The NUM-UTILS Reference Manual

Numerical Utilities, version 1.3.0

Steven Nunez `<steve@symbolics.tech>`

This manual was generated automatically by Declt 4.0b2.

Table of Contents

Copying

1 Systems

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

1.1 `num-utils`

Numerical utilities for Common Lisp

Long Name
Numerical Utilities

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

Source Control
`(GIT https://github.com/Lisp-Stat/numerical-utilities.git)`

Bug Tracker
`https://github.com/Lisp-Stat/numerical-utilities/issues`

License MS-PL

Long Description
This library implements simple numerical functions for Common Lisp, including

num=, a comparison operator for floats
simple arithmeric functions, like sum and l2norm
elementwise operations for arrays
intervals
special matrices and shorthand for their input
sample statistics
Chebyshev polynomials
univariate rootfinding

Version 1.3.0

Dependencies
- `anaphora` (system).
- `alexandria` (system).
- `array-operations` (system).
- `select` (system).
- `let-plus` (system).

Source [`num-utils.asd`], page 5.

Child Components
- [`packages.lisp`], page 5 (file).
- [`utilities.lisp`], page 5 (file).
- [`num=.lisp`], page 6 (file).
- [`arithmetic.lisp`], page 7 (file).
- [`elementwise.lisp`], page 7 (file).
- [`extended-real.lisp`], page 9 (file).
- [`interval.lisp`], page 9 (file).
- [`print-matrix.lisp`], page 10 (file).
- [`matrix.lisp`], page 11 (file).
- [`matrix-shorthand.lisp`], page 14 (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/num-utils.asd

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

ASDF Systems
[num-utils], page 3.

2.1.2 num-utils/packages.lisp

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Packages

- [num-utils.quadrature], page 19.
- [num-utils.print-matrix], page 21.
- [num-utils.chebyshev], page 21.
- [num-utils.interval], page 22.
- [num-utils.matrix-shorthand], page 23.
- [num-utils.log-exp], page 24.
- [num-utils.matrix], page 24.
- [num-utils.extended-real], page 26.
- [num-utils.utilities], page 26.
- [num-utils.test-utilities], page 27.
- [num-utils.arithmetic], page 28.
- [num-utils.elementwise], page 29.
- [num-utils.num=], page 31.
- [num-utils.polynomial], page 31.
- [num-utils.rootfinding], page 32.

2.1.3 num-utils/utilities.lisp

Dependency
[packages.lisp], page 5 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface

- [as-alist], page 48 (generic function).
- [as-bit-vector], page 36 (function).

- [as-double-float], page 37 (function).
- [as-plist], page 49 (generic function).
- [as-simple-fixnum-vector], page 37 (function).
- [bic], page 37 (function).
- [binary-search], page 37 (function).
- [check-types], page 33 (macro).
- [curry*], page 33 (macro).
- [define-with-multiple-bindings], page 34 (macro).
- [distinct], page 39 (function).
- [expanding], page 34 (macro).
- [fixnum?], page 41 (function).
- [generate-sequence], page 41 (function).
- [gethash*], page 34 (macro).
- [make-vector], page 36 (compiler macro).
- [make-vector], page 44 (function).
- [monotonicp], page 44 (function).
- [simple-boolean-vector], page 71 (type).
- [simple-double-float-vector], page 71 (type).
- [simple-fixnum-vector], page 71 (type).
- [simple-single-float-vector], page 71 (type).
- [splice-awhen], page 35 (macro).
- [splice-when], page 35 (macro).
- [with-double-floats], page 35 (macro).
- [within?], page 48 (function).

Internals

- [boolean-sequence-p], page 74 (function).
- [boolean?], page 74 (function).

2.1.4 `num-utils/num=.lisp`

Dependency
[utilities.lisp], page 5 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface

- [*num=-tolerance*], page 33 (special variable).
- [define-num=-with-accessors], page 34 (macro).
- [define-structure-num=], page 34 (macro).
- [num-delta], page 44 (function).
- [num=], page 60 (generic function).
- [num=-function], page 45 (function).

2.1.5 num-utils/arithmetic.lisp

Dependency
[num=.lisp], page 6 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface

- [1c], page 36 (function).
- [abs-diff], page 36 (function).
- [absolute-square], page 36 (function).
- [as-integer], page 37 (function).
- [ceiling*], page 37 (function).
- [cube], page 38 (function).
- [cumulative-product], page 38 (function).
- [cumulative-sum], page 38 (function).
- [divides?], page 39 (function).
- [floor*], page 41 (function).
- [ivec], page 42 (function).
- [l2norm], page 42 (function).
- [l2norm-square], page 59 (generic function).
- [log10], page 43 (function).
- [log2], page 43 (function).
- [normalize-probabilities], page 44 (function).
- [numseq], page 45 (function).
- [product], page 61 (generic function).
- [round*], page 46 (function).
- [same-sign-p], page 46 (function).
- [sequence-maximum], page 46 (function).
- [sequence-minimum], page 46 (function).
- [square], page 47 (function).
- [sum], page 62 (generic function).
- [truncate*], page 47 (function).

Internals

- [define-rounding-with-offset], page 72 (macro).
- [ln], page 77 (function).
- [similar-element-type], page 80 (function).
- [similar-sequence-type], page 81 (function).

2.1.6 num-utils/elementwise.lisp

Dependency
[arithmetic.lisp], page 7 (file).

Source [num-utils.asd], page 5.

Parent Component

[num-utils], page 3 (system).

Public Interface

- [e*], page 39 (function).
- [e+], page 39 (function).
- [e-], page 39 (function).
- [e/], page 39 (function).
- [e1-], page 49 (generic function).
- [e1/], page 50 (generic function).
- [e1log], page 50 (generic function).
- [e2*], page 50 (generic function).
- [e2+], page 52 (generic function).
- [e2-], page 53 (generic function).
- [e2/], page 53 (generic function).
- [e2<], page 54 (generic function).
- [e2<=], page 55 (generic function).
- [e2=], page 55 (generic function).
- [e2>], page 55 (generic function).
- [e2>=], page 56 (generic function).
- [e2log], page 56 (generic function).
- [eceiling], page 56 (generic function).
- [econjugate], page 56 (generic function).
- [ecos], page 57 (generic function).
- [eexp], page 57 (generic function).
- [eexpt], page 57 (generic function).
- [efloor], page 58 (generic function).
- [elementwise-float-contagion], page 39 (function).
- [elog], page 40 (function).
- [emax], page 40 (function).
- [emin], page 40 (function).
- [emod], page 58 (generic function).
- [ereduce], page 58 (generic function).
- [esin], page 58 (generic function).
- [esqrt], page 59 (generic function).

Internals

- [define-e&], page 72 (macro).
- [define-e1], page 72 (macro).
- [define-e2], page 72 (macro).
- [define-elementwise-reduction], page 72 (macro).
- [esquare], page 83 (generic function).
- [mapping-array], page 73 (macro).

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

Dependency
[`elementwise.lisp`], page 7 (file).

Source [`num-utils.asd`], page 5.

Parent Component
[`num-utils`], page 3 (system).

Public Interface

- [`<`], page 36 (function).
- [`<=`], page 36 (function).
- [`=`], page 36 (function).
- [`>`], page 36 (function).
- [`>=`], page 36 (function).
- [`extended-real`], page 71 (type).
- [`infinite?`], page 42 (function).
- [`lambda-template`], page 34 (macro).
- [`with-template`], page 35 (macro).

Internals

- [`define-comparison`], page 72 (macro).
- [`extend-pairwise-comparison`], page 75 (function).
- [`infinite`], page 88 (type).

2.1.8 `num-utils/interval.lisp`

Dependency
[`extended-real.lisp`], page 9 (file).

Source [`num-utils.asd`], page 5.

Parent Component
[`num-utils`], page 3 (system).

Public Interface

- [`&interval`], page 33 (macro).
- [`extend-interval`], page 59 (generic function).
- [`extendf-interval`], page 34 (macro).
- [`finite-interval`], page 69 (class).
- [`grid-in`], page 41 (function).
- [`in-interval?`], page 41 (function).
- [`initialize-instance`], page 63 (method).
- [`interval`], page 42 (function).
- [`interval`], page 69 (class).
- [`interval-hull`], page 42 (function).
- [`interval-length`], page 42 (function).
- [`interval-midpoint`], page 42 (function).
- [`left`], page 60 (generic function).
- [`minusinf-interval`], page 70 (class).

- [num=], page 61 (method).
- [num=], page 61 (method).
- [open-left?], page 61 (generic function).
- [open-right?], page 61 (generic function).
- [plusinf-interval], page 70 (class).
- [plusminus-interval], page 45 (function).
- [print-object], page 63 (method).
- [real-line], page 70 (class).
- [relative], page 45 (function).
- [relative], page 66 (structure).
- [right], page 62 (generic function).
- [shift-interval], page 62 (generic function).
- [shrink-interval], page 46 (function).
- [spacer], page 47 (function).
- [spacer], page 66 (structure).
- [split-interval], page 47 (function).
- [subintervals-in], page 47 (function).

Internals

- [copy-relative], page 74 (function).
- [copy-spacer], page 75 (function).
- [interval/finite-left], page 87 (class).
- [interval/finite-right], page 87 (class).
- [interval/infinite-left], page 88 (class).
- [interval/infinite-right], page 88 (class).
- [print-left-endpoint], page 83 (generic function).
- [print-right-endpoint], page 83 (generic function).
- [relative-fraction], page 79 (reader).
- [relative-p], page 79 (function).
- [spacer-p], page 81 (function).
- [spacer-weight], page 81 (reader).

2.1.9 num-utils/print-matrix.lisp

Dependency
[interval.lisp], page 9 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface

- [*print-matrix-precision*], page 33 (special variable).
- [print-length-truncate], page 45 (function).
- [print-matrix], page 45 (function).

Internals [print-matrix-formatter], page 79 (function).

2.1.10 `num-utils/matrix.lisp`

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

Source [`num-utils.asd`], page 5.

Parent Component
[`num-utils`], page 3 (system).

Public Interface

- [`as-array`], page 63 (method).
- [`as-array`], page 63 (method).
- [`as-array`], page 63 (method).
- [`as-array`], page 63 (method).
- [`diagonal-matrix`], page 38 (function).
- [`diagonal-matrix`], page 64 (structure).
- [`diagonal-matrix-elements`], page 39 (reader).
- [`(setf diagonal-matrix-elements)`], page 39 (writer).
- [`diagonal-vector`], page 49 (generic function).
- [`(setf diagonal-vector)`], page 49 (generic function).
- [`dims`], page 63 (method).
- [`dims`], page 63 (method).
- [`e1-`], page 49 (method).
- [`e1-`], page 49 (method).
- [`e1-`], page 49 (method).
- [`e1-`], page 49 (method).
- [`e1/`], page 50 (method).
- [`e1/`], page 50 (method).
- [`e1/`], page 50 (method).
- [`e1/`], page 50 (method).
- [`e1log`], page 50 (method).
- [`e1log`], page 50 (method).
- [`e1log`], page 50 (method).
- [`e1log`], page 50 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).
- [`e2*`], page 51 (method).

- [e2*], page 51 (method).
- [e2*], page 51 (method).
- [e2*], page 52 (method).
- [e2+], page 52 (method).
- [e2+], page 52 (method).
- [e2+], page 52 (method).
- [e2+], page 52 (method).
- [e2+], page 52 (method).
- [e2+], page 52 (method).
- [e2-], page 53 (method).
- [e2-], page 53 (method).
- [e2-], page 53 (method).
- [e2-], page 53 (method).
- [e2-], page 53 (method).
- [e2-], page 53 (method).
- [e2/], page 53 (method).
- [e2/], page 54 (method).
- [e2/], page 54 (method).
- [e2/], page 54 (method).
- [e2/], page 54 (method).
- [e2/], page 54 (method).
- [e2/], page 54 (method).
- [e2/], page 54 (method).
- [eexp], page 57 (method).
- [eexp], page 57 (method).
- [eexp], page 57 (method).
- [eexp], page 57 (method).
- [element-type], page 63 (method).
- [element-type], page 63 (method).
- [esqrt], page 59 (method).
- [esqrt], page 59 (method).
- [esqrt], page 59 (method).
- [esqrt], page 59 (method).
- [hermitian-matrix], page 41 (function).
- [hermitian-matrix], page 65 (structure).
- [lower-triangular-matrix], page 44 (function).
- [lower-triangular-matrix], page 65 (structure).
- [map-array], page 60 (method).
- [num=], page 60 (method).
- [num=], page 60 (method).
- [print-object], page 63 (method).
- [print-object], page 64 (method).
- [print-object], page 64 (method).

- [`select`], page 64 (method).
- [`select`], page 64 (method).
- [`select`], page 64 (method).
- [`transpose`], page 62 (generic function).
- [`triangular-matrix`], page 71 (type).
- [`upper-triangular-matrix`], page 47 (function).
- [`upper-triangular-matrix`], page 68 (structure).
- [`wrapped-matrix`], page 68 (structure).
- [`wrapped-matrix-elements`], page 48 (reader).

Internals

- [`&diagonal-matrix`], page 71 (macro).
- [`&diagonal-matrix-r/o`], page 71 (macro).
- [`above-diagonal?`], page 73 (function).
- [`below-diagonal?`], page 74 (function).
- [`copy-diagonal-matrix`], page 74 (function).
- [`copy-hermitian-matrix`], page 74 (function).
- [`copy-lower-triangular-matrix`], page 74 (function).
- [`copy-upper-triangular-matrix`], page 75 (function).
- [`copy-wrapped-matrix`], page 75 (function).
- [`define-elementwise-as-array`], page 72 (macro).
- [`define-elementwise-same-class`], page 72 (macro).
- [`define-elementwise-univariate`], page 72 (macro).
- [`define-elementwise-with-constant`], page 72 (macro).
- [`define-wrapped-matrix`], page 73 (macro).
- [`diagonal-matrix-p`], page 75 (function).
- [`ensure-valid-elements`], page 75 (function).
- [`hermitian-matrix-elements`], page 75 (function).
- [`hermitian-matrix-p`], page 76 (function).
- [`lower-triangular-matrix-elements`], page 77 (function).
- [`lower-triangular-matrix-p`], page 77 (function).
- [`make-diagonal-matrix`], page 77 (function).
- [`make-hermitian-matrix`], page 77 (function).
- [`make-lower-triangular-matrix`], page 77 (function).
- [`make-upper-triangular-matrix`], page 77 (function).
- [`make-wrapped-matrix`], page 77 (function).
- [`upper-triangular-matrix-elements`], page 82 (function).
- [`upper-triangular-matrix-p`], page 82 (function).
- [`valid-sparse-type?`], page 82 (function).
- [`wrapped-matrix-p`], page 82 (function).
- [`zero-like`], page 83 (function).

2.1.11 num-utils/matrix-shorthand.lisp

Dependency
[matrix.lisp], page 11 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface
- [diagonal-mx], page 39 (function).
- [hermitian-mx], page 34 (macro).
- [lower-triangular-mx], page 35 (macro).
- [mx], page 35 (macro).
- [upper-triangular-mx], page 35 (macro).
- [vec], page 48 (function).

Internals [pad-left-expansion], page 79 (function).

2.1.12 num-utils/chebyshev.lisp

Dependency
[matrix-shorthand.lisp], page 14 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface
- [chebyshev-approximate], page 37 (function).
- [chebyshev-regression], page 38 (function).
- [chebyshev-root], page 38 (function).
- [chebyshev-roots], page 38 (function).
- [evaluate-chebyshev], page 40 (function).

Internals
- [ab-to-cd-intercept-slope], page 73 (function).
- [ab-to-cinf], page 73 (function).
- [chebyshev-approximate-implementation], page 83 (generic function).
- [chebyshev-recursion], page 74 (function).
- [cinf-to-ab], page 74 (function).

2.1.13 num-utils/polynomial.lisp

Dependency
[chebyshev.lisp], page 14 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface
- [evaluate-polynomial], page 40 (function).
- [evaluate-rational], page 40 (function).

2.1.14 num-utils/rootfinding.lisp

Dependency
[polynomial.lisp], page 14 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface

- [*rootfinding-delta-relative*], page 33 (special variable).
- [*rootfinding-epsilon*], page 33 (special variable).
- [root-bisection], page 46 (function).

Internals

- [narrow-bracket?], page 79 (function).
- [near-root?], page 79 (function).
- [opposite-sign?], page 79 (function).
- [rootfinding-delta], page 80 (function).
- [univariate-rootfinder-loop%], page 73 (macro).

2.1.15 num-utils/quadrature.lisp

Dependency
[rootfinding.lisp], page 15 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface
[romberg-quadrature], page 46 (function).

Internals

- [copy-iterative-quadrature], page 74 (function).
- [copy-midpoint-quadrature], page 74 (function).
- [copy-richardson-extrapolation], page 75 (function).
- [copy-trapezoidal-quadrature], page 75 (function).
- [iterative-quadrature], page 84 (structure).
- [iterative-quadrature-a], page 76 (reader).
- [(setf iterative-quadrature-a)], page 76 (writer).
- [iterative-quadrature-b], page 76 (reader).
- [(setf iterative-quadrature-b)], page 76 (writer).
- [iterative-quadrature-f], page 76 (reader).
- [(setf iterative-quadrature-f)], page 76 (writer).
- [iterative-quadrature-h], page 76 (reader).
- [(setf iterative-quadrature-h)], page 76 (writer).
- [iterative-quadrature-n], page 76 (reader).
- [(setf iterative-quadrature-n)], page 76 (writer).
- [iterative-quadrature-p], page 76 (function).

- [`iterative-quadrature-sum`], page 76 (reader).
- [`(setf iterative-quadrature-sum)`], page 76 (writer).
- [`make-iterative-quadrature`], page 77 (function).
- [`midpoint-quadrature`], page 77 (function).
- [`midpoint-quadrature`], page 86 (structure).
- [`midpoint-quadrature%`], page 77 (function).
- [`midpoint-quadrature-a`], page 78 (function).
- [`(setf midpoint-quadrature-a)`], page 78 (function).
- [`midpoint-quadrature-b`], page 78 (function).
- [`(setf midpoint-quadrature-b)`], page 78 (function).
- [`midpoint-quadrature-f`], page 78 (function).
- [`(setf midpoint-quadrature-f)`], page 78 (function).
- [`midpoint-quadrature-h`], page 78 (function).
- [`(setf midpoint-quadrature-h)`], page 78 (function).
- [`midpoint-quadrature-n`], page 78 (function).
- [`(setf midpoint-quadrature-n)`], page 78 (function).
- [`midpoint-quadrature-p`], page 78 (function).
- [`midpoint-quadrature-sum`], page 78 (function).
- [`(setf midpoint-quadrature-sum)`], page 79 (function).
- [`refine-quadrature`], page 84 (generic function).
- [`richardson-coefficient`], page 84 (generic function).
- [`richardson-extrapolation`], page 79 (function).
- [`richardson-extrapolation`], page 86 (structure).
- [`richardson-extrapolation-coefficient`], page 80 (reader).
- [`(setf richardson-extrapolation-coefficient)`], page 80 (writer).
- [`richardson-extrapolation-diagonal`], page 80 (reader).
- [`(setf richardson-extrapolation-diagonal)`], page 80 (writer).
- [`richardson-extrapolation-n`], page 80 (reader).
- [`(setf richardson-extrapolation-n)`], page 80 (writer).
- [`richardson-extrapolation-p`], page 80 (function).
- [`richardson-iteration`], page 80 (function).
- [`romberg-quadrature%`], page 80 (function).
- [`transformed-quadrature`], page 84 (generic function).
- [`trapezoidal-quadrature`], page 81 (function).
- [`trapezoidal-quadrature`], page 86 (structure).
- [`trapezoidal-quadrature%`], page 81 (function).
- [`trapezoidal-quadrature-a`], page 81 (function).
- [`(setf trapezoidal-quadrature-a)`], page 81 (function).
- [`trapezoidal-quadrature-b`], page 81 (function).
- [`(setf trapezoidal-quadrature-b)`], page 81 (function).
- [`trapezoidal-quadrature-f`], page 81 (function).
- [`(setf trapezoidal-quadrature-f)`], page 82 (function).
- [`trapezoidal-quadrature-h`], page 82 (function).

- [(setf trapezoidal-quadrature-h)], page 82 (function).
- [trapezoidal-quadrature-n], page 82 (function).
- [(setf trapezoidal-quadrature-n)], page 82 (function).
- [trapezoidal-quadrature-p], page 82 (function).
- [trapezoidal-quadrature-sum], page 82 (function).
- [(setf trapezoidal-quadrature-sum)], page 82 (function).

2.1.16 num-utils/log-exp.lisp

Dependency
[quadrature.lisp], page 15 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface
- [exp-1], page 40 (function).
- [exp-1/x], page 40 (function).
- [expt-1], page 41 (function).
- [hypot], page 41 (function).
- [log1+], page 42 (function).
- [log1+/x], page 43 (function).
- [log1+exp], page 43 (function).
- [log1-], page 43 (function).
- [log1-exp], page 43 (function).
- [log1pmx], page 43 (function).
- [log2-exp], page 43 (function).
- [logexp-1], page 43 (function).

2.1.17 num-utils/test-utilities.lisp

Dependency
[log-exp.lisp], page 17 (file).

Source [num-utils.asd], page 5.

Parent Component
[num-utils], page 3 (system).

Public Interface
- [compare-fns], page 38 (function).
- [compare-vectors], page 38 (function).
- [max-error], page 44 (reader).
- [(setf max-error)], page 44 (writer).
- [mean-error], page 44 (reader).
- [(setf mean-error)], page 44 (writer).
- [min-error], page 44 (reader).
- [(setf min-error)], page 44 (writer).
- [rms], page 45 (reader).

- [`(setf rms)`], page 45 (writer).
- [`test-count`], page 47 (reader).
- [`(setf test-count)`], page 47 (writer).
- [`test-fn`], page 47 (function).
- [`test-results`], page 67 (structure).
- [`variance0`], page 48 (reader).
- [`(setf variance0)`], page 48 (writer).
- [`variance1`], page 48 (reader).
- [`(setf variance1)`], page 48 (writer).
- [`worst-case`], page 48 (reader).
- [`(setf worst-case)`], page 48 (writer).

Internals

- [`copy-test-results`], page 75 (function).
- [`make-test-results`], page 77 (function).
- [`test-results-p`], page 81 (function).

2.1.18 `num-utils/pkgdcl.lisp`

Dependency
[`test-utilities.lisp`], page 17 (file).

Source [`num-utils.asd`], page 5.

Parent Component
[`num-utils`], page 3 (system).

Packages [`num-utils`], page 31.

3 Packages

Packages are listed by definition order.

3.1 num-utils.quadrature

Source [packages.lisp], page 5.

Use List

- alexandria.
- alexandria+.
- anaphora.
- common-lisp.
- let-plus.
- [num-utils.arithmetic], page 28.
- [num-utils.interval], page 22.
- [num-utils.utilities], page 26.

Used By List

[num-utils], page 31.

Public Interface

[romberg-quadrature], page 46 (function).

Internals

- [copy-iterative-quadrature], page 74 (function).
- [copy-midpoint-quadrature], page 74 (function).
- [copy-richardson-extrapolation], page 75 (function).
- [copy-trapezoidal-quadrature], page 75 (function).
- [iterative-quadrature], page 84 (structure).
- [iterative-quadrature-a], page 76 (reader).
- [(setf iterative-quadrature-a)], page 76 (writer).
- [iterative-quadrature-b], page 76 (reader).
- [(setf iterative-quadrature-b)], page 76 (writer).
- [iterative-quadrature-f], page 76 (reader).
- [(setf iterative-quadrature-f)], page 76 (writer).
- [iterative-quadrature-h], page 76 (reader).
- [(setf iterative-quadrature-h)], page 76 (writer).
- [iterative-quadrature-n], page 76 (reader).
- [(setf iterative-quadrature-n)], page 76 (writer).
- [iterative-quadrature-p], page 76 (function).
- [iterative-quadrature-sum], page 76 (reader).
- [(setf iterative-quadrature-sum)], page 76 (writer).
- [make-iterative-quadrature], page 77 (function).
- [midpoint-quadrature], page 77 (function).
- [midpoint-quadrature], page 86 (structure).
- [midpoint-quadrature%], page 77 (function).

- [midpoint-quadrature-a], page 78 (function).
- [(setf midpoint-quadrature-a)], page 78 (function).
- [midpoint-quadrature-b], page 78 (function).
- [(setf midpoint-quadrature-b)], page 78 (function).
- [midpoint-quadrature-f], page 78 (function).
- [(setf midpoint-quadrature-f)], page 78 (function).
- [midpoint-quadrature-h], page 78 (function).
- [(setf midpoint-quadrature-h)], page 78 (function).
- [midpoint-quadrature-n], page 78 (function).
- [(setf midpoint-quadrature-n)], page 78 (function).
- [midpoint-quadrature-p], page 78 (function).
- [midpoint-quadrature-sum], page 78 (function).
- [(setf midpoint-quadrature-sum)], page 79 (function).
- [refine-quadrature], page 84 (generic function).
- [richardson-coefficient], page 84 (generic function).
- [richardson-extrapolation], page 79 (function).
- [richardson-extrapolation], page 86 (structure).
- [richardson-extrapolation-coefficient], page 80 (reader).
- [(setf richardson-extrapolation-coefficient)], page 80 (writer).
- [richardson-extrapolation-diagonal], page 80 (reader).
- [(setf richardson-extrapolation-diagonal)], page 80 (writer).
- [richardson-extrapolation-n], page 80 (reader).
- [(setf richardson-extrapolation-n)], page 80 (writer).
- [richardson-extrapolation-p], page 80 (function).
- [richardson-iteration], page 80 (function).
- [romberg-quadrature%], page 80 (function).
- [transformed-quadrature], page 84 (generic function).
- [trapezoidal-quadrature], page 81 (function).
- [trapezoidal-quadrature], page 86 (structure).
- [trapezoidal-quadrature%], page 81 (function).
- [trapezoidal-quadrature-a], page 81 (function).
- [(setf trapezoidal-quadrature-a)], page 81 (function).
- [trapezoidal-quadrature-b], page 81 (function).
- [(setf trapezoidal-quadrature-b)], page 81 (function).
- [trapezoidal-quadrature-f], page 81 (function).
- [(setf trapezoidal-quadrature-f)], page 82 (function).
- [trapezoidal-quadrature-h], page 82 (function).
- [(setf trapezoidal-quadrature-h)], page 82 (function).
- [trapezoidal-quadrature-n], page 82 (function).
- [(setf trapezoidal-quadrature-n)], page 82 (function).
- [trapezoidal-quadrature-p], page 82 (function).
- [trapezoidal-quadrature-sum], page 82 (function).
- [(setf trapezoidal-quadrature-sum)], page 82 (function).

3.2 num-utils.print-matrix

Source [packages.lisp], page 5.

Use List

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

Used By List

- lisp-stat.
- [num-utils.matrix], page 24.

Public Interface

- [*print-matrix-precision*], page 33 (special variable).
- [print-length-truncate], page 45 (function).
- [print-matrix], page 45 (function).

Internals [print-matrix-formatter], page 79 (function).

3.3 num-utils.chebyshev

Source [packages.lisp], page 5.

Use List

- alexandria.
- anaphora.
- common-lisp.
- let-plus.
- [num-utils.interval], page 22.
- [num-utils.utilities], page 26.

Used By List
[num-utils], page 31.

Public Interface

- [chebyshev-approximate], page 37 (function).
- [chebyshev-regression], page 38 (function).
- [chebyshev-root], page 38 (function).
- [chebyshev-roots], page 38 (function).
- [evaluate-chebyshev], page 40 (function).

Internals

- [ab-to-cd-intercept-slope], page 73 (function).
- [ab-to-cinf], page 73 (function).
- [chebyshev-approximate-implementation], page 83 (generic function).
- [chebyshev-recursion], page 74 (function).
- [cinf-to-ab], page 74 (function).

3.4 num-utils.interval

Source [packages.lisp], page 5.

Use List

- alexandria.
- anaphora.
- common-lisp.
- let-plus.
- [num-utils.num=], page 31.
- [num-utils.utilities], page 26.

Used By List

- [num-utils], page 31.
- [num-utils.chebyshev], page 21.
- [num-utils.quadrature], page 19.
- [num-utils.rootfinding], page 32.

Public Interface

- [&interval], page 33 (macro).
- [extend-interval], page 59 (generic function).
- [extendf-interval], page 34 (macro).
- [finite-interval], page 69 (class).
- [grid-in], page 41 (function).
- [in-interval?], page 41 (function).
- [interval], page 42 (function).
- [interval], page 69 (class).
- [interval-hull], page 42 (function).
- [interval-length], page 42 (function).
- [interval-midpoint], page 42 (function).
- [left], page 60 (generic function).
- [minusinf-interval], page 70 (class).
- [open-left?], page 61 (generic function).
- [open-right?], page 61 (generic function).
- [plusinf-interval], page 70 (class).
- [plusminus-interval], page 45 (function).
- [real-line], page 70 (class).
- [relative], page 45 (function).
- [relative], page 66 (structure).
- [right], page 62 (generic function).
- [shift-interval], page 62 (generic function).
- [shrink-interval], page 46 (function).
- [spacer], page 47 (function).
- [spacer], page 66 (structure).
- [split-interval], page 47 (function).
- [subintervals-in], page 47 (function).

Internals

- [copy-relative], page 74 (function).
- [copy-spacer], page 75 (function).
- [interval/finite-left], page 87 (class).
- [interval/finite-right], page 87 (class).
- [interval/infinite-left], page 88 (class).
- [interval/infinite-right], page 88 (class).
- [print-left-endpoint], page 83 (generic function).
- [print-right-endpoint], page 83 (generic function).
- [relative-fraction], page 79 (reader).
- [relative-p], page 79 (function).
- [spacer-p], page 81 (function).
- [spacer-weight], page 81 (reader).

3.5 num-utils.matrix-shorthand

Source [packages.lisp], page 5.

Nickname nu.mx

Use List

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

Public Interface

- [diagonal-mx], page 39 (function).
- [hermitian-mx], page 34 (macro).
- [lower-triangular-mx], page 35 (macro).
- [mx], page 35 (macro).
- [upper-triangular-mx], page 35 (macro).
- [vec], page 48 (function).

Internals [pad-left-expansion], page 79 (function).

3.6 num-utils.log-exp

Source [packages.lisp], page 5.

Use List

- common-lisp.
- let-plus.

Used By List

[num-utils], page 31.

Public Interface

- [exp-1], page 40 (function).

- [exp-1/x], page 40 (function).
- [expt-1], page 41 (function).
- [hypot], page 41 (function).
- [log1+], page 42 (function).
- [log1+/x], page 43 (function).
- [log1+exp], page 43 (function).
- [log1-], page 43 (function).
- [log1-exp], page 43 (function).
- [log1pmx], page 43 (function).
- [log2-exp], page 43 (function).
- [logexp-1], page 43 (function).

3.7 num-utils.matrix

Source [packages.lisp], page 5.

Use List

- alexandria.
- anaphora.
- common-lisp.
- let-plus.
- [num-utils.elementwise], page 29.
- [num-utils.num=], page 31.
- [num-utils.print-matrix], page 21.
- [num-utils.utilities], page 26.
- select.

Used By List

- distributions.
- [num-utils], page 31.
- [num-utils.matrix-shorthand], page 23.

Public Interface

- [diagonal-matrix], page 38 (function).
- [diagonal-matrix], page 64 (structure).
- [diagonal-matrix-elements], page 39 (reader).
- [(setf diagonal-matrix-elements)], page 39 (writer).
- [diagonal-vector], page 49 (generic function).
- [(setf diagonal-vector)], page 49 (generic function).
- [hermitian-matrix], page 41 (function).
- [hermitian-matrix], page 65 (structure).
- [lower-triangular-matrix], page 44 (function).
- [lower-triangular-matrix], page 65 (structure).
- [map-array], page 60 (generic function).
- [transpose], page 62 (generic function).
- [triangular-matrix], page 71 (type).

- [upper-triangular-matrix], page 47 (function).
- [upper-triangular-matrix], page 68 (structure).
- [wrapped-matrix], page 68 (structure).
- [wrapped-matrix-elements], page 48 (reader).

Internals

- [&diagonal-matrix], page 71 (macro).
- [&diagonal-matrix-r/o], page 71 (macro).
- [above-diagonal?], page 73 (function).
- [below-diagonal?], page 74 (function).
- [copy-diagonal-matrix], page 74 (function).
- [copy-hermitian-matrix], page 74 (function).
- [copy-lower-triangular-matrix], page 74 (function).
- [copy-upper-triangular-matrix], page 75 (function).
- [copy-wrapped-matrix], page 75 (function).
- [define-elementwise-as-array], page 72 (macro).
- [define-elementwise-same-class], page 72 (macro).
- [define-elementwise-univariate], page 72 (macro).
- [define-elementwise-with-constant], page 72 (macro).
- [define-wrapped-matrix], page 73 (macro).
- [diagonal-matrix-p], page 75 (function).
- [ensure-valid-elements], page 75 (function).
- [hermitian-matrix-elements], page 75 (function).
- [hermitian-matrix-p], page 76 (function).
- [lower-triangular-matrix-elements], page 77 (function).
- [lower-triangular-matrix-p], page 77 (function).
- [make-diagonal-matrix], page 77 (function).
- [make-hermitian-matrix], page 77 (function).
- [make-lower-triangular-matrix], page 77 (function).
- [make-upper-triangular-matrix], page 77 (function).
- [make-wrapped-matrix], page 77 (function).
- [upper-triangular-matrix-elements], page 82 (function).
- [upper-triangular-matrix-p], page 82 (function).
- [valid-sparse-type?], page 82 (function).
- [wrapped-matrix-p], page 82 (function).
- [zero-like], page 83 (function).

3.8 num-utils.extended-real

Source [packages.lisp], page 5.

Nickname xreal

Use List

- alexandria.
- common-lisp.

Public Interface

- [<], page 36 (function).
- [<=], page 36 (function).
- [=], page 36 (function).
- [>], page 36 (function).
- [>=], page 36 (function).
- [`extended-real`], page 71 (type).
- [`infinite?`], page 42 (function).
- [`lambda-template`], page 34 (macro).
- [`with-template`], page 35 (macro).

Internals

- [`define-comparison`], page 72 (macro).
- [`extend-pairwise-comparison`], page 75 (function).
- [`infinite`], page 88 (type).

3.9 `num-utils.utilities`

A collection of utilities to work with floating point values. Optimised for double-float.

Source [`packages.lisp`], page 5.

Use List

- `alexandria`.
- `anaphora`.
- `common-lisp`.
- `let-plus`.

Used By List

- `lisp-stat`.
- `nu.statistics`.
- [`num-utils`], page 31.
- [`num-utils.arithmetic`], page 28.
- [`num-utils.chebyshev`], page 21.
- [`num-utils.elementwise`], page 29.
- [`num-utils.interval`], page 22.
- [`num-utils.matrix`], page 24.
- [`num-utils.matrix-shorthand`], page 23.
- [`num-utils.polynomial`], page 31.
- [`num-utils.quadrature`], page 19.
- [`num-utils.rootfinding`], page 32.

Public Interface

- [`as-alist`], page 48 (generic function).
- [`as-bit-vector`], page 36 (function).
- [`as-double-float`], page 37 (function).
- [`as-plist`], page 49 (generic function).
- [`as-simple-fixnum-vector`], page 37 (function).

- [bic], page 37 (function).
- [binary-search], page 37 (function).
- [check-types], page 33 (macro).
- [curry*], page 33 (macro).
- [define-with-multiple-bindings], page 34 (macro).
- [distinct], page 39 (function).
- [expanding], page 34 (macro).
- [fixnum?], page 41 (function).
- [generate-sequence], page 41 (function).
- [gethash*], page 34 (macro).
- [make-vector], page 36 (compiler macro).
- [make-vector], page 44 (function).
- [monotonicp], page 44 (function).
- [simple-boolean-vector], page 71 (type).
- [simple-double-float-vector], page 71 (type).
- [simple-fixnum-vector], page 71 (type).
- [simple-single-float-vector], page 71 (type).
- [splice-awhen], page 35 (macro).
- [splice-when], page 35 (macro).
- [with-double-floats], page 35 (macro).
- [within?], page 48 (function).

Internals

- [boolean-sequence-p], page 74 (function).
- [boolean?], page 74 (function).

3.10 `num-utils.test-utilities`

Source [packages.lisp], page 5.

Use List `common-lisp`.

Used By List
[num-utils], page 31.

Public Interface

- [compare-fns], page 38 (function).
- [compare-vectors], page 38 (function).
- [max-error], page 44 (reader).
- [(setf max-error)], page 44 (writer).
- [mean-error], page 44 (reader).
- [(setf mean-error)], page 44 (writer).
- [min-error], page 44 (reader).
- [(setf min-error)], page 44 (writer).
- [rms], page 45 (reader).
- [(setf rms)], page 45 (writer).
- [test-count], page 47 (reader).

- [(setf test-count)], page 47 (writer).
- [test-fn], page 47 (function).
- [test-results], page 67 (structure).
- [variance0], page 48 (reader).
- [(setf variance0)], page 48 (writer).
- [variance1], page 48 (reader).
- [(setf variance1)], page 48 (writer).
- [worst-case], page 48 (reader).
- [(setf worst-case)], page 48 (writer).

Internals

- [copy-test-results], page 75 (function).
- [make-test-results], page 77 (function).
- [test-results-p], page 81 (function).

3.11 num-utils.arithmetic

Source [packages.lisp], page 5.

Use List

- alexandria+.
- alexandria-2.
- anaphora.
- common-lisp.
- let-plus.
- [num-utils.utilities], page 26.

Used By List

- lisp-stat.
- nu.statistics.
- [num-utils], page 31.
- [num-utils.elementwise], page 29.
- [num-utils.quadrature], page 19.
- special-functions.

Public Interface

- [1c], page 36 (function).
- [abs-diff], page 36 (function).
- [absolute-square], page 36 (function).
- [as-integer], page 37 (function).
- [ceiling*], page 37 (function).
- [cube], page 38 (function).
- [cumulative-product], page 38 (function).
- [cumulative-sum], page 38 (function).
- [divides?], page 39 (function).
- [floor*], page 41 (function).
- [ivec], page 42 (function).

- [l2norm], page 42 (function).
- [l2norm-square], page 59 (generic function).
- [log10], page 43 (function).
- [log2], page 43 (function).
- [normalize-probabilities], page 44 (function).
- [numseq], page 45 (function).
- [product], page 61 (generic function).
- [round*], page 46 (function).
- [same-sign-p], page 46 (function).
- [sequence-maximum], page 46 (function).
- [sequence-minimum], page 46 (function).
- [square], page 47 (function).
- [sum], page 85 (slot).
- [sum], page 62 (generic function).
- [truncate*], page 47 (function).

Internals

- [define-rounding-with-offset], page 72 (macro).
- [ln], page 77 (function).
- [similar-element-type], page 80 (function).
- [similar-sequence-type], page 81 (function).

3.12 num-utils.elementwise

Source [packages.lisp], page 5.

Nickname elmt

Use List

- alexandria.
- common-lisp.
- let-plus.
- [num-utils.arithmetic], page 28.
- [num-utils.utilities], page 26.

Used By List

- distributions.
- lisp-stat.
- [num-utils], page 31.
- [num-utils.matrix], page 24.

Public Interface

- [e*], page 39 (function).
- [e+], page 39 (function).
- [e-], page 39 (function).
- [e/], page 39 (function).
- [e1-], page 49 (generic function).
- [e1/], page 50 (generic function).

- [`e1log`], page 50 (generic function).
- [`e2*`], page 50 (generic function).
- [`e2+`], page 52 (generic function).
- [`e2-`], page 53 (generic function).
- [`e2/`], page 53 (generic function).
- [`e2<`], page 54 (generic function).
- [`e2<=`], page 55 (generic function).
- [`e2=`], page 55 (generic function).
- [`e2>`], page 55 (generic function).
- [`e2>=`], page 56 (generic function).
- [`e2log`], page 56 (generic function).
- [`eceiling`], page 56 (generic function).
- [`econjugate`], page 56 (generic function).
- [`ecos`], page 57 (generic function).
- [`eexp`], page 57 (generic function).
- [`eexpt`], page 57 (generic function).
- [`efloor`], page 58 (generic function).
- [`elementwise-float-contagion`], page 39 (function).
- [`elog`], page 40 (function).
- [`emax`], page 40 (function).
- [`emin`], page 40 (function).
- [`emod`], page 58 (generic function).
- [`ereduce`], page 58 (generic function).
- [`esin`], page 58 (generic function).
- [`esqrt`], page 59 (generic function).

Internals

- [`define-e&`], page 72 (macro).
- [`define-e1`], page 72 (macro).
- [`define-e2`], page 72 (macro).
- [`define-elementwise-reduction`], page 72 (macro).
- [`esquare`], page 83 (generic function).
- [`mapping-array`], page 73 (macro).

3.13 `num-utils.num=`

Source [`packages.lisp`], page 5.

Use List

- `alexandria`.
- `anaphora`.
- `common-lisp`.
- `let-plus`.

Used By List

- `distributions`.

- `nu.statistics`.
- [`num-utils`], page 31.
- [`num-utils.interval`], page 22.
- [`num-utils.matrix`], page 24.

Public Interface

- [`*num=-tolerance*`], page 33 (special variable).
- [`define-num=-with-accessors`], page 34 (macro).
- [`define-structure-num=`], page 34 (macro).
- [`num-delta`], page 44 (function).
- [`num=`], page 60 (generic function).
- [`num=-function`], page 45 (function).

3.14 `num-utils.polynomial`

Efficient evaluation of polynomial functions using Horner's method

Source [`packages.lisp`], page 5.

Nickname `poly`

Use List

- `alexandria`.
- `common-lisp`.
- [`num-utils.utilities`], page 26.

Used By List

- [`num-utils`], page 31.
- `special-functions`.

Public Interface

- [`evaluate-polynomial`], page 40 (function).
- [`evaluate-rational`], page 40 (function).

3.15 `num-utils`

Numerical utilities for Lisp-Stat

Source [`pkgdcl.lisp`], page 18.

Nickname `nu`

Use List

- `common-lisp`.
- [`num-utils.arithmetic`], page 28.
- [`num-utils.chebyshev`], page 21.
- [`num-utils.elementwise`], page 29.
- [`num-utils.interval`], page 22.
- [`num-utils.log-exp`], page 24.
- [`num-utils.matrix`], page 24.
- [`num-utils.num=`], page 31.
- [`num-utils.polynomial`], page 31.

- [`num-utils.quadrature`], page 19.
- [`num-utils.rootfinding`], page 32.
- [`num-utils.test-utilities`], page 27.
- [`num-utils.utilities`], page 26.

3.16 `num-utils.rootfinding`

Source [`packages.lisp`], page 5.

Use List

- `alexandria`.
- `common-lisp`.
- `let-plus`.
- [`num-utils.interval`], page 22.
- [`num-utils.utilities`], page 26.

Used By List

[`num-utils`], page 31.

Public Interface

- [`*rootfinding-delta-relative*`], page 33 (special variable).
- [`*rootfinding-epsilon*`], page 33 (special variable).
- [`root-bisection`], page 46 (function).

Internals

- [`narrow-bracket?`], page 79 (function).
- [`near-root?`], page 79 (function).
- [`opposite-sign?`], page 79 (function).
- [`rootfinding-delta`], page 80 (function).
- [`univariate-rootfinder-loop%`], page 73 (macro).

4 Definitions

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

4.1 Public Interface

4.1.1 Special variables

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

Default tolerance for NUM=.

Package [`num-utils.num=`], page 31.

Source [`num=.lisp`], page 6.

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

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

Package [`num-utils.print-matrix`], page 21.

Source [`print-matrix.lisp`], page 10.

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

Default relative interval width for rootfinding.

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

Source [`rootfinding.lisp`], page 15.

`*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 15.

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 22.

Source [`interval.lisp`], page 9.

`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 26.

Source [`utilities.lisp`], page 5.

`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 26.

Source [`utilities.lisp`], page 5.

`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 31.

Source [`num=.lisp`], page 6.

`define-structure-num=` (*structure* **&rest** *slots*) [Macro]
Define a NUM= method for the given structure, comparing the given slots.

Package [`num-utils.num=`], page 31.

Source [`num=.lisp`], page 6.

`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 26.

Source [`utilities.lisp`], page 5.

`expanding` (**&body** *body*) [Macro]
Expand BODY. Useful for generating code programmatically.

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

Source [`utilities.lisp`], page 5.

`extendf-interval` (*place object*) [Macro]
Apply EXTEND-INTERVAL on PLACE using OBJECT.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`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 26.

Source [`utilities.lisp`], page 5.

`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 23.

Source [`matrix-shorthand.lisp`], page 14.

`lambda-template` ((*prefix* **&rest** *variables)* **&body** *body*) [Macro]
LAMBDA with WITH-TEMPLATE in its BODY.

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`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 23.

Source [`matrix-shorthand.lisp`], page 14.

`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 23.

Source [`matrix-shorthand.lisp`], page 14.

`splice-awhen` (*test* **&body** *forms*) [Macro]

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

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

Source [`utilities.lisp`], page 5.

`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 26.

Source [`utilities.lisp`], page 5.

`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 23.

Source [`matrix-shorthand.lisp`], page 14.

`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 26.

Source [`utilities.lisp`], page 5.

`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 26.

Source [`extended-real.lisp`], page 9.

4.1.3 Compiler macros

`make-vector` (*element-type* **&rest** *initial-contents*) [Compiler Macro]

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

Source [`utilities.lisp`], page 5.

4.1.4 Ordinary functions

`1c` (*number*) [Function]

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

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`<` (*number* **&rest** *more-numbers*) [Function]

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`<=` (*number* **&rest** *more-numbers*) [Function]

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`=` (*number* **&rest** *more-numbers*) [Function]

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`>` (*number* **&rest** *more-numbers*) [Function]

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`>=` (*number* **&rest** *more-numbers*) [Function]

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`abs-diff` (*a b*) [Function]

Absolute difference of A and B.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`absolute-square` (*number*) [Function]

Number multiplied by its complex conjugate.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`as-bit-vector` (*v*) [Function]

Return a bit vector where each non-nil element of V is mapped to 1 and each NIL element is mapped to 0

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

Source [`utilities.lisp`], page 5.

`as-double-float` (*x*) [Function]

Convert argument to DOUBLE-FLOAT.

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

Source [`utilities.lisp`], page 5.

`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 28.

Source [`arithmetic.lisp`], page 7.

`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 26.

Source [`utilities.lisp`], page 5.

`bic` (*a b*) [Function]

Biconditional. Returns A <=> B.

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

Source [`utilities.lisp`], page 5.

`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 26.

Source [`utilities.lisp`], page 5.

`ceiling*` (*number* **&optional** *divisor offset*) [Function]

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

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`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 21.

Source [`chebyshev.lisp`], page 14.

`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 21.

Source [`chebyshev.lisp`], page 14.

`chebyshev-root` (*m i*) [Function]

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

Package [`num-utils.chebyshev`], page 21.

Source [`chebyshev.lisp`], page 14.

`chebyshev-roots` (*m*) [Function]

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

Package [`num-utils.chebyshev`], page 21.

Source [`chebyshev.lisp`], page 14.

`compare-fns` (*fn-1 fn-2* **&rest** *fn-params*) [Function]

Compare the values returned by two functions

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

`compare-vectors` (*reference-values computed-values*) [Function]

Compare two vectors containing the results of previous computations

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

`cube` (*number*) [Function]

Cube of number.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`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 28.

Source [`arithmetic.lisp`], page 7.

`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 28.

Source [`arithmetic.lisp`], page 7.

`diagonal-matrix` (*elements*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`diagonal-matrix-elements` (*instance*) [Reader]
`(setf diagonal-matrix-elements)` (*instance*) [Writer]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

Target Slot
[`elements`], page 65.

`diagonal-mx` (*element-type* **&rest** *elements*) [Function]
Return a DIAGONAL-MATRIX with elements coerced to ELEMENT-TYPE.

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

Source [`matrix-shorthand.lisp`], page 14.

`distinct` (*column*) [Function]
Returns the number of distinct elements in COLUMN, a symbol naming a variable. Useful for formatting columns for human output.

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

Source [`utilities.lisp`], page 5.

`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 28.

Source [`arithmetic.lisp`], page 7.

`e*` (*argument* **&rest** *more-arguments*) [Function]
Elementwise *.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`e+` (*argument* **&rest** *more-arguments*) [Function]
Elementwise +.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`e-` (*argument* **&rest** *more-arguments*) [Function]
Elementwise -.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`e/` (*argument* **&rest** *more-arguments*) [Function]
Elementwise /.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`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 29.

Source [`elementwise.lisp`], page 7.

`elog` (*a* **&optional** *base*) [Function]

Elementwise logarithm.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`emax` (*object*) [Function]

Elementwise MAX.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`emin` (*object*) [Function]

Elementwise MIN.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`evaluate-chebyshev` (*coefficients x*) [Function]

Return the sum of Chebyshev polynomials, weighted by COEFFICIENTS, at X.

Package [`num-utils.chebyshev`], page 21.

Source [`chebyshev.lisp`], page 14.

`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 31.

Source [`polynomial.lisp`], page 14.

`evaluate-rational` (*numerator denominator z*) [Function]

Evaluate a rational function using Horner's method. NUMERATOR and DENOMINATOR must be equal in size. These always have a loop and so may be less efficient than evaluating a pair of polynomials. However, there are some tricks we can use to prevent overflow that might otherwise occur in polynomial evaluation if z is large. This is important in our Lanczos code for example.

N.B. The order of coefficients for this function is NOT the same as evaluate-polynomial.

Package [`num-utils.polynomial`], page 31.

Source [`polynomial.lisp`], page 14.

`exp-1` (*x*) [Function]

Compute (- (exp x) 1) stably even when X is near 0

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`exp-1/x` (*x*) [Function]

Compute (/ (- (exp x) 1) x) stably even when X is near zero.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`expt-1` (*a z*) [Function]

Compute (a^z)-1 stably even when A is close to 1 or Z is close to zero.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`fixnum?` (*object*) [Function]

Check of type of OBJECT is fixnum.

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

Source [`utilities.lisp`], page 5.

`floor*` (*number* **&optional** *divisor offset*) [Function]

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

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`generate-sequence` (*result-type size function*) [Function]

Like MAKE-SEQUENCE, but using a function to fill the result.

Example to create a sequence of random numbers between 0-1 from the uniform distribution:
(generate-sequence '(vector double-float) 100 (lambda () (random 1.0))).
Essentially the initial values are ignored when using this function.
See also: aops:generate

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

Source [`utilities.lisp`], page 5.

`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 22.

Source [`interval.lisp`], page 9.

`hermitian-matrix` (*elements*) [Function]

Create a lower-triangular-matrix.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`hypot` (*x y*) [Function]

Compute the hypotenuse of X and Y without danger of floating-point overflow or underflow.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`in-interval?` (*interval number*) [Function]

Test if NUMBER is in INTERVAL (which can be NIL, designating the empty set).

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`infinite?` (*object*) [Function]

Test if an object represents positive or negative infinity.

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`interval` (*left right* **&key** *open-left? open-right?*) [Function]

Create an INTERVAL.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`interval-hull` (*object*) [Function]

Return the smallest connected interval that contains (elements in) OBJECT.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`interval-length` (*interval*) [Function]

Difference between left and right.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`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 22.

Source [`interval.lisp`], page 9.

`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 28.

Source [`arithmetic.lisp`], page 7.

`l2norm` (*object*) [Function]

L_2 norm of OBJECT.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`log1+` (*x*) [Function]

Compute (log (1+ x)) stably even when X is near 0.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`log1+/x` (*x*) [Function]

Compute (/ (log (+ 1 x)) x) stably even when X is near zero.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`log1+exp` (*a*) [Function]

Accurately compute log(1+exp(x)) even when A is near zero.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`log1-` (*x*) [Function]

Compute (log (- 1 x)) stably even when X is near zero.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`log1-exp` (*a*) [Function]

Compute log(1-exp(x)) stably even when A is near zero.
This is sometimes known as the E_3, the third Einstein function.
See Mächler 2008 for notes on accurate calculation. https://cran.r-project.org/web/packages/Rmpfr/vignettes/log1mexp-note.pdf

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`log10` (*number*) [Function]

Abbreviation for decimal logarithm.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`log1pmx` (*x*) [Function]

Compute (- (log (1+ x)) x)
Accuracy within ~2ulps for -0.227 < x < 0.315

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`log2` (*number*) [Function]

Abbreviation for binary logarithm.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`log2-exp` (*x*) [Function]

Compute log(2-exp(x)) stably even when X is near zero.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`logexp-1` (*a*) [Function]

Compute log(exp(a)-1) stably even when A is small.

Package [`num-utils.log-exp`], page 24.

Source [`log-exp.lisp`], page 17.

`lower-triangular-matrix` (*elements*) [Function]

Create a lower-triangular-matrix.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`make-vector` (*element-type* **&rest** *initial-contents*) [Function]

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

Source [`utilities.lisp`], page 5.

`max-error` (*instance*) [Reader]
`(setf max-error)` (*instance*) [Writer]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

Target Slot
[`max-error`], page 67.

`mean-error` (*instance*) [Reader]
`(setf mean-error)` (*instance*) [Writer]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

Target Slot
[`mean-error`], page 67.

`min-error` (*instance*) [Reader]
`(setf min-error)` (*instance*) [Writer]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

Target Slot
[`min-error`], page 67.

`monotonicp` (*column*) [Function]

Returns T if all elements of COLUMN, a SYMBOL, are increasing monotonically Useful for detecting row numbers in imported data.

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

Source [`utilities.lisp`], page 5.

`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 28.

Source [`arithmetic.lisp`], page 7.

`num-delta` (*a b*) [Function]

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

Package [`num-utils.num=`], page 31.

Source [`num=.lisp`], page 6.

`num=-function` (*tolerance*) [Function]
Curried version of num=, with given tolerance.

Package [`num-utils.num=`], page 31.

Source [`num=.lisp`], page 6.

`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 28.

Source [`arithmetic.lisp`], page 7.

`plusminus-interval` (*center half-width* **&key** *open-left? open-right?*) [Function]
A symmetric interval around CENTER.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`print-length-truncate` (*dimension*) [Function]
Return values (min dimension *print-length*) and whether the constraint is binding.

Package [`num-utils.print-matrix`], page 21.

Source [`print-matrix.lisp`], page 10.

`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 21.

Source [`print-matrix.lisp`], page 10.

`relative` (*fraction*) [Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`rms` (*instance*) [Reader]
`(setf rms)` (*instance*) [Writer]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

Target Slot
[`rms`], page 68.

`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 19.

Source [`quadrature.lisp`], page 15.

`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 15.

`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 28.

Source [`arithmetic.lisp`], page 7.

`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 28.

Source [`arithmetic.lisp`], page 7.

`sequence-maximum` (*x*) [Function]

Return the maximum value in the sequence X

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`sequence-minimum` (*x*) [Function]

Return the minimum value in the sequence X

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`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 22.

Source [`interval.lisp`], page 9.

`spacer` (**&optional** *weight*) [Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`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 22.

Source [`interval.lisp`], page 9.

`square` (*number*) [Function]

Square of number.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`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 22.

Source [`interval.lisp`], page 9.

`test-count` (*instance*) [Reader]
`(setf test-count)` (*instance*) [Writer]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

Target Slot
[`test-count`], page 67.

`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 27.

Source [`test-utilities.lisp`], page 17.

`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 28.

Source [`arithmetic.lisp`], page 7.

`upper-triangular-matrix` (*elements*) [Function]

Create a lower-triangular-matrix.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`variance0` (*instance*) [Reader]
`(setf variance0)` (*instance*) [Writer]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

Target Slot
[`variance0`], page 67.

`variance1` (*instance*) [Reader]
`(setf variance1)` (*instance*) [Writer]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

Target Slot
[`variance1`], page 68.

`vec` (*element-type* **&rest** *elements*) [Function]
Return a vector with elements coerced to ELEMENT-TYPE.

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

Source [`matrix-shorthand.lisp`], page 14.

`within?` (*left value right*) [Function]
Return non-nil iff value is in [left,right).

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

Source [`utilities.lisp`], page 5.

`worst-case` (*instance*) [Reader]
`(setf worst-case)` (*instance*) [Writer]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

Target Slot
[`worst-case`], page 67.

`wrapped-matrix-elements` (*instance*) [Reader]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

Target Slot
[`elements`], page 69.

4.1.5 Generic functions

`as-alist` (*object*) [Generic Function]
Return OBJECT as an ALIST. Semantics depends on OBJECT.

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

Source [`utilities.lisp`], page 5.

`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 26.

Source [`utilities.lisp`], page 5.

Methods

`as-plist` (*object*) [Method]

`diagonal-vector` (*matrix*) [Generic Function]

Return the diagonal elements of MATRIX as a vector.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

Methods

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

`diagonal-vector` (*matrix*) [Method]

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

Set the diagonal elements of MATRIX using VECTOR.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

Methods

`(setf diagonal-vector)` ((*matrix* `array`)) [Method]

`e1-` (*a*) [Generic Function]

Univariate elementwise -.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`e1-` ((*a* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`e1-` ((*a* [`hermitian-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e1-` ((*a* [`upper-triangular-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e1-` ((*a* [`lower-triangular-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

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

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

`e1/` (*a*) [Generic Function]

Univariate elementwise /.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`e1/` ((*a* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`e1/` ((*a* [`hermitian-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e1/` ((*a* [`upper-triangular-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e1/` ((*a* [`lower-triangular-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

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

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

`e1log` (*a*) [Generic Function]

Univariate elementwise LOG.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`e1log` ((*a* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`e1log` ((*a* [`hermitian-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e1log` ((*a* [`upper-triangular-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e1log` ((*a* [`lower-triangular-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

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

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

`e2*` (*a b*) [Generic Function]

Bivariate elementwise *.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`e2*` ((*a* [`diagonal-matrix`], *page 64*) (*b* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a [*`hermitian-matrix`*], page 65*) (*b [*`hermitian-matrix`*], page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a [*`upper-triangular-matrix`*], page 68*) (*b [*`upper-triangular-matrix`*], page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a [*`lower-triangular-matrix`*], page 65*) (*b [*`lower-triangular-matrix`*], page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a* `number`) (*b [*`diagonal-matrix`*], page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a [*`diagonal-matrix`*], page 64*) (*b* `number`)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a* `number`) (*b [*`hermitian-matrix`*], page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a [*`hermitian-matrix`*], page 65*) (*b* `number`)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a* `number`) (*b [*`upper-triangular-matrix`*], page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a [*`upper-triangular-matrix`*], page 68*) (*b* `number`)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a* `number`) (*b [*`lower-triangular-matrix`*], page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a [*`lower-triangular-matrix`*], page 65*) (*b* `number`)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` (*a* (*b [*`wrapped-matrix`*], page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e2*` ((*a [*`wrapped-matrix`*], page 68*) *b*) [Method]

Source [`matrix.lisp`], page 11.

`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 29.

Source [`elementwise.lisp`], page 7.

Methods

`e2+` ((*a* [`diagonal-matrix`], *page 64*) (*b* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`e2+` ((*a* [`hermitian-matrix`], *page 65*) (*b* [`hermitian-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2+` ((*a* [`upper-triangular-matrix`], *page 68*) (*b* [`upper-triangular-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e2+` ((*a* [`lower-triangular-matrix`], *page 65*) (*b* [`lower-triangular-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2+` (*a* (*b* [`wrapped-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e2+` ((*a* [`wrapped-matrix`], *page 68*) *b*) [Method]

Source [`matrix.lisp`], page 11.

`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 29.

Source [`elementwise.lisp`], page 7.

Methods

`e2-` ((*a* [`diagonal-matrix`], *page 64*) (*b* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`e2-` ((*a* [`hermitian-matrix`], *page 65*) (*b* [`hermitian-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2-` ((*a* [`upper-triangular-matrix`], *page 68*) (*b* [`upper-triangular-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e2-` ((*a* [`lower-triangular-matrix`], *page 65*) (*b* [`lower-triangular-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2-` (*a* (*b* [`wrapped-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e2-` ((*a* [`wrapped-matrix`], *page 68*) *b*) [Method]

Source [`matrix.lisp`], page 11.

`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 29.

Source [`elementwise.lisp`], page 7.

Methods

`e2/` ((*a* `number`) (*b* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`e2/` ((*a* [`diagonal-matrix`], *page 64*) (*b* `number`)) [Method]

Source [`matrix.lisp`], page 11.

`e2/` ((*a* `number`) (*b* [`hermitian-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2/` ((*a* [`hermitian-matrix`], *page 65*) (*b* `number`)) [Method]

Source [`matrix.lisp`], page 11.

`e2/` ((*a* `number`) (*b* [`upper-triangular-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`e2/` ((*a* [`upper-triangular-matrix`], *page 68*) (*b* `number`)) [Method]

Source [`matrix.lisp`], page 11.

`e2/` ((*a* `number`) (*b* [`lower-triangular-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`e2/` ((*a* [`lower-triangular-matrix`], *page 65*) (*b* `number`)) [Method]

Source [`matrix.lisp`], page 11.

`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 29.

Source [`elementwise.lisp`], page 7.

Methods

`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 29.

Source [`elementwise.lisp`], page 7.

Methods

`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 29.

Source [`elementwise.lisp`], page 7.

Methods

`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 29.

Source [`elementwise.lisp`], page 7.

Methods

`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 29.

Source [`elementwise.lisp`], page 7.

Methods

`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 29.

Source [`elementwise.lisp`], page 7.

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 29.

Source [`elementwise.lisp`], page 7.

Methods

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

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

`econjugate` (*a*) [Generic Function]

Univariate elementwise CONJUGATE.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

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

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

`ecos` (*a*) [Generic Function]

Univariate elementwise COS.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`ecos` ((*a* `number`)) [Method]

`ecos` ((*a* `array`)) [Method]

`eexp` (*a*) [Generic Function]

Univariate elementwise EXP.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`eexp` ((*a* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`eexp` ((*a* [`hermitian-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`eexp` ((*a* *[upper-triangular-matrix], page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`eexp` ((*a* *[lower-triangular-matrix], page 65*)) [Method]

Source [`matrix.lisp`], page 11.

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

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

`eexpt` (*a* *b*) [Generic Function]

Bivariate elementwise EXPT.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

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 29.

Source [`elementwise.lisp`], page 7.

Methods

`efloor` ((*a* `number`)) [Method]

`efloor` ((*a* `array`)) [Method]

`emod` (*a* *b*) [Generic Function]

Bivariate elementwise MOD.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`emod` ((*a* `number`) (*b* `number`)) [Method]

`emod` ((*a* `vector`) (*b* `number`)) [Method]

`emod` ((*a* `number`) (*b* `vector`)) [Method]

`emod` ((*a* `vector`) (*b* `vector`)) [Method]

`emod` ((*a* `array`) (*b* `number`)) [Method]

`emod` ((*a* `number`) (*b* `array`)) [Method]

`emod` ((*a* `array`) (*b* `array`)) [Method] `ereduce` (*function object* **&key** *key*) [Generic Function]

Elementwise reduce, traversing in row-major order.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`ereduce` (*function* (*array* `array`) **&key** *key*) [Method]

`ereduce` (*function* (*sequence* `sequence`) **&key** *key*) [Method]

`ereduce` (*function object* **&key** *key*) [Method]

`esin` (*a*) [Generic Function]

Univariate elementwise SIN.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`esin` ((*a* `number`)) [Method]

`esin` ((*a* `array`)) [Method]

`esqrt` (*a*) [Generic Function]

Univariate elementwise SQRT.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`esqrt` ((*a* [`diagonal-matrix`], *page 64*)) [Method]

Source [`matrix.lisp`], page 11.

`esqrt` ((*a* [`hermitian-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`esqrt` ((*a* [`upper-triangular-matrix`], *page 68*)) [Method]

Source [`matrix.lisp`], page 11.

`esqrt` ((*a* [`lower-triangular-matrix`], *page 65*)) [Method]

Source [`matrix.lisp`], page 11.

`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 22.

Source [`interval.lisp`], page 9.

Methods

`extend-interval` ((*interval* `null`) (*object* `null`)) [Method]

`extend-interval` ((*interval* `null`) (*number* `real`)) [Method] `extend-interval` ((*interval* [`interval`], *page 69*) (*number* `real`)) [Method]

`extend-interval` (*interval* (*object* [`interval`], *page 69*)) [Method]

`extend-interval` (*interval* (*list* `list`)) [Method]

`extend-interval` (*interval* (*array* `array`)) [Method]

`l2norm-square` (*object*) [Generic Function]

Square of the L_2 norm of OBJECT.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

Methods

`l2norm-square` ((*sequence* `sequence`)) [Method]

`left` (*interval*) [Generic Function]

Left endpoint of interval.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Methods

`left` ((*interval* [`interval/infinite-left`], *page 88*)) [Method]

`left` ((*interval/finite-left* [`interval/finite-left`], *page 87*)) [Reader Method]

automatically generated reader method

Target Slot

[`left`], page 87.

`map-array` (*array function* **&optional** *retval*) [Generic Function]

Package [`num-utils.matrix`], page 24.

Methods

`map-array` (*array function* **&optional** *retval*) [Method]

Apply FUNCTION to each element of ARRAY
Return a new array, or write into the optional 3rd argument.

Source [`matrix.lisp`], page 11.

`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 31.

Source [`num=.lisp`], page 6.

Methods

`num=` ((*a* [`diagonal-matrix`], *page 64*) (*b* [`diagonal-matrix`], *page 64*) **&optional** *tolerance*) [Method]

Source [`matrix.lisp`], page 11.

`num=` ((*a* [`wrapped-matrix`], *page 68*) (*b* [`wrapped-matrix`], *page 68*) **&optional** *tolerance*) [Method]

Source [`matrix.lisp`], page 11.

`num=` ((*a* [`finite-interval`], *page 69*) (*b* [`finite-interval`], *page 69*) **&optional** *tolerance*) [Method]

Source [`interval.lisp`], page 9.

`num=` ((*a* [`real-line`], *page 70*) (*b* [`real-line`], *page 70*) **&optional** *tolerance*) [Method]

Source [`interval.lisp`], page 9.

`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 22.

Source [`interval.lisp`], page 9.

Methods

`open-left?` ((*interval* [`interval/infinite-left`], *page 88*)) [Method]

`open-left?` ((*interval/finite-left* [`interval/finite-left`], *page 87*)) [Reader Method]

automatically generated reader method

Target Slot

[`open-left?`], page 87.

`open-right?` (*interval*) [Generic Function]

True iff the right endpoint of the interval is open.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Methods

`open-right?` ((*interval* [`interval/infinite-right`], *page 88*)) [Method]

`open-right?` ((*interval/finite-right* [`interval/finite-right`], *page 87*)) [Reader Method]

automatically generated reader method

Target Slot

[`open-right?`], page 88.

`product` (*object*) [Generic Function]

Product of elements in object.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

Methods

`product` ((*sequence* `sequence`)) [Method]

`product` ((*array* `array`)) [Method]

`right` (*interval*) [Generic Function]

Right endpoint of interval.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Methods

`right` ((*interval* [`interval/infinite-right`], *page 88*)) [Method]

`right` ((*interval/finite-right* [`interval/finite-right`], *page 87*)) [Reader Method]

automatically generated reader method

Target Slot

[`right`], page 88.

`shift-interval` (*interval offset*) [Generic Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Methods

`shift-interval` ((*interval* [`finite-interval`], *page 69*) (*offset* `real`)) [Method]

`sum` (*object* **&key** *key*) [Generic Function]

Sum of elements in object. KEY is applied to each element.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

Methods

`sum` ((*sequence* `sequence`) **&key** *key*) [Method]

`sum` ((*array* `array`) **&key** *key*) [Method]

`transpose` (*array*) [Generic Function]

Transpose.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

Methods

`transpose` ((*array* `array`)) [Method]

`transpose` ((*matrix* [`lower-triangular-matrix`], *page 65*)) [Method]

`transpose` ((*matrix* [`upper-triangular-matrix`], *page 68*)) [Method]

`transpose` ((*matrix* [`hermitian-matrix`], *page 65*)) [Method]

`transpose` ((*diagonal* [`diagonal-matrix`], *page 64*)) [Method]

4.1.6 Standalone methods

`as-array` ((*matrix0* [`upper-triangular-matrix`], *page 68*)) [Method]

Package `array-operations/generic`.

Source [`matrix.lisp`], page 11.

`as-array` ((*matrix0* [`hermitian-matrix`], *page 65*)) [Method]

Package `array-operations/generic`.

Source [`matrix.lisp`], page 11.

`as-array` ((*matrix0* [`lower-triangular-matrix`], *page 65*)) [Method]

Package `array-operations/generic`.

Source [`matrix.lisp`], page 11.

`as-array` ((*diagonal-matrix* [`diagonal-matrix`], *page 64*)) [Method]

Package `array-operations/generic`.

Source [`matrix.lisp`], page 11.

`dims` ((*wrapped-matrix* [`wrapped-matrix`], *page 68*)) [Method]

Package `array-operations/generic`.

Source [`matrix.lisp`], page 11.

`dims` ((*diagonal-matrix* [`diagonal-matrix`], *page 64*)) [Method]

Package `array-operations/generic`.

Source [`matrix.lisp`], page 11.

`element-type` ((*wrapped-matrix* [`wrapped-matrix`], *page 68*)) [Method]

Package `array-operations/generic`.

Source [`matrix.lisp`], page 11.

`element-type` ((*diagonal-matrix* [`diagonal-matrix`], *page 64*)) [Method]

Package `array-operations/generic`.

Source [`matrix.lisp`], page 11.

`initialize-instance` *:after* ((*interval* [`finite-interval`], *page 69*) **&key** **&allow-other-keys**) [Method]

Source [`interval.lisp`], page 9.

`print-object` ((*interval* [`interval`], *page 69*) *stream*) [Method]

Source [`interval.lisp`], page 9.

`print-object` ((*matrix0* [`upper-triangular-matrix`], *page 68*) *stream1*) [Method]

Source [`matrix.lisp`], page 11.

`print-object` ((*matrix0* [`hermitian-matrix`], *page 65*) *stream1*) [Method]

Source [`matrix.lisp`], page 11.

`print-object` ((*matrix0* [`lower-triangular-matrix`], *page 65*) *stream1*) [Method]

Source [`matrix.lisp`], page 11.

`select` ((*matrix0 [*`upper-triangular-matrix`*], page 68*) **&rest** *slices*) [Method]

Package `select`.

Source [`matrix.lisp`], page 11.

`select` ((*matrix0 [*`hermitian-matrix`*], page 65*) **&rest** *slices*) [Method]

Package `select`.

Source [`matrix.lisp`], page 11.

`select` ((*matrix0 [*`lower-triangular-matrix`*], page 65*) **&rest** *slices*) [Method]

Package `select`.

Source [`matrix.lisp`], page 11.

4.1.7 Structures

`diagonal-matrix` [Structure]

Diagonal matrix. The elements in the diagonal are stored in a vector.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

Direct superclasses

`structure-object`.

Direct methods

- [`as-array`], page 63.
- [`dims`], page 63.
- [`e1-`], page 49.
- [`e1/`], page 50.
- [`e1log`], page 50.
- [`e2*`], page 51.
- [`e2*`], page 51.
- [`e2*`], page 51.
- [`e2+`], page 52.
- [`e2-`], page 53.
- [`e2/`], page 53.
- [`e2/`], page 54.
- [`eexp`], page 57.
- [`element-type`], page 63.
- [`esqrt`], page 59.
- [`num=`], page 60.
- [`transpose`], page 62.

Direct slots

`elements` [Slot]

Type `vector`

Readers [`diagonal-matrix-elements`], page 39.

Writers [`(setf diagonal-matrix-elements)`], page 39.

`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 24.

Source [`matrix.lisp`], page 11.

Direct superclasses
[`wrapped-matrix`], page 68.

Direct methods

- [`as-array`], page 63.
- [`e1-`], page 49.
- [`e1/`], page 50.
- [`e1log`], page 50.
- [`e2*`], page 51.
- [`e2*`], page 51.
- [`e2*`], page 51.
- [`e2+`], page 52.
- [`e2-`], page 53.
- [`e2/`], page 54.
- [`e2/`], page 54.
- [`eexp`], page 57.
- [`esqrt`], page 59.
- [`print-object`], page 64.
- [`select`], page 64.
- [`transpose`], page 62.

`lower-triangular-matrix` [Structure]

Lower triangular matrix. ELEMENTS in the upper triangle are treated as zero.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

Direct superclasses
[`wrapped-matrix`], page 68.

Direct methods

- [`as-array`], page 63.
- [`e1-`], page 49.
- [`e1/`], page 50.
- [`e1log`], page 50.
- [`e2*`], page 51.
- [`e2*`], page 51.
- [`e2*`], page 51.
- [`e2+`], page 52.

- [e2-], page 53.
- [e2/], page 54.
- [e2/], page 54.
- [eexp], page 57.
- [esqrt], page 59.
- [print-object], page 64.
- [select], page 64.
- [transpose], page 62.

`relative` [Structure]

Relative sizes are in terms of width.

Package [num-utils.interval], page 22.

Source [interval.lisp], page 9.

Direct superclasses
`structure-object`.

Direct slots

`fraction` [Slot]

Type `(real 0)`

Readers [relative-fraction], page 79.

Writers *This slot is read-only.*

`spacer` [Structure]

Spacers divide the leftover portion of an interval.

Package [num-utils.interval], page 22.

Source [interval.lisp], page 9.

Direct superclasses
`structure-object`.

Direct slots

`weight` [Slot]

Type `(real 0)`

Initform `1`

Readers [spacer-weight], page 81.

Writers *This slot is read-only.*

`test-results` [Structure]

Differences between reference values and computed values

Package [num-utils.test-utilities], page 27.

Source [test-utilities.lisp], page 17.

Direct superclasses
`structure-object`.

Direct slots

`worst-case` [Slot]

Type `integer`

Initform 0

Readers [worst-case], page 48.

Writers [(setf worst-case)], page 48.

`min-error` [Slot]

Type `double-float`

Initform `0.0d0`

Readers [min-error], page 44.

Writers [(setf min-error)], page 44.

`max-error` [Slot]

Type `double-float`

Initform `0.0d0`

Readers [max-error], page 44.

Writers [(setf max-error)], page 44.

`mean-error` [Slot]

Type `double-float`

Initform `0.0d0`

Readers [mean-error], page 44.

Writers [(setf mean-error)], page 44.

`test-count` [Slot]

Type `integer`

Initform `0`

Readers [test-count], page 47.

Writers [(setf test-count)], page 47.

`variance0` [Slot]

Type `double-float`

Initform `0.0d0`

Readers [variance0], page 48.

Writers [(setf variance0)], page 48.

`variance1` [Slot]

Type `double-float`

Initform `0.0d0`

Readers [variance1], page 48.

Writers [(setf variance1)], page 48.

`rms` [Slot]

Type `double-float`

Initform `0.0d0`

Readers [rms], page 45.

Writers [(setf rms)], page 45.

`upper-triangular-matrix` [Structure]

Upper triangular matrix. ELEMENTS in the lower triangle are treated as zero.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

Direct superclasses
[`wrapped-matrix`], page 68.

Direct methods

- [`as-array`], page 63.
- [`e1-`], page 49.
- [`e1/`], page 50.
- [`e1log`], page 50.
- [`e2*`], page 51.
- [`e2*`], page 51.
- [`e2*`], page 51.
- [`e2+`], page 52.
- [`e2-`], page 53.
- [`e2/`], page 54.
- [`e2/`], page 54.
- [`eexp`], page 57.
- [`esqrt`], page 59.
- [`print-object`], page 63.
- [`select`], page 64.
- [`transpose`], page 62.

`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 24.

Source [`matrix.lisp`], page 11.

Direct superclasses
`structure-object`.

Direct subclasses

- [`hermitian-matrix`], page 65.
- [`lower-triangular-matrix`], page 65.
- [`upper-triangular-matrix`], page 68.

Direct methods

- [`dims`], page 63.
- [`e2*`], page 51.
- [`e2*`], page 52.
- [`e2+`], page 52.
- [`e2+`], page 52.
- [`e2-`], page 53.
- [`e2-`], page 53.

- [`element-type`], page 63.
- [`num=`], page 60.

Direct slots

`elements` [Slot]

Type	`(array * (* *))`
Readers	[`wrapped-matrix-elements`], page 48.
Writers	*This slot is read-only.*

4.1.8 Classes

`finite-interval` [Class]

Interval with finite endpoints.

Package	[`num-utils.interval`], page 22.
Source	[`interval.lisp`], page 9.

Direct superclasses

- [`interval`], page 69.
- [`interval/finite-left`], page 87.
- [`interval/finite-right`], page 87.

Direct methods

- [`chebyshev-approximate-implementation`], page 83.
- [`initialize-instance`], page 63.
- [`num=`], page 61.
- [`shift-interval`], page 62.
- [`transformed-quadrature`], page 84.

`interval` [Class]

Abstract superclass for all intervals.

Package	[`num-utils.interval`], page 22.
Source	[`interval.lisp`], page 9.

Direct subclasses

- [`finite-interval`], page 69.
- [`minusinf-interval`], page 70.
- [`plusinf-interval`], page 70.
- [`real-line`], page 70.

Direct methods

- [`extend-interval`], page 59.
- [`extend-interval`], page 59.
- [`print-object`], page 63.

`minusinf-interval` [Class]

Interval from -∞ to RIGHT.

Package	[`num-utils.interval`], page 22.
Source	[`interval.lisp`], page 9.

Direct superclasses

- [`interval`], page 69.
- [`interval/finite-right`], page 87.
- [`interval/infinite-left`], page 88.

`plusinf-interval` [Class]

Interval from LEFT to ∞.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Direct superclasses

- [`interval`], page 69.
- [`interval/finite-left`], page 87.
- [`interval/infinite-right`], page 88.

Direct methods

- [`chebyshev-approximate-implementation`], page 83.
- [`transformed-quadrature`], page 84.

`real-line` [Class]

Representing the real line (-∞,∞).

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Direct superclasses

- [`interval`], page 69.
- [`interval/infinite-left`], page 88.
- [`interval/infinite-right`], page 88.

Direct methods

[`num=`], page 61.

4.1.9 Types

`extended-real` (*&optional base*) [Type]

Extended real number.

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`simple-boolean-vector` (*&optional length*) [Type]

Vector of BOOLEAN elements.

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

Source [`utilities.lisp`], page 5.

`simple-double-float-vector` (*&optional length*) [Type]

Simple vector of double-float elements.

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

Source [`utilities.lisp`], page 5.

`simple-fixnum-vector` () [Type]
Simple vector of fixnum elements.

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

Source [`utilities.lisp`], page 5.

`simple-single-float-vector` (*&optional length*) [Type]
Simple vector of single-float elements.

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

Source [`utilities.lisp`], page 5.

`triangular-matrix` () [Type]
Triangular matrix (either lower or upper).

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

4.2 Internals

4.2.1 Macros

`&diagonal-matrix` (*elements*) [Macro]
LET+ form for slots of the structure DIAGONAL-MATRIX.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`&diagonal-matrix-r/o` (*elements*) [Macro]
LET+ form for slots of the structure DIAGONAL-MATRIX. Read-only.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`define-comparison` (*name test*) [Macro]
Define a comparison, extendeding a pairwise comparison to an arbitrary number of arguments.

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

`define-e&` (*operation* **&key** *function bivariate univariate docstring*) [Macro]

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`define-e1` (*operation* **&key** *function docstring*) [Macro]
Define an univariate elementwise operation.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`define-e2` (*operation* **&key** *function docstring*) [Macro]
Define a bivariate elementwise operation.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`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 24.

Source [`matrix.lisp`], page 11.

`define-elementwise-reduction` (*name function* **&optional** *docstring*) [Macro]

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`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 24.

Source [`matrix.lisp`], page 11.

`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 24.

Source [`matrix.lisp`], page 11.

`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 24.

Source [`matrix.lisp`], page 11.

`define-rounding-with-offset` (*name function docstring*) [Macro]

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`define-wrapped-matrix` (*type elements struct-docstring* (*masked-test masked-string*) *check-and-convert-elements regularize-elements*) [Macro]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`mapping-array` ((*ref array* **&rest** *other*) *form*) [Macro]

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

`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 15.

4.2.2 Ordinary 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 21.

Source [`chebyshev.lisp`], page 14.

`ab-to-cinf` (*z a b c*) [Function]
Inverse of cinf-to-ab.

Package [`num-utils.chebyshev`], page 21.

Source [`chebyshev.lisp`], page 14.

`above-diagonal?` (*row col*) [Function]
Test if element with indexes row and col is (strictly) above the diagonal.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`below-diagonal?` (*row col*) [Function]
Test if element with indexes row and col is (strictly) below the diagonal.

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`boolean-sequence-p` (*x*) [Function]

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

Source [`utilities.lisp`], page 5.

`boolean?` (*object*) [Function]
Check type of OBJECT is BOOLEAN.

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

Source [`utilities.lisp`], page 5.

`chebyshev-recursion` (*x value previous-value*) [Function]

Chebyshev polynomial recursion formula.

Package [`num-utils.chebyshev`], page 21.

Source [`chebyshev.lisp`], page 14.

`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 21.

Source [`chebyshev.lisp`], page 14.

`copy-diagonal-matrix` (*instance*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`copy-hermitian-matrix` (*instance*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`copy-iterative-quadrature` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`copy-lower-triangular-matrix` (*instance*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`copy-midpoint-quadrature` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`copy-relative` (*instance*) [Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`copy-richardson-extrapolation` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`copy-spacer` (*instance*) [Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`copy-test-results` (*instance*) [Function]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

`copy-trapezoidal-quadrature` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`copy-upper-triangular-matrix` (*instance*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`copy-wrapped-matrix` (*instance*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`diagonal-matrix-p` (*object*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`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 24.

Source [`matrix.lisp`], page 11.

`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 26.

Source [`extended-real.lisp`], page 9.

`hermitian-matrix-elements` (*instance*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`hermitian-matrix-p` (*object*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`iterative-quadrature-a` (*instance*) [Reader]
`(setf iterative-quadrature-a)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot

[a], page 85.

`iterative-quadrature-b` (*instance*) [Reader]
`(setf iterative-quadrature-b)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot
[`b`], page 85.

`iterative-quadrature-f` (*instance*) [Reader]
`(setf iterative-quadrature-f)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot
[`f`], page 85.

`iterative-quadrature-h` (*instance*) [Reader]
`(setf iterative-quadrature-h)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot
[`h`], page 85.

`iterative-quadrature-n` (*instance*) [Reader]
`(setf iterative-quadrature-n)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot
[`n`], page 85.

`iterative-quadrature-p` (*object*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`iterative-quadrature-sum` (*instance*) [Reader]
`(setf iterative-quadrature-sum)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot
[`sum`], page 85.

`ln` (*n*) [Function]

Natural logarithm.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`lower-triangular-matrix-elements` (*instance*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`lower-triangular-matrix-p` (*object*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`make-diagonal-matrix` (**&key** *elements*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`make-hermitian-matrix` (**&key** *elements*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`make-iterative-quadrature` (**&key** *f a b h n sum*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`make-lower-triangular-matrix` (**&key** *elements*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`make-test-results` (**&key** *worst-case min-error max-error mean-error test-count variance0 variance1 rms*) [Function]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

`make-upper-triangular-matrix` (**&key** *elements*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`make-wrapped-matrix` (**&key** *elements*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`midpoint-quadrature` (*f a b*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`midpoint-quadrature%` (**&key** *f a b h n sum*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`midpoint-quadrature-a` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf midpoint-quadrature-a)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`midpoint-quadrature-b` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf midpoint-quadrature-b)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`midpoint-quadrature-f` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf midpoint-quadrature-f)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`midpoint-quadrature-h` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf midpoint-quadrature-h)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`midpoint-quadrature-n` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf midpoint-quadrature-n)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`midpoint-quadrature-p` (*object*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`midpoint-quadrature-sum` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf midpoint-quadrature-sum)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`narrow-bracket?` (*a b delta*) [Function]

Return true iff $|a-b| < \delta$.

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

Source [`rootfinding.lisp`], page 15.

`near-root?` (*f epsilon*) [Function]
Return true iff $|f| < \epsilon$.

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

Source [`rootfinding.lisp`], page 15.

`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 15.

`pad-left-expansion` (*rows ncol*) [Function]
Pad ragged-right rows. Used internally to implement ragged right matrix specifications.

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

Source [`matrix-shorthand.lisp`], page 14.

`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 21.

Source [`print-matrix.lisp`], page 10.

`relative-fraction` (*instance*) [Reader]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Target Slot
[`fraction`], page 66.

`relative-p` (*object*) [Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`richardson-extrapolation` (*coefficient iterations*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`richardson-extrapolation-coefficient` (*instance*) [Reader]
`(setf richardson-extrapolation-coefficient)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot
[`coefficient`], page 86.

`richardson-extrapolation-diagonal` (*instance*) [Reader]
`(setf richardson-extrapolation-diagonal)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot
[`diagonal`], page 86.

`richardson-extrapolation-n` (*instance*) [Reader]
`(setf richardson-extrapolation-n)` (*instance*) [Writer]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Target Slot
[`n`], page 86.

`richardson-extrapolation-p` (*object*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`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 19.

Source [`quadrature.lisp`], page 15.

`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 19.

Source [`quadrature.lisp`], page 15.

`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 15.

`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 28.

Source [`arithmetic.lisp`], page 7.

`similar-sequence-type` (*sequence*) [Function]
Return type that sequence can be mapped to using arithmetic operations.

Package [`num-utils.arithmetic`], page 28.

Source [`arithmetic.lisp`], page 7.

`spacer-p` (*object*) [Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

`spacer-weight` (*instance*) [Reader]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Target Slot
[`weight`], page 66.

`test-results-p` (*object*) [Function]

Package [`num-utils.test-utilities`], page 27.

Source [`test-utilities.lisp`], page 17.

`trapezoidal-quadrature` (*f a b*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`trapezoidal-quadrature%` (**&key** *f a b h n sum*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`trapezoidal-quadrature-a` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf trapezoidal-quadrature-a)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`trapezoidal-quadrature-b` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf trapezoidal-quadrature-b)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`trapezoidal-quadrature-f` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf trapezoidal-quadrature-f)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`trapezoidal-quadrature-h` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf trapezoidal-quadrature-h)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`trapezoidal-quadrature-n` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf trapezoidal-quadrature-n)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`trapezoidal-quadrature-p` (*object*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`trapezoidal-quadrature-sum` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`(setf trapezoidal-quadrature-sum)` (*instance*) [Function]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

`upper-triangular-matrix-elements` (*instance*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`upper-triangular-matrix-p` (*object*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`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 24.

Source [`matrix.lisp`], page 11.

`wrapped-matrix-p` (*object*) [Function]

Package [`num-utils.matrix`], page 24.

Source [`matrix.lisp`], page 11.

`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 24.

Source [`matrix.lisp`], page 11.

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 21.

Source [`chebyshev.lisp`], page 14.

Methods

`chebyshev-approximate-implementation` (*f* (*interval* *[*`plusinf-interval`*], page 70*) *n-polynomials n-points*) [Method]

`chebyshev-approximate-implementation` (*f* (*interval* *[*`finite-interval`*], page 69*) *n-polynomials n-points*) [Method]

`esquare` (*a*) [Generic Function]

Univariate elementwise SQUARE.

Package [`num-utils.elementwise`], page 29.

Source [`elementwise.lisp`], page 7.

Methods

`esquare` ((*a* `number`)) [Method]

`esquare` ((*a* `array`)) [Method]

`print-left-endpoint` (*interval stream*) [Generic Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Methods

`print-left-endpoint` ((*interval* *[*`interval/finite-left`*], page 87*) *stream*) [Method]

`print-left-endpoint` ((*interval* *[*`interval/infinite-left`*], page 88*) *stream*) [Method]

`print-right-endpoint` (*interval stream*) [Generic Function]

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Methods

`print-right-endpoint` ((*interval* *[*`interval/finite-right`*], page 87*) *stream*) [Method]

`print-right-endpoint` ((*interval* *[*`interval/infinite-right`*], page 88*) *stream*) [Method]

`refine-quadrature` (*quadrature*) [Generic Function]

Refine quadrature with more points. Return the sum for those points.

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Methods

`refine-quadrature` ((*quadrature* *[*`midpoint-quadrature`*], page 86*)) [Method]

`refine-quadrature` ((*quadrature* *[*`trapezoidal-quadrature`*], page 86*)) [Method]

`richardson-coefficient` (*quadrature*) [Generic Function]

Return the coefficient q for Richardson approximation.

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Methods

`richardson-coefficient` ((*quadrature* [`midpoint-quadrature`], *page 86*)) [Method]

`richardson-coefficient` ((*quadrature* [`trapezoidal-quadrature`], *page 86*)) [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 19.

Source [`quadrature.lisp`], page 15.

Methods

`transformed-quadrature` (*function* (*interval* [`finite-interval`], *page 69*) (*transformation* `null`)) [Method]

`transformed-quadrature` (*function* (*interval* [`plusinf-interval`], *page 70*) (*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 19.

Source [`quadrature.lisp`], page 15.

Direct superclasses
`structure-object`.

Direct subclasses
- [`midpoint-quadrature`], page 86.
- [`trapezoidal-quadrature`], page 86.

Direct slots

`f` [Slot]

Type `(function (double-float) double-float)`

Readers [`iterative-quadrature-f`], page 76.

Writers [`(setf iterative-quadrature-f)`], page 76.

`a` [Slot]

Type `double-float`

Readers [`iterative-quadrature-a`], page 76.

Writers [`(setf iterative-quadrature-a)`], page 76.

`b` [Slot]

Type `double-float`

Readers [`iterative-quadrature-b`], page 76.

Writers [`(setf iterative-quadrature-b)`], page 76.

`h` [Slot]

Type `double-float`

Readers [`iterative-quadrature-h`], page 76.

Writers [`(setf iterative-quadrature-h)`], page 76.

`n` [Slot]

Type `fixnum`

Initform `0`

Readers [`iterative-quadrature-n`], page 76.

Writers [`(setf iterative-quadrature-n)`], page 76.

`sum` [Slot]

Package [`num-utils.arithmetic`], page 28.

Type `double-float`

Initform `0.0d0`

Readers [`iterative-quadrature-sum`], page 76.

Writers [`(setf iterative-quadrature-sum)`], page 76.

`midpoint-quadrature` [Structure]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Direct superclasses
[`iterative-quadrature`], page 84.

Direct methods
- [`refine-quadrature`], page 84.
- [`richardson-coefficient`], page 84.

`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 19.

Source [`quadrature.lisp`], page 15.

Direct superclasses
`structure-object`.

Direct slots

`coefficient` [Slot]

Type `double-float`

Readers [`richardson-extrapolation-coefficient`], page 80.

Writers [`(setf richardson-extrapolation-coefficient)`], page 80.

`n` [Slot]

Type `fixnum`

Initform `0`

Readers [`richardson-extrapolation-n`], page 80.

Writers [`(setf richardson-extrapolation-n)`], page 80.

`diagonal` [Slot]

Type `(array double-float (*))`

Readers [`richardson-extrapolation-diagonal`], page 80.

Writers [`(setf richardson-extrapolation-diagonal)`], page 80.

`trapezoidal-quadrature` [Structure]

Package [`num-utils.quadrature`], page 19.

Source [`quadrature.lisp`], page 15.

Direct superclasses
[`iterative-quadrature`], page 84.

Direct methods
- [`refine-quadrature`], page 84.
- [`richardson-coefficient`], page 84.

4.2.5 Classes

`interval/finite-left` [Class]

Interval with left endpoint.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Direct subclasses
- [`finite-interval`], page 69.
- [`plusinf-interval`], page 70.

Direct methods
- [`left`], page 60.
- [`open-left?`], page 61.
- [`print-left-endpoint`], page 83.

Direct slots

`left` [Slot]

Type `real`

Initargs `:left`

Readers [`left`], page 60.

Writers *This slot is read-only.*

`open-left?` [Slot]

Type `boolean`

Initargs `:open-left?`

Readers [`open-left?`], page 61.

Writers *This slot is read-only.*

`interval/finite-right` [Class]

Interval with right endpoint.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Direct subclasses

- [`finite-interval`], page 69.
- [`minusinf-interval`], page 70.

Direct methods

- [`open-right?`], page 61.
- [`print-right-endpoint`], page 83.
- [`right`], page 62.

Direct slots

`right` [Slot]

Type `real`

Initargs `:right`

Readers [`right`], page 62.

Writers *This slot is read-only.*

`open-right?` [Slot]

Type `boolean`

Initargs `:open-right?`

Readers [`open-right?`], page 61.

Writers *This slot is read-only.*

`interval/infinite-left` [Class]

Left endpoint is $-\infty$.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Direct subclasses

- [`minusinf-interval`], page 70.
- [`real-line`], page 70.

Direct methods

- [`left`], page 60.
- [`open-left?`], page 61.
- [`print-left-endpoint`], page 83.

`interval/infinite-right` [Class]

Right endpoint is ∞.

Package [`num-utils.interval`], page 22.

Source [`interval.lisp`], page 9.

Direct subclasses

- [`plusinf-interval`], page 70.
- [`real-line`], page 70.

Direct methods

- [`open-right?`], page 61.
- [`print-right-endpoint`], page 84.
- [`right`], page 62.

4.2.6 Types

`infinite ()` [Type]

Representing infinity (extending the real line).

Package [`num-utils.extended-real`], page 26.

Source [`extended-real.lisp`], page 9.

Appendix A Indexes

A.1 Concepts

(Index is nonexistent)