**Overview**

* [matrix, vector, cube and field classes](http://arma.sourceforge.net/docs.html#part_classes)
* [member functions & variables](http://arma.sourceforge.net/docs.html#part_membfns)

* [generated vectors / matrices / cubes](http://arma.sourceforge.net/docs.html#part_gen)
* [functions of vectors / matrices / cubes](http://arma.sourceforge.net/docs.html#part_fns)

* [decompositions, factorisations, inverses and equation solvers (dense matrices)](http://arma.sourceforge.net/docs.html#part_decompdense)
* [decompositions, factorisations, and equation solvers (sparse matrices)](http://arma.sourceforge.net/docs.html#part_decompsparse)

* [signal & image processing](http://arma.sourceforge.net/docs.html#part_sigproc)
* [statistics and clustering](http://arma.sourceforge.net/docs.html#part_stats)
* [miscellaneous (constants, configuration)](http://arma.sourceforge.net/docs.html#part_misc)

**Matrix, Vector, Cube and Field Classes**

|  |  |  |
| --- | --- | --- |
| [Mat<*type*>, mat, cx\_mat](http://arma.sourceforge.net/docs.html#Mat) |  | dense matrix class |
| [Col<*type*>, colvec, vec](http://arma.sourceforge.net/docs.html#Col) |  | dense column vector class |
| [Row<*type*>, rowvec](http://arma.sourceforge.net/docs.html#Row) |  | dense row vector class |
|  |  |  |
| [Cube<*type*>, cube, cx\_cube](http://arma.sourceforge.net/docs.html#Cube) |  | dense cube class ("3D matrix") |
| [field<*object type*>](http://arma.sourceforge.net/docs.html#field) |  | class for storing arbitrary objects in matrix-like or cube-like layouts |
| [SpMat<*type*>, sp\_mat, sp\_cx\_mat](http://arma.sourceforge.net/docs.html#SpMat) |  | sparse matrix class |
|  |  |  |
| [operators](http://arma.sourceforge.net/docs.html#operators) |  | +  -  \*  /  %  ==  !=  <=  >=  <  > |

**Member Functions & Variables**

|  |  |  |
| --- | --- | --- |
| [attributes](http://arma.sourceforge.net/docs.html#attributes) |  | .n\_rows, .n\_cols, .n\_elem, .n\_slices, ... |
| [element access](http://arma.sourceforge.net/docs.html#element_access) |  | element/object access via (), [] and .at() |
| [element initialisation](http://arma.sourceforge.net/docs.html#element_initialisation) |  | set elements via << operator or initialiser list |
|  |  |  |
| [.zeros](http://arma.sourceforge.net/docs.html#zeros_member) |  | set all elements to zero |
| [.ones](http://arma.sourceforge.net/docs.html#ones_member) |  | set all elements to one |
| [.eye](http://arma.sourceforge.net/docs.html#eye_member) |  | set elements along main diagonal to one and off-diagonal elements to zero |
| [.randu / .randn](http://arma.sourceforge.net/docs.html#randu_randn_member) |  | set all elements to random values |
|  |  |  |
| [.fill](http://arma.sourceforge.net/docs.html#fill) |  | set all elements to specified value |
| [.imbue](http://arma.sourceforge.net/docs.html#imbue) |  | imbue (fill) with values provided by functor or lambda function |
|  |  |  |
| [.replace](http://arma.sourceforge.net/docs.html#replace) |  | replace specific elements with a new value |
| [.transform](http://arma.sourceforge.net/docs.html#transform) |  | transform each element via functor or lambda function |
| [.for\_each](http://arma.sourceforge.net/docs.html#for_each) |  | apply a functor or lambda function to each element |
|  |  |  |
| [.set\_size](http://arma.sourceforge.net/docs.html#set_size) |  | change size without keeping elements (fast) |
| [.reshape](http://arma.sourceforge.net/docs.html#reshape_member) |  | change size while keeping elements |
| [.resize](http://arma.sourceforge.net/docs.html#resize_member) |  | change size while keeping elements and preserving layout |
| [.copy\_size](http://arma.sourceforge.net/docs.html#copy_size) |  | change size to be same as given object |
| [.reset](http://arma.sourceforge.net/docs.html#reset) |  | change size to empty |
|  |  |  |
| [submatrix views](http://arma.sourceforge.net/docs.html#submat) |  | read/write access to contiguous and non-contiguous submatrices |
| [subcube views](http://arma.sourceforge.net/docs.html#subcube) |  | read/write access to contiguous and non-contiguous subcubes |
| [subfield views](http://arma.sourceforge.net/docs.html#subfield) |  | read/write access to contiguous subfields |
|  |  |  |
| [.diag](http://arma.sourceforge.net/docs.html#diag) |  | read/write access to matrix diagonals |
| [.each\_col / .each\_row](http://arma.sourceforge.net/docs.html#each_colrow) |  | repeated operations on each column or row of matrix |
| [.each\_slice](http://arma.sourceforge.net/docs.html#each_slice) |  | repeated operations on each slice of cube |
|  |  |  |
| [.set\_imag / .set\_real](http://arma.sourceforge.net/docs.html#set_imag) |  | set imaginary/real part |
| [.insert\_rows/cols/slices](http://arma.sourceforge.net/docs.html#insert) |  | insert vector/matrix/cube at specified row/column/slice |
| [.shed\_rows/cols/slices](http://arma.sourceforge.net/docs.html#shed) |  | remove specified rows/columns/slices |
| [.swap\_rows/cols](http://arma.sourceforge.net/docs.html#swap_rows) |  | swap specified rows or columns |
| [.swap](http://arma.sourceforge.net/docs.html#swap) |  | swap contents with given object |
|  |  |  |
| [.memptr](http://arma.sourceforge.net/docs.html#memptr) |  | raw pointer to memory |
| [.colptr](http://arma.sourceforge.net/docs.html#colptr) |  | raw pointer to memory used by specified column |
|  |  |  |
| [iterators (matrices)](http://arma.sourceforge.net/docs.html#iterators_mat) |  | STL-style iterators and associated member functions for matrices and vectors |
| [iterators (cubes)](http://arma.sourceforge.net/docs.html#iterators_cube) |  | STL-style iterators and associated member functions for cubes |
| [STL container functions](http://arma.sourceforge.net/docs.html#stl_container_fns) |  | STL-style container functions |
|  |  |  |
| [.t / .st](http://arma.sourceforge.net/docs.html#t_st_members) |  | return matrix transpose |
| [.i](http://arma.sourceforge.net/docs.html#i_member) |  | return inverse of square matrix |
| [.min / .max](http://arma.sourceforge.net/docs.html#min_and_max_member) |  | return extremum value |
| [.index\_min / .index\_max](http://arma.sourceforge.net/docs.html#index_min_and_index_max_member) |  | return index of extremum value |
| [.eval](http://arma.sourceforge.net/docs.html#eval_member) |  | force evaluation of delayed expression |
|  |  |  |
| [.in\_range](http://arma.sourceforge.net/docs.html#in_range) |  | check whether given location or span is valid |
| [.is\_empty](http://arma.sourceforge.net/docs.html#is_empty) |  | check whether object is empty |
| [.is\_square](http://arma.sourceforge.net/docs.html#is_square) |  | check whether matrix is square sized |
| [.is\_vec](http://arma.sourceforge.net/docs.html#is_vec) |  | check whether matrix is a vector |
| [.is\_sorted](http://arma.sourceforge.net/docs.html#is_sorted) |  | check whether vector or matrix is sorted |
|  |  |  |
| [.is\_finite](http://arma.sourceforge.net/docs.html#is_finite_member) |  | check whether all elements are finite |
| [.has\_inf](http://arma.sourceforge.net/docs.html#has_inf) |  | check whether any element is +-Inf |
| [.has\_nan](http://arma.sourceforge.net/docs.html#has_nan) |  | check whether any element is NaN |
|  |  |  |
| [.print](http://arma.sourceforge.net/docs.html#print) |  | print object to *std::cout* or user specified stream |
| [.raw\_print](http://arma.sourceforge.net/docs.html#raw_print) |  | print object without formatting |
|  |  |  |
| [.save/.load (matrices & cubes)](http://arma.sourceforge.net/docs.html#save_load_mat) |  | save/load matrices and cubes in files or streams |
| [.save/.load (fields)](http://arma.sourceforge.net/docs.html#save_load_field) |  | save/load fields in files or streams |

**Generated Vectors/Matrices/Cubes**

|  |  |  |
| --- | --- | --- |
| [eye](http://arma.sourceforge.net/docs.html#eye_standalone) |  | generate identity matrix |
| [linspace](http://arma.sourceforge.net/docs.html#linspace) |  | generate vector with linearly spaced elements |
| [logspace](http://arma.sourceforge.net/docs.html#logspace) |  | generate vector with logarithmically spaced elements |
| [ones](http://arma.sourceforge.net/docs.html#ones_standalone) |  | generate object filled with ones |
| [randi](http://arma.sourceforge.net/docs.html#randi) |  | generate object with random integer values in specified interval |
| [randu / randn](http://arma.sourceforge.net/docs.html#randu_randn_standalone) |  | generate object with random values (uniform and normal distributions) |
| [randg](http://arma.sourceforge.net/docs.html#randg) |  | generate object with random values (gamma distribution) |
| [regspace](http://arma.sourceforge.net/docs.html#regspace) |  | generate vector with regularly spaced elements |
| [speye](http://arma.sourceforge.net/docs.html#speye) |  | generate sparse identity matrix |
| [spones](http://arma.sourceforge.net/docs.html#spones) |  | generate sparse matrix with non-zero elements set to one |
| [sprandu / sprandn](http://arma.sourceforge.net/docs.html#sprandu_sprandn) |  | generate sparse matrix with non-zero elements set to random values |
| [toeplitz](http://arma.sourceforge.net/docs.html#toeplitz) |  | generate Toeplitz matrix |
| [zeros](http://arma.sourceforge.net/docs.html#zeros_standalone) |  | generate object filled with zeros |

**Functions of Vectors/Matrices/Cubes**

|  |  |  |
| --- | --- | --- |
| [abs](http://arma.sourceforge.net/docs.html#abs) |  | obtain magnitude of each element |
| [accu](http://arma.sourceforge.net/docs.html#accu) |  | accumulate (sum) all elements |
| [all](http://arma.sourceforge.net/docs.html#all) |  | check whether all elements are non-zero, or satisfy a relational condition |
| [any](http://arma.sourceforge.net/docs.html#any) |  | check whether any element is non-zero, or satisfies a relational condition |
| [approx\_equal](http://arma.sourceforge.net/docs.html#approx_equal) |  | approximate equality |
| [arg](http://arma.sourceforge.net/docs.html#arg) |  | phase angle of each element |
| [as\_scalar](http://arma.sourceforge.net/docs.html#as_scalar) |  | convert 1x1 matrix to pure scalar |
| [clamp](http://arma.sourceforge.net/docs.html#clamp) |  | obtain clamped elements according to given limits |
| [cond](http://arma.sourceforge.net/docs.html#cond) |  | condition number of matrix |
| [conj](http://arma.sourceforge.net/docs.html#conj) |  | obtain complex conjugate of each element |
| [conv\_to](http://arma.sourceforge.net/docs.html#conv_to) |  | convert between matrix types |
| [cross](http://arma.sourceforge.net/docs.html#cross) |  | cross product |
| [cumsum](http://arma.sourceforge.net/docs.html#cumsum) |  | cumulative sum |
| [cumprod](http://arma.sourceforge.net/docs.html#cumprod) |  | cumulative product |
| [det](http://arma.sourceforge.net/docs.html#det) |  | Determinant |
| [diagmat](http://arma.sourceforge.net/docs.html#diagmat) |  | generate diagonal matrix from given matrix or vector |
| [diagvec](http://arma.sourceforge.net/docs.html#diagvec) |  | extract specified diagonal |
| [diff](http://arma.sourceforge.net/docs.html#diff) |  | differences between adjacent elements |
| [dot / cdot / norm\_dot](http://arma.sourceforge.net/docs.html#dot) |  | dot product |
| [eps](http://arma.sourceforge.net/docs.html#eps) |  | obtain distance of each element to next largest floating point representation |
| [expmat](http://arma.sourceforge.net/docs.html#expmat) |  | matrix exponential |
| [expmat\_sym](http://arma.sourceforge.net/docs.html#expmat_sym) |  | matrix exponential (symmetric) |
| [find](http://arma.sourceforge.net/docs.html#find) |  | find indices of non-zero elements, or elements satisfying a relational condition |
| [find\_finite](http://arma.sourceforge.net/docs.html#find_finite) |  | find indices of finite elements |
| [find\_nonfinite](http://arma.sourceforge.net/docs.html#find_nonfinite) |  | find indices of non-finite elements |
| [find\_unique](http://arma.sourceforge.net/docs.html#find_unique) |  | find indices of unique elements |
| [fliplr / flipud](http://arma.sourceforge.net/docs.html#flip) |  | reverse order of columns or rows |
| [imag / real](http://arma.sourceforge.net/docs.html#imag_real) |  | extract imaginary/real part |
| [ind2sub](http://arma.sourceforge.net/docs.html#ind2sub) |  | convert linear index to subscripts |
| [index\_min / index\_max](http://arma.sourceforge.net/docs.html#index_min_and_index_max_standalone) |  | indices of extremum values |
| [inplace\_trans](http://arma.sourceforge.net/docs.html#inplace_trans) |  | in-place transpose |
| [is\_finite](http://arma.sourceforge.net/docs.html#is_finite) |  | check whether all elements are finite |
| [join\_rows / join\_cols](http://arma.sourceforge.net/docs.html#join) |  | concatenation of matrices |
| [join\_slices](http://arma.sourceforge.net/docs.html#join_slices) |  | concatenation of cubes |
| [kron](http://arma.sourceforge.net/docs.html#kron) |  | Kronecker tensor product |
| [log\_det](http://arma.sourceforge.net/docs.html#log_det) |  | log determinant |
| [logmat](http://arma.sourceforge.net/docs.html#logmat) |  | matrix logarithm |
| [logmat\_sympd](http://arma.sourceforge.net/docs.html#logmat_sympd) |  | matrix logarithm (symmetric) |
| [min / max](http://arma.sourceforge.net/docs.html#min_and_max) |  | return extremum values |
| [nonzeros](http://arma.sourceforge.net/docs.html#nonzeros) |  | return non-zero values |
| [norm](http://arma.sourceforge.net/docs.html#norm) |  | various norms of vectors and matrices |
| [normalise](http://arma.sourceforge.net/docs.html#normalise) |  | normalise vectors to unit *p*-norm |
| [prod](http://arma.sourceforge.net/docs.html#prod) |  | product of elements |
| [rank](http://arma.sourceforge.net/docs.html#rank) |  | rank of matrix |
| [rcond](http://arma.sourceforge.net/docs.html#rcond) |  | reciprocal of condition number |
| [repmat](http://arma.sourceforge.net/docs.html#repmat) |  | replicate matrix in block-like fashion |
| [reshape](http://arma.sourceforge.net/docs.html#reshape) |  | change size while keeping elements |
| [resize](http://arma.sourceforge.net/docs.html#resize) |  | change size while keeping elements and preserving layout |
| [shift](http://arma.sourceforge.net/docs.html#shift) |  | shift elements |
| [shuffle](http://arma.sourceforge.net/docs.html#shuffle) |  | randomly shuffle elements |
| [size](http://arma.sourceforge.net/docs.html#size) |  | obtain dimensions of given object |
| [sort](http://arma.sourceforge.net/docs.html#sort) |  | sort elements |
| [sort\_index](http://arma.sourceforge.net/docs.html#sort_index) |  | vector describing sorted order of elements |
| [sqrtmat](http://arma.sourceforge.net/docs.html#sqrtmat) |  | square root of matrix |
| [sqrtmat\_sympd](http://arma.sourceforge.net/docs.html#sqrtmat_sympd) |  | square root of matrix (symmetric) |
| [sum](http://arma.sourceforge.net/docs.html#sum) |  | sum of elements |
| [sub2ind](http://arma.sourceforge.net/docs.html#sub2ind) |  | convert subscripts to linear index |
| [symmatu / symmatl](http://arma.sourceforge.net/docs.html#symmat) |  | generate symmetric matrix from given matrix |
| [trace](http://arma.sourceforge.net/docs.html#trace) |  | sum of diagonal elements |
| [trans](http://arma.sourceforge.net/docs.html#trans) |  | transpose of matrix |
| [trapz](http://arma.sourceforge.net/docs.html#trapz) |  | trapezoidal numerical integration |
| [trimatu / trimatl](http://arma.sourceforge.net/docs.html#trimat) |  | copy upper/lower triangular part |
| [unique](http://arma.sourceforge.net/docs.html#unique) |  | return unique elements |
| [vectorise](http://arma.sourceforge.net/docs.html#vectorise) |  | convert matrix to vector |
| [misc functions](http://arma.sourceforge.net/docs.html#misc_fns) |  | miscellaneous element-wise functions: exp, log, pow, sqrt, round, sign, ... |
| [trig functions](http://arma.sourceforge.net/docs.html#trig_fns) |  | trigonometric element-wise functions: cos, sin, ... |

**Decompositions, Factorisations, Inverses and Equation Solvers (Dense Matrices)**

|  |  |  |
| --- | --- | --- |
| [chol](http://arma.sourceforge.net/docs.html#chol) |  | Cholesky decomposition |
| [eig\_sym](http://arma.sourceforge.net/docs.html#eig_sym) |  | eigen decomposition of dense symmetric/hermitian matrix |
| [eig\_gen](http://arma.sourceforge.net/docs.html#eig_gen) |  | eigen decomposition of dense general square matrix |
| [eig\_pair](http://arma.sourceforge.net/docs.html#eig_pair) |  | eigen decomposition for pair of general dense square matrices |
| [inv](http://arma.sourceforge.net/docs.html#inv) |  | inverse of general square matrix |
| [inv\_sympd](http://arma.sourceforge.net/docs.html#inv_sympd) |  | inverse of symmetric positive definite matrix |
| [lu](http://arma.sourceforge.net/docs.html#lu) |  | lower-upper decomposition |
| [null](http://arma.sourceforge.net/docs.html#null) |  | orthonormal basis of null space |
| [orth](http://arma.sourceforge.net/docs.html#orth) |  | orthonormal basis of range space |
| [pinv](http://arma.sourceforge.net/docs.html#pinv) |  | pseudo-inverse |
| [qr](http://arma.sourceforge.net/docs.html#qr) |  | QR decomposition |
| [qr\_econ](http://arma.sourceforge.net/docs.html#qr_econ) |  | economical QR decomposition |
| [qz](http://arma.sourceforge.net/docs.html#qz) |  | generalised Schur decomposition |
| [schur](http://arma.sourceforge.net/docs.html#schur) |  | Schur decomposition |
| [solve](http://arma.sourceforge.net/docs.html#solve) |  | solve systems of linear equations |
| [svd](http://arma.sourceforge.net/docs.html#svd) |  | singular value decomposition |
| [svd\_econ](http://arma.sourceforge.net/docs.html#svd_econ) |  | economical singular value decomposition |
| [syl](http://arma.sourceforge.net/docs.html#syl) |  | Sylvester equation solver |

**Decompositions, Factorisations and Equation Solvers (Sparse Matrices)**

|  |  |  |
| --- | --- | --- |
| [eigs\_sym](http://arma.sourceforge.net/docs.html#eigs_sym) |  | limited number of eigenvalues & eigenvectors of sparse symmetric real matrix |
| [eigs\_gen](http://arma.sourceforge.net/docs.html#eigs_gen) |  | limited number of eigenvalues & eigenvectors of sparse general square matrix |
| [spsolve](http://arma.sourceforge.net/docs.html#spsolve) |  | solve sparse systems of linear equations |
| [svds](http://arma.sourceforge.net/docs.html#svds) |  | limited number of singular values & singular vectors of sparse matrix |

**Signal & Image Processing**

|  |  |  |
| --- | --- | --- |
| [conv](http://arma.sourceforge.net/docs.html#conv) |  | 1D convolution |
| [conv2](http://arma.sourceforge.net/docs.html#conv2) |  | 2D convolution |
| [fft / ifft](http://arma.sourceforge.net/docs.html#fft) |  | 1D fast Fourier transform and its inverse |
| [fft2 / ifft2](http://arma.sourceforge.net/docs.html#fft2) |  | 2D fast Fourier transform and its inverse |
| [interp1](http://arma.sourceforge.net/docs.html#interp1) |  | 1D interpolation |
| [polyfit](http://arma.sourceforge.net/docs.html#polyfit) |  | find polynomial coefficients for data fitting |
| [polyval](http://arma.sourceforge.net/docs.html#polyval) |  | evaluate polynomial |

**Statistics & Clustering**

|  |  |  |
| --- | --- | --- |
| [stats functions](http://arma.sourceforge.net/docs.html#stats_fns) |  | mean, median, standard deviation, variance |
| [cov](http://arma.sourceforge.net/docs.html#cov) |  | covariance |
| [cor](http://arma.sourceforge.net/docs.html#cor) |  | correlation |
| [hist](http://arma.sourceforge.net/docs.html#hist) |  | histogram of counts |
| [histc](http://arma.sourceforge.net/docs.html#histc) |  | histogram of counts with user specified edges |
| [princomp](http://arma.sourceforge.net/docs.html#princomp) |  | principal component analysis |
| [running\_stat](http://arma.sourceforge.net/docs.html#running_stat) |  | running statistics of one dimensional process/signal |
| [running\_stat\_vec](http://arma.sourceforge.net/docs.html#running_stat_vec) |  | running statistics of multi-dimensional process/signal |
| [kmeans](http://arma.sourceforge.net/docs.html#kmeans) |  | cluster data into disjoint sets |
| [gmm\_diag](http://arma.sourceforge.net/docs.html#gmm_diag) |  | model data as a Gaussian Mixture Model (GMM) |

**Miscellaneous**

|  |  |  |
| --- | --- | --- |
| [constants](http://arma.sourceforge.net/docs.html#constants) |  | pi, inf, NaN, speed of light, ... |
| [wall\_clock](http://arma.sourceforge.net/docs.html#wall_clock) |  | timer for measuring number of elapsed seconds |
| [logging of errors/warnings](http://arma.sourceforge.net/docs.html#logging) |  | how to change the streams for displaying warnings and errors |
| [uword / sword](http://arma.sourceforge.net/docs.html#uword) |  | shorthand for unsigned and signed integers |
| [cx\_double / cx\_float](http://arma.sourceforge.net/docs.html#cx_double) |  | shorthand for std::complex<double> and std::complex<float> |
| [Matlab/Armadillo syntax differences](http://arma.sourceforge.net/docs.html#syntax) |  | examples of Matlab syntax and conceptually corresponding Armadillo syntax |
| [example program](http://arma.sourceforge.net/docs.html#example_prog) |  | short example program |
| [config.hpp](http://arma.sourceforge.net/docs.html#config_hpp) |  | configuration options |
| [API additions](http://arma.sourceforge.net/docs.html#api_additions) |  | API stability and list of API additions |

Matlab 与 Armadillo对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Matlab/Octave** |  | **Armadillo** |  | **Notes** |
|  |  |  |  |  |
| A(1, 1) |  | A(0, 0) |  | indexing in Armadillo starts at 0 |
| A(k, k) |  | A(k-1, k-1) |  |  |
|  |  |  |  |  |
| size(A,1) |  | A[.n\_rows](http://arma.sourceforge.net/docs.html#attributes) |  | read only |
| size(A,2) |  | A[.n\_cols](http://arma.sourceforge.net/docs.html#attributes) |  |  |
| size(Q,3) |  | Q[.n\_slices](http://arma.sourceforge.net/docs.html#attributes) |  | Q is a [cube](http://arma.sourceforge.net/docs.html#Cube) (3D array) |
| numel(A) |  | A[.n\_elem](http://arma.sourceforge.net/docs.html#attributes) |  |  |
|  |  |  |  |  |
| A(:, k) |  | A[.col](http://arma.sourceforge.net/docs.html#submat)(k) |  | this is a conceptual example only; exact conversion from Matlab/Octave to Armadillo syntax will require taking into account that indexing starts at 0 |
| A(k, :) |  | A[.row](http://arma.sourceforge.net/docs.html#submat)(k) |  |  |
| A(:, p:q) |  | A[.cols](http://arma.sourceforge.net/docs.html#submat)(p, q) |  |  |
| A(p:q, :) |  | A[.rows](http://arma.sourceforge.net/docs.html#submat)(p, q) |  |  |
| A(p:q, r:s) |  | A( [span](http://arma.sourceforge.net/docs.html#submat)(p,q), [span](http://arma.sourceforge.net/docs.html#submat)(r,s) ) |  | A( span(first\_row, last\_row), span(first\_col, last\_col) ) |
|  |  |  |  |  |
| Q(:, :, k) |  | Q[.slice](http://arma.sourceforge.net/docs.html#subcube)(k) |  | Q is a [cube](http://arma.sourceforge.net/docs.html#Cube) (3D array) |
| Q(:, :, t:u) |  | Q[.slices](http://arma.sourceforge.net/docs.html#subcube)(t, u) |  |  |
| Q(p:q, r:s, t:u) |  | Q( [span](http://arma.sourceforge.net/docs.html#subcube)(p,q), [span](http://arma.sourceforge.net/docs.html#subcube)(r,s), [span](http://arma.sourceforge.net/docs.html#subcube)(t,u) ) |  |  |
|  |  |  |  |  |
| A' |  | A[.t()](http://arma.sourceforge.net/docs.html#t_st_members) or [trans](http://arma.sourceforge.net/docs.html#trans)(A) |  | matrix transpose / Hermitian transpose  (for complex matrices, the conjugate of each element is taken) |
|  |  |  |  |  |
| A = zeros(size(A)) |  | A[.zeros()](http://arma.sourceforge.net/docs.html#zeros_member) |  |  |
| A = ones(size(A)) |  | A.[ones()](http://arma.sourceforge.net/docs.html#ones_member) |  |  |
| A = zeros(k) |  | A = [zeros](http://arma.sourceforge.net/docs.html#zeros_standalone)<mat>(k,k) |  |  |
| A = ones(k) |  | A = [ones](http://arma.sourceforge.net/docs.html#ones_standalone)<mat>(k,k) |  |  |
|  |  |  |  |  |
| C = complex(A,B) |  | cx\_mat C = [cx\_mat](http://arma.sourceforge.net/docs.html#Mat)(A,B) |  |  |
|  |  |  |  |  |
| A .\* B |  | A % B |  | [element-wise multiplication](http://arma.sourceforge.net/docs.html#operators) |
| A ./ B |  | A / B |  | [element-wise division](http://arma.sourceforge.net/docs.html#operators) |
| A \ B |  | [solve](http://arma.sourceforge.net/docs.html#solve)(A,B) |  | conceptually similar to [inv](http://arma.sourceforge.net/docs.html#inv)(A)\*B, but more efficient |
| A = A + 1; |  | A++ |  |  |
| A = A - 1; |  | A-- |  |  |
|  |  |  |  |  |
| A = [ 1 2; 3 4; ] |  | A << 1 << 2 << endr    << 3 << 4 << endr; |  | [element initialisation](http://arma.sourceforge.net/docs.html#element_initialisation), with special element *endr* indicating *end of row* |
|  |  |  |  |  |
| X = A(:) |  | X = [vectorise](http://arma.sourceforge.net/docs.html#vectorise)(A) |  |  |
| X = [ A  B ] |  | X = [join\_horiz](http://arma.sourceforge.net/docs.html#join)(A,B) |  |  |
| X = [ A; B ] |  | X = [join\_vert](http://arma.sourceforge.net/docs.html#join)(A,B) |  |  |
|  |  |  |  |  |
| A |  | cout << A << endl;  or  A[.print](http://arma.sourceforge.net/docs.html#print)("A ="); |  |  |
|  |  |  |  |  |
| save ‑ascii 'A.dat' A |  | A[.save](http://arma.sourceforge.net/docs.html#save_load_mat)("A.dat", raw\_ascii); |  | Matlab/Octave matrices saved as ascii are readable by Armadillo (and vice-versa) |
| load ‑ascii 'A.dat' |  | A[.load](http://arma.sourceforge.net/docs.html#save_load_mat)("A.dat", raw\_ascii); |  |  |
|  |  |  |  |  |
| A = randn(2,3);  B = randn(4,5);  F = { A; B } |  | mat A = randn(2,3);  mat B = randn(4,5);  [field](http://arma.sourceforge.net/docs.html#field)<mat> F(2,1);  F(0,0) = A;  F(1,0) = B; |  | [fields](http://arma.sourceforge.net/docs.html#field) store arbitrary objects, such |