# The Numerical Utilities Reference Manual

Numerical utilities for Common Lisp, version 1.1

**Steven Nunez** `<steve@symbolics.tech>`

# Table of Contents

# 1 Systems

The main system appears first, followed by any subsystem dependency.

## 1.1 `num-utils`

**Author** Steven Nunez `<steve@symbolics.tech>`

**License** MS-PL

**Description**
Numerical utilities for Common Lisp

**Version** 1.1

**Dependencies**
- `anaphora`
- `alexandria`
- `array-operations`
- `select`
- `let-plus`

**Source** [`num-utils.asd`], page 3, (file)

**Directory** `s:/src/num-utils/`

**Components**
- [`packages.lisp`], page 3, (file)
- [`utilities.lisp`], page 3, (file)
- [`num=.lisp`], page 4, (file)
- [`arithmetic.lisp`], page 4, (file)
- [`elementwise.lisp`], page 5, (file)
- [`extended-real.lisp`], page 8, (file)
- [`interval.lisp`], page 8, (file)
- [`print-matrix.lisp`], page 10, (file)
- [`matrix.lisp`], page 10, (file)
- [`matrix-shorthand.lisp`], page 13, (file)
- [`statistics.lisp`], page 13, (file)
- [`chebyshev.lisp`], page 16, (file)
- [`polynomial.lisp`], page 16, (file)
- [`rootfinding.lisp`], page 16, (file)
- [`quadrature.lisp`], page 17, (file)
- [`test-utilities.lisp`], page 19, (file)
- [`common-package.lisp`], page 19, (file)

# 2 Files

Files are sorted by type and then listed depth-first from the systems components trees.

## 2.1 Lisp

### 2.1.1 `num-utils.asd`

**Location** `/src/num-utils/num-utils.asd`

**Systems** [`num-utils`], page 1, (system)

### 2.1.2 `num-utils/packages.lisp`

**Parent** [`num-utils`], page 1, (system)

**Location** `packages.lisp`

**Packages**

- [`num-utils.polynomial`], page 21,
- [`num-utils.interval`], page 21,
- [`num-utils.print-matrix`], page 23,
- [`num-utils.utilities`], page 23,
- [`num-utils.arithmetic`], page 24,
- [`num-utils.matrix-shorthand`], page 25,
- [`num-utils.num=`], page 26,
- [`num-utils.test-utilities`], page 26,
- [`num-utils.extended-real`], page 27,
- [`num-utils.quadrature`], page 28,
- [`num-utils.statistics`], page 30,
- [`num-utils.rootfinding`], page 32,
- [`num-utils.elementwise`], page 33,
- [`num-utils.chebyshev`], page 37,
- [`num-utils.matrix`], page 37,

### 2.1.3 `num-utils/utilities.lisp`

**Dependency**
[`packages.lisp`], page 3, (file)

**Parent** [`num-utils`], page 1, (system)

**Location** `utilities.lisp`

**Exported Definitions**

- [`as-alist`], page 56, (generic function)
- [`as-double-float`], page 45, (function)
- [`as-plist`], page 56, (generic function)
- [`as-plist`], page 57, (method)
- [`as-simple-fixnum-vector`], page 45, (function)
- [`bic`], page 45, (function)
- [`binary-search`], page 45, (function)

- [check-types], page 41, (macro)
- [curry*], page 42, (macro)
- [define-with-multiple-bindings], page 42, (macro)
- [expanding], page 42, (macro)
- [fixnum?], page 49, (function)
- [generate-sequence], page 49, (function)
- [gethash*], page 42, (macro)
- [make-vector], page 44, (compiler macro)
- [make-vector], page 51, (function)
- [sequencep], page 54, (function)
- [simple-double-float-vector], page 79, (type)
- [simple-fixnum-vector], page 79, (type)
- [simple-single-float-vector], page 79, (type)
- [splice-awhen], page 43, (macro)
- [splice-when], page 43, (macro)
- [unlessf], page 43, (macro)
- [with-double-floats], page 44, (macro)
- [within?], page 56, (function)

### 2.1.4 num-utils/num=.lisp

**Dependency**
[utilities.lisp], page 3, (file)

**Parent** [num-utils], page 1, (system)

**Location** num=.lisp

**Exported Definitions**

- [*num=-tolerance*], page 41, (special variable)
- [define-num=-with-accessors], page 42, (macro)
- [define-structure-num=], page 42, (macro)
- [num-delta], page 51, (function)
- [num=], page 66, (generic function)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=-function], page 52, (function)

### 2.1.5 num-utils/arithmetic.lisp

**Dependency**
[num=.lisp], page 4, (file)

**Parent** [num-utils], page 1, (system)

**Location** arithmetic.lisp

**Exported Definitions**

- [1c], page 44, (function)
- [abs-diff], page 45, (function)
- [absolute-square], page 45, (function)
- [as-integer], page 45, (function)
- [ceiling*], page 46, (function)
- [cumulative-product], page 47, (function)
- [cumulative-sum], page 47, (function)
- [divides?], page 47, (function)
- [floor*], page 49, (function)
- [ivec], page 50, (function)
- [l2norm], page 50, (function)
- [l2norm-square], page 65, (generic function)
- [l2norm-square], page 65, (method)
- [log10], page 50, (function)
- [log2], page 51, (function)
- [multf], page 43, (macro)
- [normalize-probabilities], page 51, (function)
- [numseq], page 52, (function)
- [product], page 67, (generic function)
- [product], page 67, (method)
- [product], page 67, (method)
- [round*], page 53, (function)
- [same-sign-p], page 53, (function)
- [sequence-maximum], page 53, (function)
- [sequence-minimum], page 53, (function)
- [square], page 54, (function)
- [sum], page 69, (generic function)
- [sum], page 69, (method)
- [sum], page 69, (method)
- [truncate*], page 55, (function)

**Internal Definitions**

- [define-rounding-with-offset], page 81, (macro)
- [similar-element-type], page 89, (function)
- [similar-sequence-type], page 89, (function)

### 2.1.6 `num-utils/elementwise.lisp`

**Dependency**
[arithmetic.lisp], page 4, (file)

**Parent** [num-utils], page 1, (system)

**Location** `elementwise.lisp`

**Exported Definitions**

- [e*], page 47, (function)

- [e2/], page 62, (method)
- [e2log], page 62, (generic function)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [eceiling], page 63, (generic function)
- [eceiling], page 63, (method)
- [eceiling], page 63, (method)
- [econjugate], page 63, (generic function)
- [econjugate], page 63, (method)
- [econjugate], page 63, (method)
- [eexp], page 63, (generic function)
- [eexp], page 63, (method)
- [eexp], page 63, (method)
- [eexpt], page 63, (generic function)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [efloor], page 64, (generic function)
- [efloor], page 64, (method)
- [efloor], page 64, (method)
- [elementwise-float-contagion], page 48, (function)
- [elog], page 48, (function)
- [emax], page 48, (function)
- [emin], page 48, (function)
- [ereduce], page 64, (generic function)
- [ereduce], page 64, (method)
- [ereduce], page 64, (method)
- [ereduce], page 64, (method)
- [esqrt], page 64, (generic function)
- [esqrt], page 65, (method)
- [esqrt], page 65, (method)

**Internal Definitions**

- [define-e&], page 80, (macro)
- [define-e1], page 80, (macro)

- [`define-e2`], page 80, (macro)
- [`define-elementwise-reduction`], page 80, (macro)
- [`esquare`], page 92, (generic function)
- [`esquare`], page 92, (method)
- [`esquare`], page 92, (method)
- [`mapping-array`], page 81, (macro)

### 2.1.7 `num-utils/extended-real.lisp`

**Dependency**
[`elementwise.lisp`], page 5, (file)

**Parent** [`num-utils`], page 1, (system)

**Location** `extended-real.lisp`

**Exported Definitions**

- [`<`], page 44, (function)
- [`<=`], page 44, (function)
- [`=`], page 44, (function)
- [`>`], page 44, (function)
- [`>=`], page 45, (function)
- [`extended-real`], page 79, (type)
- [`infinite?`], page 50, (function)
- [`lambda-template`], page 43, (macro)
- [`with-template`], page 44, (macro)

**Internal Definitions**

- [`define-comparison`], page 80, (macro)
- [`extend-pairwise-comparison`], page 85, (function)
- [`infinite`], page 98, (type)

### 2.1.8 `num-utils/interval.lisp`

**Dependency**
[`extended-real.lisp`], page 8, (file)

**Parent** [`num-utils`], page 1, (system)

**Location** `interval.lisp`

**Exported Definitions**

- [`&interval`], page 41, (macro)
- [`extend-interval`], page 65, (generic function)
- [`extend-interval`], page 65, (method)
- [`extend-interval`], page 65, (method)
- [`extend-interval`], page 65, (method)
- [`extend-interval`], page 65, (method)
- [`extend-interval`], page 65, (method)
- [`extend-interval`], page 65, (method)
- [`extendf-interval`], page 42, (macro)
- [`finite-interval`], page 77, (class)

- [`grid-in`], page 49, (function)
- [`in-interval?`], page 49, (function)
- [`interval`], page 50, (function)
- [`interval`], page 78, (class)
- [`interval-hull`], page 50, (function)
- [`interval-length`], page 50, (function)
- [`interval-midpoint`], page 50, (function)
- [`left`], page 65, (generic function)
- [`left`], page 66, (method)
- [`left`], page 66, (method)
- [`minusinf-interval`], page 78, (class)
- [`num=`], page 67, (method)
- [`num=`], page 67, (method)
- [`open-left?`], page 67, (generic function)
- [`open-left?`], page 67, (method)
- [`open-left?`], page 67, (method)
- [`open-right?`], page 67, (generic function)
- [`open-right?`], page 67, (method)
- [`open-right?`], page 67, (method)
- [`plusinf-interval`], page 78, (class)
- [`plusminus-interval`], page 52, (function)
- [`real-line`], page 78, (class)
- [`relative`], page 52, (function)
- [`relative`], page 73, (structure)
- [`right`], page 68, (generic function)
- [`right`], page 68, (method)
- [`right`], page 68, (method)
- [`shift-interval`], page 68, (generic function)
- [`shift-interval`], page 68, (method)
- [`shrink-interval`], page 54, (function)
- [`spacer`], page 54, (function)
- [`spacer`], page 74, (structure)
- [`split-interval`], page 54, (function)
- [`subintervals-in`], page 54, (function)

**Internal Definitions**

- [`copy-relative`], page 83, (function)
- [`copy-spacer`], page 84, (function)
- [`interval/finite-left`], page 96, (class)
- [`interval/finite-right`], page 97, (class)
- [`interval/infinite-left`], page 97, (class)
- [`interval/infinite-right`], page 98, (class)
- [`print-left-endpoint`], page 93, (generic function)
- [`print-left-endpoint`], page 93, (method)

- [print-left-endpoint], page 93, (method)
- [print-right-endpoint], page 93, (generic function)
- [print-right-endpoint], page 93, (method)
- [print-right-endpoint], page 93, (method)
- [relative-fraction], page 88, (function)
- [relative-p], page 88, (function)
- [spacer-p], page 90, (function)
- [spacer-weight], page 90, (function)

### 2.1.9 num-utils/print-matrix.lisp

**Dependency**
[interval.lisp], page 8, (file)

**Parent** [num-utils], page 1, (system)

**Location** print-matrix.lisp

**Exported Definitions**
- [*print-matrix-precision*], page 41, (special variable)
- [print-length-truncate], page 52, (function)
- [print-matrix], page 52, (function)

**Internal Definitions**
[print-matrix-formatter], page 88, (function)

### 2.1.10 num-utils/matrix.lisp

**Dependency**
[print-matrix.lisp], page 10, (file)

**Parent** [num-utils], page 1, (system)

**Location** matrix.lisp

**Exported Definitions**
- [diagonal-matrix], page 47, (function)
- [diagonal-matrix], page 71, (structure)
- [diagonal-matrix-elements], page 47, (function)
- [(setf diagonal-matrix-elements)], page 47, (function)
- [diagonal-vector], page 58, (generic function)
- [diagonal-vector], page 58, (method)
- [diagonal-vector], page 58, (method)
- [(setf diagonal-vector)], page 58, (generic function)
- [e1-], page 58, (method)
- [e1-], page 58, (method)
- [e1-], page 58, (method)
- [e1-], page 58, (method)
- [e1/], page 58, (method)
- [e1/], page 58, (method)
- [e1/], page 59, (method)
- [e1/], page 59, (method)

- [e1log], page 59, (method)
- [e1log], page 59, (method)
- [e1log], page 59, (method)
- [e1log], page 59, (method)
- [e2*], page 59, (method)
- [e2*], page 59, (method)
- [e2*], page 59, (method)
- [e2*], page 59, (method)
- [e2*], page 59, (method)
- [e2*], page 60, (method)
- [e2*], page 60, (method)
- [e2*], page 60, (method)
- [e2*], page 60, (method)
- [e2*], page 60, (method)
- [e2*], page 60, (method)
- [e2*], page 60, (method)
- [e2*], page 60, (method)
- [e2*], page 60, (method)
- [e2+], page 60, (method)
- [e2+], page 60, (method)
- [e2+], page 60, (method)
- [e2+], page 61, (method)
- [e2+], page 61, (method)
- [e2+], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [eexp], page 63, (method)
- [eexp], page 63, (method)
- [eexp], page 63, (method)
- [eexp], page 63, (method)
- [esqrt], page 64, (method)

- [make-lower-triangular-matrix], page 86, (function)
- [make-upper-triangular-matrix], page 87, (function)
- [make-wrapped-matrix], page 87, (function)
- [upper-triangular-matrix-elements], page 91, (function)
- [upper-triangular-matrix-p], page 91, (function)
- [valid-sparse-type?], page 91, (function)
- [wrapped-matrix-p], page 92, (function)
- [zero-like], page 92, (function)

### 2.1.11 num-utils/matrix-shorthand.lisp

**Dependency**
[matrix.lisp], page 10, (file)

**Parent** [num-utils], page 1, (system)

**Location** `matrix-shorthand.lisp`

**Exported Definitions**
- [diagonal-mx], page 47, (function)
- [hermitian-mx], page 42, (macro)
- [lower-triangular-mx], page 43, (macro)
- [mx], page 43, (macro)
- [upper-triangular-mx], page 43, (macro)
- [vec], page 55, (function)

**Internal Definitions**
[pad-left-expansion], page 88, (function)

### 2.1.12 num-utils/statistics.lisp

**Dependency**
[matrix-shorthand.lisp], page 13, (file)

**Parent** [num-utils], page 1, (system)

**Location** `statistics.lisp`

**Exported Definitions**
- [*central-sample-moments-default-degree*], page 41, (special variable)
- [add], page 56, (generic function)
- [add], page 56, (method)
- [add], page 56, (method)
- [add], page 56, (method)
- [add], page 56, (method)
- [as-alist], page 56, (method)
- [central-m2], page 57, (generic function)
- [central-m2], page 57, (method)
- [central-m2], page 57, (method)
- [central-m3], page 57, (generic function)
- [central-m3], page 57, (method)
- [central-m3], page 57, (method)

- [sorted-reals], page 73, (structure)
- [sorted-reals-elements], page 54, (function)
- [sparse-counter], page 74, (structure)
- [sparse-counter-count], page 54, (function)
- [sparse-counter-table], page 54, (function)
- [tabulate], page 55, (function)
- [tally], page 69, (generic function)
- [tally], page 69, (method)
- [tally], page 69, (method)
- [variance], page 69, (generic function)
- [variance], page 69, (method)
- [variance], page 69, (method)
- [weighted-quantiles], page 55, (function)

**Internal Definitions**

- [&sorted-reals], page 79, (macro)
- [&sorted-reals-r/o], page 80, (macro)
- [central-sample-moments-m], page 82, (function)
- [(setf central-sample-moments-m)], page 82, (function)
- [central-sample-moments-p], page 82, (function)
- [central-sample-moments-s2], page 82, (function)
- [(setf central-sample-moments-s2)], page 82, (function)
- [central-sample-moments-s3], page 82, (function)
- [(setf central-sample-moments-s3)], page 82, (function)
- [central-sample-moments-s4], page 82, (function)
- [(setf central-sample-moments-s4)], page 82, (function)
- [central-sample-moments-w], page 83, (function)
- [(setf central-sample-moments-w)], page 83, (function)
- [copy-central-sample-moments], page 83, (function)
- [copy-sorted-reals], page 84, (function)
- [copy-sparse-counter], page 84, (function)
- [copy-tally-mixin], page 84, (function)
- [define-central-sample-moment], page 80, (macro)
- [make-central-sample-moments], page 86, (function)
- [make-sorted-reals], page 86, (function)
- [make-sparse-counter%], page 86, (function)
- [make-tally-mixin], page 86, (function)
- [pool2], page 92, (generic function)
- [pool2], page 92, (method)
- [sort-reals], page 89, (function)
- [sorted-reals-ordered-elements], page 89, (function)
- [(setf sorted-reals-ordered-elements)], page 89, (function)
- [sorted-reals-p], page 90, (function)
- [sorted-reals-unordered-elements], page 90, (function)

- [`(setf sorted-reals-unordered-elements)`], page 90, (function)
- [`sparse-counter-p`], page 90, (function)
- [`tally-mixin`], page 96, (structure)
- [`tally-mixin-p`], page 90, (function)
- [`tally-mixin-w`], page 90, (function)
- [`(setf tally-mixin-w)`], page 90, (function)
- [`weighted-empirical-quantile`], page 91, (function)
- [`weighted-quantile-p-table`], page 92, (function)

### 2.1.13 `num-utils/chebyshev.lisp`

**Dependency**
[`statistics.lisp`], page 13, (file)

**Parent** [`num-utils`], page 1, (system)

**Location** `chebyshev.lisp`

**Exported Definitions**
- [`chebyshev-approximate`], page 46, (function)
- [`chebyshev-regression`], page 46, (function)
- [`chebyshev-root`], page 46, (function)
- [`chebyshev-roots`], page 46, (function)
- [`evaluate-chebyshev`], page 49, (function)

**Internal Definitions**
- [`ab-to-cd-intercept-slope`], page 82, (function)
- [`ab-to-cinf`], page 82, (function)
- [`chebyshev-approximate-implementation`], page 92, (generic function)
- [`chebyshev-approximate-implementation`], page 92, (method)
- [`chebyshev-approximate-implementation`], page 92, (method)
- [`chebyshev-recursion`], page 83, (function)
- [`cinf-to-ab`], page 83, (function)

### 2.1.14 `num-utils/polynomial.lisp`

**Dependency**
[`chebyshev.lisp`], page 16, (file)

**Parent** [`num-utils`], page 1, (system)

**Location** `polynomial.lisp`

**Exported Definitions**
[`evaluate-polynomial`], page 49, (function)

### 2.1.15 `num-utils/rootfinding.lisp`

**Dependency**
[`polynomial.lisp`], page 16, (file)

**Parent** [`num-utils`], page 1, (system)

**Location** `rootfinding.lisp`

**Exported Definitions**

- [*rootfinding-delta-relative*], page 41, (special variable)
- [*rootfinding-epsilon*], page 41, (special variable)
- [root-bisection], page 53, (function)

**Internal Definitions**

- [narrow-bracket?], page 88, (function)
- [near-root?], page 88, (function)
- [opposite-sign?], page 88, (function)
- [rootfinding-delta], page 89, (function)
- [univariate-rootfinder-loop%], page 81, (macro)

### 2.1.16 num-utils/quadrature.lisp

**Dependency**
[rootfinding.lisp], page 16, (file)

**Parent** [num-utils], page 1, (system)

**Location** `quadrature.lisp`

**Exported Definitions**
[romberg-quadrature], page 53, (function)

**Internal Definitions**

- [copy-iterative-quadrature], page 83, (function)
- [copy-midpoint-quadrature], page 83, (function)
- [copy-richardson-extrapolation], page 83, (function)
- [copy-trapezoidal-quadrature], page 84, (function)
- [iterative-quadrature], page 94, (structure)
- [iterative-quadrature-a], page 85, (function)
- [(setf iterative-quadrature-a)], page 85, (function)
- [iterative-quadrature-b], page 85, (function)
- [(setf iterative-quadrature-b)], page 85, (function)
- [iterative-quadrature-f], page 85, (function)
- [(setf iterative-quadrature-f)], page 85, (function)
- [iterative-quadrature-h], page 85, (function)
- [(setf iterative-quadrature-h)], page 85, (function)
- [iterative-quadrature-n], page 85, (function)
- [(setf iterative-quadrature-n)], page 85, (function)
- [iterative-quadrature-p], page 85, (function)
- [iterative-quadrature-sum], page 85, (function)
- [(setf iterative-quadrature-sum)], page 85, (function)
- [make-iterative-quadrature], page 86, (function)
- [midpoint-quadrature], page 87, (function)
- [midpoint-quadrature], page 95, (structure)
- [midpoint-quadrature%], page 87, (function)
- [midpoint-quadrature-a], page 87, (function)
- [(setf midpoint-quadrature-a)], page 87, (function)

- [(setf trapezoidal-quadrature-n)], page 91, (function)
- [trapezoidal-quadrature-p], page 91, (function)
- [trapezoidal-quadrature-sum], page 91, (function)
- [(setf trapezoidal-quadrature-sum)], page 91, (function)

### 2.1.17 num-utils/test-utilities.lisp

**Dependency**
[quadrature.lisp], page 17, (file)

**Parent** [num-utils], page 1, (system)

**Location** test-utilities.lisp

**Exported Definitions**

- [compare-fns], page 46, (function)
- [compare-vectors], page 46, (function)
- [max-error], page 51, (function)
- [(setf max-error)], page 51, (function)
- [mean-error], page 51, (function)
- [(setf mean-error)], page 51, (function)
- [min-error], page 51, (function)
- [(setf min-error)], page 51, (function)
- [rms], page 53, (function)
- [(setf rms)], page 53, (function)
- [test-count], page 55, (function)
- [(setf test-count)], page 55, (function)
- [test-fn], page 55, (function)
- [test-results], page 75, (structure)
- [variance0], page 55, (function)
- [(setf variance0)], page 55, (function)
- [variance1], page 55, (function)
- [(setf variance1)], page 55, (function)
- [worst-case], page 56, (function)
- [(setf worst-case)], page 56, (function)

**Internal Definitions**

- [copy-test-results], page 84, (function)
- [make-test-results], page 86, (function)
- [test-results-p], page 90, (function)

### 2.1.18 num-utils/common-package.lisp

**Dependency**
[test-utilities.lisp], page 19, (file)

**Parent** [num-utils], page 1, (system)

**Location** common-package.lisp

**Packages** [num-utils], page 39,

# 3 Packages

Packages are listed by definition order.

## 3.1 num-utils.polynomial

**Source** [packages.lisp], page 3, (file)

**Nickname** poly

**Use List**

- [num-utils.utilities], page 23,
- alexandria
- common-lisp

**Exported Definitions**
[evaluate-polynomial], page 49, (function)

## 3.2 num-utils.interval

**Source** [packages.lisp], page 3, (file)

**Use List**

- let-plus
- [num-utils.utilities], page 23,
- [num-utils.num=], page 26,
- anaphora
- alexandria
- common-lisp

**Used By List**

- [num-utils.rootfinding], page 32,
- [num-utils.quadrature], page 28,
- [num-utils.chebyshev], page 37,

**Exported Definitions**

- [&interval], page 41, (macro)
- [extend-interval], page 65, (generic function)
- [extend-interval], page 65, (method)
- [extend-interval], page 65, (method)
- [extend-interval], page 65, (method)
- [extend-interval], page 65, (method)
- [extend-interval], page 65, (method)
- [extend-interval], page 65, (method)
- [extendf-interval], page 42, (macro)
- [finite-interval], page 77, (class)
- [grid-in], page 49, (function)
- [in-interval?], page 49, (function)
- [interval], page 50, (function)
- [interval], page 78, (class)

- [`interval-hull`], page 50, (function)
- [`interval-length`], page 50, (function)
- [`interval-midpoint`], page 50, (function)
- [`left`], page 65, (generic function)
- [`left`], page 66, (method)
- [`left`], page 66, (method)
- [`minusinf-interval`], page 78, (class)
- [`open-left?`], page 67, (generic function)
- [`open-left?`], page 67, (method)
- [`open-left?`], page 67, (method)
- [`open-right?`], page 67, (generic function)
- [`open-right?`], page 67, (method)
- [`open-right?`], page 67, (method)
- [`plusinf-interval`], page 78, (class)
- [`plusminus-interval`], page 52, (function)
- [`real-line`], page 78, (class)
- [`relative`], page 52, (function)
- [`relative`], page 73, (structure)
- [`right`], page 68, (generic function)
- [`right`], page 68, (method)
- [`right`], page 68, (method)
- [`shift-interval`], page 68, (generic function)
- [`shift-interval`], page 68, (method)
- [`shrink-interval`], page 54, (function)
- [`spacer`], page 54, (function)
- [`spacer`], page 74, (structure)
- [`split-interval`], page 54, (function)
- [`subintervals-in`], page 54, (function)

**Internal Definitions**

- [`copy-relative`], page 83, (function)
- [`copy-spacer`], page 84, (function)
- [`interval/finite-left`], page 96, (class)
- [`interval/finite-right`], page 97, (class)
- [`interval/infinite-left`], page 97, (class)
- [`interval/infinite-right`], page 98, (class)
- [`print-left-endpoint`], page 93, (generic function)
- [`print-left-endpoint`], page 93, (method)
- [`print-left-endpoint`], page 93, (method)
- [`print-right-endpoint`], page 93, (generic function)
- [`print-right-endpoint`], page 93, (method)
- [`print-right-endpoint`], page 93, (method)
- [`relative-fraction`], page 88, (function)
- [`relative-p`], page 88, (function)
- [`spacer-p`], page 90, (function)
- [`spacer-weight`], page 90, (function)

## 3.3 num-utils.print-matrix

**Source** [packages.lisp], page 3, (file)

**Use List**

- let-plus
- anaphora
- alexandria
- common-lisp

**Used By List**
[num-utils.matrix], page 37,

**Exported Definitions**

- [*print-matrix-precision*], page 41, (special variable)
- [print-length-truncate], page 52, (function)
- [print-matrix], page 52, (function)

**Internal Definitions**
[print-matrix-formatter], page 88, (function)

## 3.4 num-utils.utilities

**Source** [packages.lisp], page 3, (file)

**Use List**

- let-plus
- anaphora
- alexandria
- common-lisp

**Used By List**

- [num-utils.statistics], page 30,
- [num-utils.rootfinding], page 32,
- [num-utils.quadrature], page 28,
- [num-utils.matrix-shorthand], page 25,
- [num-utils.matrix], page 37,
- [num-utils.elementwise], page 33,
- [num-utils.polynomial], page 21,
- [num-utils.chebyshev], page 37,
- [num-utils.interval], page 21,
- [num-utils.arithmetic], page 24,

**Exported Definitions**

- [as-alist], page 56, (generic function)
- [as-alist], page 56, (method)
- [as-double-float], page 45, (function)
- [as-plist], page 56, (generic function)
- [as-plist], page 57, (method)
- [as-simple-fixnum-vector], page 45, (function)
- [bic], page 45, (function)

- [binary-search], page 45, (function)
- [check-types], page 41, (macro)
- [curry*], page 42, (macro)
- [define-with-multiple-bindings], page 42, (macro)
- [expanding], page 42, (macro)
- [fixnum?], page 49, (function)
- [generate-sequence], page 49, (function)
- [gethash*], page 42, (macro)
- [make-vector], page 44, (compiler macro)
- [make-vector], page 51, (function)
- [sequencep], page 54, (function)
- [simple-double-float-vector], page 79, (type)
- [simple-fixnum-vector], page 79, (type)
- [simple-single-float-vector], page 79, (type)
- [splice-awhen], page 43, (macro)
- [splice-when], page 43, (macro)
- [unlessf], page 43, (macro)
- [with-double-floats], page 44, (macro)
- [within?], page 56, (function)

## 3.5 num-utils.arithmetic

**Source** [packages.lisp], page 3, (file)

**Use List**

- let-plus
- [num-utils.utilities], page 23,
- anaphora
- alexandria-2
- common-lisp

**Used By List**

- [num-utils.statistics], page 30,
- [num-utils.quadrature], page 28,
- [num-utils.elementwise], page 33,

**Exported Definitions**

- [1c], page 44, (function)
- [abs-diff], page 45, (function)
- [absolute-square], page 45, (function)
- [as-integer], page 45, (function)
- [ceiling*], page 46, (function)
- [cumulative-product], page 47, (function)
- [cumulative-sum], page 47, (function)
- [divides?], page 47, (function)
- [floor*], page 49, (function)

- [ivec], page 50, (function)
- [l2norm], page 50, (function)
- [l2norm-square], page 65, (generic function)
- [l2norm-square], page 65, (method)
- [log10], page 50, (function)
- [log2], page 51, (function)
- [multf], page 43, (macro)
- [normalize-probabilities], page 51, (function)
- [numseq], page 52, (function)
- [product], page 67, (generic function)
- [product], page 67, (method)
- [product], page 67, (method)
- [round*], page 53, (function)
- [same-sign-p], page 53, (function)
- [sequence-maximum], page 53, (function)
- [sequence-minimum], page 53, (function)
- [square], page 54, (function)
- [sum], page 69, (generic function)
- [sum], page 69, (method)
- [sum], page 69, (method)
- [truncate*], page 55, (function)

**Internal Definitions**

- [define-rounding-with-offset], page 81, (macro)
- [similar-element-type], page 89, (function)
- [similar-sequence-type], page 89, (function)

## 3.6 `num-utils.matrix-shorthand`

**Source** [packages.lisp], page 3, (file)

**Nickname** `nu.mx`

**Use List**

- `let-plus`
- [num-utils.utilities], page 23,
- [num-utils.matrix], page 37,
- `anaphora`
- `alexandria`
- `common-lisp`

**Exported Definitions**

- [diagonal-mx], page 47, (function)
- [hermitian-mx], page 42, (macro)
- [lower-triangular-mx], page 43, (macro)
- [mx], page 43, (macro)
- [upper-triangular-mx], page 43, (macro)

- [vec], page 55, (function)

**Internal Definitions**
[pad-left-expansion], page 88, (function)

## 3.7 num-utils.num=

**Source** [packages.lisp], page 3, (file)

**Use List**

- let-plus
- anaphora
- alexandria
- common-lisp

**Used By List**

- [num-utils.statistics], page 30,
- [num-utils.matrix], page 37,
- [num-utils.interval], page 21,

**Exported Definitions**

- [*num=-tolerance*], page 41, (special variable)
- [define-num=-with-accessors], page 42, (macro)
- [define-structure-num=], page 42, (macro)
- [num-delta], page 51, (function)
- [num=], page 66, (generic function)
- [num=], page 66, (method)
- [num=], page 66, (method)
- [num=], page 66, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=], page 67, (method)
- [num=-function], page 52, (function)

## 3.8 num-utils.test-utilities

**Source** [packages.lisp], page 3, (file)

**Use List** common-lisp

**Exported Definitions**

- [compare-fns], page 46, (function)
- [compare-vectors], page 46, (function)
- [max-error], page 51, (function)
- [(setf max-error)], page 51, (function)
- [mean-error], page 51, (function)

- [`(setf mean-error)`], page 51, (function)
- [`min-error`], page 51, (function)
- [`(setf min-error)`], page 51, (function)
- [`rms`], page 53, (function)
- [`(setf rms)`], page 53, (function)
- [`test-count`], page 55, (function)
- [`(setf test-count)`], page 55, (function)
- [`test-fn`], page 55, (function)
- [`test-results`], page 75, (structure)
- [`variance0`], page 55, (function)
- [`(setf variance0)`], page 55, (function)
- [`variance1`], page 55, (function)
- [`(setf variance1)`], page 55, (function)
- [`worst-case`], page 56, (function)
- [`(setf worst-case)`], page 56, (function)

**Internal Definitions**

- [`copy-test-results`], page 84, (function)
- [`make-test-results`], page 86, (function)
- [`test-results-p`], page 90, (function)

## 3.9 `num-utils.extended-real`

**Source** [`packages.lisp`], page 3, (file)

**Nickname** `xreal`

**Use List**

- `alexandria`
- `common-lisp`

**Exported Definitions**

- [`<`], page 44, (function)
- [`<=`], page 44, (function)
- [`=`], page 44, (function)
- [`>`], page 44, (function)
- [`>=`], page 45, (function)
- [`extended-real`], page 79, (type)
- [`infinite?`], page 50, (function)
- [`lambda-template`], page 43, (macro)
- [`with-template`], page 44, (macro)

**Internal Definitions**

- [`define-comparison`], page 80, (macro)
- [`extend-pairwise-comparison`], page 85, (function)
- [`infinite`], page 98, (type)

## 3.10 num-utils.quadrature

**Source** [packages.lisp], page 3, (file)

**Use List**

- let-plus
- [num-utils.utilities], page 23,
- [num-utils.interval], page 21,
- [num-utils.arithmetic], page 24,
- anaphora
- alexandria
- common-lisp

**Exported Definitions**
[romberg-quadrature], page 53, (function)

**Internal Definitions**

- [copy-iterative-quadrature], page 83, (function)
- [copy-midpoint-quadrature], page 83, (function)
- [copy-richardson-extrapolation], page 83, (function)
- [copy-trapezoidal-quadrature], page 84, (function)
- [iterative-quadrature], page 94, (structure)
- [iterative-quadrature-a], page 85, (function)
- [(setf iterative-quadrature-a)], page 85, (function)
- [iterative-quadrature-b], page 85, (function)
- [(setf iterative-quadrature-b)], page 85, (function)
- [iterative-quadrature-f], page 85, (function)
- [(setf iterative-quadrature-f)], page 85, (function)
- [iterative-quadrature-h], page 85, (function)
- [(setf iterative-quadrature-h)], page 85, (function)
- [iterative-quadrature-n], page 85, (function)
- [(setf iterative-quadrature-n)], page 85, (function)
- [iterative-quadrature-p], page 85, (function)
- [iterative-quadrature-sum], page 85, (function)
- [(setf iterative-quadrature-sum)], page 85, (function)
- [make-iterative-quadrature], page 86, (function)
- [midpoint-quadrature], page 87, (function)
- [midpoint-quadrature], page 95, (structure)
- [midpoint-quadrature%], page 87, (function)
- [midpoint-quadrature-a], page 87, (function)
- [(setf midpoint-quadrature-a)], page 87, (function)
- [midpoint-quadrature-b], page 87, (function)
- [(setf midpoint-quadrature-b)], page 87, (function)
- [midpoint-quadrature-f], page 87, (function)
- [(setf midpoint-quadrature-f)], page 87, (function)
- [midpoint-quadrature-h], page 87, (function)

- [`(setf midpoint-quadrature-h)`], page 87, (function)
- [`midpoint-quadrature-n`], page 87, (function)
- [`(setf midpoint-quadrature-n)`], page 87, (function)
- [`midpoint-quadrature-p`], page 87, (function)
- [`midpoint-quadrature-sum`], page 87, (function)
- [`(setf midpoint-quadrature-sum)`], page 87, (function)
- [`refine-quadrature`], page 93, (generic function)
- [`refine-quadrature`], page 93, (method)
- [`refine-quadrature`], page 93, (method)
- [`richardson-coefficient`], page 93, (generic function)
- [`richardson-coefficient`], page 93, (method)
- [`richardson-coefficient`], page 93, (method)
- [`richardson-extrapolation`], page 88, (function)
- [`richardson-extrapolation`], page 95, (structure)
- [`richardson-extrapolation-coefficient`], page 88, (function)
- [`(setf richardson-extrapolation-coefficient)`], page 88, (function)
- [`richardson-extrapolation-diagonal`], page 88, (function)
- [`(setf richardson-extrapolation-diagonal)`], page 88, (function)
- [`richardson-extrapolation-n`], page 89, (function)
- [`(setf richardson-extrapolation-n)`], page 89, (function)
- [`richardson-extrapolation-p`], page 89, (function)
- [`richardson-iteration`], page 89, (function)
- [`romberg-quadrature%`], page 89, (function)
- [`transformed-quadrature`], page 93, (generic function)
- [`transformed-quadrature`], page 94, (method)
- [`transformed-quadrature`], page 94, (method)
- [`trapezoidal-quadrature`], page 90, (function)
- [`trapezoidal-quadrature`], page 96, (structure)
- [`trapezoidal-quadrature%`], page 90, (function)
- [`trapezoidal-quadrature-a`], page 90, (function)
- [`(setf trapezoidal-quadrature-a)`], page 90, (function)
- [`trapezoidal-quadrature-b`], page 91, (function)
- [`(setf trapezoidal-quadrature-b)`], page 91, (function)
- [`trapezoidal-quadrature-f`], page 91, (function)
- [`(setf trapezoidal-quadrature-f)`], page 91, (function)
- [`trapezoidal-quadrature-h`], page 91, (function)
- [`(setf trapezoidal-quadrature-h)`], page 91, (function)
- [`trapezoidal-quadrature-n`], page 91, (function)
- [`(setf trapezoidal-quadrature-n)`], page 91, (function)
- [`trapezoidal-quadrature-p`], page 91, (function)
- [`trapezoidal-quadrature-sum`], page 91, (function)
- [`(setf trapezoidal-quadrature-sum)`], page 91, (function)

## 3.11 num-utils.statistics

**Source** [packages.lisp], page 3, (file)

**Nickname** nu.stats

**Use List**

- let-plus
- [num-utils.utilities], page 23,
- [num-utils.num=], page 26,
- [num-utils.arithmetic], page 24,
- alexandria
- anaphora
- common-lisp

**Exported Definitions**

- [*central-sample-moments-default-degree*], page 41, (special variable)
- [add], page 56, (generic function)
- [add], page 56, (method)
- [add], page 56, (method)
- [add], page 56, (method)
- [add], page 56, (method)
- [central-m2], page 57, (generic function)
- [central-m2], page 57, (method)
- [central-m2], page 57, (method)
- [central-m3], page 57, (generic function)
- [central-m3], page 57, (method)
- [central-m3], page 57, (method)
- [central-m4], page 57, (generic function)
- [central-m4], page 57, (method)
- [central-m4], page 57, (method)
- [central-sample-moments], page 57, (generic function)
- [central-sample-moments], page 57, (method)
- [central-sample-moments], page 57, (method)
- [central-sample-moments], page 57, (method)
- [central-sample-moments], page 70, (structure)
- [central-sample-moments-degree], page 46, (function)
- [cross-tabulate], page 46, (function)
- [empirical-quantile], page 48, (function)
- [empirical-quantile-probabilities], page 48, (function)
- [empty-accumulator], page 70, (condition)
- [ensure-sorted-reals], page 64, (generic function)
- [ensure-sorted-reals], page 64, (method)
- [ensure-sorted-reals], page 64, (method)
- [ensure-sorted-reals], page 64, (method)
- [ensure-sorted-vector], page 48, (function)

- [information-not-collected-in-accumulator], page 70, (condition)
- [kurtosis], page 65, (generic function)
- [kurtosis], page 65, (method)
- [kurtosis], page 65, (method)
- [make-sparse-counter], page 51, (function)
- [mean], page 66, (generic function)
- [mean], page 66, (method)
- [mean], page 66, (method)
- [median], page 66, (generic function)
- [median], page 66, (method)
- [median], page 66, (method)
- [not-enough-elements-in-accumulator], page 70, (condition)
- [pool], page 52, (function)
- [quantile], page 68, (generic function)
- [quantile], page 68, (method)
- [quantile], page 68, (method)
- [quantiles], page 68, (generic function)
- [quantiles], page 68, (method)
- [quantiles], page 68, (method)
- [sd], page 68, (generic function)
- [sd], page 68, (method)
- [skewness], page 69, (generic function)
- [skewness], page 69, (method)
- [skewness], page 69, (method)
- [sorted-reals], page 73, (structure)
- [sorted-reals-elements], page 54, (function)
- [sparse-counter], page 74, (structure)
- [sparse-counter-count], page 54, (function)
- [sparse-counter-table], page 54, (function)
- [tabulate], page 55, (function)
- [tally], page 69, (generic function)
- [tally], page 69, (method)
- [tally], page 69, (method)
- [variance], page 69, (generic function)
- [variance], page 69, (method)
- [variance], page 69, (method)
- [weighted-quantiles], page 55, (function)

**Internal Definitions**

- [&sorted-reals], page 79, (macro)
- [&sorted-reals-r/o], page 80, (macro)
- [central-sample-moments-m], page 82, (function)
- [(setf central-sample-moments-m)], page 82, (function)
- [central-sample-moments-p], page 82, (function)

- [central-sample-moments-s2], page 82, (function)
- [(setf central-sample-moments-s2)], page 82, (function)
- [central-sample-moments-s3], page 82, (function)
- [(setf central-sample-moments-s3)], page 82, (function)
- [central-sample-moments-s4], page 82, (function)
- [(setf central-sample-moments-s4)], page 82, (function)
- [central-sample-moments-w], page 83, (function)
- [(setf central-sample-moments-w)], page 83, (function)
- [copy-central-sample-moments], page 83, (function)
- [copy-sorted-reals], page 84, (function)
- [copy-sparse-counter], page 84, (function)
- [copy-tally-mixin], page 84, (function)
- [define-central-sample-moment], page 80, (macro)
- [make-central-sample-moments], page 86, (function)
- [make-sorted-reals], page 86, (function)
- [make-sparse-counter%], page 86, (function)
- [make-tally-mixin], page 86, (function)
- [pool2], page 92, (generic function)
- [pool2], page 92, (method)
- [sort-reals], page 89, (function)
- [sorted-reals-ordered-elements], page 89, (function)
- [(setf sorted-reals-ordered-elements)], page 89, (function)
- [sorted-reals-p], page 90, (function)
- [sorted-reals-unordered-elements], page 90, (function)
- [(setf sorted-reals-unordered-elements)], page 90, (function)
- [sparse-counter-p], page 90, (function)
- [tally-mixin], page 96, (structure)
- [tally-mixin-p], page 90, (function)
- [tally-mixin-w], page 90, (function)
- [(setf tally-mixin-w)], page 90, (function)
- [weighted-empirical-quantile], page 91, (function)
- [weighted-quantile-p-table], page 92, (function)

## 3.12 num-utils.rootfinding

**Source** [packages.lisp], page 3, (file)

**Use List**

- let-plus
- [num-utils.utilities], page 23,
- [num-utils.interval], page 21,
- alexandria
- common-lisp

**Exported Definitions**

- [*rootfinding-delta-relative*], page 41, (special variable)

- [*rootfinding-epsilon*], page 41, (special variable)
- [root-bisection], page 53, (function)

**Internal Definitions**

- [narrow-bracket?], page 88, (function)
- [near-root?], page 88, (function)
- [opposite-sign?], page 88, (function)
- [rootfinding-delta], page 89, (function)
- [univariate-rootfinder-loop%], page 81, (macro)

## 3.13 num-utils.elementwise

**Source** [packages.lisp], page 3, (file)

**Nickname** elmt

**Use List**

- let-plus
- [num-utils.utilities], page 23,
- [num-utils.arithmetic], page 24,
- alexandria
- common-lisp

**Used By List**

[num-utils.matrix], page 37,

**Exported Definitions**

- [e*], page 47, (function)
- [e+], page 47, (function)
- [e-], page 47, (function)
- [e/], page 48, (function)
- [e1-], page 58, (generic function)
- [e1-], page 58, (method)
- [e1-], page 58, (method)
- [e1-], page 58, (method)
- [e1-], page 58, (method)
- [e1-], page 58, (method)
- [e1-], page 58, (method)
- [e1/], page 58, (generic function)
- [e1/], page 58, (method)
- [e1/], page 58, (method)
- [e1/], page 59, (method)
- [e1/], page 59, (method)
- [e1/], page 59, (method)
- [e1/], page 59, (method)
- [e1log], page 59, (generic function)
- [e1log], page 59, (method)
- [e1log], page 59, (method)

- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2-], page 61, (method)
- [e2/], page 62, (generic function)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2/], page 62, (method)
- [e2log], page 62, (generic function)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [e2log], page 62, (method)
- [eceiling], page 63, (generic function)
- [eceiling], page 63, (method)
- [eceiling], page 63, (method)
- [econjugate], page 63, (generic function)
- [econjugate], page 63, (method)
- [econjugate], page 63, (method)
- [eexp], page 63, (generic function)
- [eexp], page 63, (method)

- [eexp], page 63, (method)
- [eexp], page 63, (method)
- [eexp], page 63, (method)
- [eexp], page 63, (method)
- [eexp], page 63, (method)
- [eexpt], page 63, (generic function)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [eexpt], page 64, (method)
- [efloor], page 64, (generic function)
- [efloor], page 64, (method)
- [efloor], page 64, (method)
- [elementwise-float-contagion], page 48, (function)
- [elog], page 48, (function)
- [emax], page 48, (function)
- [emin], page 48, (function)
- [ereduce], page 64, (generic function)
- [ereduce], page 64, (method)
- [ereduce], page 64, (method)
- [ereduce], page 64, (method)
- [esqrt], page 64, (generic function)
- [esqrt], page 64, (method)
- [esqrt], page 65, (method)
- [esqrt], page 65, (method)
- [esqrt], page 65, (method)
- [esqrt], page 65, (method)
- [esqrt], page 65, (method)

**Internal Definitions**

- [define-e&], page 80, (macro)
- [define-e1], page 80, (macro)
- [define-e2], page 80, (macro)
- [define-elementwise-reduction], page 80, (macro)
- [esquare], page 92, (generic function)
- [esquare], page 92, (method)
- [esquare], page 92, (method)
- [mapping-array], page 81, (macro)

## 3.14 `num-utils.chebyshev`

**Source** [`packages.lisp`], page 3, (file)

**Use List**

- `let-plus`
- [`num-utils.utilities`], page 23,
- [`num-utils.interval`], page 21,
- `anaphora`
- `alexandria`
- `common-lisp`

**Exported Definitions**

- [`chebyshev-approximate`], page 46, (function)
- [`chebyshev-regression`], page 46, (function)
- [`chebyshev-root`], page 46, (function)
- [`chebyshev-roots`], page 46, (function)
- [`evaluate-chebyshev`], page 49, (function)

**Internal Definitions**

- [`ab-to-cd-intercept-slope`], page 82, (function)
- [`ab-to-cinf`], page 82, (function)
- [`chebyshev-approximate-implementation`], page 92, (generic function)
- [`chebyshev-approximate-implementation`], page 92, (method)
- [`chebyshev-approximate-implementation`], page 92, (method)
- [`chebyshev-recursion`], page 83, (function)
- [`cinf-to-ab`], page 83, (function)

## 3.15 `num-utils.matrix`

**Source** [`packages.lisp`], page 3, (file)

**Use List**

- `let-plus`
- `select`
- [`num-utils.utilities`], page 23,
- [`num-utils.print-matrix`], page 23,
- [`num-utils.num=`], page 26,
- [`num-utils.elementwise`], page 33,
- `anaphora`
- `alexandria`
- `common-lisp`

**Used By List**

[`num-utils.matrix-shorthand`], page 25,

**Exported Definitions**

- [`diagonal-matrix`], page 47, (function)
- [`diagonal-matrix`], page 71, (structure)
- [`diagonal-matrix-elements`], page 47, (function)

- [make-lower-triangular-matrix], page 86, (function)
- [make-upper-triangular-matrix], page 87, (function)
- [make-wrapped-matrix], page 87, (function)
- [upper-triangular-matrix-elements], page 91, (function)
- [upper-triangular-matrix-p], page 91, (function)
- [valid-sparse-type?], page 91, (function)
- [wrapped-matrix-p], page 92, (function)
- [zero-like], page 92, (function)

## 3.16 num-utils

|  |  |
|---|---|
| **Source** | [common-package.lisp], page 19, (file) |
| **Nickname** | nu |
| **Use List** | common-lisp |

# 4 Definitions

Definitions are sorted by export status, category, package, and then by lexicographic order.

## 4.1 Exported definitions

### 4.1.1 Special variables

`*central-sample-moments-default-degree*` [Special Variable]

Default degree for (weighted) central sample moments.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`*num=-tolerance*` [Special Variable]

Default tolerance for NUM=.

**Package** [`num-utils.num=`], page 26,

**Source** [`num=.lisp`], page 4, (file)

`*print-matrix-precision*` [Special Variable]

Number of digits after the decimal point when printing numeric matrices.

**Package** [`num-utils.print-matrix`], page 23,

**Source** [`print-matrix.lisp`], page 10, (file)

`*rootfinding-delta-relative*` [Special Variable]

Default relative interval width for rootfinding.

**Package** [`num-utils.rootfinding`], page 32,

**Source** [`rootfinding.lisp`], page 16, (file)

`*rootfinding-epsilon*` [Special Variable]

Default maximum for the absolute value of the function, used for rootfinding.

**Package** [`num-utils.rootfinding`], page 32,

**Source** [`rootfinding.lisp`], page 16, (file)

### 4.1.2 Macros

`&interval` *LEFT RIGHT* [Macro]

LET+ expansion for interval endpoints. If given a list of two values, the second value is an indicator for whether the endpoint is open.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`check-types` (**&rest** *ARGUMENTS*) *TYPE* [Macro]

CHECK-TYPE for multiple places of the same type. Each argument is either a place, or a list of a place and a type-string.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file) `curry*` *FUNCTION* **&rest** *ARGUMENTS* [Macro]

Currying in all variables that are not *. Note that this is a macro, so * should not be quoted, and FUNCTION will be used as is, ie it can be a LAMBDA form.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`define-num=-with-accessors` *CLASS ACCESSORS* [Macro]

Define a method for NUM=, specialized to the given class, comparing values obtained with accessors.

**Package** [`num-utils.num=`], page 26,

**Source** [`num=.lisp`], page 4, (file)

`define-structure-num=` *STRUCTURE* **&rest** *SLOTS* [Macro]

Define a NUM= method for the given structure, comparing the given slots.

**Package** [`num-utils.num=`], page 26,

**Source** [`num=.lisp`], page 4, (file)

`define-with-multiple-bindings` *MACRO* **&key** *PLURAL DOCSTRING* [Macro]

Define a version of MACRO with multiple arguments, given as a list. Application of MACRO will be nested. The new name is the plural of the old one (generated using format by default).

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`expanding` **&body** *BODY* [Macro]

Expand BODY. Useful for generating code programmatically.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`extendf-interval` *PLACE OBJECT* [Macro]

Apply EXTEND-INTERVAL on PLACE using OBJECT.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`gethash*` *KEY HASH-TABLE* **&optional** *DATUM* **&rest** *ARGUMENTS* [Macro]

Like GETHASH, but checking that KEY is present and raising the given error if not.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`hermitian-mx` *ELEMENT-TYPE* **&body** *ROWS* [Macro]

Macro for creating a lower triangular matrix. ROWS should be a list of lists, elements are evaluated. Masked elements (above the diagonal) are ignored at the expansion, rows which don't have enough elements are padded with zeros.

**Package** [`num-utils.matrix-shorthand`], page 25,

**Source** [`matrix-shorthand.lisp`], page 13, (file) `lambda-template` (*PREFIX* **&rest** *VARIABLES*) **&body** *BODY* [Macro]

LAMBDA with WITH-TEMPLATE in its BODY.

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`lower-triangular-mx` *ELEMENT-TYPE* **&body** *ROWS* [Macro]

Macro for creating a lower triangular matrix. ROWS should be a list of lists, elements are evaluated. Masked elements (above the diagonal) are ignored at the expansion, rows which don't have enough elements are padded with zeros.

**Package** [`num-utils.matrix-shorthand`], page 25,

**Source** [`matrix-shorthand.lisp`], page 13, (file)

`multf` *PLACE COEFFICIENT* [Macro]

Multiply place by COEFFICIENT.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`mx` *ELEMENT-TYPE* **&body** *ROWS* [Macro]

Macro for creating a (dense) matrix (ie a rank 2 array). ROWS should be a list of lists (or atoms, which are treated as lists), elements are evaluated.

**Package** [`num-utils.matrix-shorthand`], page 25,

**Source** [`matrix-shorthand.lisp`], page 13, (file)

`splice-awhen` *TEST* **&body** *FORMS* [Macro]

Similar to splice-when, but binds IT to test.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`splice-when` *TEST* **&body** *FORMS* [Macro]

Similar to when, but wraps the result in list.

Example: ‘(,foo ,@(splice-when add-bar? bar))

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`unlessf` *PLACE VALUE-FORM* [Macro]

When PLACE is NIL, evaluate VALUE-FORM and save it there.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`upper-triangular-mx` *ELEMENT-TYPE* **&body** *ROWS* [Macro]

Macro for creating an upper triangular matrix. ROWS should be a list of lists, elements are evaluated. Masked elements (below the diagonal) are ignored at the expansion.

**Package** [`num-utils.matrix-shorthand`], page 25,

**Source** [`matrix-shorthand.lisp`], page 13, (file) `with-double-floats` *BINDINGS* **&body** *BODY* [Macro]

For each binding = (variable value), coerce VALUE to DOUBLE-FLOAT and bind it to VARIABLE for BODY. When VALUE is omitted, VARIABLE is used instead. When BINDING is an atom, it is used for both the value and the variable.

Example:
(with-double-floats (a
(b)
(c 1))
...)

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`with-template` (*PREFIX* **&rest** *VARIABLES*) **&body** *BODY* [Macro]

Define the function (PREFIX &rest VARIABLES) which can be used to match variables using :PLUSINF, :MINUSINF, REAL, or T.

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

### 4.1.3 Compiler macros

`make-vector` *ELEMENT-TYPE* **&rest** *INITIAL-CONTENTS* [Compiler Macro]

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

### 4.1.4 Functions

`1c` *NUMBER* [Function]

Return 1-number. The mnemonic is "1 complement", 1- is already a CL library function.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`<` *NUMBER* **&rest** *MORE-NUMBERS* [Function]

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`<=` *NUMBER* **&rest** *MORE-NUMBERS* [Function]

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`=` *NUMBER* **&rest** *MORE-NUMBERS* [Function]

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`>` *NUMBER* **&rest** *MORE-NUMBERS* [Function]

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`>=` *NUMBER* **&rest** *MORE-NUMBERS* [Function]

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`abs-diff` *A B* [Function]

Absolute difference of A and B.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`absolute-square` *NUMBER* [Function]

Number multiplied by its complex conjugate.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`as-double-float` *V* [Function]

Convert argument to DOUBLE-FLOAT.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`as-integer` *NUMBER* [Function]

If NUMBER represents an integer (as an integer, complex, or float, etc), return it as an integer, otherwise signal an error. Floats are converted with RATIONALIZE.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`as-simple-fixnum-vector` *SEQUENCE* **&optional** *COPY?* [Function]

Convert SEQUENCE to a SIMPLE-FIXNUM-VECTOR. When COPY?, make sure that the they don't share structure.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`bic` *A B* [Function]

Biconditional. Returns A <=> B.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`binary-search` *SORTED-REALS VALUE* [Function]

Return INDEX such that

(WITHIN? (AREF SORTED-REALS INDEX) VALUE (AREF SORTED-REALS (1+ INDEX)).

SORTED-REALS is assumed to be reals sorted in ascending order (not checked, if this does not hold the result may be nonsensical, though the algorithm will terminate).

If value is below (or above) the first (last) break, NIL (T) is returned.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file) `ceiling*` *NUMBER* **&optional** *DIVISOR OFFSET* [Function]

Find the lowest A=I*DIVISOR+OFFSET >= NUMBER, return (values A (- A NUMBER).

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`central-sample-moments-degree` *CENTRAL-SAMPLE-MOMENTS* [Function]

Return the degree of CENTRAL-SAMPLE-MOMENTS.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`chebyshev-approximate` *F INTERVAL N-POLYNOMIALS* **&key** *N-POINTS* [Function]

Return a closure approximating F on the given INTERVAL (may be infinite on either end) using the given number of Chebyshev polynomials.

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`chebyshev-regression` *F N-POLYNOMIALS* **&optional** *N-POINTS* [Function]

Chebyshev polynomial regression using the given number of polynomials and points (zeroes of the corresponding Chebyshev polynomial).

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`chebyshev-root` *M I* [Function]

Return the iTH root of the Mth Chebyshev polynomial as double-float.

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`chebyshev-roots` *M* [Function]

Return the roots of the Mth Chebyshev polynomial as a vector of double-floats.

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`compare-fns` *FN-1 FN-2* **&rest** *FN-PARAMS* [Function]

Compare the values returned by two functions

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`compare-vectors` *REFERENCE-VALUES COMPUTED-VALUES* [Function]

Compare two vectors containing the results of previous computations

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`cross-tabulate` *SEQUENCE1 SEQUENCE2* **&key** *TEST* [Function]

Cross-tabulate two sequences (using a SPARSE-COUNTER with the given TEST). TEST is used to compare conses.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`cumulative-product` *SEQUENCE* **&key** *RESULT-TYPE* [Function]

Cumulative product of sequence. Return a sequence of the same kind and length; last element is the total product. The latter is also returned as the second value.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`cumulative-sum` *SEQUENCE* **&key** *RESULT-TYPE* [Function]

Cumulative sum of sequence. Return a sequence of the same kind and length; last element is the total. The latter is returned as the second value.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`diagonal-matrix` *ELEMENTS* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`diagonal-matrix-elements` *INSTANCE* [Function]
`(setf diagonal-matrix-elements)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`diagonal-mx` *ELEMENT-TYPE* **&rest** *ELEMENTS* [Function]

Return a DIAGONAL-MATRIX with elements coerced to ELEMENT-TYPE.

**Package** [`num-utils.matrix-shorthand`], page 25,

**Source** [`matrix-shorthand.lisp`], page 13, (file)

`divides?` *NUMBER DIVISOR* [Function]

Test if DIVISOR divides NUMBER without remainder, and if so, return the quotient. Works generally, but makes most sense for rationals.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`e*` *ARGUMENT* **&rest** *MORE-ARGUMENTS* [Function]

Elementwise *.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`e+` *ARGUMENT* **&rest** *MORE-ARGUMENTS* [Function]

Elementwise +.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`e-` *ARGUMENT* **&rest** *MORE-ARGUMENTS* [Function]

Elementwise -.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`e/` *ARGUMENT* **&rest** *MORE-ARGUMENTS* [Function]
Elementwise /.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`elementwise-float-contagion` **&rest** *OBJECTS* [Function]
Return the resulting float type when objects (or their elements) are combined using arithmetic operations.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`elog` *A* **&optional** *BASE* [Function]
Elementwise logarithm.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`emax` *OBJECT* [Function]
Elementwise MAX.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`emin` *OBJECT* [Function]
Elementwise MIN.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`empirical-quantile` *SORTED-VECTOR Q* [Function]
Return the empirical quantile of a vector of real numbers, sorted in ascending order (not checked). Uses a 0.5 correction.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`empirical-quantile-probabilities` *N* [Function]
Probabilities that correspond to the empirical quantiles of a vector of length N. That is to say,

(== (quantiles sample (empirical-quantile-probabilities (length sample)))
sample)

for any vector SAMPLE.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`ensure-sorted-vector` *OBJECT* [Function]
Return the elements of OBJECT as a vector (or reals) sorted in ascending order.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`evaluate-chebyshev` *COEFFICIENTS X* [Function]
Return the sum of Chebyshev polynomials, weighted by COEFFICIENTS, at X.

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`evaluate-polynomial` *COEFFICIENTS X* [Function]
Return the sum of polynomials, weighted by COEFFICIENTS, at X. COFFICIENTS are ordered from the highest degree down to the constant term. X must be of the same type as COEFFICIENTS.

**Package** [`num-utils.polynomial`], page 21,

**Source** [`polynomial.lisp`], page 16, (file)

`fixnum?` *OBJECT* [Function]
Check of type of OBJECT is fixnum.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`floor*` *NUMBER* **&optional** *DIVISOR OFFSET* [Function]
Find the highest A=I*DIVISOR+OFFSET <= NUMBER, return (values A (- A NUMBER).

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`generate-sequence` *RESULT-TYPE SIZE FUNCTION* [Function]
Like MAKE-SEQUENCE, but using a function to fill the result.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`grid-in` *INTERVAL SIZE* **&optional** *SEQUENCE-TYPE* [Function]
Return an arithmetic sequence of the given size (length) between the endpoints of the interval. The endpoints of the sequence coincide with the respective endpoint of the interval iff it is closed. RESULT-TYPE determines the result type (eg list), if not given it is a simple-array (of rank 1), narrowing to the appropriate float type or fixnum if possible.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`hermitian-matrix` *ELEMENTS* [Function]
Create a lower-triangular-matrix.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`in-interval?` *INTERVAL NUMBER* [Function]
Test if NUMBER is in INTERVAL (which can be NIL, designating the empty set).

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`infinite?` *OBJECT* [Function]
Test if an object represents positive or negative infinity.

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`interval` *LEFT RIGHT* **&key** *OPEN-LEFT? OPEN-RIGHT?* [Function]
Create an INTERVAL.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`interval-hull` *OBJECT* [Function]
Return the smallest connected interval that contains (elements in) OBJECT.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`interval-length` *INTERVAL* [Function]
Difference between left and right.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`interval-midpoint` *INTERVAL* **&optional** *ALPHA* [Function]
Convex combination of left and right, with alpha (defaults to 0.5) weight on right.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`ivec` *END-OR-START* **&optional** *END BY STRICT-DIRECTION?* [Function]
Return a vector of fixnums.

(ivec end) => #(0 ... end-1) (or #(0 ... end+1) when end is negative).

(ivec start end) => #(start ... end-1) or to end+1 when end is negative.

When BY is given it determines the increment, adjusted to match the direction unless STRICT-DIRECTION, in which case an error is signalled.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`l2norm` *OBJECT* [Function]
$L_2$ norm of OBJECT.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`log10` *NUMBER* [Function]
Abbreviation for decimal logarithm.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file) `log2` *NUMBER* [Function]

Abbreviation for binary logarithm.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`lower-triangular-matrix` *ELEMENTS* [Function]

Create a lower-triangular-matrix.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`make-sparse-counter` **&key** *TEST* [Function]

Create a sparse counter. Elements are compared with TEST (should be accepted by HASH-TABLE).

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`make-vector` *ELEMENT-TYPE* **&rest** *INITIAL-CONTENTS* [Function]

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`max-error` *INSTANCE* [Function]
`(setf max-error)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`mean-error` *INSTANCE* [Function]
`(setf mean-error)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`min-error` *INSTANCE* [Function]
`(setf min-error)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`normalize-probabilities` *VECTOR* **&key** *ELEMENT-TYPE RESULT* [Function]

Verify that each element of VECTOR is nonnegative and return a vector multiplied so that they sum to 1. ELEMENT-TYPE can be used to specify the element-type of the result. When RESULT is given, the result is placed there. When RESULT is NIL, VECTOR is modified instead.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`num-delta` *A B* [Function]

|a-b|/max(1,|a|,|b|). Useful for comparing numbers.

**Package** [`num-utils.num=`], page 26,

**Source** [`num=.lisp`], page 4, (file) `num=-function` *TOLERANCE* [Function]

Curried version of num=, with given tolerance.

**Package** [`num-utils.num=`], page 26,

**Source** [`num=.lisp`], page 4, (file)

`numseq` *FROM TO* **&key** *LENGTH BY TYPE* [Function]

Return a sequence between FROM and TO, progressing by BY, of the given LENGTH. Only 3 of these a parameters should be given, the missing one (NIL) should be inferred automatically. The sign of BY is adjusted if necessary. If TYPE is LIST, the result is a list, otherwise it determines the element type of the resulting simple array. If TYPE is nil, it as autodetected from the arguments (as a FIXNUM, a RATIONAL, or some subtype of FLOAT). Note that the implementation may upgrade the element type.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`plusminus-interval` *CENTER HALF-WIDTH* **&key** *OPEN-LEFT? OPEN-RIGHT?* [Function]

A symmetric interval around CENTER.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`pool` **&rest** *ACCUMULATORS* [Function]

Pool ACCUMULATORS.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`print-length-truncate` *DIMENSION* [Function]

Return values (min dimension *print-length*) and whether the constraint is binding.

**Package** [`num-utils.print-matrix`], page 23,

**Source** [`print-matrix.lisp`], page 10, (file)

`print-matrix` *MATRIX STREAM* **&key** *FORMATTER MASKED-FN ALIGNED? PADDING INDENT* [Function]

Format and print the elements of MATRIX (a 2d array) to STREAM, using PADDING between columns.

MASKED-FN is called on row and column indices. If it returns nil, the corresponding element is formatted using FORMATTER and printed. Otherwise, it should return a string, which is printed as is. INDENT is printed before each row.

If ALIGNED?, columns will be right-aligned. At most *PRINT-LENGTH* rows and columns are printed, more is indicated with ellipses (...).

**Package** [`num-utils.print-matrix`], page 23,

**Source** [`print-matrix.lisp`], page 10, (file)

`relative` *FRACTION* [Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`rms` *INSTANCE* [Function]
`(setf rms)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`romberg-quadrature` *F INTERVAL* **&key** *EPSILON MIN-ITER MAX-ITER TRANSFORMATION* [Function]

Romberg quadrature of F on the interval. The iteration stops if the relative change is below EPSILON, but only after MIN-ITER refinements (to avoid spurious premature convergence). An error occurs when MAX-ITER iterations are reached without convergence.

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`root-bisection` *F BRACKET* **&key** *DELTA EPSILON* [Function]

Find the root of f bracketed between a and b using bisection.
The algorithm stops when either the root is bracketed in an interval of length TOLERANCE (relative to the initial |a-b|), or root is found such that abs(f(root)) <= epsilon.

Return five values: the root, the value of the function at the root, and a boolean which is true iff abs(f(root)) <= epsilon. If the third value is true, the fourth and fifth values are the endpoints of the bracketing interval, otherwise they are undefined.

**Package** [`num-utils.rootfinding`], page 32,

**Source** [`rootfinding.lisp`], page 16, (file)

`round*` *NUMBER* **&optional** *DIVISOR OFFSET* [Function]

Find A=I*DIVISOR+OFFSET that minimizes |A-NUMBER|, return (values A (- A NUMBER). When NUMBER is exactly in between two possible A's, the rounding rule of ROUND is used on NUMBER-OFFSET.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`same-sign-p` **&rest** *ARGUMENTS* [Function]

Test whether all arguments have the same sign (ie all are positive, negative, or zero).

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`sequence-maximum` *X* [Function]

Return the maximum value in the sequence

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`sequence-minimum` *X* [Function]

Return the minimum value in the sequence

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`sequencep` *X* [Function]
Return T if X is type SEQUENCE.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`shrink-interval` *INTERVAL LEFT* **&optional** *RIGHT CHECK-FLIP?* [Function]
Shrink interval by given magnitudes (which may be REAL or RELATIVE). When check-flip?, the result is checked for endpoints being in a different order than the original. Negative LEFT and RIGHT extend the interval.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`sorted-reals-elements` *SORTED-REALS* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`spacer` **&optional** *WEIGHT* [Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`sparse-counter-count` *SPARSE-COUNTER OBJECT* [Function]
Return the count for OBJECT.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`sparse-counter-table` *INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`split-interval` *INTERVAL DIVISIONS* [Function]
Return a vector of subintervals (same length as DIVISIONS), splitting the interval using the sequence DIVISIONS, which can be nonnegative real numbers (or RELATIVE specifications) and SPACERs which divide the leftover proportionally. If there are no spacers and the divisions don't fill up the interval, and error is signalled.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`square` *NUMBER* [Function]
Square of number.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`subintervals-in` *INTERVAL COUNT* **&optional** *MID-OPEN-RIGHT?* [Function]
Return INTERVAL evenly divided into COUNT subintervals as a vector. When MID-OPEN-RIGHT?, subintervals in the middle are open on the right and closed on the left, otherwise the opposite; openness of endpoints on the edge follows INTERVAL.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`tabulate` *SEQUENCE* **&key** *TEST* [Function]

Tabulate a sequence (using a SPARSE-COUNTER with the given TEST).

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`test-count` *INSTANCE* [Function]
`(setf test-count)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`test-fn` *EXPECTED-COLUMN FN* **&rest** *FN-PARAM-COLUMNS* [Function]

Test the differences between expected values and the given function

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`truncate*` *NUMBER* **&optional** *DIVISOR OFFSET* [Function]

Find A=I*DIVISOR+OFFSET that maximizes `|A|<=|NUMBER|` with the same sign, return (values A (- A NUMBER).

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`upper-triangular-matrix` *ELEMENTS* [Function]

Create a lower-triangular-matrix.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`variance0` *INSTANCE* [Function]
`(setf variance0)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`variance1` *INSTANCE* [Function]
`(setf variance1)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`vec` *ELEMENT-TYPE* **&rest** *ELEMENTS* [Function]

Return a vector with elements coerced to ELEMENT-TYPE.

**Package** [`num-utils.matrix-shorthand`], page 25,

**Source** [`matrix-shorthand.lisp`], page 13, (file)

`weighted-quantiles` *VALUES WEIGHTS QS* [Function]

Calculate quantiles QS of weighted observations. Uses a 0.5 correction.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`within?` *LEFT VALUE RIGHT* [Function]

Return non-nil iff value is in [left,right).

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`worst-case` *INSTANCE* [Function]

`(setf worst-case)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`wrapped-matrix-elements` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

## 4.1.5 Generic functions

`add` *ACCUMULATOR OBJECT* **&key** *WEIGHT* [Generic Function]

Add OBJECT to ACCUMULATOR. Return OBJECT. NILs are ignored by the accumulator, unless a specialized method decides otherwise. Keywords may be used to specify additional information (eg weight).

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`add` (*ACCUMULATOR* `sparse-counter`) *OBJECT* **&key** *WEIGHT* [Method]

`add` (*ACCUMULATOR* `sorted-reals`) *OBJECT* **&key** [Method]

`add` (*MOMENTS* `central-sample-moments`) (*Y* `real`) **&key** *WEIGHT* [Method]

`add` *ACCUMULATOR* (*OBJECT* `null`) **&key** [Method]

`as-alist` *OBJECT* [Generic Function]

Return OBJECT as an ALIST. Semantics depends on OBJECT.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

**Methods**

`as-alist` (*OBJECT* `sparse-counter`) [Method]

Return (OBJECT . COUNT) pairs as an alist.

**Source** [`statistics.lisp`], page 13, (file)

`as-plist` *OBJECT* [Generic Function]

Return OBJECT as a PLIST. Semantics depends on OBJECT. The default method uses AS-ALIST.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

**Methods**

`as-plist` *OBJECT* [Method]

`central-m2` *OBJECT* **&key** *WEIGHTS* [Generic Function]

Second central moment. For samples, normalized by the total weight (and thus not the unbiased estimator, see VARIANCE).

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`central-m2` *OBJECT* **&key** *WEIGHTS* [Method]

`central-m2` (*OBJECT* `central-sample-moments`) **&key** *WEIGHTS* [Method]

`central-m3` *OBJECT* **&key** *WEIGHTS* [Generic Function]

Third central moment.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`central-m3` *OBJECT* **&key** *WEIGHTS* [Method]

`central-m3` (*OBJECT* `central-sample-moments`) **&key** *WEIGHTS* [Method]

`central-m4` *OBJECT* **&key** *WEIGHTS* [Generic Function]

Fourth central moment.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`central-m4` *OBJECT* **&key** *WEIGHTS* [Method]

`central-m4` (*OBJECT* `central-sample-moments`) **&key** *WEIGHTS* [Method]

`central-sample-moments` *OBJECT* **&key** *DEGREE WEIGHTS* [Generic Function]

Return a CENTRAL-SAMPLE-MOMENTS object that allows the
calculation of the central sample moments of OBJECT up to the given DEGREE.

When WEIGHTS are given, they need to be a sequence of matching length.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`central-sample-moments` (*OBJECT* `null`) **&key** *DEGREE WEIGHTS* [Method]

`central-sample-moments` (*MOMENTS* `central-sample-moments`) **&key** *DEGREE WEIGHTS* [Method]

`central-sample-moments` (*SEQUENCE* `sequence`) **&key** *DEGREE WEIGHTS* [Method] `diagonal-vector` *MATRIX* [Generic Function]

Return the diagonal elements of MATRIX as a vector.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

**Writer** [`(setf diagonal-vector)`], page 58, (generic function)

**Methods**

`diagonal-vector (`*MATRIX* `array)` [Method]

`diagonal-vector` *MATRIX* [Method]

`(setf diagonal-vector)` *VECTOR MATRIX* [Generic Function]

Set the diagonal elements of MATRIX using VECTOR.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

**Reader** [`diagonal-vector`], page 58, (generic function)

`e1-` *A* [Generic Function]

Univariate elementwise -.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`e1- (`*A* `diagonal-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e1- (`*A* `hermitian-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e1- (`*A* `upper-triangular-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e1- (`*A* `lower-triangular-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e1- (`*A* `number)` [Method]

`e1- (`*A* `array)` [Method]

`e1/` *A* [Generic Function]

Univariate elementwise /.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`e1/ (`*A* `diagonal-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e1/ (`*A* `hermitian-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e1/` (*A* `upper-triangular-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e1/` (*A* `lower-triangular-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e1/` (*A* `number`) [Method]

`e1/` (*A* `array`) [Method]

`e1log` *A* [Generic Function]

Univariate elementwise LOG.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`e1log` (*A* `diagonal-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e1log` (*A* `hermitian-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e1log` (*A* `upper-triangular-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e1log` (*A* `lower-triangular-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e1log` (*A* `number`) [Method]

`e1log` (*A* `array`) [Method]

`e2*` *A B* [Generic Function]

Bivariate elementwise *.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`e2*` (*A* `diagonal-matrix`) (*B* `diagonal-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `hermitian-matrix`) (*B* `hermitian-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `upper-triangular-matrix`) (*B* `upper-triangular-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `lower-triangular-matrix`) (*B* `lower-triangular-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `number`) (*B* `diagonal-matrix`) [Method]
: **Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `diagonal-matrix`) (*B* `number`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `number`) (*B* `hermitian-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `hermitian-matrix`) (*B* `number`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `number`) (*B* `upper-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `upper-triangular-matrix`) (*B* `number`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `number`) (*B* `lower-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `lower-triangular-matrix`) (*B* `number`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` *A* (*B* `wrapped-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `wrapped-matrix`) *B* [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2*` (*A* `number`) (*B* `number`) [Method]

`e2*` (*A* `vector`) (*B* `number`) [Method]

`e2*` (*A* `number`) (*B* `vector`) [Method]

`e2*` (*A* `vector`) (*B* `vector`) [Method]

`e2*` (*A* `array`) (*B* `number`) [Method]

`e2*` (*A* `number`) (*B* `array`) [Method]

`e2*` (*A* `array`) (*B* `array`) [Method]

`e2+` *A* *B* [Generic Function]

Bivariate elementwise `+`.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`e2+` (*A* `diagonal-matrix`) (*B* `diagonal-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2+` (*A* `hermitian-matrix`) (*B* `hermitian-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2+` (*A* `upper-triangular-matrix`) (*B* `upper-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2+` (*A* `lower-triangular-matrix`) (*B* `lower-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2+` *A* (*B* `wrapped-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2+` (*A* `wrapped-matrix`) *B* [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2+` (*A* `number`) (*B* `number`) [Method]

`e2+` (*A* `vector`) (*B* `number`) [Method]

`e2+` (*A* `number`) (*B* `vector`) [Method]

`e2+` (*A* `vector`) (*B* `vector`) [Method]

`e2+` (*A* `array`) (*B* `number`) [Method]

`e2+` (*A* `number`) (*B* `array`) [Method]

`e2+` (*A* `array`) (*B* `array`) [Method]

`e2-` *A B* [Generic Function]

Bivariate elementwise -.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`e2-` (*A* `diagonal-matrix`) (*B* `diagonal-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2-` (*A* `hermitian-matrix`) (*B* `hermitian-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2-` (*A* `upper-triangular-matrix`) (*B* `upper-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2-` (*A* `lower-triangular-matrix`) (*B* `lower-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2-` *A* (*B* `wrapped-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2-` (*A* `wrapped-matrix`) *B* [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2-` (*A* `number`) (*B* `number`) [Method]

`e2-` (*A* `vector`) (*B* `number`) [Method]

`e2-` (*A* `number`) (*B* `vector`) [Method]

`e2-` (*A* `vector`) (*B* `vector`) [Method]

`e2-` (*A* `array`) (*B* `number`) [Method]

`e2-` (*A* `number`) (*B* `array`) [Method]

`e2-` (*A* `array`) (*B* `array`) [Method] `e2/` *A B* [Generic Function]

Bivariate elementwise /.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`e2/` (*A* `number`) (*B* `diagonal-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2/` (*A* `diagonal-matrix`) (*B* `number`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2/` (*A* `number`) (*B* `hermitian-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2/` (*A* `hermitian-matrix`) (*B* `number`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2/` (*A* `number`) (*B* `upper-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2/` (*A* `upper-triangular-matrix`) (*B* `number`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2/` (*A* `number`) (*B* `lower-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2/` (*A* `lower-triangular-matrix`) (*B* `number`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`e2/` (*A* `number`) (*B* `number`) [Method]

`e2/` (*A* `vector`) (*B* `number`) [Method]

`e2/` (*A* `number`) (*B* `vector`) [Method]

`e2/` (*A* `vector`) (*B* `vector`) [Method]

`e2/` (*A* `array`) (*B* `number`) [Method]

`e2/` (*A* `number`) (*B* `array`) [Method]

`e2/` (*A* `array`) (*B* `array`) [Method]

`e2log` *A B* [Generic Function]

Bivariate elementwise LOG.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`e2log` (*A* `number`) (*B* `number`) [Method]

`e2log` (*A* `vector`) (*B* `number`) [Method]

`e2log` (*A* `number`) (*B* `vector`) [Method]

`e2log` (*A* `vector`) (*B* `vector`) [Method]

`e2log` (*A* `array`) (*B* `number`) [Method]

`e2log` (*A* `number`) (*B* `array`) [Method]

`e2log` (*A* `array`) (*B* `array`) [Method] `eceiling` *A* [Generic Function]

Univariate elementwise CEILING.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`eceiling (`*A* `number)` [Method]

`eceiling (`*A* `array)` [Method]

`econjugate` *A* [Generic Function]

Univariate elementwise CONJUGATE.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`econjugate (`*A* `number)` [Method]

`econjugate (`*A* `array)` [Method]

`eexp` *A* [Generic Function]

Univariate elementwise EXP.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`eexp (`*A* `diagonal-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`eexp (`*A* `hermitian-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`eexp (`*A* `upper-triangular-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`eexp (`*A* `lower-triangular-matrix)` [Method]

**Source** [`matrix.lisp`], page 10, (file)

`eexp (`*A* `number)` [Method]

`eexp (`*A* `array)` [Method]

`eexpt` *A B* [Generic Function]

Bivariate elementwise EXPT.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`eexpt` (*A* `number`) (*B* `number`) [Method]

`eexpt` (*A* `vector`) (*B* `number`) [Method]

`eexpt` (*A* `number`) (*B* `vector`) [Method]

`eexpt` (*A* `vector`) (*B* `vector`) [Method]

`eexpt` (*A* `array`) (*B* `number`) [Method]

`eexpt` (*A* `number`) (*B* `array`) [Method]

`eexpt` (*A* `array`) (*B* `array`) [Method]

`efloor` *A* [Generic Function]

Univariate elementwise FLOOR.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`efloor` (*A* `number`) [Method]

`efloor` (*A* `array`) [Method]

`ensure-sorted-reals` *OBJECT* [Generic Function]

Return the contents of OBJECT as a SORTED-REALS.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`ensure-sorted-reals` (*SORTED-REALS* `sorted-reals`) [Method]

`ensure-sorted-reals` (*ARRAY* `array`) [Method]

`ensure-sorted-reals` (*LIST* `list`) [Method]

`ereduce` *FUNCTION OBJECT* **&key** *KEY* [Generic Function]

Elementwise reduce, traversing in row-major order.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`ereduce` *FUNCTION* (*ARRAY* `array`) **&key** *KEY* [Method]

`ereduce` *FUNCTION* (*SEQUENCE* `sequence`) **&key** *KEY* [Method]

`ereduce` *FUNCTION OBJECT* **&key** *KEY* [Method]

`esqrt` *A* [Generic Function]

Univariate elementwise SQRT.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`esqrt` (*A* `diagonal-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`esqrt` (*A* `hermitian-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`esqrt` (*A* `upper-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`esqrt` (*A* `lower-triangular-matrix`) [Method]

**Source** [`matrix.lisp`], page 10, (file)

`esqrt` (*A* `number`) [Method]

`esqrt` (*A* `array`) [Method]

`extend-interval` *INTERVAL OBJECT* [Generic Function]

Return an interval that includes INTERVAL and OBJECT. NIL stands for the empty set.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Methods**

`extend-interval` (*INTERVAL* `null`) (*OBJECT* `null`) [Method]

`extend-interval` (*INTERVAL* `null`) (*NUMBER* `real`) [Method]

`extend-interval` (*INTERVAL* `interval`) (*NUMBER* `real`) [Method]

`extend-interval` *INTERVAL* (*OBJECT* `interval`) [Method]

`extend-interval` *INTERVAL* (*LIST* `list`) [Method]

`extend-interval` *INTERVAL* (*ARRAY* `array`) [Method]

`kurtosis` *OBJECT* **&key** *WEIGHTS* [Generic Function]

Kurtosis FIXME talk about bias, maybe implement unbiased?

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`kurtosis` *OBJECT* **&key** *WEIGHTS* [Method]

`kurtosis` (*OBJECT* `central-sample-moments`) **&key** *WEIGHTS* [Method]

`l2norm-square` *OBJECT* [Generic Function]

Square of the $L_2$ norm of OBJECT.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

**Methods**

`l2norm-square` (*SEQUENCE* `sequence`) [Method]

`left` *INTERVAL* [Generic Function]

Left endpoint of interval.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Methods**

`left` (*INTERVAL* `interval/infinite-left`) [Method]

`left` (*INTERVAL/FINITE-LEFT* `interval/finite-left`) [Method]
automatically generated reader method

`mean` *OBJECT* **&key** *WEIGHTS* [Generic Function]
The mean of elements in OBJECT.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`mean` *OBJECT* **&key** *WEIGHTS* [Method]

`mean` (*OBJECT* `central-sample-moments`) **&key** *WEIGHTS* [Method]

`median` *OBJECT* [Generic Function]
Median of OBJECT.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`median` (*OBJECT* `sequence`) [Method]

`median` *OBJECT* [Method]

`num=` *A B* **&optional** *TOLERANCE* [Generic Function]
Compare A and B for approximate equality, checking corresponding elements when applicable (using TOLERANCE).

Two numbers A and B are NUM= iff |a-b|/max(1,|a|,|b|) <= tolerance.

Unless a method is defined for them, two objects are compared with EQUALP.

Generally, methods should be defined so that two objects are NUM= if they the same class, same dimensions, and all their elements are NUM=.

**Package** [`num-utils.num=`], page 26,

**Source** [`num=.lisp`], page 4, (file)

**Methods**

`num=` (*A* `central-sample-moments`) (*B* `central-sample-moments`) **&optional** *TOLERANCE* [Method]

**Source** [`statistics.lisp`], page 13, (file)

`num=` (*A* `diagonal-matrix`) (*B* `diagonal-matrix`) **&optional** *TOLERANCE* [Method]

**Source** [`matrix.lisp`], page 10, (file)

`num=` (*A* `wrapped-matrix`) (*B* `wrapped-matrix`) **&optional** *TOLERANCE* [Method]

**Source** [`matrix.lisp`], page 10, (file)

`num=` (*A* `finite-interval`) (*B* `finite-interval`) **&optional** *TOLERANCE* [Method]

**Source** [`interval.lisp`], page 8, (file)

`num=` (*A* `real-line`) (*B* `real-line`) **&optional** *TOLERANCE* [Method]

**Source** [`interval.lisp`], page 8, (file)

`num=` *A B* **&optional** *TOLERANCE* [Method]

`num=` (*A* `number`) (*B* `number`) **&optional** *TOLERANCE* [Method]

`num=` (*A* `array`) (*B* `array`) **&optional** *TOLERANCE* [Method]

`num=` (*A* `cons`) (*B* `cons`) **&optional** *TOLERANCE* [Method]

`num=` (*A* `null`) (*B* `null`) **&optional** *TOLERANCE* [Method]

`open-left?` *INTERVAL* [Generic Function]

True iff the left endpoint of the interval is open.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Methods**

`open-left?` (*INTERVAL* `interval/infinite-left`) [Method]

`open-left?` (*INTERVAL/FINITE-LEFT* `interval/finite-left`) [Method]

automatically generated reader method

`open-right?` *INTERVAL* [Generic Function]

True iff the right endpoint of the interval is open.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Methods**

`open-right?` (*INTERVAL* `interval/infinite-right`) [Method]

`open-right?` (*INTERVAL/FINITE-RIGHT* `interval/finite-right`) [Method]

automatically generated reader method

`product` *OBJECT* [Generic Function]

Product of elements in object.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

**Methods**

`product` (*SEQUENCE* `sequence`) [Method]

`product` (*ARRAY* `array`) [Method]

`quantile` *OBJECT Q* [Generic Function]

Return an element at quantile Q. May be an interpolation or an approximation, depending on OBJECT and Q. NOTE: Extensions should define methods for QUANTILES, not QUANTILE.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`quantile` (*OBJECT* `sequence`) *Q* [Method]

`quantile` *OBJECT Q* [Method]

`quantiles` *OBJECT QS* [Generic Function]

Multiple quantiles (see QUANTILE). NOTE: Extensions should define methods for QUANTILES, not QUANTILE.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`quantiles` (*OBJECT* `sequence`) *QS* [Method]

`quantiles` (*ACCUMULATOR* `sorted-reals`) *Q* [Method]

`right` *INTERVAL* [Generic Function]

Right endpoint of interval.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Methods**

`right` (*INTERVAL* `interval/infinite-right`) [Method]

`right` (*INTERVAL/FINITE-RIGHT* `interval/finite-right`) [Method]

automatically generated reader method

`sd` *OBJECT* **&key** *WEIGHTS* [Generic Function]

Standard deviation. For samples, the square root of the unbiased estimator (see VARIANCE).

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`sd` *OBJECT* **&key** *WEIGHTS* [Method]

`shift-interval` *INTERVAL OFFSET* [Generic Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Methods**

`shift-interval` (*INTERVAL* `finite-interval`) (*OFFSET* `real`) [Method] `skewness` *OBJECT* **&key** *WEIGHTS* [Generic Function]

Skewness FIXME talk about bias, maybe implement unbiased?

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`skewness` *OBJECT* **&key** *WEIGHTS* [Method]

`skewness` (*OBJECT* `central-sample-moments`) **&key** *WEIGHTS* [Method]

`sum` *OBJECT* **&key** *KEY* [Generic Function]

Sum of elements in object. KEY is applied to each element.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

**Methods**

`sum` (*SEQUENCE* `sequence`) **&key** *KEY* [Method]

`sum` (*ARRAY* `array`) **&key** *KEY* [Method]

`tally` *ACCUMULATOR* [Generic Function]

The total weight of elements in ACCUMULATOR.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`tally` (*ACCUMULATOR* `sparse-counter`) [Method]

`tally` (*ACCUMULATOR* `tally-mixin`) [Method]

`transpose` *ARRAY* [Generic Function]

Transpose.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

**Methods**

`transpose` (*ARRAY* `array`) [Method]

`transpose` (*MATRIX* `lower-triangular-matrix`) [Method]

`transpose` (*MATRIX* `upper-triangular-matrix`) [Method]

`transpose` (*MATRIX* `hermitian-matrix`) [Method]

`transpose` (*DIAGONAL* `diagonal-matrix`) [Method]

`variance` *OBJECT* **&key** *WEIGHTS* [Generic Function]

Variance of OBJECT. For samples, normalized by the weight-1 (and thus unbiased if certain assumptions hold, eg weights that count frequencies).

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`variance` *OBJECT* **&key** *WEIGHTS* [Method]

`variance` (*OBJECT* `central-sample-moments`) **&key** *WEIGHTS* [Method]

### 4.1.6 Conditions

`empty-accumulator` () [Condition]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Direct superclasses**
`error` (condition)

`information-not-collected-in-accumulator` () [Condition]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Direct superclasses**
`error` (condition)

`not-enough-elements-in-accumulator` () [Condition]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Direct superclasses**
`error` (condition)

### 4.1.7 Structures

`central-sample-moments` () [Structure]

Central sample moments calculated on-line/single-pass.

M weighted mean
S2 weighted sum of squared deviations from the mean, not calculated when NIL S3 weighted sum of cubed deviations from the mean, not calculated when NIL S4 weighted sum of 4th power deviations from the mean, not calculated when NIL

Allows on-line, numerically stable calculation of moments. See cite{bennett2009numerically} and cite{pebay2008formulas} for the description of the algorithm. M_2, ..., M_4 in the paper are s2, ..., s4 in the code.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Direct superclasses**
[`tally-mixin`], page 96, (structure)

**Direct methods**

- [`kurtosis`], page 65, (method)
- [`skewness`], page 69, (method)
- [`central-m4`], page 57, (method)
- [`central-m3`], page 57, (method)
- [`central-m2`], page 57, (method)
- [`variance`], page 69, (method)
- [`mean`], page 66, (method)
- [`central-sample-moments`], page 57, (method)
- [`pool2`], page 92, (method)

- [`add`], page 56, (method)
- [`num=`], page 66, (method)

**Direct slots**

`m` [Slot]

| | |
|---|---|
| **Type** | `real` |
| **Initform** | `0.0d0` |
| **Readers** | [`central-sample-moments-m`], page 82, (function) |
| **Writers** | [`(setf central-sample-moments-m)`], page 82, (function) |

`s2` [Slot]

| | |
|---|---|
| **Type** | `(or (real 0) null)` |
| **Initform** | `0.0d0` |
| **Readers** | [`central-sample-moments-s2`], page 82, (function) |
| **Writers** | [`(setf central-sample-moments-s2)`], page 82, (function) |

`s3` [Slot]

| | |
|---|---|
| **Type** | `(or real null)` |
| **Initform** | `0.0d0` |
| **Readers** | [`central-sample-moments-s3`], page 82, (function) |
| **Writers** | [`(setf central-sample-moments-s3)`], page 82, (function) |

`s4` [Slot]

| | |
|---|---|
| **Type** | `(or (real 0) null)` |
| **Initform** | `0.0d0` |
| **Readers** | [`central-sample-moments-s4`], page 82, (function) |
| **Writers** | [`(setf central-sample-moments-s4)`], page 82, (function) |

`diagonal-matrix ()` [Structure]

Diagonal matrix. The elements in the diagonal are stored in a vector.

| | |
|---|---|
| **Package** | [`num-utils.matrix`], page 37, |
| **Source** | [`matrix.lisp`], page 10, (file) |

**Direct superclasses**

`structure-object` (structure)

**Direct methods**

- [`transpose`], page 69, (method)
- [`num=`], page 66, (method)
- [`esqrt`], page 64, (method)
- [`e1log`], page 59, (method)
- [`eexp`], page 63, (method)
- [`e1/`], page 58, (method)
- [`e1-`], page 58, (method)
- [`e2*`], page 59, (method)

- [`e2-`], page 61, (method)
- [`e2+`], page 60, (method)
- [`e2/`], page 62, (method)
- [`e2/`], page 62, (method)
- [`e2*`], page 59, (method)
- [`e2*`], page 60, (method)
- `dims` (method)
- `element-type` (method)
- `as-array` (method)

**Direct slots**

`elements` [Slot]

| | |
|---|---|
| **Type** | `vector` |
| **Readers** | [`diagonal-matrix-elements`], page 47, (function) |
| **Writers** | [`(setf diagonal-matrix-elements)`], page 47, (function) |

`hermitian-matrix ()` [Structure]

Hermitian/symmetric matrix, with elements stored in the _lower_ triangle.

Implements _both_ real symmetric and complex Hermitian matrices — as technically, real symmetric matrices are also Hermitian. Complex symmetric matrices are _not_ implemented as a special matrix type, as they don't have any special properties (eg real eigenvalues, etc).

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

**Direct superclasses**
[`wrapped-matrix`], page 77, (structure)

**Direct methods**

- [`transpose`], page 69, (method)
- [`esqrt`], page 65, (method)
- [`e1log`], page 59, (method)
- [`eexp`], page 63, (method)
- [`e1/`], page 58, (method)
- [`e1-`], page 58, (method)
- [`e2*`], page 59, (method)
- [`e2-`], page 61, (method)
- [`e2+`], page 60, (method)
- [`e2/`], page 62, (method)
- [`e2/`], page 62, (method)
- [`e2*`], page 60, (method)
- [`e2*`], page 60, (method)
- `select` (method)
- `print-object` (method)
- `as-array` (method)

`lower-triangular-matrix ()` [Structure]

Lower triangular matrix. ELEMENTS in the upper triangle are treated as zero.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

**Direct superclasses**
[`wrapped-matrix`], page 77, (structure)

**Direct methods**
- [`transpose`], page 69, (method)
- [`esqrt`], page 65, (method)
- [`e1log`], page 59, (method)
- [`eexp`], page 63, (method)
- [`e1/`], page 59, (method)
- [`e1-`], page 58, (method)
- [`e2*`], page 59, (method)
- [`e2-`], page 61, (method)
- [`e2+`], page 61, (method)
- [`e2/`], page 62, (method)
- [`e2/`], page 62, (method)
- [`e2*`], page 60, (method)
- [`e2*`], page 60, (method)
- `select` (method)
- `print-object` (method)
- `as-array` (method)

`relative ()` [Structure]

Relative sizes are in terms of width.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**
`structure-object` (structure)

**Direct slots**

`fraction` [Slot]

**Type** `(real 0)`

**Readers** [`relative-fraction`], page 88, (function)

**Writers** `(setf relative-fraction)` (function)

`sorted-reals ()` [Structure]

Accumulator which sorts elements. ELEMENTS return the sorted elements.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Direct superclasses**
`structure-object` (structure)

**Direct methods**
- [`quantiles`], page 68, (method)
- [`ensure-sorted-reals`], page 64, (method)
- `print-object` (method)
- [`add`], page 56, (method)

**Direct slots**

`ordered-elements` [Slot]

| | |
|---|---|
| **Type** | `vector` |
| **Initform** | `#()` |
| **Readers** | [`sorted-reals-ordered-elements`], page 89, (function) |
| **Writers** | [`(setf sorted-reals-ordered-elements)`], page 89, (function) |

`unordered-elements` [Slot]

| | |
|---|---|
| **Type** | `list` |
| **Readers** | [`sorted-reals-unordered-elements`], page 90, (function) |
| **Writers** | [`(setf sorted-reals-unordered-elements)`], page 90, (function) |

`spacer` () [Structure]

Spacers divide the leftover portion of an interval.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**
`structure-object` (structure)

**Direct slots**

`weight` [Slot]

| | |
|---|---|
| **Type** | `(real 0)` |
| **Initform** | `1` |
| **Readers** | [`spacer-weight`], page 90, (function) |
| **Writers** | `(setf spacer-weight)` (function) |

`sparse-counter` () [Structure]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Direct superclasses**
`structure-object` (structure)

**Direct methods**
- `print-object` (method)
- [`as-alist`], page 56, (method)
- [`tally`], page 69, (method)
- [`add`], page 56, (method)

**Direct slots**

`table` [Slot]

| | |
|---|---|
| **Type** | `hash-table` |
| **Readers** | [`sparse-counter-table`], page 54, (function) |
| **Writers** | `(setf sparse-counter-table)` (function) |

`test-results` () [Structure]

Differences between reference values and computed values

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

**Direct superclasses**

`structure-object` (structure)

**Direct slots**

`worst-case` [Slot]

| | |
|---|---|
| **Type** | `integer` |
| **Initform** | `0` |
| **Readers** | [`worst-case`], page 56, (function) |
| **Writers** | [`(setf worst-case)`], page 56, (function) |

`min-error` [Slot]

| | |
|---|---|
| **Type** | `double-float` |
| **Initform** | `0.0d0` |
| **Readers** | [`min-error`], page 51, (function) |
| **Writers** | [`(setf min-error)`], page 51, (function) |

`max-error` [Slot]

| | |
|---|---|
| **Type** | `double-float` |
| **Initform** | `0.0d0` |
| **Readers** | [`max-error`], page 51, (function) |
| **Writers** | [`(setf max-error)`], page 51, (function) |

`mean-error` [Slot]

| | |
|---|---|
| **Type** | `double-float` |
| **Initform** | `0.0d0` |
| **Readers** | [`mean-error`], page 51, (function) |
| **Writers** | [`(setf mean-error)`], page 51, (function) |

`test-count` [Slot]

| | |
|---|---|
| **Type** | `integer` |
| **Initform** | `0` |
| **Readers** | [`test-count`], page 55, (function) |
| **Writers** | [`(setf test-count)`], page 55, (function) |

`variance0` [Slot]

**Type** `double-float`

**Initform** `0.0d0`

**Readers** [`variance0`], page 55, (function)

**Writers** [`(setf variance0)`], page 55, (function)

`variance1` [Slot]

**Type** `double-float`

**Initform** `0.0d0`

**Readers** [`variance1`], page 55, (function)

**Writers** [`(setf variance1)`], page 55, (function)

`rms` [Slot]

**Type** `double-float`

**Initform** `0.0d0`

**Readers** [`rms`], page 53, (function)

**Writers** [`(setf rms)`], page 53, (function)

`upper-triangular-matrix ()` [Structure]

Upper triangular matrix. ELEMENTS in the lower triangle are treated as zero.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

**Direct superclasses**
[`wrapped-matrix`], page 77, (structure)

**Direct methods**

- [`transpose`], page 69, (method)
- [`esqrt`], page 65, (method)
- [`e1log`], page 59, (method)
- [`eexp`], page 63, (method)
- [`e1/`], page 59, (method)
- [`e1-`], page 58, (method)
- [`e2*`], page 59, (method)
- [`e2-`], page 61, (method)
- [`e2+`], page 60, (method)
- [`e2/`], page 62, (method)
- [`e2/`], page 62, (method)
- [`e2*`], page 60, (method)
- [`e2*`], page 60, (method)
- `select` (method)
- `print-object` (method)
- `as-array` (method)

`wrapped-matrix` () [Structure]

A matrix that has some special structure (eg triangular, symmetric/hermitian). ELEMENTS is always a matrix. Not used directly, not exported.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

**Direct superclasses**

`structure-object` (structure)

**Direct subclasses**

- [`lower-triangular-matrix`], page 73, (structure)
- [`upper-triangular-matrix`], page 76, (structure)
- [`hermitian-matrix`], page 72, (structure)

**Direct methods**

- [`num=`], page 66, (method)
- [`e2*`], page 60, (method)
- [`e2*`], page 60, (method)
- [`e2-`], page 61, (method)
- [`e2-`], page 61, (method)
- [`e2+`], page 61, (method)
- [`e2+`], page 61, (method)
- `dims` (method)
- `element-type` (method)

**Direct slots**

`elements` [Slot]

**Type** `(array * (* *))`

**Readers** [`wrapped-matrix-elements`], page 56, (function)

**Writers** `(setf wrapped-matrix-elements)` (function)

### 4.1.8 Classes

`finite-interval` () [Class]

Interval with finite endpoints.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**

- [`interval/finite-right`], page 97, (class)
- [`interval/finite-left`], page 96, (class)
- [`interval`], page 78, (class)

**Direct methods**

- [`transformed-quadrature`], page 94, (method)
- [`chebyshev-approximate-implementation`], page 92, (method)
- [`shift-interval`], page 68, (method)
- [`num=`], page 67, (method)
- `initialize-instance` (method)

`interval` () [Class]

Abstract superclass for all intervals.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**
`standard-object` (class)

**Direct subclasses**
- [`finite-interval`], page 77, (class)
- [`plusinf-interval`], page 78, (class)
- [`minusinf-interval`], page 78, (class)
- [`real-line`], page 78, (class)

**Direct methods**
- [`extend-interval`], page 65, (method)
- [`extend-interval`], page 65, (method)
- `print-object` (method)

`minusinf-interval` () [Class]

Interval from -$\infty$ to RIGHT.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**
- [`interval/finite-right`], page 97, (class)
- [`interval/infinite-left`], page 97, (class)
- [`interval`], page 78, (class)

`plusinf-interval` () [Class]

Interval from LEFT to $\infty$.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**
- [`interval/infinite-right`], page 98, (class)
- [`interval/finite-left`], page 96, (class)
- [`interval`], page 78, (class)

**Direct methods**
- [`transformed-quadrature`], page 94, (method)
- [`chebyshev-approximate-implementation`], page 92, (method)

`real-line` () [Class]

Representing the real line (-$\infty$,$\infty$).

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**
- [`interval/infinite-right`], page 98, (class)
- [`interval/infinite-left`], page 97, (class)

- [`interval`], page 78, (class)

**Direct methods**
[`num=`], page 67, (method)

### 4.1.9 Types

`extended-real` *&optional BASE* [Type]
Extended real number.

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`simple-double-float-vector` *&optional LENGTH* [Type]
Simple vector of double-float elements.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`simple-fixnum-vector` () [Type]
Simple vector or fixnum elements.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`simple-single-float-vector` *&optional LENGTH* [Type]
Simple vector of single-float elements.

**Package** [`num-utils.utilities`], page 23,

**Source** [`utilities.lisp`], page 3, (file)

`triangular-matrix` () [Type]
Triangular matrix (either lower or upper).

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

## 4.2 Internal definitions

### 4.2.1 Macros

`&diagonal-matrix` *ELEMENTS* [Macro]
LET+ form for slots of the structure DIAGONAL-MATRIX.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`&diagonal-matrix-r/o` *ELEMENTS* [Macro]
LET+ form for slots of the structure DIAGONAL-MATRIX. Read-only.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`&sorted-reals` *ORDERED-ELEMENTS UNORDERED-ELEMENTS* [Macro]
LET+ form for slots of the structure SORTED-REALS.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`&sorted-reals-r/o` *ORDERED-ELEMENTS UNORDERED-ELEMENTS* [Macro]
LET+ form for slots of the structure SORTED-REALS. Read-only.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`define-central-sample-moment` *FUNCTION* (*VARIABLE DEGREE*) **&body** *BODY* [Macro]
FIXME documentation, factor out general part

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`define-comparison` *NAME TEST* [Macro]
Define a comparison, extendeding a pairwise comparison to an arbitrary number of arguments.

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`define-e&` *OPERATION* **&key** *FUNCTION BIVARIATE UNIVARIATE DOCSTRING* [Macro]

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`define-e1` *OPERATION* **&key** *FUNCTION DOCSTRING* [Macro]
Define an univariate elementwise operation.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`define-e2` *OPERATION* **&key** *FUNCTION DOCSTRING* [Macro]
Define an univariate elementwise operation.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`define-elementwise-as-array` *TYPE* **&key** *FUNCTIONS* [Macro]
Define binary elementwise operations for FUNCTION, implemented by converting them to arrays.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`define-elementwise-reduction` *NAME FUNCTION* **&optional** *DOCSTRING* [Macro]

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`define-elementwise-same-class` *TYPE* **&key** *FUNCTIONS ELEMENTS-ACCESSOR* [Macro]
Define binary elementwise operations for FUNCTION for two arguments of the same class.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`define-elementwise-univariate` *TYPE* **&key** *FUNCTIONS ELEMENTS-ACCESSOR* [Macro]

Define unary elementwise operations for FUNCTION for all subclasses of wrapped-elements.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`define-elementwise-with-constant` *TYPE* **&key** *FUNCTIONS ELEMENTS-ACCESSOR* [Macro]

Define binary elementwise operations for FUNCTION for all subclasses of wrapped-elements.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`define-rounding-with-offset` *NAME FUNCTION DOCSTRING* [Macro]

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`define-wrapped-matrix` *TYPE ELEMENTS STRUCT-DOCSTRING* (*MASKED-TEST MASKED-STRING*) *CHECK-AND-CONVERT-ELEMENTS REGULARIZE-ELEMENTS* [Macro]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`mapping-array` (*REF ARRAY* **&rest** *OTHER*) *FORM* [Macro]

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

`univariate-rootfinder-loop%` ((*F A B FA FB*) (*F-TESTED TEST-BRACKET DELTA EPSILON*)) **&body** *BODY* [Macro]

Common parts for univariate rootfinder functions.

Sets up the following:

- function OPPOSITE-SIGN-P for checking that two numbers are on the opposite side of 0

- function EVALUATE-AND-RETURN-IF-WITHIN-EPSILON which checks that `|f(x)| <=` EPSILON, if so, returns from the block with (VALUES X FX T), otherwise simply returns the value

- function RETURN-IF-WITHIN-TOLERANCE checks if the interval [A,B] bracketing X is small enough (smaller than TOLERANCE) and if so, returns (X FX NIL (INTERVAL A B))

- variables FA and FB to hold function values at A and B

Initially, it checks for either \$f(a)\$ or \$f(b)\$ being a root, and establishes \$a leq b\$ by exchanging \$a,f(a)\$ and \$b,f(b)\$ if necessary. Also checks that \$f(a)\$ and \$f(b)\$ are of opposite sign. Checks that both tolerance and epsilon are nonnegative.

**Package** [`num-utils.rootfinding`], page 32,

**Source** [`rootfinding.lisp`], page 16, (file)

### 4.2.2 Functions

`ab-to-cd-intercept-slope` *A B C D* [Function]

Return (values INTERCEPT SLOPE) for linear mapping x:-> intercept+slope*x from [a,b] to [c,d].

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`ab-to-cinf` *Z A B C* [Function]

Inverse of cinf-to-ab.

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`above-diagonal?` *ROW COL* [Function]

Test if element with indexes row and col is (strictly) above the diagonal.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`below-diagonal?` *ROW COL* [Function]

Test if element with indexes row and col is (strictly) below the diagonal.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`central-sample-moments-m` *INSTANCE* [Function]
`(setf central-sample-moments-m)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`central-sample-moments-p` *OBJECT* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`central-sample-moments-s2` *INSTANCE* [Function]
`(setf central-sample-moments-s2)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`central-sample-moments-s3` *INSTANCE* [Function]
`(setf central-sample-moments-s3)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`central-sample-moments-s4` *INSTANCE* [Function]
`(setf central-sample-moments-s4)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`central-sample-moments-w` *INSTANCE* [Function]
`(setf central-sample-moments-w)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`chebyshev-recursion` *X VALUE PREVIOUS-VALUE* [Function]

Chebyshev polynomial recursion formula.

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`cinf-to-ab` *X A B C* [Function]

Map x in [c,plus-infinity) to z in [a,b] using x -> (x-c)/(1+x-c)+(b-a)+a.

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

`copy-central-sample-moments` *INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`copy-diagonal-matrix` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`copy-hermitian-matrix` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`copy-iterative-quadrature` *INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`copy-lower-triangular-matrix` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`copy-midpoint-quadrature` *INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`copy-relative` *INSTANCE* [Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`copy-richardson-extrapolation` *INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`copy-sorted-reals` *INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`copy-spacer` *INSTANCE* [Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`copy-sparse-counter` *INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`copy-tally-mixin` *INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`copy-test-results` *INSTANCE* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`copy-trapezoidal-quadrature` *INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`copy-upper-triangular-matrix` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`copy-wrapped-matrix` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`diagonal-matrix-p` *OBJECT* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`ensure-valid-elements` *ARRAY RANK* **&rest** *PREDICATES* [Function]

Convert OBJECT to an array, check that it

1. has the required rank,

2. has a valid sparse element type, and
3. that it satisfies PREDICATES.

Return the array.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`extend-pairwise-comparison` *TEST FIRST REST* [Function]

Extend TEST (a pairwise comparison) to an arbitrary number of arguments (but at least one, FIRST).

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

`hermitian-matrix-elements` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`hermitian-matrix-p` *OBJECT* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`iterative-quadrature-a` *INSTANCE* [Function]
`(setf iterative-quadrature-a)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`iterative-quadrature-b` *INSTANCE* [Function]
`(setf iterative-quadrature-b)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`iterative-quadrature-f` *INSTANCE* [Function]
`(setf iterative-quadrature-f)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`iterative-quadrature-h` *INSTANCE* [Function]
`(setf iterative-quadrature-h)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`iterative-quadrature-n` *INSTANCE* [Function]
`(setf iterative-quadrature-n)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`iterative-quadrature-p` *OBJECT* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`iterative-quadrature-sum` *INSTANCE* [Function]
`(setf iterative-quadrature-sum)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`lower-triangular-matrix-elements` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`lower-triangular-matrix-p` *OBJECT* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`make-central-sample-moments` **&key** (*W* **W**) (*M* **M**) (*S2* **S2**) (*S3* **S3**) (*S4* **S4**) [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`make-diagonal-matrix` **&key** (*ELEMENTS* **ELEMENTS**) [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`make-hermitian-matrix` **&key** (*ELEMENTS* **ELEMENTS**) [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`make-iterative-quadrature` **&key** (*F* **F**) (*A* **A**) (*B* **B**) (*H* **H**) (*N* **N**) (*SUM* **SUM**) [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`make-lower-triangular-matrix` **&key** (*ELEMENTS* **ELEMENTS**) [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`make-sorted-reals` **&key** (*ORDERED-ELEMENTS* **ORDERED-ELEMENTS**) (*UNORDERED-ELEMENTS* **UNORDERED-ELEMENTS**) [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`make-sparse-counter%` **&key** (*TABLE* **TABLE**) [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`make-tally-mixin` **&key** (*W* **W**) [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`make-test-results` **&key** (*WORST-CASE* **WORST-CASE**) (*MIN-ERROR* **MIN-ERROR**) (*MAX-ERROR* **MAX-ERROR**) (*MEAN-ERROR* **MEAN-ERROR**) (*TEST-COUNT* **TEST-COUNT**) (*VARIANCE0* **VARIANCE0**) (*VARIANCE1* **VARIANCE1**) (*RMS* **RMS**) [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file) `make-upper-triangular-matrix` **&key** (*ELEMENTS* **ELEMENTS**) [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`make-wrapped-matrix` **&key** (*ELEMENTS* **ELEMENTS**) [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`midpoint-quadrature` *F A B* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`midpoint-quadrature%` **&key** (*F* **F**) (*A* **A**) (*B* **B**) (*H* **H**) (*N* **N**) (*SUM* **SUM**) [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`midpoint-quadrature-a` *INSTANCE* [Function]
`(setf midpoint-quadrature-a)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`midpoint-quadrature-b` *INSTANCE* [Function]
`(setf midpoint-quadrature-b)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`midpoint-quadrature-f` *INSTANCE* [Function]
`(setf midpoint-quadrature-f)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`midpoint-quadrature-h` *INSTANCE* [Function]
`(setf midpoint-quadrature-h)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`midpoint-quadrature-n` *INSTANCE* [Function]
`(setf midpoint-quadrature-n)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`midpoint-quadrature-p` *OBJECT* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`midpoint-quadrature-sum` *INSTANCE* [Function]
`(setf midpoint-quadrature-sum)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`narrow-bracket?` *A B DELTA* [Function]

Return true iff $|a-b| < \delta$.

**Package** [`num-utils.rootfinding`], page 32,

**Source** [`rootfinding.lisp`], page 16, (file)

`near-root?` *F EPSILON* [Function]

Return true iff $|f| < \epsilon$.

**Package** [`num-utils.rootfinding`], page 32,

**Source** [`rootfinding.lisp`], page 16, (file)

`opposite-sign?` *A B* [Function]

Return true iff A and B are on opposite sides of 0.

**Package** [`num-utils.rootfinding`], page 32,

**Source** [`rootfinding.lisp`], page 16, (file)

`pad-left-expansion` *ROWS NCOL* [Function]

Pad ragged-right rows. Used internally to implement ragged right matrix specifications.

**Package** [`num-utils.matrix-shorthand`], page 25,

**Source** [`matrix-shorthand.lisp`], page 13, (file)

`print-matrix-formatter` *X* [Function]

Standard formatter for matrix printing. Respects *print-precision*, and formats complex numbers as a+bi, eg 0.0+1.0i.

**Package** [`num-utils.print-matrix`], page 23,

**Source** [`print-matrix.lisp`], page 10, (file)

`relative-fraction` *INSTANCE* [Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`relative-p` *OBJECT* [Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`richardson-extrapolation` *COEFFICIENT ITERATIONS* **&aux** *DIAGONAL* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`richardson-extrapolation-coefficient` *INSTANCE* [Function]
`(setf richardson-extrapolation-coefficient)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`richardson-extrapolation-diagonal` *INSTANCE* [Function]
`(setf richardson-extrapolation-diagonal)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`richardson-extrapolation-n` *INSTANCE* [Function]
`(setf richardson-extrapolation-n)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`richardson-extrapolation-p` *OBJECT* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`richardson-iteration` *EXTRAPOLATION STEP* [Function]

Add STEP (= $A(h q^{-k}$) to an existing Richardson EXTRAPOLATION. See the documentation of RICHARDSON-EXTRAPOLATION for details.

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`romberg-quadrature%` *QUADRATURE EPSILON MIN-ITER MAX-ITER* [Function]

Internal function implementing Romberg quadrature. Requires an iterative quadrature instance, a relative EPSILON and MIN-ITER for the stopping criterion, and the maximum number of iterations allowed. Works on finite intervals.

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`rootfinding-delta` *INTERVAL* **&optional** *DELTA-RELATIVE* [Function]

Default DELTA for rootfinding methods, uses bracket width.

**Package** [`num-utils.rootfinding`], page 32,

**Source** [`rootfinding.lisp`], page 16, (file)

`similar-element-type` *ELEMENT-TYPE* [Function]

Return a type that is a supertype of ELEMENT-TYPE and is closed under arithmetic operations. May not be the narrowest.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`similar-sequence-type` *SEQUENCE* [Function]

Return type that sequence can be mapped to using arithmetic operations.

**Package** [`num-utils.arithmetic`], page 24,

**Source** [`arithmetic.lisp`], page 4, (file)

`sort-reals` *SEQUENCE* [Function]

Return a SORTED-REALS structure.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`sorted-reals-ordered-elements` *INSTANCE* [Function]
`(setf sorted-reals-ordered-elements)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`sorted-reals-p` *OBJECT* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`sorted-reals-unordered-elements` *INSTANCE* [Function]
`(setf sorted-reals-unordered-elements)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`spacer-p` *OBJECT* [Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`spacer-weight` *INSTANCE* [Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

`sparse-counter-p` *OBJECT* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`tally-mixin-p` *OBJECT* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`tally-mixin-w` *INSTANCE* [Function]
`(setf tally-mixin-w)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`test-results-p` *OBJECT* [Function]

**Package** [`num-utils.test-utilities`], page 26,

**Source** [`test-utilities.lisp`], page 19, (file)

`trapezoidal-quadrature` *F A B* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`trapezoidal-quadrature%` **&key** (*F* **F**) (*A* **A**) (*B* **B**) (*H* **H**) (*N* **N**) (*SUM* **SUM**) [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`trapezoidal-quadrature-a` *INSTANCE* [Function]
`(setf trapezoidal-quadrature-a)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`trapezoidal-quadrature-b` *INSTANCE* [Function]
`(setf trapezoidal-quadrature-b)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`trapezoidal-quadrature-f` *INSTANCE* [Function]
`(setf trapezoidal-quadrature-f)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`trapezoidal-quadrature-h` *INSTANCE* [Function]
`(setf trapezoidal-quadrature-h)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`trapezoidal-quadrature-n` *INSTANCE* [Function]
`(setf trapezoidal-quadrature-n)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`trapezoidal-quadrature-p` *OBJECT* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`trapezoidal-quadrature-sum` *INSTANCE* [Function]
`(setf trapezoidal-quadrature-sum)` *VALUE INSTANCE* [Function]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

`upper-triangular-matrix-elements` *INSTANCE* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`upper-triangular-matrix-p` *OBJECT* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`valid-sparse-type?` *TYPE* [Function]

Check if TYPE is a valid type for sparse matrices. Only supertypes and subtypes of NUMBER are allowed.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`weighted-empirical-quantile` *SORTED-REALS P-TABLE Q* [Function]

Return the empirical quantile of a vector of real numbers, sorted in ascending order (not checked). Uses a 0.5 correction.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`weighted-quantile-p-table` *WEIGHTS* [Function]

Return table of probability brackets for weighted quantile calculations., built from the weights (which should be positive reals, not checked). Uses a 0.5 correction.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

`wrapped-matrix-p` *OBJECT* [Function]

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

`zero-like` *ARRAY* [Function]

Return 0 coerced to the element type of ARRAY. It is assumed that the latter satisfies VALID-SPARSE-TYPE?.

**Package** [`num-utils.matrix`], page 37,

**Source** [`matrix.lisp`], page 10, (file)

### 4.2.3 Generic functions

`chebyshev-approximate-implementation` *F INTERVAL N-POLYNOMIALS N-POINTS* [Generic Function]

Implementation of CHEBYSHEV-APPROXIMATE.

**Package** [`num-utils.chebyshev`], page 37,

**Source** [`chebyshev.lisp`], page 16, (file)

**Methods**

`chebyshev-approximate-implementation` *F* (*INTERVAL* `plusinf-interval`) *N-POLYNOMIALS N-POINTS* [Method]

`chebyshev-approximate-implementation` *F* (*INTERVAL* `finite-interval`) *N-POLYNOMIALS N-POINTS* [Method]

`esquare` *A* [Generic Function]

Univariate elementwise SQUARE.

**Package** [`num-utils.elementwise`], page 33,

**Source** [`elementwise.lisp`], page 5, (file)

**Methods**

`esquare` (*A* `number`) [Method]

`esquare` (*A* `array`) [Method]

`pool2` *ACCUMULATOR1 ACCUMULATOR2* [Generic Function]

Pool two accumulators. When they are of a different type, the resulting accumulator will be downgraded to the level afforded by the information available in the accumulators.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Methods**

`pool2` (*MOMENTS-A* `central-sample-moments`) (*MOMENTS-B* `central-sample-moments`) [Method] `print-left-endpoint` *INTERVAL STREAM* [Generic Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Methods**

`print-left-endpoint` (*INTERVAL* `interval/finite-left`) *STREAM* [Method]

`print-left-endpoint` (*INTERVAL* `interval/infinite-left`) *STREAM* [Method]

`print-right-endpoint` *INTERVAL STREAM* [Generic Function]

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Methods**

`print-right-endpoint` (*INTERVAL* `interval/finite-right`) *STREAM* [Method]

`print-right-endpoint` (*INTERVAL* `interval/infinite-right`) *STREAM* [Method]

`refine-quadrature` *QUADRATURE* [Generic Function]

Refine quadrature with more points. Return the sum for those points.

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

**Methods**

`refine-quadrature` (*QUADRATURE* `midpoint-quadrature`) [Method]

`refine-quadrature` (*QUADRATURE* `trapezoidal-quadrature`) [Method]

`richardson-coefficient` *QUADRATURE* [Generic Function]

Return the coefficient $q$ for Richardson approximation.

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

**Methods**

`richardson-coefficient` (*QUADRATURE* `midpoint-quadrature`) [Method]

`richardson-coefficient` (*QUADRATURE* `trapezoidal-quadrature`) [Method]

`transformed-quadrature` *FUNCTION INTERVAL TRANSFORMATION* [Generic Function]

Return a quadrature for integrating FUNCTION on INTERVAL, which may be infinite, in which case FUNCTION will be transformed. TRANSFORMATION can be used to select the transformation when applicable, otherwise it is NIL.

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

**Methods**

`transformed-quadrature` *FUNCTION* (*INTERVAL* `finite-interval`) (*TRANSFORMATION* `null`) [Method]

`transformed-quadrature` *FUNCTION* (*INTERVAL* `plusinf-interval`) (*TRANSFORMATION* `null`) [Method]

### 4.2.4 Structures

`iterative-quadrature` () [Structure]

Quadrature building block.

F is the function.

A and B are the endpoints.

H is the stepsize.

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

**Direct superclasses**
`structure-object` (structure)

**Direct subclasses**

- [`trapezoidal-quadrature`], page 96, (structure)
- [`midpoint-quadrature`], page 95, (structure)

**Direct slots**

`f` [Slot]

| | |
|---|---|
| **Type** | `(function (double-float) double-float)` |
| **Readers** | [`iterative-quadrature-f`], page 85, (function) |
| **Writers** | [`(setf iterative-quadrature-f)`], page 85, (function) |

`a` [Slot]

| | |
|---|---|
| **Type** | `double-float` |
| **Readers** | [`iterative-quadrature-a`], page 85, (function) |
| **Writers** | [`(setf iterative-quadrature-a)`], page 85, (function) |

`b` [Slot]

| | |
|---|---|
| **Type** | `double-float` |
| **Readers** | [`iterative-quadrature-b`], page 85, (function) |
| **Writers** | [`(setf iterative-quadrature-b)`], page 85, (function) |

`h` [Slot]

| | |
|---|---|
| **Type** | `double-float` |
| **Readers** | [`iterative-quadrature-h`], page 85, (function) |
| **Writers** | [`(setf iterative-quadrature-h)`], page 85, (function) |

`n` [Slot]

| | |
|---|---|
| **Type** | `fixnum` |
| **Initform** | `0` |
| **Readers** | [`iterative-quadrature-n`], page 85, (function) |
| **Writers** | [`(setf iterative-quadrature-n)`], page 85, (function) |

`sum` [Slot]

**Type** `double-float`

**Initform** `0.0d0`

**Readers** [`iterative-quadrature-sum`], page 85, (function)

**Writers** [`(setf iterative-quadrature-sum)`], page 85, (function)

`midpoint-quadrature ()` [Structure]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

**Direct superclasses**
[`iterative-quadrature`], page 94, (structure)

**Direct methods**

- [`richardson-coefficient`], page 93, (method)
- [`refine-quadrature`], page 93, (method)

`richardson-extrapolation ()` [Structure]

Given A(h)=A_0 + sum_{k=1}^infty a_k h^{kp}, calculate approximations for A given A(h q^{-k}), where the latter can be incorporated using RICHARDSON-ITERATION with consecutive values for k=1,...,max_iter, which returns the latest A(0) as the first and the largest relative change, which can be used to test termination.

The algorithm uses Richardson extrapolation, the required coefficient is q^k.

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

**Direct superclasses**
`structure-object` (structure)

**Direct slots**

`coefficient` [Slot]

**Type** `double-float`

**Readers** [`richardson-extrapolation-coefficient`], page 88, (function)

**Writers** [`(setf richardson-extrapolation-coefficient)`], page 88, (function)

`n` [Slot]

**Type** `fixnum`

**Initform** `0`

**Readers** [`richardson-extrapolation-n`], page 89, (function)

**Writers** [`(setf richardson-extrapolation-n)`], page 89, (function)

`diagonal` [Slot]

**Type** `(array double-float (*))`

**Readers** [`richardson-extrapolation-diagonal`], page 88, (function)

**Writers** [`(setf richardson-extrapolation-diagonal)`], page 88, (function) `tally-mixin` () [Structure]

Mixin structure that contains a tally. Not exported. W is the total weight.

**Package** [`num-utils.statistics`], page 30,

**Source** [`statistics.lisp`], page 13, (file)

**Direct superclasses**
`structure-object` (structure)

**Direct subclasses**
[`central-sample-moments`], page 70, (structure)

**Direct methods**
[`tally`], page 69, (method)

**Direct slots**

`w` [Slot]

**Type** `(real 0)`

**Initform** `0`

**Readers** [`tally-mixin-w`], page 90, (function)

**Writers** [`(setf tally-mixin-w)`], page 90, (function)

`trapezoidal-quadrature` () [Structure]

**Package** [`num-utils.quadrature`], page 28,

**Source** [`quadrature.lisp`], page 17, (file)

**Direct superclasses**
[`iterative-quadrature`], page 94, (structure)

**Direct methods**
- [`richardson-coefficient`], page 93, (method)
- [`refine-quadrature`], page 93, (method)

### 4.2.5 Classes

`interval/finite-left` () [Class]

Interval with left endpoint.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**
`standard-object` (class)

**Direct subclasses**
- [`finite-interval`], page 77, (class)
- [`plusinf-interval`], page 78, (class)

**Direct methods**
- [`print-left-endpoint`], page 93, (method)
- [`open-left?`], page 67, (method)
- [`left`], page 66, (method)

**Direct slots**

`left` [Slot]

**Type** `real`

**Initargs** `:left`

**Readers** [`left`], page 65, (generic function)

`open-left?` [Slot]

**Type** `boolean`

**Initargs** `:open-left?`

**Readers** [`open-left?`], page 67, (generic function)

`interval/finite-right ()` [Class]

Interval with right endpoint.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**

`standard-object` (class)

**Direct subclasses**

- [`finite-interval`], page 77, (class)
- [`minusinf-interval`], page 78, (class)

**Direct methods**

- [`print-right-endpoint`], page 93, (method)
- [`open-right?`], page 67, (method)
- [`right`], page 68, (method)

**Direct slots**

`right` [Slot]

**Type** `real`

**Initargs** `:right`

**Readers** [`right`], page 68, (generic function)

`open-right?` [Slot]

**Type** `boolean`

**Initargs** `:open-right?`

**Readers** [`open-right?`], page 67, (generic function)

`interval/infinite-left ()` [Class]

Left endpoint is $-\infty$.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**

`standard-object` (class)

**Direct subclasses**

- [`minusinf-interval`], page 78, (class)
- [`real-line`], page 78, (class)

**Direct methods**

- [`print-left-endpoint`], page 93, (method)
- [`open-left?`], page 67, (method)
- [`left`], page 66, (method)

`interval/infinite-right ()` [Class]

Right endpoint is $\infty$.

**Package** [`num-utils.interval`], page 21,

**Source** [`interval.lisp`], page 8, (file)

**Direct superclasses**
`standard-object` (class)

**Direct subclasses**
- [`plusinf-interval`], page 78, (class)
- [`real-line`], page 78, (class)

**Direct methods**
- [`print-right-endpoint`], page 93, (method)
- [`open-right?`], page 67, (method)
- [`right`], page 68, (method)

### 4.2.6 Types

`infinite ()` [Type]

Representing infinity (extending the real line).

**Package** [`num-utils.extended-real`], page 27,

**Source** [`extended-real.lisp`], page 8, (file)

# Appendix A Indexes

## A.1 Concepts

## A.2 Functions

## A.3 Variables

## A.4 Data types