannumeral`&&@` (`^` being used as letter, had to find another character usable with catcode 7).

## Bug fixes

- in **xintexpr**: recent release 1.2 introduced a bad bug in the parsing of decimal numbers and as a result `\xinttheexpr 0.01\relax` expanded to `0!` (sigh...)

## 1.2 (2015/10/10)

### Removed

- the macros `\xintAdd`, `\xintSub`, `\xintMul`, `\xintMax`, `\xintMin`, `\xintMaxof`, `\xintMinof` are removed from package **xint**, and only exist in the versions from **xintfrac**. With only **xintcore** or **xint** loaded, one *must* use `\xintiiAdd`, `\xintiiSub`, ..., or `\xintiAdd`, `\xintiSub`, etc...

### Improvements and new features

- the basic arithmetic implemented in **xintcore** has been entirely rewritten. The mathematics remains the elementary school one, but the TeX implementation achieves higher speed (except, regarding addition/subtraction, for numbers up to about thirty digits), the gains becoming quite significant for numbers with hundreds of digits.
- the inputs must have less than 19959 digits. But computations with thousands of digits take time.
- a previously standing limitation of `\xintexpr`, `\xintiiexpr`, and of `\xintfloatexpr` to numbers of less than 5000 digits has been lifted.
- a *qint* function is provided to help the parser gather huge integers in one-go, as an exception to its normal mode of operation which expands token by token.
- `\xintFloatFac` macro for computing the factorials of integers as floating point numbers to a given precision. The `!` postfix operator inside `\xintfloatexpr` maps to this new macro rather than to the exact factorial as used by `\xintexpr` and `\xintiiexpr`.
- there is more flexibility in the parsing done by the macros from **xintfrac** on fractional input: the decimal parts of both the numerator and the denominator may arise from a separate expansion via `\romannumeral-`0`. Also the strict `A/B[N]` format is a bit relaxed: `N` may be anything understood by `\numexpr` (it could even be empty but that possibility has been removed by later 1.2f release.)
- on the other hand an isolated dot `.` is not legal syntax anymore inside the expression parsers: there must be digits either before or after. It remains legal input for the macros of **xintfrac**.
- added `\ht`, `\dp`, `\wd`, `\fontcharht`, etc... to the tokens recognized by the parsers and expanded by `\number`.
- an obscure bug in package **xintkernel** has been fixed, regarding the sanitization of catcodes: under certain circumstances (which could not occur in a normal LaTeX context), unusual catcodes could end up being propagated to the external world.
- an effort at randomly shuffling around various pieces of the documentation has been done.

## 1.1c (2015/09/12)

- bugfix regarding macro `\xintAssign` from **xinttools** which did not behave correctly in some circumstances (if there was a space before `\to`, in particular).
- very minor code improvements, and correction of some issues regarding the source code formatting in `sourcexint.pdf`, and minor issues in `Makefile.mk`.

## 1.1b (2015/08/31)

- bugfix: some macros needed by the integer division routine from **xintcore** had been left in **xint.sty** since release 1.1. This for example broke the `\xintGCD` from **xintgcd** if package **xint** was not loaded.
- Slight enhancements to the documentation, particularly in the `Read this first` section.

## 1.1a (2014/11/07)

- fixed a bug which prevented `\xintNewExpr` from producing correctly working macros from a comma separated replacement text.
- `\xintiiSqrtR` for rounded integer square root; former `\xintiiSqrt` already produced truncated integer square root; corresponding function `sqrtr` added to `\xintiiexpr.. \relax` syntax.
- use of straight quotes in the documentation for better legibility.
- added `\xintiiIsOne`, `\xintiiifOne`, `\xintiiifCmp`, `\xintiiifEq`, `\xintiiifGt`, `\xintiiifLt`, `\xintiiifOdd`, `\xintiiCmp`, `\xintiiEq`, `\xintiiGt`, `\xintiiLt`, `\xintiiLtorEq`, `\xintiiGtorEq`, `\xintiiNeq`, mainly for efficiency of `\xintiiexpr`.
- for the same reason, added `\xintiiGCD` and `\xintiiLCM`.
- added the previously mentioned `ii` macros, and some others from 1.1, to the user manual. But their main usage is internal to `\xintiiexpr`, to skip unnecessary overheads.
- various typographical fixes throughout the documentation, and a bit of clean up of the code comments. Improved `\Factors` example of nested `subs`, `rseq`, `iter` in `\xintiiexpr`.

## 1.1 (2014/10/28)

### Incompatible changes

- in `\xintiiexpr`, `/` does *rounded* division, rather than the Euclidean division (for positive arguments, this is truncated division). The `//` operator does truncated division,
- the `:` operator for three-way branching is gone, replaced with `??`,
- `1e(3+5)` is now illegal. The number parser identifies `e` and `E` in the same way it does for the decimal mark, earlier versions treated `e` as `E` rather as infix operators of highest precedence,
- the `add` and `mul` have a new syntax, old syntax is with ``+`` and ``*`` (left quotes mandatory), `sum` and `prd` are gone,
- no more special treatment for encountered brace pairs `{. . }` by the number scanner, `a/b[N]` notation can be used without use of braces (the `N` will end up as is in a `\numexpr`, it is not parsed by the `\xintexpr-ession` scanner),
- in earlier releases, place holders for `\xintNewExpr` could either be denoted `#1`, `#2`, `...` or also `$1`, `$2`, `...`. Only the usual `#` form is now accepted and the special cases previously treated via the second form are now managed via a `protect(...)` function.

### Removed

- `\xintnumexpr`, `\xintthenumexpr`, `\xintNewNumExpr`: use `\xintiexpr`, `\xinttheiexpr`, `\xintNewIExpr`.

## Deprecated

- `\xintDivision`, `\xintQuo`, `\xintRem`: use `\xintiDivision`, `\xintiQuo`, `\xintiRem`.
- `\xintMax`, `\xintMin`, `\xintAdd`, `\xintSub`, `\xintMul` (**xint**): their usage without **xintfrac** is deprecated; use `\xintiMax`, `\xintiMin`, `\xintiAdd`, `\xintiSub`, `\xintiMul`.
- the `&` and `|` as Boolean operators in `xintexpr`-essions are deprecated in favour of `&&` and `||`. The single letter operators might be assigned some other meaning in some later release (bitwise operations, perhaps). Do not use them.

## Improvements and new features

- new package **xintcore** has been split off **xint**. It contains the core arithmetic macros (it is loaded by LaTeX package **bnumexpr**),
- neither **xint** nor **xintfrac** load **xinttools**. Only **xintexpr** does,
- whenever some portion of code has been revised, often use has been made of the `\xint_dothis` and `\xint_orthat` pair of macros for expandably branching,
- these tiny helpful macros, and a few others are in package **xintkernel** which contains also the catcode and loading order management code, initially inspired by code found in Heiko Oberdiek's packages,
- the source code, which was suppressed from `xint.pdf` in release 1.09n, is now compiled into a separate file `sourcexint.pdf`,
- faster handling by `\xintAdd`, `\xintSub`, `\xintMul`, ... of the case where one of the arguments is zero,
- the `\xintAdd` and `\xintSub` macros from package **xintfrac** check if one of the denominators is a multiple of the other, and only if this is not the case do they multiply the denominators. But systematic reduction would be too costly,
- this naturally will be also the case for the `+` and `-` operations in `\xintexpr`,
- macros `\xintiiDivRound`, `\xintiiDivTrunc` and `\xintiiMod` for rounded and truncated division of big integers (now in **xintcore**), alongside the earlier `\xintiiQuo` and `\xintiiRem`,
- with **xintfrac** loaded, the `\xintNum` macro does `\xintTTrunc` (which is truncation to an integer, same as `\xintiTrunc {0}`),
- macro `\xintMod` in **xintfrac** for modulo operation with fractional numbers,
- `\xintiexpr`, `\xinttheiexpr` admit an optional argument within brackets [*d*], they round the computation result (or results, if comma separated) to *d* digits after decimal mark, (the whole computation is done exactly, as in `xintexpr`),
- `\xintfloatexpr`, `\xintthefloatexpr` similarly admit an optional argument which serves to keep only *d* digits of precision, getting rid of cumulated uncertainties in the last digits (the whole computation is done according to the precision set via `\xintDigits`),
- `\xinttheexpr` and `\xintthefloatexpr` *pretty-print* if possible, the former removing unit denominator or [0] brackets, the latter avoiding scientific notation if decimal notation is practical,
- the `//` does truncated division and `/:` is the associated modulo,
- multi-character operators `&&`, `||`, `==`, `<=`, `>=`, `!=`, `**`,
- multi-letter infix binary words 'and', 'or', 'xor', 'mod' (straight quotes mandatory),

- functions `even`, `odd`,
- `\xintdefvar A3:=3.1415`; for variable definitions (non expandable, naturally), usable in subsequent expressions; variable names may contain letters, digits, underscores. They should not start with a digit, the `@` is reserved, and single lowercase and uppercase Latin letters are predefined to work as dummy variables (see next),
- generation of comma separated lists `a..b`, `a..[d]..b`,
- Python syntax-like list extractors `[list][n:]`, `[list][:n]`, `[list][a:b]` allowing negative indices, but no optional step argument, and `[list][n]` (`n=0` for the number of items in the list),
- functions `first`, `last`, `reversed`,
- itemwise operations on comma separated lists `a*[list]`, etc., possible on both sides `a*[list]^b`, and obeying the same precedence rules as with numbers,
- `add` and `mul` must use a dummy variable: `add(x(x+1)(x-1), x=-10..10)`,
- variable substitutions with `subs`: `subs(subs(add(x^2+y^2, x=1..y), y=t), t=20)`,
- sequence generation using `seq` with a dummy variable: `seq(x^3, x=-10..10)`,
- simple recursive lists with `rseq`, with `@` given the last value, `rseq(1;2@+1, i=1..10)`,
- higher recursion with `rrseq`, `@1`, `@2`, `@3`, `@4`, and `@@n` for earlier values, up to `n=K` where `K` is the number of terms of the initial stretch `rrseq(0, 1;@1+@2, i=2..100)`,
- iteration with `iter` which is like `rrseq` but outputs only the last `K` terms, where `K` was the number of initial terms,
- inside `seq`, `rseq`, `rrseq`, `iter`, possibility to use `omit`, `abort` and `break` to control termination,
- `n++` potentially infinite index generation for `seq`, `rseq`, `rrseq`, and `iter`, it is advised to use `abort` or `break(..)` at some point,
- the `add`, `mul`, `seq`, ... are nestable,
- `\xintthecoords` converts a comma separated list of an even number of items to the format expected by the TikZ coordinates syntax,
- completely new version `\xintNewExpr`, `protect` function to handle external macros. The dollar sign `$` for place holders is not accepted anymore, only the standard macro parameter `#`. Not all constructs are compatible with `\xintNewExpr`.

## Bug fixes

- `\xintZapFirstSpaces` hence also `\xintZapSpaces` from package **xinttools** were buggy when used with an argument either empty or containing only space tokens.
- `\xintiexpr` did not strip leading zeroes, hence `\xinttheiexpr 001+1\relax` did not obtain the expected result ...
- `\xinttheexpr \xintiexpr 1.23\relax\relax` should have produced 1, but it produced 1.23
- the catcode of `;` was not set at package launching time.
- the `\XINTinFloatPrd:csv` macro name had a typo, hence `prd` was non-functional in `\xintfloatexpr`.

### 1.09n (2014/04/01)

- the user manual does not include by default the source code anymore: the `\NoSourceCode` toggle in file `xint.tex` has to be set to 0 before compilation to get source code inclusion (later release 1.1 made source code available as `sourcexint.pdf`).
- bug fix (**xinttools**) in `\XINT_nthelt_finish` (this bug was introduced in 1.09i of 2013/12/18 and showed up when the index  $N$  was larger than the number of elements of the list).

### 1.09m (2014/02/26)

- new in **xinttools**: `\xintKeep` keeps the first  $N$  or last  $N$  elements of a list (sequence of braced items); `\xintTrim` cuts out either the first  $N$  or the last  $N$  elements from a list.
- new in **xintfrac**: `\xintFGtoC` finds the initial partial quotients common to two numbers or fractions  $f$  and  $g$ ; `\xintGGCFrac` is a clone of `\xintGCFrac` which however does not assume that the coefficients of the generalized continued fraction are numeric quantities. Some other minor changes.

### 1.09kb (2014/02/13)

- bug fix (**xintexpr**): an aloof modification done by 1.09i to `\xintNewExpr` had resulted in a spurious trailing space present in the outputs of all macros created by `\xintNewExpr`, making nesting of such macros impossible.
- bug fix (**xinttools**): `\xintBreakFor` and `\xintBreakForAndDo` were buggy when used in the last iteration of an `\xintFor` loop.
- bug fix (**xinttools**): `\xintSeq` from 1.09k needed a `\chardef` which was missing from `xinttools.sty`, it was in `xint.sty`.

### 1.09k (2014/01/21)

- inside `\xintexpr.. \relax` (and its variants) tacit multiplication is implied when a number or operand is followed directly with an opening parenthesis,
- the " for denoting (arbitrarily big) hexadecimal numbers is recognized by `\xintexpr` and its variants (package **xintbinhex** is required); a fractional hexadecimal part introduced by a dot `.` is allowed.
- re-organization of the first sections of the user manual.
- bug fix (**xinttools**, **xint**, ...): forgotten catcode check of " at loading time has been added.

### 1.09j (2014/01/09)

- (**xint**) the core division routines have been re-written for some (limited) efficiency gain, more pronounced for small divisors. As a result the *computation of one thousand digits of  $\pi$*  is close to three times faster than with earlier releases.
- some various other small improvements, particularly in the power routines.
- (**xintfrac**) a macro `\xintXTrunc` is designed to produce thousands or even tens of thousands of digits of the decimal expansion of a fraction. Although completely expandable it has its use limited to inside an `\edef`, `\write`, `\message`, .... It can thus not be nested as argument to another package macro.

- (**xintexpr**) the tacit multiplication done in `\xintexpr...\relax` on encountering a count register or variable, or a `\numexpr`, while scanning a (decimal) number, is extended to the case of a sub `\xintexpr`-ession.
- `\xintexpr` can now be used in an `\edef` with no `\xintthe` prefix; it will execute completely the computation, and the error message about a missing `\xintthe` will be inhibited. Previously, in the absence of `\xintthe`, expansion could only be a full one (with `\romannumeral-`0`), not a complete one (with `\edef`). Note that this differs from the behavior of the non-expandable `\numexpr`: `\the` or `\number` (or `\romannumeral`) are needed not only to print but also to trigger the computation, whereas `\xintthe` is mandatory only for the printing step.
- the default behavior of `\xintAssign` is changed, it now does not do any further expansion beyond the initial full-expansion which provided the list of items to be assigned to macros.
- bug fix (**xintfrac**): 1.09i did an unexplainable change to `\XINT_infloat_zero` which broke the floating point routines for vanishing operands `=:(((`
- bug fix: the 1.09i `xint.ins` file produced a buggy `xint.tex` file.

### 1.09i (2013/12/18)

- (**xintexpr**) `\xintiexpr` is a variant of `\xintexpr` which is optimized to deal only with (long) integers, `/` does a euclidean quotient.
- *deprecated*: `\xintnumexpr`, `\xintthenumexpr`, `\xintNewNumExpr` are renamed, respectively, `\xintiexpr`, `\xinttheiexpr`, `\xintNewIExpr`. The earlier denominations are kept but are to be removed at some point.
- it is now possible within `\xintexpr...\relax` and its variants to use count, dimen, and skip registers or variables without explicit `\the/\number`: the parser inserts automatically `\number` and a tacit multiplication is implied when a register or variable immediately follows a number or fraction. Regarding dimensions and `\number`, see the further discussion in *Dimensions*.
- (**xintfrac**) conditional `\xintifOne`; `\xintifTrueFalse` renamed to `\xintifTrueAelseB`; macros `\xintTFrac` (fractional part, mapped to function `frac` in `\xintexpr`-essions), `\xintFloatE`.
- (**xinttools**) `\xintAssign` admits an optional argument to specify the expansion type to be used: `[]` (none, default), `[o]` (once), `[oo]` (twice), `[f]` (full), `[e]` (`\edef`),... to define the macros
- **xinttools** defines `\odef`, `\oodef`, `\fdef` (if the names have already been assigned, it uses `\xintoodef` etc...). These tools are provided for the case one uses the package macros in a non-expandable context. `\oodef` expands twice the macro replacement text, and `\fdef` applies full expansion. They are useful in situations where one does not want a full `\edef`. `\fdef` appears to be faster than `\oodef` in almost all cases (with less than thousand digits in the result), and even faster than `\edef` for expanding the package macros when the result has a few dozens of digits. `\oodef` needs that expansion ends up in thousands of digits to become competitive with the other two.
- some across the board slight efficiency improvement as a result of modifications of various types to *fork macros* and *branching conditionals* which are used internally.
- bug fix (**xint**): `\xintAND` and `\xintOR` inserted a space token in some cases and did not expand as promised in two steps `:-((` (bug dating back to 1.09a I think; this bug was without consequences when using `&` and `|` in `\xintexpr`-essions, it affected only the macro form).
- bug fix (**xintfrac**): `\xintFtoCCv` still ended fractions with the `[0]`'s which were supposed to have been removed since release 1.09b.
- *deprecated*: `\xintifTrueFalse`, `\xintifTrue`; use `\xintifTrueAelseB`.

### 1.09h (2013/11/28)

- parts of the documentation have been re-written or re-organized, particularly the discussion of expansion issues and of input and output formats.
- the expansion types of macro arguments are documented in the margin of the macro descriptions, with conventions mainly taken over from those in the LaTeX3 documentation.
- a dependency of **xinttools** on **xint** (inside `\xintSeq`) has been removed.
- (**xintgcd**) `\xintTypesetEuclideanAlgorithm` and `\xintTypesetBezoutAlgorithm` have been slightly modified (regarding indentation).
- (**xint**) macros `xintiSum` and `xintiPrd` are renamed to `\xintiiSum` and `\xintiiPrd`.
- (**xinttools**) a count register used in 1.09g in the `\xintFor` loops for parsing purposes has been removed and replaced by use of a `\numexpr`.
- the few uses of `\loop` have been replaced by `\xintloop`/`\xintiloop`.
- all macros of **xinttools** for which it makes sense are now declared `\long`.

### 1.09g (2013/11/22)

- a package **xinttools** is detached from **xint**, to make tools such as `\xintFor`, `\xintApplyUnbraced`, and `\xintiloop` available without the **xint** overhead.
- expandable nestable loops `\xintloop` and `\xintiloop`.
- bugfix: `\xintFor` and `\xintFor*` do not modify anymore the value of `\count 255`.

### 1.09f (2013/11/04)

- (**xint**) `\xintZapFirstSpaces`, `\xintZapLastSpaces`, `\xintZapSpaces`, `\xintZapSpacesB`, for expandably stripping away leading and/or ending spaces.
- `\xintCSVtoList` by default uses `\xintZapSpacesB` to strip away spaces around commas (or at the start and end of the comma separated list).
- also the `\xintFor` loop will strip out all spaces around commas and at the start and the end of its list argument; and similarly for `\xintForpair`, `\xintForthree`, `\xintForfour`.
- `\xintFor` *et al.* accept all macro parameters from #1 to #9.
- for reasons of inner coherence some macros previously with one extra *i* in their names (e.g. `\xintiMON`) now have a doubled *ii* (`\xintiiMON`) to indicate that they skip the overhead of parsing their inputs via `\xintNum`. Macros with a *single* *i* such as `\xintiAdd` are those which maintain the non-**xintfrac** output format for big integers, but do parse their inputs via `\xintNum` (since release 1.09a). They too may have doubled-*i* variants for matters of programming optimization when working only with (big) integers and not fractions or decimal numbers.

### 1.09e (2013/10/29)

- (**xint**) `\xintintegers`, `\xintdimensions`, `\xintrationals` for infinite `\xintFor` loops, interrupted with `\xintBreakFor` and `\xintBreakForAndDo`.
- `\xintifForFirst`, `\xintifForLast` for the `\xintFor` and `\xintFor*` loops,



- the `\xintFor` and `xintFor*` loops are now `\long`, the replacement text and the items may contain explicit `\par`'s.
- conditionals `\xintifCmp`, `\xintifInt`, `\xintifOdd`.
- bug fix (**xint**): the `\xintFor` loop (not `\xintFor*`) did not correctly detect an empty list.
- bug fix (**xint**): `\xintiSqrt {0}` crashed. `:-((`
- the documentation has been enriched with various additional examples, such as the *the quick sort algorithm illustrated* or the various ways of *computing prime numbers*.
- the documentation explains with more details various expansion related issues, particularly in relation to conditionals.

### 1.09d (2013/10/22)

- bug fix (**xint**): `\xintFor*` is modified to gracefully handle a space token (or more than one) located at the very end of its list argument (as the space before `\do` in `\xintFor* #1 in {{a}{b}{c}<space> \do {stuff}}`; spaces at other locations were already harmless). Furthermore this new version *f-expands* the un-braced list items. After `\def\x{{1}{2}}` and `\def\y{{a}\x {b}{c}\x }`, `\y` will appear to `\xintFor*` exactly as if it had been defined as `\def\y{{a}{1}{2}{b}{c}{1}{2}}`.
- same bug fix for `\xintApplyInline`.

### 1.09c (2013/10/09)

- (**xintexpr**) added `bool` and `togl` to the `\xintexpr` syntax; also added `\xintboolexpr` and `\xintifboolexpr`.
- added `\xintNewNumExpr`.
- the factorial `!` and branching `?`, `:`, operators (in `\xintexpr... \relax`) have now less precedence than a function name located just before,
- (**xint**) `\xintFor` is a new type of loop, whose replacement text inserts the comma separated values or list items via macro parameters, rather than encapsulated in macros; the loops are nestable up to four levels (nine levels since 1.09f) and their replacement texts are allowed to close groups as happens with the tabulation in alignments,
- `\xintForpair`, `\xintForthree`, `\xintForfour` are experimental variants of `\xintFor`,
- `\xintApplyInline` has been enhanced in order to be usable for generating rows (partially or completely) in an alignment,
- command `\xintSeq` to generate (expandably) arithmetic sequences of (short) integers,
- again various improvements and changes in the documentation.

### 1.09b (2013/10/03)

- various improvements in the documentation,
- more economical catcode management and re-loading handling,
- removal of all those `[0]`'s previously forcefully added at the end of fractions by various macros of **xintfrac**,
- `\xintNthElt` with a negative index returns from the tail of the list,
- macro `\xintPRaw` to have something like what `\xintFrac` does in math mode; i.e. a `\xintRaw` which does not print the denominator if it is one.

## 1.09a (2013/09/24)

- (**xintexpr**) `\xintexpr..\relax` and `\xintfloatexpr..\relax` admit functions in their syntax, with comma separated values as arguments, among them `reduce`, `sqr`, `sqrt`, `abs`, `sgn`, `floor`, `ceil`, `quo`, `rem`, `round`, `trunc`, `float`, `gcd`, `lcm`, `max`, `min`, `sum`, `prd`, `add`, `mul`, `not`, `all`, `any`, `xor`.
- comparison (`<`, `>`, `=`) and logical (`|`, `&`) operators.
- the command `\xintthe` which converts `\xint` expressions into printable format (like `\the` with `\numexpr`) is more efficient, for example one can do `\xintthe\x` if `\x` was defined to be an `\xintexpr..\relax`:

```
\def\x{\xintexpr 3^57\relax}
\def\y{\xintexpr \x^(-2)\relax}
\def\z{\xintexpr \y-3^114\relax}
\xintthe\z
```
- `\xintnumexpr .. \relax` (now renamed `\xintiexpr`) is `\xintexpr round( .. ) \relax`.
- `\xintNewExpr` now works with the standard macro parameter character `#`.
- both regular `\xint` expressions and commands defined by `\xintNewExpr` will work with comma separated lists of expressions,
- commands `\xintFloor`, `\xintCeil`, `\xintMaxof`, `\xintMinof` (package **xintfrac**), `\xintGCDof`, `\xintLCM`, `\xintLCMof` (package **xintgcd**), `\xintifLt`, `\xintifGt`, `\xintifSgn`, `\xintANDof`, ...
- The arithmetic macros from package **xint** now filter their operands via `\xintNum` which means that they may use directly count registers and `\numexpr`-essions without having to prefix them by `\the`. This is thus similar to the situation holding previously already when **xintfrac** was loaded.
- a bug (**xintfrac**) introduced in 1.08b made `\xintCmp` crash when one of its arguments was zero.  
:-((

## 1.08b (2013/06/14)

- (**xintexpr**) Correction of a problem with spaces inside `\xint` expressions.
- (**xintfrac**) Additional improvements to the handling of floating point numbers.
- section *Use of count registers* documenting how count registers may be directly used in arguments to the macros of **xintfrac**.

## 1.08a (2013/06/11)

- (**xintfrac**) Improved efficiency of the basic conversion from exact fractions to floating point numbers, with ensuing speed gains especially for the power function macros `\xintFloatPow` and `\xintFloatPower`,
- Better management by `\xintCmp`, `\xintMax`, `\xintMin` and `\xintGeq` of inputs having big powers of ten in them.
- Macros for floating point numbers added to the **xintseries** package.

## 1.08 (2013/06/07)

- (**xint** and **xintfrac**) Macros for extraction of square roots, for floating point numbers (`\xintFloatSqrt`), and integers (`\xintiSqrt`).
- new package **xintbinhex** providing *conversion routines* to and from binary and hexadecimal bases.

## 1.07 (2013/05/25)

- The **xintexpr** package is a new core constituent (which loads automatically **xintfrac** and **xint**) and implements the expandable expanding parser

```
\xintexpr . . . \relax,
```

and its variant

```
\xintfloatexpr . . . \relax
```

allowing on input formulas using the infix operators `+`, `-`, `*`, `/`, and `^`, and arbitrary levels of parenthesizing. Within a float expression the operations are executed according to the current value set by `\xintDigits`. Within an `\xintexpr`-ession the binary operators are computed exactly.

To write the `\xintexpr` parser I benefited from the commented source of the `l3fp` parser; the `\xintexpr` parser has its own features and peculiarities. *See its documentation.*

- The floating point precision `D` is set (this is a local assignment to a `\mathchar` variable) with `\xintDigits := D`; and queried with `\xinttheDigits`. It may be set to anything up to 32767.<sup>1</sup> The macro incarnations of the binary operations admit an optional argument which will replace pointwise `D`; this argument may exceed the 32767 bound.
- The **xintfrac** macros now accept numbers written in scientific notation, the `\xintFloat` command serves to output its argument with a given number `D` of significant figures. The value of `D` is either given as optional argument to `\xintFloat` or set with `\xintDigits := D`; . The default value is 16.

## 1.06b (2013/05/14)

- Minor code and documentation improvements. Everywhere in the source code, a more modern underscore has replaced the `@` sign.

## 1.06 (2013/05/07)

- Some code improvements, particularly for macros of **xint** doing loops.
- New utilities in **xint** for expandable manipulations of lists:

```
\xintNthElt, \xintCSVtoList, \xintRevWithBraces
```

- The macros did only a double expansion of their arguments. They now fully expand them (using `\romannumeral-`0`). Furthermore, in the case of arguments constrained to obey the TeX bounds they will be inserted inside a `\numexpr . . \relax`, hence completely expanded, one may use count registers, even infix arithmetic operations, etc. . .

---

<sup>1</sup>but values higher than 100 or 200 will presumably give too slow evaluations.

## 1.05 (2013/05/01)

Minor changes and additions to **xintfrac** and **xintcfrac**.

## 1.04 (2013/04/25)

- New component **xintcfrac** devoted to continued fractions.
- bug fix (**xintfrac**): `\xintIrr {0}` crashed.
- faster division routine in **xint**, new macros to deal expandably with token lists.
- `\xintRound` added.
- **xintseries** has a new implementation of `\xintPowerSeries` based on a Horner scheme, and new macro `\xintRationalSeries`. Both to help deal with the *denominator buildup* plague.
- `tex xint.dtx` extracts style files (no need for a `xint.ins`).

## 1.03 (2013/04/14)

- new modules **xintfrac** (expandable operations on fractions) and **xintseries** (expandable partial sums with xint package).
- slightly improved division and faster multiplication (the best ordering of the arguments is chosen automatically).
- added illustration of Machin algorithm to the documentation.

## 1.0 (2013/03/28)

Initial announcement:

The **xint** package implements with expandable TeX macros the basic arithmetic operations of addition, subtraction, multiplication and division, as applied to arbitrarily long numbers represented as chains of digits with an optional minus sign.

The **xintgcd** package provides implementations of the Euclidean algorithm and of its type-setting.

The packages may be used with Plain and with LaTeX.