

# The LINEAR-ALGEBRA Reference Manual

---

Linear Algebra for Common Lisp, version 0.1.1

Steve Nunez <steve@symbolics.tech>

Thomas M. Hermann <thomas.m.hermann@odonata-research.com>

This manual was generated automatically by Declt 4.0b2.

Copyright © 2019-2023 Steve Nunez Copyright © 2019-2023 Thomas M. Hermann  
Permission is granted to make and distribute verbatim copies of this manual provided  
the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under  
the conditions for verbatim copying, provided also that the section entitled “Copy-  
ing” is included exactly as in the original.

Permission is granted to copy and distribute translations of this manual into an-  
other language, under the above conditions for modified versions, except that this  
permission notice may be translated as well.

# Table of Contents

Copying .....	1
<b>1 Systems .....</b>	<b>3</b>
1.1 linear-algebra .....	3
<b>2 Modules .....</b>	<b>5</b>
2.1 linear-algebra/kernel .....	5
2.2 linear-algebra/interface .....	5
2.3 linear-algebra/sequence .....	5
<b>3 Files .....</b>	<b>7</b>
3.1 Lisp .....	7
3.1.1 linear-algebra/linear-algebra.asd .....	7
3.1.2 linear-algebra/pkgdcl.lisp .....	7
3.1.3 linear-algebra/kernel/pkgdcl.lisp .....	7
3.1.4 linear-algebra/kernel/utility.lisp .....	7
3.1.5 linear-algebra/kernel/permute.lisp .....	8
3.1.6 linear-algebra/kernel/unary-operations.lisp .....	8
3.1.7 linear-algebra/kernel/binary-operations.lisp .....	8
3.1.8 linear-algebra/kernel/rotation.lisp .....	9
3.1.9 linear-algebra/kernel/gauss.lisp .....	9
3.1.10 linear-algebra/kernel/cholesky.lisp .....	10
3.1.11 linear-algebra/kernel/conjugate-gradient.lisp .....	10
3.1.12 linear-algebra/kernel/tridiagonal.lisp .....	10
3.1.13 linear-algebra/interface/fundamental-ops.lisp .....	11
3.1.14 linear-algebra/interface/vector.lisp .....	11
3.1.15 linear-algebra/interface/matrix.lisp .....	12
3.1.16 linear-algebra/interface/identity-matrix.lisp .....	12
3.1.17 linear-algebra/interface/permutation-matrix.lisp .....	13
3.1.18 linear-algebra/sequence/list.lisp .....	13
3.1.19 linear-algebra/sequence/vector.lisp .....	14
3.1.20 linear-algebra/sequence/array.lisp .....	14
3.1.21 linear-algebra/data-vector.lisp .....	15
3.1.22 linear-algebra/dense-matrix.lisp .....	16
3.1.23 linear-algebra/square-matrix.lisp .....	18
3.1.24 linear-algebra/hermitian-matrix.lisp .....	18
3.1.25 linear-algebra/symmetric-matrix.lisp .....	19
<b>4 Packages .....</b>	<b>21</b>
4.1 linear-algebra-kernel .....	21
4.2 linear-algebra .....	23
<b>5 Definitions .....</b>	<b>27</b>
5.1 Public Interface .....	27
5.1.1 Macros .....	27
5.1.2 Ordinary functions .....	27

5.1.3	Generic functions .....	32
5.1.4	Standalone methods .....	57
5.1.5	Classes .....	57
5.2	Internals .....	65
5.2.1	Ordinary functions .....	65
5.2.2	Generic functions .....	69
<b>Appendix A Indexes .....</b>		<b>75</b>
A.1	Concepts .....	75
A.2	Functions .....	76
A.3	Variables .....	80
A.4	Data types .....	81

## Copying

This program is distributed under the terms of the Microsoft Public License.



# 1 Systems

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

## 1.1 linear-algebra

Linear Algebra for Common Lisp

### Long Name

Linear Algebra for Common Lisp

### Maintainer

Steve Nunez <steve@symbolics.tech>

### Author

Thomas M. Hermann <thomas.m.hermann@odonata-research.com>

### Home Page

<https://lisp-stat.dev/docs/manuals/lla>

### Source Control

(GIT <https://github.com/Lisp-Stat/linear-algebra.git>)

### Bug Tracker

<https://github.com/Lisp-Stat/linear-algebra/issues>

### License

MS-PL

### Long Description

This system is a high level interface for linear algebra and matrix manipulation. It was forked from Thomas Hermann's linear-algebra library (<https://github.com/OdonataResearchLLC/linear-algebra>) and currently maintained by Brian Eberman and Steve Nunez.

Current goals are to implement backends that use BLAS/LAPACK and CUDA.

**Version** 0.1.1

### Dependencies

- `closer-mop` (system).
- `floating-point` (system).

**Source** [linear-algebra.asd], page 7.

### Child Components

- [pkgdcl.lisp], page 7 (file).
- [kernel], page 5 (module).
- [interface], page 5 (module).
- [sequence], page 5 (module).
- [data-vector.lisp], page 15 (file).
- [dense-matrix.lisp], page 16 (file).
- [square-matrix.lisp], page 18 (file).
- [hermitian-matrix.lisp], page 18 (file).
- [symmetric-matrix.lisp], page 19 (file).





## 2 Modules

Modules are listed depth-first from the system components tree.

### 2.1 linear-algebra/kernel

**Source** [linear-algebra.asd], page 7.

**Parent Component**  
[linear-algebra], page 3 (system).

**Child Components**

- [pkgdcl.lisp], page 7 (file).
- [utility.lisp], page 7 (file).
- [permute.lisp], page 8 (file).
- [unary-operations.lisp], page 8 (file).
- [binary-operations.lisp], page 8 (file).
- [rotation.lisp], page 9 (file).
- [gauss.lisp], page 9 (file).
- [cholesky.lisp], page 10 (file).
- [conjugate-gradient.lisp], page 10 (file).
- [tridiagonal.lisp], page 10 (file).

### 2.2 linear-algebra/interface

**Dependency**  
[kernel], page 5 (module).

**Source** [linear-algebra.asd], page 7.

**Parent Component**  
[linear-algebra], page 3 (system).

**Child Components**

- [fundamental-ops.lisp], page 11 (file).
- [vector.lisp], page 11 (file).
- [matrix.lisp], page 12 (file).
- [identity-matrix.lisp], page 12 (file).
- [permutation-matrix.lisp], page 13 (file).

### 2.3 linear-algebra/sequence

**Dependency**  
[interface], page 5 (module).

**Source** [linear-algebra.asd], page 7.

**Parent Component**  
[linear-algebra], page 3 (system).

**Child Components**

- [list.lisp], page 13 (file).
- [vector.lisp], page 14 (file).
- [array.lisp], page 14 (file).



## 3 Files

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

### 3.1 Lisp

#### 3.1.1 linear-algebra/linear-algebra.asd

**Source** [linear-algebra.asd], page 7.

**Parent Component**  
[linear-algebra], page 3 (system).

**ASDF Systems**  
[linear-algebra], page 3.

#### 3.1.2 linear-algebra/pkgdcl.lisp

**Dependency**  
[kernel], page 5 (module).

**Source** [linear-algebra.asd], page 7.

**Parent Component**  
[linear-algebra], page 3 (system).

**Packages** [linear-algebra], page 23.

#### 3.1.3 linear-algebra/kernel/pkgdcl.lisp

**Source** [linear-algebra.asd], page 7.

**Parent Component**  
[kernel], page 5 (module).

**Packages** [linear-algebra-kernel], page 21.

#### 3.1.4 linear-algebra/kernel/utility.lisp

**Dependency**  
[pkgdcl.lisp], page 7 (file).

**Source** [linear-algebra.asd], page 7.

**Parent Component**  
[kernel], page 5 (module).

#### Public Interface

- [common-array-element-type], page 27 (function).
- [common-class-of], page 27 (function).
- [complex-equal], page 27 (function).
- [copy-array], page 34 (generic function).
- [number-equal], page 30 (function).
- [specific-array-element-type], page 31 (function).

#### Internals

- [zero-array], page 69 (function).
- [zero-vector], page 69 (function).

### 3.1.5 linear-algebra/kernel/permute.lisp

#### Dependency

[pkgdcl.lisp], page 7 (file).

**Source** [linear-algebra.asd], page 7.

#### Parent Component

[kernel], page 5 (module).

#### Public Interface

- [left-permute], page 36 (generic function).
- [right-permute], page 50 (generic function).

### 3.1.6 linear-algebra/kernel/unary-operations.lisp

#### Dependency

[pkgdcl.lisp], page 7 (file).

**Source** [linear-algebra.asd], page 7.

#### Parent Component

[kernel], page 5 (module).

#### Public Interface

- [norm-array], page 42 (generic function).
- [norm-vector], page 43 (generic function).
- [sump], page 54 (generic function).
- [sumsq], page 55 (generic function).
- [sumsq-column], page 31 (function).
- [sumsq-row], page 31 (function).
- [sumsq2], page 32 (function).
- [sumsq3], page 32 (function).

**Internals** [%abs-vector], page 65 (function).

### 3.1.7 linear-algebra/kernel/binary-operations.lisp

#### Dependency

[pkgdcl.lisp], page 7 (file).

**Source** [linear-algebra.asd], page 7.

#### Parent Component

[kernel], page 5 (module).

#### Public Interface

- [add-array], page 27 (function).
- [add-vector], page 27 (function).
- [compatible-dimensions-p], page 34 (generic function).
- [inner-product-vector], page 29 (function).
- [nadd-array], page 29 (function).
- [nadd-vector], page 30 (function).
- [nsubtract-array], page 30 (function).
- [nsubtract-vector], page 30 (function).
- [product-array-array], page 30 (function).

- [product-array-vector], page 30 (function).
- [product-vector-array], page 30 (function).
- [scaled-binary-op], page 50 (generic function).
- [subtract-array], page 31 (function).
- [subtract-vector], page 31 (function).

#### Internals

- [%array1<-array1-op-array2], page 65 (function).
- [%array<-array1-op-array2], page 65 (function).
- [%product-array-array], page 66 (function).
- [%product-array-vector], page 66 (function).
- [%product-vector-array], page 66 (function).
- [%scaled-product-array-array], page 67 (function).
- [%scaled-product-array-vector], page 67 (function).
- [%scaled-product-vector-array], page 67 (function).
- [%vector1<-vector1-op-vector2], page 68 (function).
- [%vector<-vector1-op-vector2], page 68 (function).

### 3.1.8 linear-algebra/kernel/rotation.lisp

#### Dependency

[unary-operations.lisp], page 8 (file).

#### Source

[linear-algebra.asd], page 7.

#### Parent Component

[kernel], page 5 (module).

#### Public Interface

- [givens-rotation], page 28 (function).
- [householder-reflection], page 29 (function).
- [jacobi-rotation], page 29 (function).

### 3.1.9 linear-algebra/kernel/gauss.lisp

#### Dependency

[pkgdcl.lisp], page 7 (file).

#### Source

[linear-algebra.asd], page 7.

#### Parent Component

[kernel], page 5 (module).

#### Public Interface

- [gauss-invert], page 28 (function).
- [gauss-solver], page 28 (function).

#### Internals

- [column-pivot], page 68 (function).
- [column-pivot-search], page 68 (function).
- [gauss-backsubstitution], page 68 (function).
- [gauss-factorization], page 68 (function).
- [gauss-update], page 68 (function).

- [initialize-pivot-selection-vector], page 68 (function).
- [swap-rows], page 68 (function).
- [unit-pivot-value], page 69 (function).

### 3.1.10 linear-algebra/kernel/cholesky.lisp

#### Dependency

[unary-operations.lisp], page 8 (file).

**Source** [linear-algebra.asd], page 7.

#### Parent Component

[kernel], page 5 (module).

#### Public Interface

- [hermitian-cholesky-decomposition], page 28 (function).
- [hermitian-cholesky-invert], page 28 (function).
- [hermitian-cholesky-solver], page 28 (function).
- [root-free-hermitian-cholesky-decomposition], page 30 (function).
- [root-free-symmetric-cholesky-decomposition], page 31 (function).
- [symmetric-cholesky-decomposition], page 32 (function).
- [symmetric-cholesky-invert], page 32 (function).
- [symmetric-cholesky-solver], page 32 (function).

### 3.1.11 linear-algebra/kernel/conjugate-gradient.lisp

#### Dependency

[binary-operations.lisp], page 8 (file).

**Source** [linear-algebra.asd], page 7.

#### Parent Component

[kernel], page 5 (module).

#### Public Interface

[conjugate-gradient-solver], page 28 (function).

#### Internals

- [%default-cg-epsilon], page 65 (function).
- [%initialize-cg-residual], page 65 (function).
- [%initialize-cg-solution], page 65 (function).
- [%negative-residual], page 66 (function).

### 3.1.12 linear-algebra/kernel/tridiagonal.lisp

#### Dependency

[pkgdcl.lisp], page 7 (file).

**Source** [linear-algebra.asd], page 7.

#### Parent Component

[kernel], page 5 (module).

#### Public Interface

[tridiagonal-solver], page 32 (function).

#### Internals

- [tridiagonal-backsubstitution], page 69 (function).

- [tridiagonal-factorization], page 69 (function).
- [tridiagonal-update], page 69 (function).

### 3.1.13 linear-algebra/interface/fundamental-ops.lisp

**Source** [linear-algebra.asd], page 7.

**Parent Component**

[interface], page 5 (module).

**Public Interface**

- [add], page 32 (generic function).
- [invert], page 35 (generic function).
- [nadd], page 40 (generic function).
- [ninvert], page 41 (generic function).
- [norm], page 42 (generic function).
- [nscale], page 43 (generic function).
- [nsolve], page 43 (generic function).
- [nsubtract], page 44 (generic function).
- [ntranspose], page 45 (generic function).
- [permute], page 46 (generic function).
- [product], page 47 (generic function).
- [scale], page 50 (generic function).
- [solve], page 51 (generic function).
- [subtract], page 53 (generic function).
- [transpose], page 55 (generic function).

### 3.1.14 linear-algebra/interface/vector.lisp

**Dependency**

[fundamental-ops.lisp], page 11 (file).

**Source** [linear-algebra.asd], page 7.

**Parent Component**

[interface], page 5 (module).

**Public Interface**

- [apply-rotation], page 33 (generic function).
- [copy-vector], page 35 (generic function).
- [dovector], page 27 (macro).
- [make-vector], page 29 (function).
- [map-into-vector], page 36 (generic function).
- [map-vector], page 36 (generic function).
- [napply-rotation], page 41 (generic function).
- [replace-vector], page 49 (generic function).
- [subvector], page 54 (generic function).
- [(setf subvector)], page 54 (generic function).
- [vector-element-type], page 56 (generic function).
- [vector-in-bounds-p], page 56 (generic function).

- `[vector-length]`, page 56 (generic function).
- `[vref]`, page 56 (generic function).
- `[(setf vref)]`, page 57 (generic function).

### 3.1.15 `linear-algebra/interface/matrix.lisp`

#### Dependency

`[fundamental-ops.lisp]`, page 11 (file).

#### Source

`[linear-algebra.asd]`, page 7.

#### Parent Component

`[interface]`, page 5 (module).

#### Public Interface

- `[copy-matrix]`, page 35 (generic function).
- `[make-matrix]`, page 29 (function).
- `[matrix-column-dimension]`, page 37 (generic function).
- `[matrix-dimensions]`, page 37 (generic function).
- `[matrix-element-type]`, page 38 (generic function).
- `[matrix-in-bounds-p]`, page 38 (generic function).
- `[matrix-object]`, page 62 (class).
- `[matrix-row-dimension]`, page 38 (generic function).
- `[matrix-validated-range]`, page 29 (function).
- `[matrixp]`, page 29 (function).
- `[mref]`, page 39 (generic function).
- `[(setf mref)]`, page 39 (generic function).
- `[replace-matrix]`, page 49 (generic function).
- `[submatrix]`, page 52 (generic function).
- `[(setf submatrix)]`, page 52 (generic function).

**Internals** `[initialize-matrix-contents]`, page 71 (generic function).

### 3.1.16 `linear-algebra/interface/identity-matrix.lisp`

#### Dependency

`[matrix.lisp]`, page 12 (file).

#### Source

`[linear-algebra.asd]`, page 7.

#### Parent Component

`[interface]`, page 5 (module).

#### Public Interface

- `[copy-matrix]`, page 35 (method).
- `[identity-matrix]`, page 61 (class).
- `[identity-matrix-p]`, page 29 (function).
- `[initialize-instance]`, page 57 (method).
- `[matrix-column-dimension]`, page 37 (reader method).
- `[matrix-dimensions]`, page 37 (method).
- `[matrix-element-type]`, page 38 (method).
- `[matrix-in-bounds-p]`, page 38 (method).



- `[matrix-row-dimension]`, page 39 (reader method).
- `[mref]`, page 39 (method).

#### Internals

- `[contents]`, page 70 (reader method).
- `[size]`, page 73 (reader method).

### 3.1.17 `linear-algebra/interface/permutation-matrix.lisp`

#### Dependency

`[matrix.lisp]`, page 12 (file).

**Source** `[linear-algebra.asd]`, page 7.

#### Parent Component

`[interface]`, page 5 (module).

#### Public Interface

- `[copy-matrix]`, page 35 (method).
- `[initialize-instance]`, page 57 (method).
- `[matrix-column-dimension]`, page 37 (method).
- `[matrix-dimensions]`, page 37 (method).
- `[matrix-element-type]`, page 38 (method).
- `[matrix-in-bounds-p]`, page 38 (method).
- `[matrix-row-dimension]`, page 39 (method).
- `[mref]`, page 39 (method).
- `[(setf mref)]`, page 40 (method).
- `[permutation-matrix]`, page 62 (class).
- `[permutation-matrix-p]`, page 30 (function).
- `[transpose]`, page 56 (method).

#### Internals

- `[%initialize-permutation-matrix-with-seq]`, page 66 (function).
- `[contents]`, page 70 (reader method).
- `[(setf contents)]`, page 71 (writer method).
- `[initialize-matrix-contents]`, page 72 (method).
- `[initialize-matrix-contents]`, page 72 (method).
- `[initialize-matrix-contents]`, page 72 (method).

### 3.1.18 `linear-algebra/sequence/list.lisp`

**Source** `[linear-algebra.asd]`, page 7.

#### Parent Component

`[sequence]`, page 5 (module).

#### Public Interface

- `[add]`, page 33 (method).
- `[nadd]`, page 41 (method).
- `[norm]`, page 42 (method).
- `[nscale]`, page 43 (method).
- `[nsubtract]`, page 45 (method).

- [ntranspose], page 46 (method).
- [permute], page 47 (method).
- [permute], page 47 (method).
- [product], page 48 (method).
- [scale], page 50 (method).
- [subtract], page 54 (method).
- [transpose], page 55 (method).

#### Internals

- [%norm], page 69 (method).
- [%norm], page 69 (method).
- [%norm], page 69 (method).
- [%norm], page 70 (method).

### 3.1.19 linear-algebra/sequence/vector.lisp

**Source** [linear-algebra.asd], page 7.

#### Parent Component

[sequence], page 5 (module).

#### Public Interface

- [add], page 33 (method).
- [nadd], page 41 (method).
- [norm], page 42 (method).
- [nscale], page 43 (method).
- [nsubtract], page 45 (method).
- [ntranspose], page 46 (method).
- [permute], page 47 (method).
- [permute], page 47 (method).
- [product], page 48 (method).
- [scale], page 50 (method).
- [subtract], page 54 (method).
- [transpose], page 55 (method).

### 3.1.20 linear-algebra/sequence/array.lisp

**Source** [linear-algebra.asd], page 7.

#### Parent Component

[sequence], page 5 (module).

#### Public Interface

- [add], page 33 (method).
- [compatible-dimensions-p], page 34 (method).
- [invert], page 36 (method).
- [nadd], page 41 (method).
- [ninvert], page 42 (method).
- [norm], page 42 (method).
- [nscale], page 43 (method).

- [nsolve], page 44 (method).
- [nsubtract], page 45 (method).
- [ntranspose], page 46 (method).
- [permute], page 47 (method).
- [permute], page 47 (method).
- [product], page 48 (method).
- [product], page 48 (method).
- [product], page 48 (method).
- [scale], page 50 (method).
- [solve], page 52 (method).
- [subtract], page 54 (method).
- [transpose], page 55 (method).

### 3.1.21 linear-algebra/data-vector.lisp

#### Dependency

[interface], page 5 (module).

#### Source

[linear-algebra.asd], page 7.

#### Parent Component

[linear-algebra], page 3 (system).

#### Public Interface

- [add], page 33 (method).
- [add], page 33 (method).
- [add], page 33 (method).
- [apply-rotation], page 33 (method).
- [apply-rotation], page 33 (method).
- [column-vector], page 27 (function).
- [column-vector], page 57 (class).
- [column-vector-p], page 27 (function).
- [copy-vector], page 35 (method).
- [data-vector], page 58 (class).
- [initialize-instance], page 57 (method).
- [map-into-vector], page 36 (method).
- [map-into-vector], page 36 (method).
- [map-vector], page 36 (method).
- [map-vector], page 37 (method).
- [nadd], page 40 (method).
- [nadd], page 40 (method).
- [nadd], page 41 (method).
- [napply-rotation], page 41 (method).
- [napply-rotation], page 41 (method).
- [norm], page 42 (method).
- [nscale], page 43 (method).
- [nsubtract], page 45 (method).

- [nsubtract], page 45 (method).
- [nsubtract], page 45 (method).
- [ntranspose], page 45 (method).
- [ntranspose], page 45 (method).
- [permute], page 46 (method).
- [permute], page 46 (method).
- [permute], page 46 (method).
- [permute], page 47 (method).
- [product], page 48 (method).
- [product], page 48 (method).
- [replace-vector], page 50 (method).
- [row-vector], page 31 (function).
- [row-vector], page 63 (class).
- [row-vector-p], page 31 (function).
- [scale], page 50 (method).
- [subtract], page 53 (method).
- [subtract], page 53 (method).
- [subtract], page 53 (method).
- [subvector], page 54 (method).
- [(setf subvector)], page 54 (method).
- [transpose], page 55 (method).
- [transpose], page 55 (method).
- [vector-element-type], page 56 (method).
- [vector-in-bounds-p], page 56 (method).
- [vector-length], page 56 (method).
- [vref], page 56 (method).
- [(setf vref)], page 57 (method).

### Internals

- [%map-data-vector], page 66 (function).
- [%map-into-data-vector], page 66 (function).
- [contents], page 70 (reader method).
- [(setf contents)], page 70 (writer method).

### 3.1.22 linear-algebra/dense-matrix.lisp

#### Dependency

[data-vector.lisp], page 15 (file).

#### Source

[linear-algebra.asd], page 7.

#### Parent Component

[linear-algebra], page 3 (system).

#### Public Interface

- [add], page 32 (method).
- [add], page 33 (method).
- [compatible-dimensions-p], page 34 (method).

- [copy-matrix], page 35 (method).
- [dense-matrix], page 59 (class).
- [dense-matrix-p], page 28 (function).
- [initialize-instance], page 57 (method).
- [invert], page 36 (method).
- [matrix-column-dimension], page 37 (method).
- [matrix-dimensions], page 37 (method).
- [matrix-element-type], page 38 (method).
- [matrix-in-bounds-p], page 38 (method).
- [matrix-row-dimension], page 39 (method).
- [mref], page 39 (method).
- [(setf mref)], page 40 (method).
- [nadd], page 40 (method).
- [nadd], page 40 (method).
- [ninvert], page 42 (method).
- [norm], page 42 (method).
- [nscale], page 43 (method).
- [nsolve], page 44 (method).
- [nsolve], page 44 (method).
- [nsubtract], page 44 (method).
- [nsubtract], page 44 (method).
- [ntranspose], page 45 (method).
- [permute], page 46 (method).
- [permute], page 46 (method).
- [product], page 47 (method).
- [product], page 47 (method).
- [product], page 48 (method).
- [product], page 48 (method).
- [product], page 48 (method).
- [product], page 48 (method).
- [product], page 48 (method).
- [replace-matrix], page 49 (method).
- [scale], page 50 (method).
- [solve], page 51 (method).
- [solve], page 51 (method).
- [submatrix], page 52 (method).
- [(setf submatrix)], page 53 (method).
- [subtract], page 53 (method).
- [subtract], page 53 (method).
- [transpose], page 55 (method).

### Internals

- [contents], page 70 (reader method).
- [(setf contents)], page 70 (writer method).

- `[initialize-matrix-contents]`, page 72 (method).
- `[initialize-matrix-contents]`, page 72 (method).
- `[initialize-matrix-contents]`, page 72 (method).
- `[initialize-matrix-contents]`, page 72 (method).

### 3.1.23 `linear-algebra/square-matrix.lisp`

#### Dependency

`[dense-matrix.lisp]`, page 16 (file).

**Source**      `[linear-algebra.asd]`, page 7.

#### Parent Component

`[linear-algebra]`, page 3 (system).

#### Public Interface

- `[compatible-dimensions-p]`, page 34 (method).
- `[invert]`, page 35 (method).
- `[ninvert]`, page 42 (method).
- `[square-matrix]`, page 64 (class).
- `[square-matrix-p]`, page 31 (function).
- `[submatrix]`, page 52 (method).

**Internals**   `[initialize-matrix-contents]`, page 72 (method).

### 3.1.24 `linear-algebra/hermitian-matrix.lisp`

#### Dependency

`[square-matrix.lisp]`, page 18 (file).

**Source**      `[linear-algebra.asd]`, page 7.

#### Parent Component

`[linear-algebra]`, page 3 (system).

#### Public Interface

- `[hermitian-matrix]`, page 61 (class).
- `[hermitian-matrix-p]`, page 28 (function).
- `[invert]`, page 35 (method).
- `[(setf mref)]`, page 40 (method).
- `[ninvert]`, page 41 (method).
- `[nsolve]`, page 44 (method).
- `[ntranspose]`, page 45 (method).
- `[permute]`, page 46 (method).
- `[permute]`, page 46 (method).
- `[replace-matrix]`, page 49 (method).
- `[replace-matrix]`, page 49 (method).
- `[solve]`, page 51 (method).
- `[submatrix]`, page 52 (method).
- `[(setf submatrix)]`, page 53 (method).
- `[(setf submatrix)]`, page 53 (method).
- `[transpose]`, page 55 (method).

**Internals**

- [%initialize-hermitian-matrix-with-seq], page 65 (function).
- [%replace-hermitian-matrix-off-diagonal], page 66 (function).
- [%replace-hermitian-matrix-on-diagonal], page 67 (function).
- [%setf-hermitian-submatrix-off-diagonal], page 67 (function).
- [%setf-hermitian-submatrix-on-diagonal], page 67 (function).
- [initialize-matrix-contents], page 71 (method).
- [initialize-matrix-contents], page 71 (method).
- [initialize-matrix-contents], page 71 (method).
- [initialize-matrix-contents], page 72 (method).

**3.1.25 linear-algebra/symmetric-matrix.lisp****Dependency**

[square-matrix.lisp], page 18 (file).

**Source**

[linear-algebra.asd], page 7.

**Parent Component**

[linear-algebra], page 3 (system).

**Public Interface**

- [invert], page 35 (method).
- [(setf mref)], page 39 (method).
- [nadd], page 40 (method).
- [nadd], page 40 (method).
- [ninvert], page 41 (method).
- [nsolve], page 44 (method).
- [nsubtract], page 44 (method).
- [nsubtract], page 44 (method).
- [replace-matrix], page 49 (method).
- [replace-matrix], page 49 (method).
- [solve], page 51 (method).
- [submatrix], page 52 (method).
- [(setf submatrix)], page 52 (method).
- [(setf submatrix)], page 52 (method).
- [symmetric-matrix], page 64 (class).
- [symmetric-matrix-p], page 32 (function).

**Internals**

- [%initialize-symmetric-matrix-with-seq], page 66 (function).
- [%replace-symmetric-matrix-off-diagonal], page 67 (function).
- [%replace-symmetric-matrix-on-diagonal], page 67 (function).
- [%setf-symmetric-submatrix-off-diagonal], page 67 (function).
- [%setf-symmetric-submatrix-on-diagonal], page 68 (function).
- [initialize-matrix-contents], page 71 (method).
- [initialize-matrix-contents], page 71 (method).
- [initialize-matrix-contents], page 71 (method).





## 4 Packages

Packages are listed by definition order.

### 4.1 linear-algebra-kernel

**Source**      [pkgdcl.lisp], page 7.

**Use List**

- common-lisp.
- floating-point.

**Used By List**

[linear-algebra], page 23.

**Public Interface**

- [add-array], page 27 (function).
- [add-vector], page 27 (function).
- [common-array-element-type], page 27 (function).
- [common-class-of], page 27 (function).
- [compatible-dimensions-p], page 34 (generic function).
- [complex-equal], page 27 (function).
- [conjugate-gradient-solver], page 28 (function).
- [copy-array], page 34 (generic function).
- [gauss-invert], page 28 (function).
- [gauss-solver], page 28 (function).
- [givens-rotation], page 28 (function).
- [hermitian-cholesky-decomposition], page 28 (function).
- [hermitian-cholesky-invert], page 28 (function).
- [hermitian-cholesky-solver], page 28 (function).
- [householder-reflection], page 29 (function).
- [inner-product-vector], page 29 (function).
- [jacobi-rotation], page 29 (function).
- [left-permute], page 36 (generic function).
- [nadd-array], page 29 (function).
- [nadd-vector], page 30 (function).
- [norm-array], page 42 (generic function).
- [norm-vector], page 43 (generic function).
- [nsubtract-array], page 30 (function).
- [nsubtract-vector], page 30 (function).
- [number-equal], page 30 (function).
- [product-array-array], page 30 (function).
- [product-array-vector], page 30 (function).
- [product-vector-array], page 30 (function).
- [right-permute], page 50 (generic function).
- [root-free-hermitian-cholesky-decomposition], page 30 (function).

- [root-free-symmetric-cholesky-decomposition], page 31 (function).
- [scaled-binary-op], page 50 (generic function).
- [specific-array-element-type], page 31 (function).
- [subtract-array], page 31 (function).
- [subtract-vector], page 31 (function).
- [sump], page 54 (generic function).
- [sumsq], page 55 (generic function).
- [sumsq-column], page 31 (function).
- [sumsq-row], page 31 (function).
- [sumsq2], page 32 (function).
- [sumsq3], page 32 (function).
- [symmetric-cholesky-decomposition], page 32 (function).
- [symmetric-cholesky-invert], page 32 (function).
- [symmetric-cholesky-solver], page 32 (function).
- [tridiagonal-solver], page 32 (function).

## Internals

- [%abs-vector], page 65 (function).
- [%array1<-array1-op-array2], page 65 (function).
- [%array<-array1-op-array2], page 65 (function).
- [%default-cg-epsilon], page 65 (function).
- [%initialize-cg-residual], page 65 (function).
- [%initialize-cg-solution], page 65 (function).
- [%negative-residual], page 66 (function).
- [%product-array-array], page 66 (function).
- [%product-array-vector], page 66 (function).
- [%product-vector-array], page 66 (function).
- [%scaled-product-array-array], page 67 (function).
- [%scaled-product-array-vector], page 67 (function).
- [%scaled-product-vector-array], page 67 (function).
- [%vector1<-vector1-op-vector2], page 68 (function).
- [%vector<-vector1-op-vector2], page 68 (function).
- [column-pivot], page 68 (function).
- [column-pivot-search], page 68 (function).
- [gauss-backsubstitution], page 68 (function).
- [gauss-factorization], page 68 (function).
- [gauss-update], page 68 (function).
- [initialize-pivot-selection-vector], page 68 (function).
- [swap-rows], page 68 (function).
- [tridiagonal-backsubstitution], page 69 (function).
- [tridiagonal-factorization], page 69 (function).
- [tridiagonal-update], page 69 (function).
- [unit-pivot-value], page 69 (function).
- [zero-array], page 69 (function).
- [zero-vector], page 69 (function).

## 4.2 linear-algebra

**Source** [pkgdcl.lisp], page 7.

**Use List**

- common-lisp.
- floating-point.
- [linear-algebra-kernel], page 21.

**Public Interface**

- [add], page 32 (generic function).
- [apply-rotation], page 33 (generic function).
- [column-vector], page 27 (function).
- [column-vector], page 57 (class).
- [column-vector-p], page 27 (function).
- [copy-matrix], page 35 (generic function).
- [copy-vector], page 35 (generic function).
- [data-vector], page 58 (class).
- [dense-matrix], page 59 (class).
- [dense-matrix-p], page 28 (function).
- [dovector], page 27 (macro).
- [hermitian-matrix], page 61 (class).
- [hermitian-matrix-p], page 28 (function).
- [identity-matrix], page 61 (class).
- [identity-matrix-p], page 29 (function).
- [invert], page 35 (generic function).
- [make-matrix], page 29 (function).
- [make-vector], page 29 (function).
- [map-into-vector], page 36 (generic function).
- [map-vector], page 36 (generic function).
- [matrix-column-dimension], page 37 (generic function).
- [matrix-dimensions], page 37 (generic function).
- [matrix-element-type], page 38 (generic function).
- [matrix-in-bounds-p], page 38 (generic function).
- [matrix-object], page 62 (class).
- [matrix-row-dimension], page 38 (generic function).
- [matrix-validated-range], page 29 (function).
- [matrixp], page 29 (function).
- [mref], page 39 (generic function).
- [(setf mref)], page 39 (generic function).
- [nadd], page 40 (generic function).
- [napply-rotation], page 41 (generic function).
- [ninvert], page 41 (generic function).
- [norm], page 42 (generic function).
- [nscale], page 43 (generic function).

- [nsolve], page 43 (generic function).
- [nsubtract], page 44 (generic function).
- [ntranspose], page 45 (generic function).
- [permutation-matrix], page 62 (class).
- [permutation-matrix-p], page 30 (function).
- [permute], page 46 (generic function).
- [product], page 47 (generic function).
- [replace-matrix], page 49 (generic function).
- [replace-vector], page 49 (generic function).
- [row-vector], page 31 (function).
- [row-vector], page 63 (class).
- [row-vector-p], page 31 (function).
- [scale], page 50 (generic function).
- [solve], page 51 (generic function).
- [square-matrix], page 64 (class).
- [square-matrix-p], page 31 (function).
- [submatrix], page 52 (generic function).
- [(setf submatrix)], page 52 (generic function).
- [subtract], page 53 (generic function).
- [subvector], page 54 (generic function).
- [(setf subvector)], page 54 (generic function).
- [symmetric-matrix], page 64 (class).
- [symmetric-matrix-p], page 32 (function).
- [transpose], page 55 (generic function).
- [vector-element-type], page 56 (generic function).
- [vector-in-bounds-p], page 56 (generic function).
- [vector-length], page 56 (generic function).
- [vref], page 56 (generic function).
- [(setf vref)], page 57 (generic function).

## Internals

- [%initialize-hermitian-matrix-with-seq], page 65 (function).
- [%initialize-permutation-matrix-with-seq], page 66 (function).
- [%initialize-symmetric-matrix-with-seq], page 66 (function).
- [%map-data-vector], page 66 (function).
- [%map-into-data-vector], page 66 (function).
- [%norm], page 69 (generic function).
- [%replace-hermitian-matrix-off-diagonal], page 66 (function).
- [%replace-hermitian-matrix-on-diagonal], page 67 (function).
- [%replace-symmetric-matrix-off-diagonal], page 67 (function).
- [%replace-symmetric-matrix-on-diagonal], page 67 (function).
- [%setf-hermitian-submatrix-off-diagonal], page 67 (function).
- [%setf-hermitian-submatrix-on-diagonal], page 67 (function).

- `[%setf-symmetric-submatrix-off-diagonal]`, page 67 (function).
- `[%setf-symmetric-submatrix-on-diagonal]`, page 68 (function).
- `[contents]`, page 70 (generic reader).
- `[(setf contents)]`, page 70 (generic writer).
- `[initialize-matrix-contents]`, page 71 (generic function).
- `[size]`, page 72 (generic reader).



## 5 Definitions

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

### 5.1 Public Interface

#### 5.1.1 Macros

**dovector** (*(element vector &optional result) &body body*) [Macro]  
 Iterate over vector returning result.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

#### 5.1.2 Ordinary functions

**add-array** (*array1 array2 scalar1 scalar2*) [Function]  
 Array binary addition.

**Package** [linear-algebra-kernel], page 21.

**Source** [binary-operations.lisp], page 8.

**add-vector** (*vector1 vector2 scalar1 scalar2*) [Function]  
 Vector binary addition.

**Package** [linear-algebra-kernel], page 21.

**Source** [binary-operations.lisp], page 8.

**column-vector** (*&rest numbers*) [Function]  
 Create a column vector from the numbers.

**Package** [linear-algebra], page 23.

**Source** [data-vector.lisp], page 15.

**column-vector-p** (*object*) [Function]  
 Return true if object is a column-vector, NIL otherwise.

**Package** [linear-algebra], page 23.

**Source** [data-vector.lisp], page 15.

**common-array-element-type** (*array1 array2*) [Function]  
 Return the array type common to both arrays.

**Package** [linear-algebra-kernel], page 21.

**Source** [utility.lisp], page 7.

**common-class-of** (*object1 object2*) [Function]  
 Return the common class of the 2 objects or default-class.

**Package** [linear-algebra-kernel], page 21.

**Source** [utility.lisp], page 7.

**complex-equal** (*complex1 complex2 &optional epsilon*) [Function]  
 Return true if both numbers are complex and equal.

**Package** [linear-algebra-kernel], page 21.

**Source** [utility.lisp], page 7.

- conjugate-gradient-solver** (*array vector* &**optional** *epsilon limit*) [Function]  
 Linear system solver using the conjugate gradient method.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [conjugate-gradient.lisp], page 10.
- dense-matrix-p** (*object*) [Function]  
 Return true if object is a dense matrix.  
**Package** [linear-algebra], page 23.  
**Source** [dense-matrix.lisp], page 16.
- gauss-invert** (*array*) [Function]  
 Find  $A^{-1}$  via Gauss algorithm with partial column pivot search.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [gauss.lisp], page 9.
- gauss-solver** (*array vector*) [Function]  
 Gauss algorithm with column pivot search.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [gauss.lisp], page 9.
- givens-rotation** (*f g*) [Function]  
 Return c,s,r defined from the Givens rotation.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [rotation.lisp], page 9.
- hermitian-cholesky-decomposition** (*array*) [Function]  
 Factor  $A = LL^T$ .  
**Package** [linear-algebra-kernel], page 21.  
**Source** [cholesky.lisp], page 10.
- hermitian-cholesky-invert** (*array*) [Function]  
 Invert a positive definite matrices using the root-free Cholesky decomposition.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [cholesky.lisp], page 10.
- hermitian-cholesky-solver** (*array vector*) [Function]  
 Linear system solver for positive definite matrices using the root-free Cholesky decomposition.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [cholesky.lisp], page 10.
- hermitian-matrix-p** (*object*) [Function]  
 Return true if object is a hermitian-matrix, NIL otherwise.  
**Package** [linear-algebra], page 23.  
**Source** [hermitian-matrix.lisp], page 18.



<b>householder-reflection</b> ( <i>alpha vector</i> )	[Function]
Return Beta, Tau and the Householder vector.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [rotation.lisp], page 9.	
<b>identity-matrix-p</b> ( <i>object</i> )	[Function]
Return true if object is an identity-matrix.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [identity-matrix.lisp], page 12.	
<b>inner-product-vector</b> ( <i>vector1 vector2 scalar</i> )	[Function]
Return the vector inner product.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	
<b>jacobi-rotation</b> ( <i>x y z</i> )	[Function]
Return a, b, cos(theta) and sin(theta) terms from the Jacobi rotation.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [rotation.lisp], page 9.	
<b>make-matrix</b> ( <i>rows columns &amp;key matrix-type element-type initial-element initial-contents</i> )	[Function]
Return a new matrix instance.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [matrix.lisp], page 12.	
<b>make-vector</b> ( <i>size &amp;key vector-type element-type initial-element initial-contents</i> )	[Function]
Create the data structure to represent a vector.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [vector.lisp], page 11.	
<b>matrix-validated-range</b> ( <i>matrix start-row start-column &amp;optional end-row end-column</i> )	[Function]
Returns a validated range of rows and columns for the matrix.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [matrix.lisp], page 12.	
<b>matrixp</b> ( <i>object</i> )	[Function]
Return true if object is a matrix, NIL otherwise.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [matrix.lisp], page 12.	
<b>nadd-array</b> ( <i>array1 array2 scalar1 scalar2</i> )	[Function]
Destructive array binary addition.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	

- nadd-vector** (*vector1 vector2 scalar1 scalar2*) [Function]  
 Destructive vector binary addition.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [binary-operations.lisp], page 8.
- nsubtract-array** (*array1 array2 scalar1 scalar2*) [Function]  
 Destructive array binary subtraction.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [binary-operations.lisp], page 8.
- nsubtract-vector** (*vector1 vector2 scalar1 scalar2*) [Function]  
 Destructive vector binary subtraction.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [binary-operations.lisp], page 8.
- number-equal** (*number1 number2 &optional epsilon*) [Function]  
 Return true if the numbers are equal using the appropriate comparison.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [utility.lisp], page 7.
- permutation-matrix-p** (*object*) [Function]  
 Return true if object is a permutation-matrix.  
**Package** [linear-algebra], page 23.  
**Source** [permutation-matrix.lisp], page 13.
- product-array-array** (*array1 array2 &optional scalar result*) [Function]  
 Return the scaled result of the product of 2 arrays.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [binary-operations.lisp], page 8.
- product-array-vector** (*array vector &optional scalar result*) [Function]  
 Return the result of the array postmultiplied by the vector and scaled.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [binary-operations.lisp], page 8.
- product-vector-array** (*vector array &optional scalar result*) [Function]  
 Return the result of the array premultiplied by the vector and scaled.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [binary-operations.lisp], page 8.
- root-free-hermitian-cholesky-decomposition** (*array*) [Function]  
 Factor  $A = LDL^t$ .  
**Package** [linear-algebra-kernel], page 21.  
**Source** [cholesky.lisp], page 10.

<b>root-free-symmetric-cholesky-decomposition</b> ( <i>array</i> )	[Function]
Factor $A = LDL^t$ .	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [cholesky.lisp], page 10.	
<b>row-vector</b> ( <b>&amp;rest</b> <i>numbers</i> )	[Function]
Create a row vector from the numbers.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [data-vector.lisp], page 15.	
<b>row-vector-p</b> ( <i>object</i> )	[Function]
Return true if object is a row-vector, NIL otherwise.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [data-vector.lisp], page 15.	
<b>specific-array-element-type</b> ( <i>array</i> <b>&amp;rest</b> <i>subscripts</i> )	[Function]
Return the specific type of the element specified by subscripts.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [utility.lisp], page 7.	
<b>square-matrix-p</b> ( <i>object</i> )	[Function]
Return true if OBJECT is a square matrix.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [square-matrix.lisp], page 18.	
<b>subtract-array</b> ( <i>array1 array2 scalar1 scalar2</i> )	[Function]
Array binary subtraction.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	
<b>subtract-vector</b> ( <i>vector1 vector2 scalar1 scalar2</i> )	[Function]
Vector binary subtraction.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	
<b>sumsq-column</b> ( <i>array column</i> <b>&amp;key</b> <i>scale sumsq start end</i> )	[Function]
Return the scaling parameter and the sum of the squares of the array column.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [unary-operations.lisp], page 8.	
<b>sumsq-row</b> ( <i>array row</i> <b>&amp;key</b> <i>scale sumsq start end</i> )	[Function]
Return the scaling parameter and the sum of the squares of the array row.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [unary-operations.lisp], page 8.	

- sumsq2** (*x y*) [Function]  
 Return the square root of  $|x|^2 + |y|^2$ .  
**Package** [linear-algebra-kernel], page 21.  
**Source** [unary-operations.lisp], page 8.
- sumsq3** (*x y z*) [Function]  
 Return the square root of  $|x|^2 + |y|^2 + |z|^2$ .  
**Package** [linear-algebra-kernel], page 21.  
**Source** [unary-operations.lisp], page 8.
- symmetric-cholesky-decomposition** (*array*) [Function]  
 Factor  $A = LL^T$ .  
**Package** [linear-algebra-kernel], page 21.  
**Source** [cholesky.lisp], page 10.
- symmetric-cholesky-invert** (*array*) [Function]  
 Invert a positive definite matrices using the root-free Cholesky decomposition.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [cholesky.lisp], page 10.
- symmetric-cholesky-solver** (*array vector*) [Function]  
 Linear system solver for positive definite matrices using the root-free Cholesky decomposition.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [cholesky.lisp], page 10.
- symmetric-matrix-p** (*object*) [Function]  
 Return true if object is a symmetric-matrix, NIL otherwise.  
**Package** [linear-algebra], page 23.  
**Source** [symmetric-matrix.lisp], page 19.
- tridiagonal-solver** (*array vector*) [Function]  
 Linear equation solver for a tridiagonal matrix.  
**Package** [linear-algebra-kernel], page 21.  
**Source** [tridiagonal.lisp], page 10.

### 5.1.3 Generic functions

- add** (*vector-or-matrix-1 vector-or-matrix-2 &key scalar1 scalar2*) [Generic Function]  
 Vector or matrix binary addition.  
**Package** [linear-algebra], page 23.  
**Source** [fundamental-ops.lisp], page 11.  
**Methods**
- add** ((*matrix1* [dense-matrix], page 59) (*matrix2* [dense-matrix], page 59) &key *scalar1 scalar2*) [Method]  
 Return the addition of the 2 matrices.  
**Source** [dense-matrix.lisp], page 16.

`add :before ((matrix1 [dense-matrix], page 59) (matrix2 [dense-matrix], page 59) &key scalar1 scalar2)` [Method]  
 Audit the input data.

**Source** [*dense-matrix.lisp*], page 16.

`add ((vector1 [row-vector], page 63) (vector2 [row-vector], page 63) &key scalar1 scalar2)` [Method]  
 Return the addition of *scalar1*\**vector1* with *scalar2*\**vector2*.

**Source** [*data-vector.lisp*], page 15.

`add ((vector1 [column-vector], page 57) (vector2 [column-vector], page 57) &key scalar1 scalar2)` [Method]  
 Return the addition of *scalar1*\**vector1* with *scalar2*\**vector2*.

**Source** [*data-vector.lisp*], page 15.

`add :before ((vector1 [data-vector], page 58) (vector2 [data-vector], page 58) &key scalar1 scalar2)` [Method]  
 Verify that the dimensions are equal.

**Source** [*data-vector.lisp*], page 15.

`add ((array1 array) (array2 array) &key scalar1 scalar2)` [Method]  
 Return the addition of the 2 arrays.

**Source** [*array.lisp*], page 14.

`add ((vector1 vector) (vector2 vector) &key scalar1 scalar2)` [Method]  
 Return the addition of *scalar1*\**vector1* with *scalar2*\**vector2*

**Source** [*vector.lisp*], page 14.

`add ((list1 list) (list2 list) &key scalar1 scalar2)` [Method]  
 Return the addition of *scalar1*\**list1* with *scalar2*\**list2*

**Source** [*list.lisp*], page 13.

`apply-rotation (vector1 vector2 cc ss)` [Generic Function]  
 Return the plane rotations of *vector1* and *vector2* by *cc* and *ss*.

**Package** [*linear-algebra*], page 23.

**Source** [*vector.lisp*], page 11.

**Methods**

`apply-rotation ((vector1 [data-vector], page 58) (vector2 [data-vector], page 58) cc ss)` [Method]  
 Return the plane rotations of *vector1* and *vector2* by *cc* and *ss*.

**Source** [*data-vector.lisp*], page 15.

`apply-rotation :before ((vector1 [data-vector], page 58) (vector2 [data-vector], page 58) cc ss)` [Method]  
 Verify the input to `apply-rotation`.

**Source** [*data-vector.lisp*], page 15.

**compatible-dimensions-p** (*operation vector-or-matrix-1* [Generic Function]  
*vector-or-matrix-2*)

Return true if the vector and matrix dimensions are compatible for the operation.

**Package** [linear-algebra-kernel], page 21.

**Source** [binary-operations.lisp], page 8.

#### Methods

**compatible-dimensions-p** ((*operation* (eq1 :solve)) [Method]  
 (*matrix* [square-matrix], page 64) (*vector* [column-vector],  
 page 57))

Return true if the array dimensions are compatible for product.

**Source** [square-matrix.lisp], page 18.

**compatible-dimensions-p** ((*operation* (eq1 :solve)) [Method]  
 (*matrix* [dense-matrix], page 59) (*vector* [column-vector],  
 page 57))

Return true if the array dimensions are compatible for product.

**Source** [dense-matrix.lisp], page 16.

**compatible-dimensions-p** ((*operation* (eq1 :solve)) [Method]  
 (*array array*) (*vector vector*))

Return true if the array dimensions are compatible for product.

**Source** [array.lisp], page 14.

**compatible-dimensions-p** ((*operation* (eq1 :product)) [Method]  
 (*array1 array*) (*array2 array*))

Return true if the array dimensions are compatible for product.

**compatible-dimensions-p** ((*operation* (eq1 :add)) (*array1* [Method]  
*array*) (*array2 array*))

Return true if the array dimensions are compatible for an addition.

**compatible-dimensions-p** ((*operation* (eq1 :product)) [Method]  
 (*array array*) (*vector vector*))

Return true if the array dimensions are compatible for product.

**compatible-dimensions-p** ((*operation* (eq1 :product)) [Method]  
 (*vector vector*) (*array array*))

Return true if the array dimensions are compatible for product.

**compatible-dimensions-p** ((*operation* (eq1 :add)) (*vector1* [Method]  
*vector*) (*vector2 vector*))

Return true if the vector dimensions are compatible for an addition.

**copy-array** (*array*) [Generic Function]

Return an element-wise copy of the original array.

**Package** [linear-algebra-kernel], page 21.

**Source** [utility.lisp], page 7.

#### Methods

**copy-array** ((*original array*) [Method]

Return an element-wise copy of the original array.

- copy-array** ((*original vector*)) [Method]  
 Return an element-wise copy of the original vector.
- copy-matrix** (*matrix*) [Generic Function]  
 Return a copy of the matrix.
- Package** [linear-algebra], page 23.
- Source** [matrix.lisp], page 12.
- Methods**
- copy-matrix** ((*matrix* [*dense-matrix*], page 59)) [Method]  
 Return a copy of the dense matrix.
- Source** [dense-matrix.lisp], page 16.
- copy-matrix** ((*matrix* [*permutation-matrix*], page 62)) [Method]  
 Return a copy of the permutation matrix.
- Source** [permutation-matrix.lisp], page 13.
- copy-matrix** ((*matrix* [*identity-matrix*], page 61)) [Method]  
 Return a copy of the matrix.
- Source** [identity-matrix.lisp], page 12.
- copy-vector** (*vector*) [Generic Function]  
 Return a copy of the vector.
- Package** [linear-algebra], page 23.
- Source** [vector.lisp], page 11.
- Methods**
- copy-vector** ((*vector* [*data-vector*], page 58)) [Method]  
 Return a copy of the vector.
- Source** [data-vector.lisp], page 15.
- invert** (*matrix*) [Generic Function]  
 Return the invert of the matrix.
- Package** [linear-algebra], page 23.
- Source** [fundamental-ops.lisp], page 11.
- Methods**
- invert** ((*matrix* [*symmetric-matrix*], page 64)) [Method]  
 Return the invert of the symmetric matrix.
- Source** [symmetric-matrix.lisp], page 19.
- invert** ((*matrix* [*hermitian-matrix*], page 61)) [Method]  
 Return the invert of the hermitian matrix.
- Source** [hermitian-matrix.lisp], page 18.
- invert** ((*matrix* [*square-matrix*], page 64)) [Method]  
 Return the invert of the square matrix.
- Source** [square-matrix.lisp], page 18.

**invert** ((*matrix* [*dense-matrix*], page 59)) [Method]  
Return the invert of the dense matrix.

**Source** [dense-matrix.lisp], page 16.

**invert** ((*array* *array*)) [Method]  
Return the invert of the array.

**Source** [array.lisp], page 14.

**left-permute** (*permutation* *vector-or-array*) [Generic Function]  
Permute the column vector or rows of the array.

**Package** [linear-algebra-kernel], page 21.

**Source** [permute.lisp], page 8.

#### Methods

**left-permute** ((*permutation* *vector*) (*data* *array*)) [Method]  
Permute the rows of the array.

**left-permute** ((*permutation* *vector*) (*data* *vector*)) [Method]  
Permute the column vector to create a row vector.

**map-into-vector** (*result-vector* *function* &rest *vectors*) [Generic Function]  
Destructively modifies the result vector with the result of applying the function to each element of the vectors.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

#### Methods

**map-into-vector** ((*result-vector* [*data-vector*], page 58) (*function* *function*) &rest *vectors*) [Method]  
Destructively modifies the result vector with the result of applying the function to each element of the vectors.

**Source** [data-vector.lisp], page 15.

**map-into-vector:verify** ((*result-vector* [*data-vector*], page 58) (*function* *function*) &rest *vectors*) [Method]  
Verify the arguments to map-into-vector.

**Source** [data-vector.lisp], page 15.

**map-vector** (*result-type* *function* *first-vector* &rest *more-vectors*) [Generic Function]  
Calls function on successive sets of vector objects.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

#### Methods

**map-vector** (*result-type* (*function* *function*) (*first-vector* [*data-vector*], page 58) &rest *more-vectors*) [Method]  
Calls function on successive sets of data vectors.

**Source** [data-vector.lisp], page 15.



**map-vector** *:before (result-type (function function) [Method]*  
*(first-vector [data-vector], page 58) &rest more-vectors)*  
 Verify the arguments to map-vector.

**Source** [data-vector.lisp], page 15.

**matrix-column-dimension** (*matrix*) [Generic Function]  
 Return the number of columns in MATRIX.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**matrix-column-dimension** ((*matrix* [dense-matrix], [Method]  
*page 59*))

Return the number of columns in matrix.

**Source** [dense-matrix.lisp], page 16.

**matrix-column-dimension** ((*matrix* [permutation-matrix], [Method]  
*page 62*))

Return the number of columns in matrix.

**Source** [permutation-matrix.lisp], page 13.

**matrix-column-dimension** ((*identity-matrix* [Reader Method]  
*[identity-matrix], page 61*))  
 automatically generated reader method

**Source** [identity-matrix.lisp], page 12.

**Target Slot**  
 [size], page 62.

**matrix-dimensions** (*matrix*) [Generic Function]  
 Return the number of rows and columns in MATRIX.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**matrix-dimensions** ((*matrix* [dense-matrix], page 59)) [Method]  
 Return the number of rows and columns in matrix.

**Source** [dense-matrix.lisp], page 16.

**matrix-dimensions** ((*matrix* [permutation-matrix], [Method]  
*page 62*))

Return the number of rows and columns in matrix.

**Source** [permutation-matrix.lisp], page 13.

**matrix-dimensions** ((*matrix* [identity-matrix], page 61)) [Method]  
 Return the number of rows and columns in matrix.

**Source** [identity-matrix.lisp], page 12.

**matrix-element-type** (*matrix*) [Generic Function]  
 Return the element type of MATRIX.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**matrix-element-type** ((*matrix* [*dense-matrix*], *page 59*)) [Method]  
 Return the element type of the matrix.

**Source** [dense-matrix.lisp], page 16.

**matrix-element-type** ((*matrix* [*permutation-matrix*], *page 62*)) [Method]  
 Element type of the permutation matrix.

**Source** [permutation-matrix.lisp], page 13.

**matrix-element-type** ((*matrix* [*identity-matrix*], *page 61*)) [Method]  
 Return the element type of the identity matrix.

**Source** [identity-matrix.lisp], page 12.

**matrix-in-bounds-p** (*matrix row column*) [Generic Function]  
 Return true if ROW and COLUMN do not exceed the dimensions of MATRIX.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**matrix-in-bounds-p** ((*matrix* [*dense-matrix*], *page 59*) (row integer) (column integer)) [Method]  
 Return true if row and column do not exceed the dimensions of matrix.

**Source** [dense-matrix.lisp], page 16.

**matrix-in-bounds-p** ((*matrix* [*permutation-matrix*], *page 62*) (row integer) (column integer)) [Method]  
 Return true if row and column do not exceed the dimensions of matrix.

**Source** [permutation-matrix.lisp], page 13.

**matrix-in-bounds-p** ((*matrix* [*identity-matrix*], *page 61*) (row integer) (column integer)) [Method]  
 Return true if row and column do not exceed the dimensions of matrix.

**Source** [identity-matrix.lisp], page 12.

**matrix-row-dimension** (*matrix*) [Generic Function]  
 Return the number of rows in MATRIX.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**matrix-row-dimension** ((*matrix* [*dense-matrix*], *page 59*)) [Method]  
Return the number of rows in matrix.

**Source** [dense-matrix.lisp], page 16.

**matrix-row-dimension** ((*matrix* [*permutation-matrix*], *page 62*)) [Method]  
Return the number of rows in matrix.

**Source** [permutation-matrix.lisp], page 13.

**matrix-row-dimension** ((*identity-matrix* [*identity-matrix*], *page 61*)) [Reader Method]  
automatically generated reader method

**Source** [identity-matrix.lisp], page 12.

**Target Slot**  
[size], page 62.

**mref** (*matrix row column*) [Generic Function]  
Return the matrix element at ROW,COLUMN.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**mref** ((*matrix* [*dense-matrix*], *page 59*) (*row integer*) (*column integer*)) [Method]  
Return the element of matrix at row,column.

**Source** [dense-matrix.lisp], page 16.

**mref** ((*matrix* [*permutation-matrix*], *page 62*) (*row integer*) (*column integer*)) [Method]  
Return 1 if a permutation and 0 otherwise.

**Source** [permutation-matrix.lisp], page 13.

**mref** ((*matrix* [*identity-matrix*], *page 61*) (*row integer*) (*column integer*)) [Method]  
Return the element of the matrix at row,column.

**Source** [identity-matrix.lisp], page 12.

**(setf mref)** (*matrix row column*) [Generic Function]  
Set the element at row,column of matrix to data.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**(setf mref)** ((*matrix* [*symmetric-matrix*], *page 64*) (*row integer*) (*column integer*)) [Method]  
Set the element of matrix at row,column.

**Source** [symmetric-matrix.lisp], page 19.

(setf mref) ((matrix [hermitian-matrix], page 61) (row integer) (column integer)) [Method]

Set the element at row,column of matrix to data.

**Source** [hermitian-matrix.lisp], page 18.

(setf mref) ((matrix [dense-matrix], page 59) (row integer) (column integer)) [Method]

Set the element of matrix at row,column.

**Source** [dense-matrix.lisp], page 16.

(setf mref) ((matrix [permutation-matrix], page 62) (row integer) (column integer)) [Method]

Swap rows of the permutation matrix.

**Source** [permutation-matrix.lisp], page 13.

nadd (vector-or-matrix-1 vector-or-matrix-2 &key scalar1 scalar2) [Generic Function]  
Destructive vector or matrix addition.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

#### Methods

nadd ((matrix1 [symmetric-matrix], page 64) (matrix2 [symmetric-matrix], page 64) &key scalar1 scalar2) [Method]

**Source** [symmetric-matrix.lisp], page 19.

nadd ((matrix1 [symmetric-matrix], page 64) (matrix2 [dense-matrix], page 59) &key scalar1 scalar2) [Method]

Generate an error if a non-symmetric matrix is destructively added to a symmetric matrix.

**Source** [symmetric-matrix.lisp], page 19.

nadd ((matrix1 [dense-matrix], page 59) (matrix2 [dense-matrix], page 59) &key scalar1 scalar2) [Method]

Return the addition of the 2 matrices.

**Source** [dense-matrix.lisp], page 16.

nadd :before ((matrix1 [dense-matrix], page 59) (matrix2 [dense-matrix], page 59) &key scalar1 scalar2) [Method]

Audit the input data.

**Source** [dense-matrix.lisp], page 16.

nadd ((vector1 [row-vector], page 63) (vector2 [row-vector], page 63) &key scalar1 scalar2) [Method]

Return the addition of scalar2\*vector2 to scalar1\*vector1.

**Source** [data-vector.lisp], page 15.

nadd ((vector1 [column-vector], page 57) (vector2 [column-vector], page 57) &key scalar1 scalar2) [Method]

Return the addition of scalar2\*vector2 to scalar1\*vector1.

**Source** [data-vector.lisp], page 15.

**nadd** *:before* ((*vector1* [*data-vector*], page 58) (*vector2* [*data-vector*], page 58) &**key** *scalar1 scalar2*) [Method]  
 Verify that the dimensions are equal.

**Source** [data-vector.lisp], page 15.

**nadd** ((*array1 array*) (*array2 array*) &**key** *scalar1 scalar2*) [Method]  
 Destructively add array2 to array1.

**Source** [array.lisp], page 14.

**nadd** ((*vector1 vector*) (*vector2 vector*) &**key** *scalar1 scalar2*) [Method]  
 Return the addition of scalar2\*vector2 to scalar1\*vector1.

**Source** [vector.lisp], page 14.

**nadd** ((*list1 list*) (*list2 list*) &**key** *scalar1 scalar2*) [Method]  
 Return the addition of scalar2\*list2 to scalar1\*list1.

**Source** [list.lisp], page 13.

**napply-rotation** (*vector1 vector2 cc ss*) [Generic Function]  
 Return the plane rotations of vector1 and vector2 by cc and ss.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

#### Methods

**napply-rotation** ((*vector1* [*data-vector*], page 58) (*vector2* [*data-vector*], page 58) *cc ss*) [Method]  
 Return the plane rotations of vector1 and vector2 by cc and ss.

**Source** [data-vector.lisp], page 15.

**napply-rotation** *:before* ((*vector1* [*data-vector*], page 58) (*vector2* [*data-vector*], page 58) *cc ss*) [Method]  
 Verify the input to napply-rotation.

**Source** [data-vector.lisp], page 15.

**ninvert** (*matrix*) [Generic Function]  
 Return the invert of the matrix with in-place decomposition.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

#### Methods

**ninvert** ((*matrix* [*symmetric-matrix*], page 64)) [Method]  
 Return the invert of the symmetric matrix.

**Source** [symmetric-matrix.lisp], page 19.

**ninvert** ((*matrix* [*hermitian-matrix*], page 61)) [Method]  
 Return the invert of the hermitian matrix.

**Source** [hermitian-matrix.lisp], page 18.

**ninvert** ((*matrix* [*square-matrix*], page 64)) [Method]  
Return the invert of the square matrix.

**Source** [square-matrix.lisp], page 18.

**ninvert** ((*matrix* [*dense-matrix*], page 59)) [Method]  
Return the invert of the dense matrix.

**Source** [dense-matrix.lisp], page 16.

**ninvert** ((*array* *array*)) [Method]  
Return the invert of the array.

**Source** [array.lisp], page 14.

**norm** (*vector-or-matrix* **&optional** *measure*) [Generic Function]  
Return the norm according to measure.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

#### Methods

**norm** ((*matrix* [*dense-matrix*], page 59) **&optional** *measure*) [Method]  
Return the norm of the matrix.

**Source** [dense-matrix.lisp], page 16.

**norm** ((*vector* [*data-vector*], page 58) **&optional** *measure*) [Method]  
Return the p-norm of the vector.

**Source** [data-vector.lisp], page 15.

**norm** ((*data* *array*) **&optional** *measure*) [Method]  
Return the norm of the array.

**Source** [array.lisp], page 14.

**norm** ((*data* *vector*) **&optional** *measure*) [Method]  
**Source** [vector.lisp], page 14.

**norm** ((*data* *list*) **&optional** *measure*) [Method]  
**Source** [list.lisp], page 13.

**norm-array** (*data* *measure*) [Generic Function]  
Return the norm of the array according to the measure.

**Package** [linear-algebra-kernel], page 21.

**Source** [unary-operations.lisp], page 8.

#### Methods

**norm-array** ((*data* *array*) (*measure* (eql :infinity))) [Method]  
Return the infinity norm of the array.

**norm-array** ((*data* *array*) (*measure* (eql :frobenius))) [Method]  
Return the Frobenius norm of the array.

**norm-array** ((*data* *array*) (*measure* (eql :max))) [Method]  
Return the max norm of the array.

<b>norm-array</b> (( <i>data array</i> ) ( <i>measure</i> (eq1 1)))	[Method]
Return the 1 norm of the array.	
<b>norm-vector</b> ( <i>data measure</i> )	[Generic Function]
Return the norm of the vector according to the measure.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [unary-operations.lisp], page 8.	
<b>Methods</b>	
<b>norm-vector</b> (( <i>data vector</i> ) ( <i>measure</i> (eq1 :infinity)))	[Method]
Return the infinity, or maximum, norm of vector.	
<b>norm-vector</b> (( <i>data vector</i> ) ( <i>measure</i> integer))	[Method]
Return the p-norm of the vector.	
<b>norm-vector</b> (( <i>data vector</i> ) ( <i>measure</i> (eq1 2)))	[Method]
Return the Euclidean norm of the vector.	
<b>norm-vector</b> (( <i>data vector</i> ) ( <i>measure</i> (eq1 1)))	[Method]
Return the Taxicab norm of the list.	
<b>nscale</b> ( <i>scalar vector-or-matrix</i> )	[Generic Function]
Destructively scale each element by the scalar.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [fundamental-ops.lisp], page 11.	
<b>Methods</b>	
<b>nscale</b> (( <i>scalar number</i> ) ( <i>matrix</i> [dense-matrix], page 59))	[Method]
Scale each element of the dense matrix.	
<b>Source</b> [dense-matrix.lisp], page 16.	
<b>nscale</b> (( <i>scalar number</i> ) ( <i>vector</i> [data-vector], page 58))	[Method]
Return the vector destructively scaled by scalar.	
<b>Source</b> [data-vector.lisp], page 15.	
<b>nscale</b> (( <i>scalar number</i> ) ( <i>data array</i> ))	[Method]
Scale each element of the array.	
<b>Source</b> [array.lisp], page 14.	
<b>nscale</b> (( <i>scalar number</i> ) ( <i>data vector</i> ))	[Method]
Return the vector destructively scaled by scalar.	
<b>Source</b> [vector.lisp], page 14.	
<b>nscale</b> (( <i>scalar number</i> ) ( <i>data list</i> ))	[Method]
Return the list destructively scaled by scalar.	
<b>Source</b> [list.lisp], page 13.	
<b>nsolve</b> ( <i>matrix vector</i> )	[Generic Function]
Return the solution to the system of equations in-place.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [fundamental-ops.lisp], page 11.	
<b>Methods</b>	

**nsolve** ((*matrix* [*symmetric-matrix*], page 64) (*vector* [*column-vector*], page 57)) [Method]

Return the solution to the system of equations.

**Source** [*symmetric-matrix.lisp*], page 19.

**nsolve** ((*matrix* [*hermitian-matrix*], page 61) (*vector* [*column-vector*], page 57)) [Method]

Return the solution to the system of equations.

**Source** [*hermitian-matrix.lisp*], page 18.

**nsolve** ((*matrix* [*dense-matrix*], page 59) (*vector* [*column-vector*], page 57)) [Method]

Return the solution to the system of equations.

**Source** [*dense-matrix.lisp*], page 16.

**nsolve** :before ((*matrix* [*dense-matrix*], page 59) (*vector* [*column-vector*], page 57)) [Method]

Return the solution to the system of equations.

**Source** [*dense-matrix.lisp*], page 16.

**nsolve** ((*array* *array*) (*vector* *vector*)) [Method]

Return the solution to the system of equations.

**Source** [*array.lisp*], page 14.

**nsubtract** (*vector-or-matrix-1* *vector-or-matrix-2* &key *scalar1* *scalar2*) [Generic Function]

Destructive vector or matrix subtraction.

**Package** [*linear-algebra*], page 23.

**Source** [*fundamental-ops.lisp*], page 11.

#### Methods

**nsubtract** ((*matrix1* [*symmetric-matrix*], page 64) (*matrix2* [*symmetric-matrix*], page 64) &key *scalar1* *scalar2*) [Method]

**Source** [*symmetric-matrix.lisp*], page 19.

**nsubtract** ((*matrix1* [*symmetric-matrix*], page 64) (*matrix2* [*dense-matrix*], page 59) &key *scalar1* *scalar2*) [Method]

Generate an error if a non-symmetric matrix is destructively subtracted to a symmetric matrix.

**Source** [*symmetric-matrix.lisp*], page 19.

**nsubtract** ((*matrix1* [*dense-matrix*], page 59) (*matrix2* [*dense-matrix*], page 59) &key *scalar1* *scalar2*) [Method]

Return the addition of the 2 matrices.

**Source** [*dense-matrix.lisp*], page 16.

**nsubtract** :before ((*matrix1* [*dense-matrix*], page 59) (*matrix2* [*dense-matrix*], page 59) &key *scalar1* *scalar2*) [Method]

Audit the input data.

**Source** [*dense-matrix.lisp*], page 16.



- nsubtract** ((*vector1* [*row-vector*], page 63) (*vector2* [*row-vector*], page 63) &key *scalar1 scalar2*) [Method]  
 Return the subtraction of  $\text{scalar2} * \text{vector2}$  from  $\text{scalar1} * \text{vector1}$ .  
**Source** [data-vector.lisp], page 15.
- nsubtract** ((*vector1* [*column-vector*], page 57) (*vector2* [*column-vector*], page 57) &key *scalar1 scalar2*) [Method]  
 Return the subtraction of  $\text{scalar2} * \text{vector2}$  from  $\text{scalar1} * \text{vector1}$ .  
**Source** [data-vector.lisp], page 15.
- nsubtract** :before ((*vector1* [*data-vector*], page 58) (*vector2* [*data-vector*], page 58) &key *scalar1 scalar2*) [Method]  
 Verify that the dimensions are equal.  
**Source** [data-vector.lisp], page 15.
- nsubtract** ((*array1* array) (*array2* array) &key *scalar1 scalar2*) [Method]  
 Destructively subtract *array2* from *array1*.  
**Source** [array.lisp], page 14.
- nsubtract** ((*vector1* vector) (*vector2* vector) &key *scalar1 scalar2*) [Method]  
 Return the subtraction of  $\text{scalar2} * \text{vector2}$  from  $\text{scalar1} * \text{vector1}$ .  
**Source** [vector.lisp], page 14.
- nsubtract** ((*list1* list) (*list2* list) &key *scalar1 scalar2*) [Method]  
 Return the subtraction of  $\text{scalar2} * \text{list2}$  from  $\text{scalar1} * \text{list1}$ .  
**Source** [list.lisp], page 13.
- ntranspose** (*vector-or-matrix*) [Generic Function]  
 Destructively transpose the vector or matrix.  
**Package** [linear-algebra], page 23.  
**Source** [fundamental-ops.lisp], page 11.
- Methods**
- ntranspose** ((*matrix* [*hermitian-matrix*], page 61)) [Method]  
 The destructive transpose of a Hermitian matrix is itself.  
**Source** [hermitian-matrix.lisp], page 18.
- ntranspose** ((*matrix* [*dense-matrix*], page 59)) [Method]  
 Replace the contents of the dense matrix with the transpose.  
**Source** [dense-matrix.lisp], page 16.
- ntranspose** ((*vector* [*row-vector*], page 63)) [Method]  
 Return a column vector destructively.  
**Source** [data-vector.lisp], page 15.
- ntranspose** ((*vector* [*column-vector*], page 57)) [Method]  
 Return a row vector destructively.  
**Source** [data-vector.lisp], page 15.

**ntranspose** ((*data array*)) [Method]

Replace the contents of the array with the transpose.

**Source** [array.lisp], page 14.

**ntranspose** ((*data vector*)) [Method]

Return a row vector destructively.

**Source** [vector.lisp], page 14.

**ntranspose** ((*data list*)) [Method]

Return a row vector destructively.

**Source** [list.lisp], page 13.

**permute** (*vector-or-matrix-1 vector-or-matrix-2*) [Generic Function]

Permute the vector or matrix.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

#### Methods

**permute** ((*permutation* [*permutation-matrix*], *page 62*) (*matrix* [*hermitian-matrix*], *page 61*)) [Method]

**Source** [hermitian-matrix.lisp], page 18.

**permute** ((*matrix* [*hermitian-matrix*], *page 61*) (*permutation* [*permutation-matrix*], *page 62*)) [Method]

**Source** [hermitian-matrix.lisp], page 18.

**permute** ((*permutation* [*permutation-matrix*], *page 62*) (*matrix* [*dense-matrix*], *page 59*)) [Method]

**Source** [dense-matrix.lisp], page 16.

**permute** ((*matrix* [*dense-matrix*], *page 59*) (*permutation* [*permutation-matrix*], *page 62*)) [Method]

**Source** [dense-matrix.lisp], page 16.

**permute** ((*matrix* [*permutation-matrix*], *page 62*) (*vector* [*column-vector*], *page 57*)) [Method]

Return the permutation of the column vector.

**Source** [data-vector.lisp], page 15.

**permute** :before ((*matrix* [*permutation-matrix*], *page 62*) (*vector* [*column-vector*], *page 57*)) [Method]

Verify that the dimensions are compatible.

**Source** [data-vector.lisp], page 15.

**permute** ((*vector* [*row-vector*], *page 63*) (*matrix* [*permutation-matrix*], *page 62*)) [Method]

Return the permutation of the row vector.

**Source** [data-vector.lisp], page 15.

`permute` *:before* ((*vector* [*row-vector*], page 63) (*matrix* [*permutation-matrix*], page 62)) [Method]

Verify that the dimensions are compatible.

**Source** [data-vector.lisp], page 15.

`permute` ((*matrix* [*permutation-matrix*], page 62) (*data array*)) [Method]

**Source** [array.lisp], page 14.

`permute` ((*data array*) (*matrix* [*permutation-matrix*], page 62)) [Method]

**Source** [array.lisp], page 14.

`permute` ((*matrix* [*permutation-matrix*], page 62) (*data vector*)) [Method]

Return the permutation of the list.

**Source** [vector.lisp], page 14.

`permute` ((*data vector*) (*matrix* [*permutation-matrix*], page 62)) [Method]

Return the permutation of the list.

**Source** [vector.lisp], page 14.

`permute` ((*matrix* [*permutation-matrix*], page 62) (*data list*)) [Method]

Return the permutation of the list.

**Source** [list.lisp], page 13.

`permute` ((*data list*) (*matrix* [*permutation-matrix*], page 62)) [Method]

Return the permutation of the list.

**Source** [list.lisp], page 13.

`product` (*vector-or-matrix-1* *vector-or-matrix-2* **&optional** *scalar*) [Generic Function]

Return the vector-vector, matrix-vector or matrix-matrix product.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

#### Methods

`product` ((*matrix1* [*dense-matrix*], page 59) (*matrix2* [*dense-matrix*], page 59) **&optional** *scalar*) [Method]

Return the product of the dense matrices.

**Source** [dense-matrix.lisp], page 16.

`product` *:before* ((*matrix1* [*dense-matrix*], page 59) (*matrix2* [*dense-matrix*], page 59) **&optional** *scalar*) [Method]

Verify the input.

**Source** [dense-matrix.lisp], page 16.

- product** ((*matrix* [*dense-matrix*], page 59) (*vector* [column-vector], page 57) &optional *scalar*) [Method]  
 Return a column vector generated by the multiplication of the dense matrix with a column vector.  
**Source** [dense-matrix.lisp], page 16.
- product** :before ((*matrix* [*dense-matrix*], page 59) (*vector* [column-vector], page 57) &optional *scalar*) [Method]  
 Verify the input.  
**Source** [dense-matrix.lisp], page 16.
- product** ((*vector* [row-vector], page 63) (*matrix* [*dense-matrix*], page 59) &optional *scalar*) [Method]  
 Return a row vector generated by the pre-multiplication of a dense matrix by a row vector.  
**Source** [dense-matrix.lisp], page 16.
- product** :before ((*vector* [row-vector], page 63) (*matrix* [*dense-matrix*], page 59) &optional *scalar*) [Method]  
 Verify the inputs.  
**Source** [dense-matrix.lisp], page 16.
- product** ((*vector1* [row-vector], page 63) (*vector2* [column-vector], page 57) &optional *scalar*) [Method]  
 Return the dot product of vector1 and vector2.  
**Source** [data-vector.lisp], page 15.
- product** :before ((*vector1* [row-vector], page 63) (*vector2* [column-vector], page 57) &optional *scalar*) [Method]  
 Verify that the dimensions are equal.  
**Source** [data-vector.lisp], page 15.
- product** ((*array1* array) (*array2* array) &optional *scalar*) [Method]  
 Return the product of the arrays.  
**Source** [array.lisp], page 14.
- product** ((*array* array) (*vector* vector) &optional *scalar*) [Method]  
 Return a vector generated by the multiplication of the array with a vector.  
**Source** [array.lisp], page 14.
- product** ((*vector* vector) (*array* array) &optional *scalar*) [Method]  
 Return a vector generated by the pre-multiplication of a array by a vector.  
**Source** [array.lisp], page 14.
- product** ((*vector1* vector) (*vector2* vector) &optional *scalar*) [Method]  
 Return the dot product of vector1 and vector2.  
**Source** [vector.lisp], page 14.
- product** ((*list1* list) (*list2* list) &optional *scalar*) [Method]  
 Return the dot product of list1 and list2.  
**Source** [list.lisp], page 13.

**replace-matrix** (*matrix1 matrix2 &key start-row1 end-row1* [Generic Function]  
*start-column1 end-column1 start-row2 end-row2 start-column2 end-column2*)

Destructively replace elements of *matrix1* with *matrix2*.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

**Methods**

**replace-matrix** ((*matrix1* [symmetric-matrix], page 64) [Method]  
(*matrix2* [dense-matrix], page 59) &key *start-row1 end-row1*  
*start-column1 end-column1 start-row2 end-row2 start-column2*  
*end-column2*)

Replace the elements of *MATRIX1* with *MATRIX2*.

**Source** [symmetric-matrix.lisp], page 19.

**replace-matrix** ((*matrix1* [symmetric-matrix], page 64) [Method]  
(*matrix2* [symmetric-matrix], page 64) &key *start-row1*  
*end-row1 start-column1 end-column1 start-row2 end-row2*  
*start-column2 end-column2*)

Replace the elements of *MATRIX1* with *MATRIX2*.

**Source** [symmetric-matrix.lisp], page 19.

**replace-matrix** ((*matrix1* [hermitian-matrix], page 61) [Method]  
(*matrix2* [dense-matrix], page 59) &key *start-row1 end-row1*  
*start-column1 end-column1 start-row2 end-row2 start-column2*  
*end-column2*)

Replace the elements of *matrix1* with *matrix2*.

**Source** [hermitian-matrix.lisp], page 18.

**replace-matrix** ((*matrix1* [hermitian-matrix], page 61) [Method]  
(*matrix2* [hermitian-matrix], page 61) &key *start-row1*  
*end-row1 start-column1 end-column1 start-row2 end-row2*  
*start-column2 end-column2*)

Replace the elements of *matrix1* with *matrix2*.

**Source** [hermitian-matrix.lisp], page 18.

**replace-matrix** ((*matrix1* [dense-matrix], page 59) [Method]  
(*matrix2* [dense-matrix], page 59) &key *start-row1 end-row1*  
*start-column1 end-column1 start-row2 end-row2 start-column2*  
*end-column2*)

Replace the elements of *matrix1* with *matrix2*.

**Source** [dense-matrix.lisp], page 16.

**replace-vector** (*vector1 vector2 &key start1 end1 start2 end2*) [Generic Function]

Destructively replace the elements of *vector1* with *vector2*.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

**Methods**

**replace-vector** ((*vector1* [*data-vector*], page 58) (*vector2* [*data-vector*], page 58) &**key** *start1 end1 start2 end2*) [Method]

Destructively replace the elements of *vector1* with *vector2*.

**Source** [data-vector.lisp], page 15.

**right-permute** (*vector-or-array permutation*) [Generic Function]

Permute the row vector or columns of the array.

**Package** [linear-algebra-kernel], page 21.

**Source** [permute.lisp], page 8.

**Methods**

**right-permute** ((*data array*) (*permutation vector*)) [Method]

Permute the columns of the array.

**right-permute** ((*data vector*) (*permutation vector*)) [Method]

Permute the row vector to create a column vector.

**scale** (*scalar vector-or-matrix*) [Generic Function]

Scale each element by the scalar.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

**Methods**

**scale** ((*scalar number*) (*matrix* [*dense-matrix*], page 59)) [Method]

Scale each element of the dense matrix.

**Source** [dense-matrix.lisp], page 16.

**scale** ((*scalar number*) (*vector* [*data-vector*], page 58)) [Method]

Return the vector scaled by scalar.

**Source** [data-vector.lisp], page 15.

**scale** ((*scalar number*) (*data array*)) [Method]

Scale each element of the array.

**Source** [array.lisp], page 14.

**scale** ((*scalar number*) (*data vector*)) [Method]

Return the vector scaled by scalar.

**Source** [vector.lisp], page 14.

**scale** ((*scalar number*) (*data list*)) [Method]

Return the list scaled by scalar.

**Source** [list.lisp], page 13.

**scaled-binary-op** (*op scalar1 scalar2*) [Generic Function]

Compile and return a scaled binary operation.

**Package** [linear-algebra-kernel], page 21.

**Source** [binary-operations.lisp], page 8.

**Methods**

`scaled-binary-op ((op (eq1 #<function ->)) (scalar1 number) (scalar2 number))` [Method]

Return the scaled operation.

`scaled-binary-op ((op (eq1 #<function +>)) (scalar1 number) (scalar2 number))` [Method]

Return the scaled operation.

`scaled-binary-op ((op (eq1 #<function ->)) (scalar1 (eq1 nil)) (scalar2 number))` [Method]

Return the scaled operation.

`scaled-binary-op ((op (eq1 #<function +>)) (scalar1 (eq1 nil)) (scalar2 number))` [Method]

Return the scaled operation.

`scaled-binary-op ((op (eq1 #<function ->)) (scalar1 number) (scalar2 (eq1 nil)))` [Method]

Return the scaled operation.

`scaled-binary-op ((op (eq1 #<function +>)) (scalar1 number) (scalar2 (eq1 nil)))` [Method]

Return the scaled operation.

`scaled-binary-op (op (scalar1 (eq1 nil)) (scalar2 (eq1 nil)))` [Method]

Return the operation.

`solve (matrix vector)` [Generic Function]

Return the solution to the system of equations.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

**Methods**

`solve ((matrix [symmetric-matrix], page 64) (vector [column-vector], page 57))` [Method]

Return the solution to the system of equations.

**Source** [symmetric-matrix.lisp], page 19.

`solve ((matrix [hermitian-matrix], page 61) (vector [column-vector], page 57))` [Method]

Return the solution to the system of equations.

**Source** [hermitian-matrix.lisp], page 18.

`solve ((matrix [dense-matrix], page 59) (vector [column-vector], page 57))` [Method]

Return the solution to the system of equations.

**Source** [dense-matrix.lisp], page 16.

`solve :before ((matrix [dense-matrix], page 59) (vector [column-vector], page 57))` [Method]

Return the solution to the system of equations.

**Source** [dense-matrix.lisp], page 16.

**solve** ((*array array*) (*vector vector*)) [Method]

Return the solution to the system of equations.

**Source** [array.lisp], page 14.

**submatrix** (*matrix start-row start-column &key end-row end-column*) [Generic Function]

Return a submatrix of the matrix.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**submatrix** ((*matrix* [*symmetric-matrix*], page 64) (*start-row integer*) (*start-column integer*) &key *end-row end-column*) [Method]

Return a matrix created from the submatrix of matrix.

**Source** [symmetric-matrix.lisp], page 19.

**submatrix** ((*matrix* [*hermitian-matrix*], page 61) (*start-row integer*) (*start-column integer*) &key *end-row end-column*) [Method]

Return a matrix created from the submatrix of matrix.

**Source** [hermitian-matrix.lisp], page 18.

**submatrix** ((*matrix* [*square-matrix*], page 64) (*start-row integer*) (*start-column integer*) &key *end-row end-column*) [Method]

Return a matrix created from the submatrix of matrix.

**Source** [square-matrix.lisp], page 18.

**submatrix** ((*matrix* [*dense-matrix*], page 59) (*start-row integer*) (*start-column integer*) &key *end-row end-column*) [Method]

Return a dense matrix created from the submatrix of a matrix.

**Source** [dense-matrix.lisp], page 16.

**(setf submatrix)** (*matrix start-row start-column &key end-row end-column*) [Generic Function]

Set the submatrix of the matrix.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

#### Methods

**(setf submatrix)** ((*matrix* [*symmetric-matrix*], page 64) (*start-row integer*) (*start-column integer*) &key *end-row end-column*) [Method]

Set a submatrix of MATRIX.

**Source** [symmetric-matrix.lisp], page 19.

**(setf submatrix)** ((*matrix* [*symmetric-matrix*], page 64) (*start-row integer*) (*start-column integer*) &key *end-row end-column*) [Method]

Set a submatrix of the matrix.

**Source** [symmetric-matrix.lisp], page 19.



(setf submatrix) ((matrix [hermitian-matrix], page 61) [Method]  
 (start-row integer) (start-column integer) &key end-row  
 end-column)

Set a submatrix of the matrix.

**Source** [hermitian-matrix.lisp], page 18.

(setf submatrix) ((matrix [hermitian-matrix], page 61) [Method]  
 (start-row integer) (start-column integer) &key end-row  
 end-column)

Set a submatrix of the matrix.

**Source** [hermitian-matrix.lisp], page 18.

(setf submatrix) ((matrix [dense-matrix], page 59) [Method]  
 (start-row integer) (start-column integer) &key end-row  
 end-column)

Set the submatrix of matrix.

**Source** [dense-matrix.lisp], page 16.

subtract (vector-or-matrix-1 vector-or-matrix-2 &key scalar1 [Generic Function]  
 scalar2)

Vector or matrix binary subtraction.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

**Methods**

subtract ((matrix1 [dense-matrix], page 59) (matrix2 [Method]  
 [dense-matrix], page 59) &key scalar1 scalar2)

Return the addition of the 2 matrices.

**Source** [dense-matrix.lisp], page 16.

subtract :before ((matrix1 [dense-matrix], page 59) [Method]  
 (matrix2 [dense-matrix], page 59) &key scalar1 scalar2)

Audit the input data.

**Source** [dense-matrix.lisp], page 16.

subtract ((vector1 [row-vector], page 63) (vector2 [Method]  
 [row-vector], page 63) &key scalar1 scalar2)

Return the subtraction of scalar2\*vector2 from scalar1\*vector1.

**Source** [data-vector.lisp], page 15.

subtract ((vector1 [column-vector], page 57) (vector2 [Method]  
 [column-vector], page 57) &key scalar1 scalar2)

Return the subtraction of scalar2\*vector2 from scalar1\*vector1.

**Source** [data-vector.lisp], page 15.

subtract :before ((vector1 [data-vector], page 58) (vector2 [Method]  
 [data-vector], page 58) &key scalar1 scalar2)

Verify that the dimensions are equal.

**Source** [data-vector.lisp], page 15.

**subtract** ((*array1 array*) (*array2 array*) **&key** *scalar1* *scalar2*) [Method]

Return the subtraction of the 2 arrays.

**Source** [array.lisp], page 14.

**subtract** ((*vector1 vector*) (*vector2 vector*) **&key** *scalar1* *scalar2*) [Method]

Return the subtraction of *scalar2*\**vector2* from *scalar1*\**vector1*.

**Source** [vector.lisp], page 14.

**subtract** ((*list1 list*) (*list2 list*) **&key** *scalar1* *scalar2*) [Method]

Return the subtraction of *scalar2*\**list2* from *scalar1*\**list1*.

**Source** [list.lisp], page 13.

**subvector** (*vector start &optional end*) [Generic Function]

Return a new vector that is a subvector of the vector.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

**Methods**

**subvector** ((*vector [data-vector]*, page 58) *start &optional end*) [Method]

Return a new data vector that is a subset of vector.

**Source** [data-vector.lisp], page 15.

**(setf subvector)** (*vector start &optional end*) [Generic Function]

Set the subvector of the vector.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

**Methods**

**(setf subvector)** ((*vector [data-vector]*, page 58) *start &optional end*) [Method]

Set the subvector of the data vector.

**Source** [data-vector.lisp], page 15.

**sump** (*vector-or-array p &optional scale sump*) [Generic Function]

Return the scaling parameter and the sum of the P powers.

**Package** [linear-algebra-kernel], page 21.

**Source** [unary-operations.lisp], page 8.

**Methods**

**sump** ((*data array*) *p &optional scale sump*) [Method]

Return the scaling parameter and the sum of the P powers of the matrix.

**sump** ((*data vector*) *p &optional scale sump*) [Method]

Return the scaling parameter and the sum of the powers of p of the vector.

**sump** ((*data list*) (*p real*) &optional *scale sump*) [Method]  
 Return the scaling parameter and the sum of the powers of *p* of the data.

**sumsq** (*vector-or-array* &optional *scale sumsq*) [Generic Function]  
 Return the scaling parameter and the sum of the squares.

**Package** [linear-algebra-kernel], page 21.

**Source** [unary-operations.lisp], page 8.

#### Methods

**sumsq** ((*data array*) &optional *scale sumsq*) [Method]  
 Return the scaling parameter and the sum of the squares of the array.

**sumsq** ((*data vector*) &optional *scale sumsq*) [Method]  
 Return the scaling parameter and the sum of the squares of the vector.

**sumsq** ((*data list*) &optional *scale sumsq*) [Method]  
 Return the scaling parameter and the sum of the squares of the list.

**transpose** (*vector-or-matrix*) [Generic Function]  
 Transpose the vector or matrix.

**Package** [linear-algebra], page 23.

**Source** [fundamental-ops.lisp], page 11.

#### Methods

**transpose** ((*matrix* [*hermitian-matrix*], page 61)) [Method]  
 The transpose of a Hermitian matrix is itself.

**Source** [hermitian-matrix.lisp], page 18.

**transpose** ((*matrix* [*dense-matrix*], page 59)) [Method]  
 Return the transpose of the matrix.

**Source** [dense-matrix.lisp], page 16.

**transpose** ((*vector* [*row-vector*], page 63)) [Method]  
 Return a column vector.

**Source** [data-vector.lisp], page 15.

**transpose** ((*vector* [*column-vector*], page 57)) [Method]  
 Return a row vector.

**Source** [data-vector.lisp], page 15.

**transpose** ((*data array*)) [Method]  
 Return the transpose of the array.

**Source** [array.lisp], page 14.

**transpose** ((*data vector*)) [Method]  
 Return a row vector.

**Source** [vector.lisp], page 14.

**transpose** ((*data list*)) [Method]  
 Return a row vector.

**Source** [list.lisp], page 13.

**transpose** ((*matrix* [*permutation-matrix*], *page 62*)) [Method]  
 Transpose the permutation matrix.

**Source** [permutation-matrix.lisp], page 13.

**vector-element-type** (*vector*) [Generic Function]  
 Return the element type of vector.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

**Methods**

**vector-element-type** ((*vector* [*data-vector*], *page 58*)) [Method]  
 Return the element type of vector.

**Source** [data-vector.lisp], page 15.

**vector-in-bounds-p** (*vector index*) [Generic Function]  
 Return true if index does not exceed the dimensions of vector.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

**Methods**

**vector-in-bounds-p** ((*vector* [*data-vector*], *page 58*)  
 (*index integer*)) [Method]  
 Return true if index does not exceed the dimensions of vector.

**Source** [data-vector.lisp], page 15.

**vector-length** (*vector*) [Generic Function]  
 Return the length of the vector.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

**Methods**

**vector-length** ((*vector* [*data-vector*], *page 58*)) [Method]  
 Return the length of the vector.

**Source** [data-vector.lisp], page 15.

**vref** (*vector index*) [Generic Function]  
 Return the element of vector at index.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

**Methods**

**vref** ((*vector* [*data-vector*], *page 58*) (*index integer*)) [Method]  
 Return the element of vector at index.

**Source** [data-vector.lisp], page 15.

**(setf vref)** (*vector index*) [Generic Function]  
 Set the element of vector at index to data.

**Package** [linear-algebra], page 23.

**Source** [vector.lisp], page 11.

#### Methods

**(setf vref)** ((*vector [data-vector]*, page 58) (*index integer*)) [Method]

Set the element of vector at index to data.

**Source** [data-vector.lisp], page 15.

### 5.1.4 Standalone methods

**initialize-instance** :after ((*self [data-vector]*, page 58) **&rest** *initargs* [Method]  
**&key** *size element-type initial-element initial-contents*)

**Source** [data-vector.lisp], page 15.

**initialize-instance** :after ((*self [identity-matrix]*, page 61) **&rest** [Method]  
*initargs &key dimensions element-type initial-element initial-contents*)

Initialize the identity matrix.

**Source** [identity-matrix.lisp], page 12.

**initialize-instance** :after ((*self [dense-matrix]*, page 59) **&rest** [Method]  
*initargs &key dimensions element-type initial-element initial-contents*)

**Source** [dense-matrix.lisp], page 16.

**initialize-instance** :after ((*self [permutation-matrix]*, page 62) **&rest** [Method]  
*initargs &key dimensions element-type initial-element initial-contents*)

Verify that the element-type was not set and that rows equals columns.

**Source** [permutation-matrix.lisp], page 13.

### 5.1.5 Classes

**column-vector** [Class]  
 A column vector.

**Package** [linear-algebra], page 23.

**Source** [data-vector.lisp], page 15.

#### Direct superclasses

[data-vector], page 58.

#### Direct methods

- [add], page 33.
- [compatible-dimensions-p], page 34.
- [compatible-dimensions-p], page 34.
- [nadd], page 40.
- [nsolve], page 44.
- [nsolve], page 44.
- [nsolve], page 44.
- [nsolve], page 44.

- [nsubtract], page 45.
- [ntranspose], page 45.
- [permute], page 46.
- [permute], page 46.
- [product], page 48.
- [product], page 48.
- [product], page 48.
- [product], page 48.
- [solve], page 51.
- [solve], page 51.
- [solve], page 51.
- [solve], page 51.
- [subtract], page 53.
- [transpose], page 55.

**data-vector**

[Class]

A data vector.

**Package** [linear-algebra], page 23.**Source** [data-vector.lisp], page 15.**Direct subclasses**

- [column-vector], page 57.
- [row-vector], page 63.

**Direct methods**

- [add], page 33.
- [apply-rotation], page 33.
- [apply-rotation], page 33.
- [(setf contents)], page 70.
- [contents], page 70.
- [copy-vector], page 35.
- float-equal.
- float-equal.
- [initialize-instance], page 57.
- [map-into-vector], page 36.
- [map-into-vector], page 36.
- [map-vector], page 36.
- [map-vector], page 37.
- [nadd], page 41.
- [napply-rotation], page 41.
- [napply-rotation], page 41.
- [norm], page 42.
- [nscale], page 43.
- [nsubtract], page 45.
- rational-equal.

- `rational-equal`.
- `[replace-vector]`, page 50.
- `[scale]`, page 50.
- `[subtract]`, page 53.
- `[(setf subvector)]`, page 54.
- `[subvector]`, page 54.
- `[vector-element-type]`, page 56.
- `[vector-in-bounds-p]`, page 56.
- `[vector-length]`, page 56.
- `[(setf vref)]`, page 57.
- `[vref]`, page 56.

**Direct slots**

<code>contents</code>		[Slot]
<b>Type</b>	<code>(array * (*))</code>	
<b>Initargs</b>	<code>:contents</code>	
<b>Readers</b>	<code>[contents]</code> , page 70.	
<b>Writers</b>	<code>[(setf contents)]</code> , page 70.	

`dense-matrix` [Class]

Dense matrix object.

**Package** `[linear-algebra]`, page 23.

**Source** `[dense-matrix.lisp]`, page 16.

**Direct superclasses**

`[matrix-object]`, page 62.

**Direct subclasses**

`[square-matrix]`, page 64.

**Direct methods**

- `[add]`, page 32.
- `[add]`, page 33.
- `[compatible-dimensions-p]`, page 34.
- `[(setf contents)]`, page 70.
- `[contents]`, page 70.
- `[copy-matrix]`, page 35.
- `float-equal`.
- `float-equal`.
- `float-equal`.
- `float-equal`.
- `[initialize-instance]`, page 57.
- `[initialize-matrix-contents]`, page 72.
- `[initialize-matrix-contents]`, page 72.
- `[initialize-matrix-contents]`, page 72.
- `[initialize-matrix-contents]`, page 72.

- [invert], page 36.
- [matrix-column-dimension], page 37.
- [matrix-dimensions], page 37.
- [matrix-element-type], page 38.
- [matrix-in-bounds-p], page 38.
- [matrix-row-dimension], page 39.
- [(setf mref)], page 40.
- [mref], page 39.
- [nadd], page 40.
- [nadd], page 40.
- [nadd], page 40.
- [ninvert], page 42.
- [norm], page 42.
- [nscale], page 43.
- [nsolve], page 44.
- [nsolve], page 44.
- [nsubtract], page 44.
- [nsubtract], page 44.
- [nsubtract], page 44.
- [ntranspose], page 45.
- [permute], page 46.
- [permute], page 46.
- [product], page 47.
- [product], page 47.
- [product], page 48.
- [product], page 48.
- [product], page 48.
- [product], page 48.
- rational-equal.
- rational-equal.
- rational-equal.
- rational-equal.
- [replace-matrix], page 49.
- [replace-matrix], page 49.
- [replace-matrix], page 49.
- [scale], page 50.
- [solve], page 51.
- [solve], page 51.
- [(setf submatrix)], page 52.
- [(setf submatrix)], page 53.
- [(setf submatrix)], page 53.
- [submatrix], page 52.
- [subtract], page 53.



- [subtract], page 53.
- [transpose], page 55.

**Direct slots**

<b>contents</b>		[Slot]
<b>Type</b>	(array * (* *))	
<b>Initargs</b>	:contents	
<b>Readers</b>	[contents], page 70.	
<b>Writers</b>	[(setf contents)], page 70.	

**hermitian-matrix** [Class]

Hermitian matrix object.

**Package** [linear-algebra], page 23.

**Source** [hermitian-matrix.lisp], page 18.

**Direct superclasses**

[square-matrix], page 64.

**Direct subclasses**

[symmetric-matrix], page 64.

**Direct methods**

- [initialize-matrix-contents], page 71.
- [initialize-matrix-contents], page 71.
- [initialize-matrix-contents], page 71.
- [initialize-matrix-contents], page 72.
- [invert], page 35.
- [(setf mref)], page 40.
- [ninvert], page 41.
- [nsolve], page 44.
- [ntranspose], page 45.
- [permute], page 46.
- [permute], page 46.
- [replace-matrix], page 49.
- [replace-matrix], page 49.
- [solve], page 51.
- [(setf submatrix)], page 53.
- [(setf submatrix)], page 53.
- [submatrix], page 52.
- [transpose], page 55.

**identity-matrix** [Class]

Identity matrix object.

**Package** [linear-algebra], page 23.

**Source** [identity-matrix.lisp], page 12.

**Direct superclasses**

[matrix-object], page 62.

**Direct methods**

- [contents], page 70.
- [copy-matrix], page 35.
- [initialize-instance], page 57.
- [matrix-column-dimension], page 37.
- [matrix-dimensions], page 37.
- [matrix-element-type], page 38.
- [matrix-in-bounds-p], page 38.
- [matrix-row-dimension], page 39.
- [mref], page 39.
- [size], page 73.

**Direct slots**

<b>size</b>		[Slot]
<b>Type</b>	fixnum	
<b>Initargs</b>	:size	
<b>Readers</b>	<ul style="list-style-type: none"> <li>• [matrix-column-dimension], page 37.</li> <li>• [matrix-row-dimension], page 39.</li> <li>• [size], page 73.</li> </ul>	
<b>Writers</b>	<i>This slot is read-only.</i>	
<b>contents</b>		[Slot]
<b>Type</b>	(array * (2))	
<b>Initargs</b>	:contents	
<b>Readers</b>	[contents], page 70.	
<b>Writers</b>	<i>This slot is read-only.</i>	

**matrix-object** [Class]

A superclass for all matrices.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

**Direct subclasses**

- [dense-matrix], page 59.
- [identity-matrix], page 61.
- [permutation-matrix], page 62.

**permutation-matrix** [Class]

Permutation matrix object.

**Package** [linear-algebra], page 23.

**Source** [permutation-matrix.lisp], page 13.

**Direct superclasses**

[matrix-object], page 62.

**Direct methods**

- [(setf contents)], page 71.
- [contents], page 70.
- [copy-matrix], page 35.
- [initialize-instance], page 57.
- [initialize-matrix-contents], page 72.
- [initialize-matrix-contents], page 72.
- [initialize-matrix-contents], page 72.
- [matrix-column-dimension], page 37.
- [matrix-dimensions], page 37.
- [matrix-element-type], page 38.
- [matrix-in-bounds-p], page 38.
- [matrix-row-dimension], page 39.
- [(setf mref)], page 40.
- [mref], page 39.
- [permute], page 46.
- [permute], page 46.
- [permute], page 46.
- [permute], page 46.
- [permute], page 46.
- [permute], page 46.
- [permute], page 46.
- [permute], page 47.
- [permute], page 47.
- [permute], page 47.
- [permute], page 47.
- [permute], page 47.
- [permute], page 47.
- [permute], page 47.
- [transpose], page 56.

**Direct slots**

<b>contents</b>		[Slot]
<b>Type</b>	(array fixnum (*))	
<b>Initargs</b>	:contents	
<b>Readers</b>	[contents], page 70.	
<b>Writers</b>	[(setf contents)], page 71.	

**row-vector** [Class]

A row vector.

**Package** [linear-algebra], page 23.

**Source** [data-vector.lisp], page 15.

**Direct superclasses**

[data-vector], page 58.

**Direct methods**

- [add], page 33.
- [nadd], page 40.
- [nsubtract], page 45.
- [ntranspose], page 45.
- [permute], page 46.
- [permute], page 47.
- [product], page 48.
- [product], page 48.
- [product], page 48.
- [product], page 48.
- [subtract], page 53.
- [transpose], page 55.

**square-matrix**

[Class]

Square matrix object.

**Package** [linear-algebra], page 23.**Source** [square-matrix.lisp], page 18.**Direct superclasses**

[dense-matrix], page 59.

**Direct subclasses**

[hermitian-matrix], page 61.

**Direct methods**

- [compatible-dimensions-p], page 34.
- [initialize-matrix-contents], page 72.
- [invert], page 35.
- [ninvert], page 42.
- [submatrix], page 52.

**symmetric-matrix**

[Class]

Symmetric matrix object.

**Package** [linear-algebra], page 23.**Source** [symmetric-matrix.lisp], page 19.**Direct superclasses**

[hermitian-matrix], page 61.

**Direct methods**

- [initialize-matrix-contents], page 71.
- [initialize-matrix-contents], page 71.
- [initialize-matrix-contents], page 71.
- [invert], page 35.
- [(setf mref)], page 39.
- [nadd], page 40.
- [nadd], page 40.
- [ninvert], page 41.

- [nsolve], page 44.
- [nsubtract], page 44.
- [nsubtract], page 44.
- [replace-matrix], page 49.
- [replace-matrix], page 49.
- [solve], page 51.
- [(setf submatrix)], page 52.
- [(setf submatrix)], page 52.
- [submatrix], page 52.

## 5.2 Internals

### 5.2.1 Ordinary functions

<b>%abs-vector</b> ( <i>vector</i> )	[Function]
Return a vector containing absolute value of each element.	
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[unary-operations.lisp], page 8.
<b>%array1&lt;-array1-op-array2</b> ( <i>operation array1 array2</i> )	[Function]
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[binary-operations.lisp], page 8.
<b>%array&lt;-array1-op-array2</b> ( <i>operation array1 array2</i> )	[Function]
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[binary-operations.lisp], page 8.
<b>%default-cg-epsilon</b> ( <i>array vector</i> )	[Function]
Return a default epsilon for the conjugate gradient method.	
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[conjugate-gradient.lisp], page 10.
<b>%initialize-cg-residual</b> ( <i>array vector solution</i> )	[Function]
Return the initial residual vector for the conjugate gradient.	
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[conjugate-gradient.lisp], page 10.
<b>%initialize-cg-solution</b> ( <i>array</i> )	[Function]
Return an initial solution vector for the conjugate gradient.	
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[conjugate-gradient.lisp], page 10.
<b>%initialize-hermitian-matrix-with-seq</b> ( <i>matrix data dimensions element-type</i> )	[Function]
Initialize and validate a Hermitian matrix with a sequence.	
<b>Package</b>	[linear-algebra], page 23.
<b>Source</b>	[hermitian-matrix.lisp], page 18.

<code>%initialize-permutation-matrix-with-seq</code> ( <i>matrix data size</i> )	[Function]
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [permutation-matrix.lisp], page 13.	
<code>%initialize-symmetric-matrix-with-seq</code> ( <i>matrix data dimensions element-type</i> )	[Function]
Initialize and validate a symmetric matrix with a sequence.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [symmetric-matrix.lisp], page 19.	
<code>%map-data-vector</code> ( <i>result-type function first-vector &amp;rest more-vectors</i> )	[Function]
Non-validating version of map-vector.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [data-vector.lisp], page 15.	
<code>%map-into-data-vector</code> ( <i>result-vector function &amp;rest vectors</i> )	[Function]
Non-validating version of map-into-vector.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [data-vector.lisp], page 15.	
<code>%negative-residual</code> ( <i>residual</i> )	[Function]
Return the negative of the residual.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [conjugate-gradient.lisp], page 10.	
<code>%product-array-array</code> ( <i>array1 array2 &amp;optional result</i> )	[Function]
Return the result of the product of 2 arrays.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	
<code>%product-array-vector</code> ( <i>array vector &amp;optional result</i> )	[Function]
Return the result of the array postmultiplied by the vector.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	
<code>%product-vector-array</code> ( <i>vector array &amp;optional result</i> )	[Function]
Return the result of the array premultiplied by the vector.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	
<code>%replace-hermitian-matrix-off-diagonal</code> ( <i>matrix1 matrix2 row1 column1 row2 column2 numRows numcols</i> )	[Function]
Destructively replace a subset off the diagonal of matrix1 with matrix2.	
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [hermitian-matrix.lisp], page 18.	

<b>%replace-hermitian-matrix-on-diagonal</b> ( <i>matrix1 matrix2 row1 column1 row2 column2 numrows numcols</i> )	[Function]
Destructively replace a subset on the diagonal of <i>matrix1</i> with <i>matrix2</i> .	
<b>Package</b>	[linear-algebra], page 23.
<b>Source</b>	[hermitian-matrix.lisp], page 18.
<b>%replace-symmetric-matrix-off-diagonal</b> ( <i>matrix1 matrix2 row1 column1 row2 column2 numrows numcols</i> )	[Function]
Destructively replace a subset off the diagonal of <i>matrix1</i> with <i>matrix2</i> .	
<b>Package</b>	[linear-algebra], page 23.
<b>Source</b>	[symmetric-matrix.lisp], page 19.
<b>%replace-symmetric-matrix-on-diagonal</b> ( <i>matrix1 matrix2 row1 column1 row2 column2 numrows numcols</i> )	[Function]
Destructively replace a subset on the diagonal of <i>matrix1</i> with <i>matrix2</i> .	
<b>Package</b>	[linear-algebra], page 23.
<b>Source</b>	[symmetric-matrix.lisp], page 19.
<b>%scaled-product-array-array</b> ( <i>scalar array1 array2 &amp;optional result</i> )	[Function]
Return the scaled result of the product of 2 arrays.	
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[binary-operations.lisp], page 8.
<b>%scaled-product-array-vector</b> ( <i>scalar array vector &amp;optional result</i> )	[Function]
Return the result of the array postmultiplied by the vector and scaled.	
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[binary-operations.lisp], page 8.
<b>%scaled-product-vector-array</b> ( <i>scalar vector array &amp;optional result</i> )	[Function]
Return the result of the array premultiplied by the vector and scaled.	
<b>Package</b>	[linear-algebra-kernel], page 21.
<b>Source</b>	[binary-operations.lisp], page 8.
<b>%setf-hermitian-submatrix-off-diagonal</b> ( <i>matrix data row column numrows numcols</i> )	[Function]
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [hermitian-matrix.lisp], page 18.	
<b>%setf-hermitian-submatrix-on-diagonal</b> ( <i>matrix data row numrows</i> )	[Function]
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [hermitian-matrix.lisp], page 18.	
<b>%setf-symmetric-submatrix-off-diagonal</b> ( <i>matrix data row column numrows numcols</i> )	[Function]
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [symmetric-matrix.lisp], page 19.	

<b>%setf-symmetric-submatrix-on-diagonal</b> ( <i>matrix data row numrows</i> )	[Function]
<b>Package</b> [linear-algebra], page 23.	
<b>Source</b> [symmetric-matrix.lisp], page 19.	
<b>%vector1&lt;-vector1-op-vector2</b> ( <i>operation vector1 vector2</i> )	[Function]
Store the result of the binary operation in vector1.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	
<b>%vector&lt;-vector1-op-vector2</b> ( <i>operation vector1 vector2</i> )	[Function]
Store the result of the binary operation in a new vector.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [binary-operations.lisp], page 8.	
<b>column-pivot</b> ( <i>array pivot-selection-vector column</i> )	[Function]
Return the LR pivot of the array.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [gauss.lisp], page 9.	
<b>column-pivot-search</b> ( <i>array column</i> )	[Function]
Return the row index of the maximum value in the column.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [gauss.lisp], page 9.	
<b>gauss-backsubstitution</b> ( <i>factored solution</i> )	[Function]
Calculate the solution by backsubstitution.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [gauss.lisp], page 9.	
<b>gauss-factorization</b> ( <i>array</i> )	[Function]
Return the Gauss factorization of the array.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [gauss.lisp], page 9.	
<b>gauss-update</b> ( <i>factored pivot-selection-vector vector</i> )	[Function]
Update the solution vector.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [gauss.lisp], page 9.	
<b>initialize-pivot-selection-vector</b> ( <i>size</i> )	[Function]
Return a new, initialized, pivot vector.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [gauss.lisp], page 9.	
<b>swap-rows</b> ( <i>array i0 jth</i> )	[Function]
Interchange the	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [gauss.lisp], page 9.	



<b>tridiagonal-backsubstitution</b> ( <i>array vector</i> )	[Function]
Perform backsubstitution to obtain the solution.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [tridiagonal.lisp], page 10.	
<b>tridiagonal-factorization</b> ( <i>array</i> )	[Function]
Return the factorization of the tridiagonal array.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [tridiagonal.lisp], page 10.	
<b>tridiagonal-update</b> ( <i>array vector</i> )	[Function]
Update the solution vector using the factored array.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [tridiagonal.lisp], page 10.	
<b>unit-pivot-value</b> ( <i>pivot-selection-vector row column &amp;optional array-type</i> )	[Function]
Return 1.0 if column equals the value at row of the pivot selection vector, otherwise 0.0.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [gauss.lisp], page 9.	
<b>zero-array</b> ( <i>rows columns &amp;optional element-type</i> )	[Function]
Return an array of zeros.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [utility.lisp], page 7.	
<b>zero-vector</b> ( <i>size &amp;optional element-type</i> )	[Function]
Return a vector of zeros.	
<b>Package</b> [linear-algebra-kernel], page 21.	
<b>Source</b> [utility.lisp], page 7.	

### 5.2.2 Generic functions

<b>%norm</b> ( <i>data measure</i> )	[Generic Function]
<b>Package</b> [linear-algebra], page 23.	
<b>Methods</b>	
<b>%norm</b> (( <i>data list</i> ) ( <i>measure</i> (eql :infinity)))	[Method]
Return the infinity, or maximum, norm of vector.	
<b>Source</b> [list.lisp], page 13.	
<b>%norm</b> (( <i>data list</i> ) ( <i>measure integer</i> ))	[Method]
Return the p-norm of the vector.	
<b>Source</b> [list.lisp], page 13.	
<b>%norm</b> (( <i>data list</i> ) ( <i>measure</i> (eql 2)))	[Method]
Return the Euclidean norm of the vector.	
<b>Source</b> [list.lisp], page 13.	

<b>%norm</b> (( <i>data list</i> ) ( <i>measure</i> (eq1 1)))		[Method]
Return the Taxicab norm of the list.		
<b>Source</b> [list.lisp], page 13.		
<b>contents</b> ( <i>object</i> )		[Generic Reader]
<b>Package</b> [linear-algebra], page 23.		
<b>Methods</b>		
<b>contents</b> (( <i>dense-matrix</i> [ <i>dense-matrix</i> ], page 59))		[Reader Method]
automatically generated reader method		
<b>Source</b> [dense-matrix.lisp], page 16.		
<b>Target Slot</b>		
[contents], page 61.		
<b>contents</b> (( <i>data-vector</i> [ <i>data-vector</i> ], page 58))		[Reader Method]
automatically generated reader method		
<b>Source</b> [data-vector.lisp], page 15.		
<b>Target Slot</b>		
[contents], page 59.		
<b>contents</b> (( <i>permutation-matrix</i> [ <i>permutation-matrix</i> ], page 62))		[Reader Method]
automatically generated reader method		
<b>Source</b> [permutation-matrix.lisp], page 13.		
<b>Target Slot</b>		
[contents], page 63.		
<b>contents</b> (( <i>identity-matrix</i> [ <i>identity-matrix</i> ], page 61))		[Reader Method]
automatically generated reader method		
<b>Source</b> [identity-matrix.lisp], page 12.		
<b>Target Slot</b>		
[contents], page 62.		
<b>(setf contents)</b> ( <i>object</i> )		[Generic Writer]
<b>Package</b> [linear-algebra], page 23.		
<b>Methods</b>		
<b>(setf contents)</b> (( <i>dense-matrix</i> [ <i>dense-matrix</i> ], page 59))		[Writer Method]
automatically generated writer method		
<b>Source</b> [dense-matrix.lisp], page 16.		
<b>Target Slot</b>		
[contents], page 61.		
<b>(setf contents)</b> (( <i>data-vector</i> [ <i>data-vector</i> ], page 58))		[Writer Method]
automatically generated writer method		
<b>Source</b> [data-vector.lisp], page 15.		

**Target Slot**

[contents], page 59.

(setf contents) ((*permutation-matrix* [Writer Method]  
 [*permutation-matrix*], page 62))  
 automatically generated writer method

**Source** [permutation-matrix.lisp], page 13.

**Target Slot**

[contents], page 63.

**initialize-matrix-contents** (*matrix initial-contents initargs*) [Generic Function]  
 Initialize the matrix with data.

**Package** [linear-algebra], page 23.

**Source** [matrix.lisp], page 12.

**Methods**

**initialize-matrix-contents** ((*matrix* [Method]  
 [*symmetric-matrix*], page 64) (*initial-contents array*) *initargs*)  
 Initialize a symmetric matrix.

**Source** [symmetric-matrix.lisp], page 19.

**initialize-matrix-contents** ((*matrix* [Method]  
 [*symmetric-matrix*], page 64) (*initial-contents vector*)  
*initargs*)  
 Initialize a symmetric matrix.

**Source** [symmetric-matrix.lisp], page 19.

**initialize-matrix-contents** ((*matrix* [Method]  
 [*symmetric-matrix*], page 64) (*initial-contents list*) *initargs*)  
 Initialize a symmetric matrix.

**Source** [symmetric-matrix.lisp], page 19.

**initialize-matrix-contents** ((*matrix* [Method]  
 [*hermitian-matrix*], page 61) (*initial-contents array*) *initargs*)  
 Initialize the Hermitian matrix with a 2D array.

**Source** [hermitian-matrix.lisp], page 18.

**initialize-matrix-contents** ((*matrix* [Method]  
 [*hermitian-matrix*], page 61) (*initial-contents vector*)  
*initargs*)  
 Initialize the Hermitian matrix with a nested sequence.

**Source** [hermitian-matrix.lisp], page 18.

**initialize-matrix-contents** ((*matrix* [Method]  
 [*hermitian-matrix*], page 61) (*initial-contents list*) *initargs*)  
 Initialize the Hermitian matrix with a nested sequence.

**Source** [hermitian-matrix.lisp], page 18.

`initialize-matrix-contents` ((*matrix* [Method]  
[*hermitian-matrix*], page 61) (*initial-element* **complex**)  
*initargs*)

It is an error to initialize a Hermitian matrix with a complex element.

**Source** [hermitian-matrix.lisp], page 18.

`initialize-matrix-contents :before` ((*matrix* [Method]  
[*square-matrix*], page 64) (*initial-contents* *initargs*)

Verify that the number of rows and columns are equal.

**Source** [square-matrix.lisp], page 18.

`initialize-matrix-contents` ((*matrix* [*dense-matrix*], [Method]  
page 59) (*initial-contents* **array**) *initargs*)

Verify that the size of the data is valid.

**Source** [dense-matrix.lisp], page 16.

`initialize-matrix-contents` ((*matrix* [*dense-matrix*], [Method]  
page 59) (*initial-contents* **vector**) *initargs*)

Initialize the dense matrix with a nested sequence.

**Source** [dense-matrix.lisp], page 16.

`initialize-matrix-contents` ((*matrix* [*dense-matrix*], [Method]  
page 59) (*initial-contents* **list**) *initargs*)

Initialize the dense matrix with a nested sequence.

**Source** [dense-matrix.lisp], page 16.

`initialize-matrix-contents` ((*matrix* [*dense-matrix*], [Method]  
page 59) (*initial-element* **number**) *initargs*)

Initialize the dense matrix with an initial element.

**Source** [dense-matrix.lisp], page 16.

`initialize-matrix-contents` ((*matrix* [Method]  
[*permutation-matrix*], page 62) (*initial-contents* **array**)  
*initargs*)

Initialize the permutation matrix with a 2D array.

**Source** [permutation-matrix.lisp], page 13.

`initialize-matrix-contents` ((*matrix* [Method]  
[*permutation-matrix*], page 62) (*initial-contents* **vector**)  
*initargs*)

Initialize the permutation matrix with a list.

**Source** [permutation-matrix.lisp], page 13.

`initialize-matrix-contents` ((*matrix* [Method]  
[*permutation-matrix*], page 62) (*initial-contents* **list**)  
*initargs*)

Initialize the permutation matrix with a list.

**Source** [permutation-matrix.lisp], page 13.

`size` (*object*) [Generic Reader]

**Package** [linear-algebra], page 23.

**Methods**

**size** ((*identity-matrix* [*identity-matrix*], page 61)) [Reader Method]  
automatically generated reader method

**Source** [identity-matrix.lisp], page 12.

**Target Slot**  
[size], page 62.



## Appendix A Indexes

### A.1 Concepts

(Index is nonexistent)

## A.2 Functions

### %

%abs-vector	65
%array<-array1-op-array2	65
%array1<-array1-op-array2	65
%default-cg-epsilon	65
%initialize-cg-residual	65
%initialize-cg-solution	65
%initialize-hermitian-matrix-with-seq	65
%initialize-permutation-matrix-with-seq	66
%initialize-symmetric-matrix-with-seq	66
%map-data-vector	66
%map-into-data-vector	66
%negative-residual	66
%norm	69, 70
%product-array-array	66
%product-array-vector	66
%product-vector-array	66
%replace-hermitian-matrix-off-diagonal	66
%replace-hermitian-matrix-on-diagonal	67
%replace-symmetric-matrix-off-diagonal	67
%replace-symmetric-matrix-on-diagonal	67
%scaled-product-array-array	67
%scaled-product-array-vector	67
%scaled-product-vector-array	67
%setf-hermitian-submatrix-off-diagonal	67
%setf-hermitian-submatrix-on-diagonal	67
%setf-symmetric-submatrix-off-diagonal	67
%setf-symmetric-submatrix-on-diagonal	68
%vector<-vector1-op-vector2	68
%vector1<-vector1-op-vector2	68

### (

(setf contents)	70, 71
(setf mref)	39, 40
(setf submatrix)	52, 53
(setf subvector)	54
(setf vref)	57

### A

add	32, 33
add-array	27
add-vector	27
apply-rotation	33

### C

column-pivot	68
column-pivot-search	68
column-vector	27
column-vector-p	27
common-array-element-type	27
common-class-of	27
compatible-dimensions-p	34
complex-equal	27
conjugate-gradient-solver	28
contents	70
copy-array	34, 35
copy-matrix	35
copy-vector	35

### D

dense-matrix-p	28
dovector	27

### F

Function, %abs-vector	65
Function, %array<-array1-op-array2	65
Function, %array1<-array1-op-array2	65
Function, %default-cg-epsilon	65
Function, %initialize-cg-residual	65
Function, %initialize-cg-solution	65
Function, %initialize-hermitian-matrix-with-seq	65
Function, %initialize-permutation-matrix-with-seq	66
Function, %initialize-symmetric-matrix-with-seq	66
Function, %map-data-vector	66
Function, %map-into-data-vector	66
Function, %negative-residual	66
Function, %product-array-array	66
Function, %product-array-vector	66
Function, %product-vector-array	66
Function, %replace-hermitian-matrix-off-diagonal	66
Function, %replace-hermitian-matrix-on-diagonal	67
Function, %replace-symmetric-matrix-off-diagonal	67
Function, %replace-symmetric-matrix-on-diagonal	67
Function, %scaled-product-array-array	67
Function, %scaled-product-array-vector	67
Function, %scaled-product-vector-array	67
Function, %setf-hermitian-submatrix-off-diagonal	67
Function, %setf-hermitian-submatrix-on-diagonal	67
Function, %setf-symmetric-submatrix-off-diagonal	67
Function, %setf-symmetric-submatrix-on-diagonal	68
Function, %vector<-vector1-op-vector2	68
Function, %vector1<-vector1-op-vector2	68
Function, add-array	27
Function, add-vector	27
Function, column-pivot	68
Function, column-pivot-search	68
Function, column-vector	27
Function, column-vector-p	27
Function, common-array-element-type	27
Function, common-class-of	27
Function, complex-equal	27
Function, conjugate-gradient-solver	28
Function, dense-matrix-p	28
Function, gauss-backsubstitution	68
Function, gauss-factorization	68
Function, gauss-invert	28
Function, gauss-solver	28
Function, gauss-update	68
Function, givens-rotation	28



Function, hermitian-cholesky-decomposition...	28
Function, hermitian-cholesky-invert.....	28
Function, hermitian-cholesky-solver.....	28
Function, hermitian-matrix-p.....	28
Function, householder-reflection.....	29
Function, identity-matrix-p.....	29
Function, initialize-pivot-selection-vector..	68
Function, inner-product-vector.....	29
Function, jacobi-rotation.....	29
Function, make-matrix.....	29
Function, make-vector.....	29
Function, matrix-validated-range.....	29
Function, matrixp.....	29
Function, nadd-array.....	29
Function, nadd-vector.....	30
Function, nsubtract-array.....	30
Function, nsubtract-vector.....	30
Function, number-equal.....	30
Function, permutation-matrix-p.....	30
Function, product-array-array.....	30
Function, product-array-vector.....	30
Function, product-vector-array.....	30
Function, root-free-hermitian-cholesky-decomposition.....	30
Function, root-free-symmetric-cholesky-decomposition.....	31
Function, row-vector.....	31
Function, row-vector-p.....	31
Function, specific-array-element-type.....	31
Function, square-matrix-p.....	31
Function, subtract-array.....	31
Function, subtract-vector.....	31
Function, sumsq-column.....	31
Function, sumsq-row.....	31
Function, sumsq2.....	32
Function, sumsq3.....	32
Function, swap-rows.....	68
Function, symmetric-cholesky-decomposition...	32
Function, symmetric-cholesky-invert.....	32
Function, symmetric-cholesky-solver.....	32
Function, symmetric-matrix-p.....	32
Function, tridiagonal-backsubstitution.....	69
Function, tridiagonal-factorization.....	69
Function, tridiagonal-solver.....	32
Function, tridiagonal-update.....	69
Function, unit-pivot-value.....	69
Function, zero-array.....	69
Function, zero-vector.....	69

## G

gauss-backsubstitution.....	68
gauss-factorization.....	68
gauss-invert.....	28
gauss-solver.....	28
gauss-update.....	68
Generic Function, %norm.....	69
Generic Function, (setf contents).....	70
Generic Function, (setf mref).....	39
Generic Function, (setf submatrix).....	52
Generic Function, (setf subvector).....	54
Generic Function, (setf vref).....	57
Generic Function, add.....	32
Generic Function, apply-rotation.....	33

Generic Function, compatible-dimensions-p.....	34
Generic Function, contents.....	70
Generic Function, copy-array.....	34
Generic Function, copy-matrix.....	35
Generic Function, copy-vector.....	35
Generic Function, initialize-matrix-contents.....	71
Generic Function, invert.....	35
Generic Function, left-permute.....	36
Generic Function, map-into-vector.....	36
Generic Function, map-vector.....	36
Generic Function, matrix-column-dimension.....	37
Generic Function, matrix-dimensions.....	37
Generic Function, matrix-element-type.....	38
Generic Function, matrix-in-bounds-p.....	38
Generic Function, matrix-row-dimension.....	38
Generic Function, mref.....	39
Generic Function, nadd.....	40
Generic Function, napply-rotation.....	41
Generic Function, ninvert.....	41
Generic Function, norm.....	42
Generic Function, norm-array.....	42
Generic Function, norm-vector.....	43
Generic Function, nscale.....	43
Generic Function, nsolve.....	43
Generic Function, nsubtract.....	44
Generic Function, ntranspose.....	45
Generic Function, permute.....	46
Generic Function, product.....	47
Generic Function, replace-matrix.....	49
Generic Function, replace-vector.....	49
Generic Function, right-permute.....	50
Generic Function, scale.....	50
Generic Function, scaled-binary-op.....	50
Generic Function, size.....	72
Generic Function, solve.....	51
Generic Function, submatrix.....	52
Generic Function, subtract.....	53
Generic Function, subvector.....	54
Generic Function, sump.....	54
Generic Function, sumsq.....	55
Generic Function, transpose.....	55
Generic Function, vector-element-type.....	56
Generic Function, vector-in-bounds-p.....	56
Generic Function, vector-length.....	56
Generic Function, vref.....	56
givens-rotation.....	28

## H

hermitian-cholesky-decomposition.....	28
hermitian-cholesky-invert.....	28
hermitian-cholesky-solver.....	28
hermitian-matrix-p.....	28
householder-reflection.....	29

## I

identity-matrix-p.....	29
initialize-instance.....	57
initialize-matrix-contents.....	71, 72
initialize-pivot-selection-vector.....	68
inner-product-vector.....	29
invert.....	35, 36

**J**

jacobi-rotation..... 29

**L**

left-permute..... 36

**M**

Macro, dovector..... 27  
 make-matrix..... 29  
 make-vector..... 29  
 map-into-vector..... 36  
 map-vector..... 36, 37  
 matrix-column-dimension..... 37  
 matrix-dimensions..... 37  
 matrix-element-type..... 38  
 matrix-in-bounds-p..... 38  
 matrix-row-dimension..... 38, 39  
 matrix-validated-range..... 29  
 matrixp..... 29  
 Method, %norm..... 69, 70  
 Method, (setf contents)..... 70, 71  
 Method, (setf mref)..... 39, 40  
 Method, (setf submatrix)..... 52, 53  
 Method, (setf subvector)..... 54  
 Method, (setf vref)..... 57  
 Method, add..... 32, 33  
 Method, apply-rotation..... 33  
 Method, compatible-dimensions-p..... 34  
 Method, contents..... 70  
 Method, copy-array..... 34, 35  
 Method, copy-matrix..... 35  
 Method, copy-vector..... 35  
 Method, initialize-instance..... 57  
 Method, initialize-matrix-contents..... 71, 72  
 Method, invert..... 35, 36  
 Method, left-permute..... 36  
 Method, map-into-vector..... 36  
 Method, map-vector..... 36, 37  
 Method, matrix-column-dimension..... 37  
 Method, matrix-dimensions..... 37  
 Method, matrix-element-type..... 38  
 Method, matrix-in-bounds-p..... 38  
 Method, matrix-row-dimension..... 39  
 Method, mref..... 39  
 Method, nadd..... 40, 41  
 Method, napply-rotation..... 41  
 Method, ninvert..... 41, 42  
 Method, norm..... 42  
 Method, norm-array..... 42, 43  
 Method, norm-vector..... 43  
 Method, nscale..... 43  
 Method, nsolve..... 44  
 Method, nsubtract..... 44, 45  
 Method, ntranspose..... 45, 46  
 Method, permute..... 46, 47  
 Method, product..... 47, 48  
 Method, replace-matrix..... 49  
 Method, replace-vector..... 50  
 Method, right-permute..... 50  
 Method, scale..... 50  
 Method, scaled-binary-op..... 51  
 Method, size..... 73

Method, solve..... 51, 52  
 Method, submatrix..... 52  
 Method, subtract..... 53, 54  
 Method, subvector..... 54  
 Method, sump..... 54, 55  
 Method, sumsq..... 55  
 Method, transpose..... 55, 56  
 Method, vector-element-type..... 56  
 Method, vector-in-bounds-p..... 56  
 Method, vector-length..... 56  
 Method, vref..... 56  
 mref..... 39

**N**

nadd..... 40, 41  
 nadd-array..... 29  
 nadd-vector..... 30  
 napply-rotation..... 41  
 ninvert..... 41, 42  
 norm..... 42  
 norm-array..... 42, 43  
 norm-vector..... 43  
 nscale..... 43  
 nsolve..... 43, 44  
 nsubtract..... 44, 45  
 nsubtract-array..... 30  
 nsubtract-vector..... 30  
 ntranspose..... 45, 46  
 number-equal..... 30

**P**

permutation-matrix-p..... 30  
 permute..... 46, 47  
 product..... 47, 48  
 product-array-array..... 30  
 product-array-vector..... 30  
 product-vector-array..... 30

**R**

replace-matrix..... 49  
 replace-vector..... 49, 50  
 right-permute..... 50  
 root-free-hermitian-cholesky-  
   decomposition..... 30  
 root-free-symmetric-cholesky-  
   decomposition..... 31  
 row-vector..... 31  
 row-vector-p..... 31

**S**

scale .....	50
scaled-binary-op .....	50, 51
size .....	72, 73
solve .....	51, 52
specific-array-element-type .....	31
square-matrix-p .....	31
submatrix .....	52
subtract .....	53, 54
subtract-array .....	31
subtract-vector .....	31
subvector .....	54
sump .....	54, 55
sumsq .....	55
sumsq-column .....	31
sumsq-row .....	31
sumsq2 .....	32
sumsq3 .....	32
swap-rows .....	68
symmetric-cholesky-decomposition .....	32
symmetric-cholesky-invert .....	32
symmetric-cholesky-solver .....	32
symmetric-matrix-p .....	32

**T**

transpose .....	55, 56
tridiagonal-backsubstitution .....	69
tridiagonal-factorization .....	69
tridiagonal-solver .....	32
tridiagonal-update .....	69

**U**

unit-pivot-value .....	69
------------------------	----

**V**

vector-element-type .....	56
vector-in-bounds-p .....	56
vector-length .....	56
vref .....	56

**Z**

zero-array .....	69
zero-vector .....	69

## A.3 Variables

### C

`contents` ..... 59, 61, 62, 63

### S

`size` ..... 62  
`Slot, contents` ..... 59, 61, 62, 63  
`Slot, size` ..... 62

## A.4 Data types

### A

array.lisp ..... 14

### B

binary-operations.lisp ..... 8

### C

cholesky.lisp ..... 10  
 Class, column-vector ..... 57  
 Class, data-vector ..... 58  
 Class, dense-matrix ..... 59  
 Class, hermitian-matrix ..... 61  
 Class, identity-matrix ..... 61  
 Class, matrix-object ..... 62  
 Class, permutation-matrix ..... 62  
 Class, row-vector ..... 63  
 Class, square-matrix ..... 64  
 Class, symmetric-matrix ..... 64  
 column-vector ..... 57  
 conjugate-gradient.lisp ..... 10

### D

data-vector ..... 58  
 data-vector.lisp ..... 15  
 dense-matrix ..... 59  
 dense-matrix.lisp ..... 16

### F

File, array.lisp ..... 14  
 File, binary-operations.lisp ..... 8  
 File, cholesky.lisp ..... 10  
 File, conjugate-gradient.lisp ..... 10  
 File, data-vector.lisp ..... 15  
 File, dense-matrix.lisp ..... 16  
 File, fundamental-ops.lisp ..... 11  
 File, gauss.lisp ..... 9  
 File, hermitian-matrix.lisp ..... 18  
 File, identity-matrix.lisp ..... 12  
 File, linear-algebra.asd ..... 7  
 File, list.lisp ..... 13  
 File, matrix.lisp ..... 12  
 File, permutation-matrix.lisp ..... 13  
 File, permute.lisp ..... 8  
 File, pkgdcl.lisp ..... 7  
 File, rotation.lisp ..... 9  
 File, square-matrix.lisp ..... 18  
 File, symmetric-matrix.lisp ..... 19  
 File, tridiagonal.lisp ..... 10  
 File, unary-operations.lisp ..... 8  
 File, utility.lisp ..... 7  
 File, vector.lisp ..... 11, 14  
 fundamental-ops.lisp ..... 11

### G

gauss.lisp ..... 9

### H

hermitian-matrix ..... 61  
 hermitian-matrix.lisp ..... 18

### I

identity-matrix ..... 61  
 identity-matrix.lisp ..... 12  
 interface ..... 5

### K

kernel ..... 5

### L

linear-algebra ..... 3, 23  
 linear-algebra-kernel ..... 21  
 linear-algebra.asd ..... 7  
 list.lisp ..... 13

### M

matrix-object ..... 62  
 matrix.lisp ..... 12  
 Module, interface ..... 5  
 Module, kernel ..... 5  
 Module, sequence ..... 5

### P

Package, linear-algebra ..... 23  
 Package, linear-algebra-kernel ..... 21  
 permutation-matrix ..... 62  
 permutation-matrix.lisp ..... 13  
 permute.lisp ..... 8  
 pkgdcl.lisp ..... 7

### R

rotation.lisp ..... 9  
 row-vector ..... 63

### S

sequence ..... 5  
 square-matrix ..... 64  
 square-matrix.lisp ..... 18  
 symmetric-matrix ..... 64  
 symmetric-matrix.lisp ..... 19  
 System, linear-algebra ..... 3

### T

tridiagonal.lisp ..... 10

### U

unary-operations.lisp ..... 8  
 utility.lisp ..... 7

**V**

`vector.lisp`..... 11, 14