Linear Algebra Commands in MAPLE

Comparing the *linalg* and *LinearAlgebra* Packages

linalg	Linear Algebra	Description
1. General		
with(linalg):	with(LinearAlgebra):	read in the linear algebra package
evalm(a);	n/a	display the matrix a in matrix form (on the screen)
type(*expr*,matrix);	type(*expr*,Matrix);	gives *true* if *expr* has the form of a matrix
rowdim(a);	RowDimension(a);	the number of rows of the matrix a
coldim(a);	ColumnDimension(a);	the number of columns of a
rowdim(a), coldim(a);	Dimensions(a);	the number of rows and columns
2. Basic matrix operations		
$a+b$;	$a+b$;	the sum of the matrices a and b
$c*a$;	$c*a$;	multiplying the matrix a by a scalar c
a &* b;	a . b; [note the spaces!]	matrix multiplication
a^n;	a^n;	the n-th power of a matrix
inverse(a); or $1/a$; or a^(-1);	$1/a$; or a^(-1);	the inverse of a matrix
map(f, a);	map(f, a);	the matrix obtained by applying the function f to each entry of a
transpose(a);	Transpose(a); or a^%T;	the transpose of a
det(a);	Determinant(a);	the determinant of a
trace(a);	Trace(a);	the trace of a
rank(a);	Rank(a);	the rank of a
adjoint(a); or adj(a);	Adjoint(a);	the adjoint matrix (matrix of minors) of a
3. Constructing matrices		
matrix($[[a_{11},\ldots,a_{1n}],\ldots]$);	Matrix($[[a_{11},\ldots,a_{1n}],\ldots]$); or $<<a_{11}\|\ldots\|a_{1n}>,\ldots,<a_{m1}\|\ldots\|a_{mn}>>$;	build an $m\times n$ matrix $a=(a_{ij})$ by listing its elements
matrix(m, n, f);	Matrix(m, n, f)	build an $m\times n$ matrix whose ij-th entry is $f(i,j)$
matrix($m, n, (i,j)\to expr$);	Matrix($m, n, (i,j)\to expr$);	build an $m\times n$ matrix with ij-th entry $expr(i,j)$
matrix(*list*);	matrix(*list*);	build a matrix whose i-th row is the i-th entry in *list* (a list of lists)
augment($v_1,\ldots,v_n$);	$<v_1\|\ldots\|v_n>$;	build a matrix whose j-th column is the vector v_j

4. Special matrices		
matrix$(m, n, 0)$;	ZeroMatrix(m, n);	the $m \times n$ zero matrix
diag$(list)$	DiagonalMatrix$(list)$;	generate a diagonal matrix with $list$ as the diagonal entries
band$([1], n)$;	IdentityMatrix(n);	generate an $n \times n$ identity matrix
vandermonde(lis);	VandermondeMatrix(lis);	generate a Vandermonde matrix with 2nd column the list lis
band$(list, n)$	BandMatrix$(list)$;	create a tri–diagonal matrix (or an arbitrary band matrix)
JordanBlock(c, n);	JordanBlockMatrix$([[c, n]])$;	generate an $n \times n$ Jordan block with eigenvalue c
diag(JordanBlock(c_1, n_1), ..., JordanBlock(c_r, n_r);	JordanBlockMatrix$([[c_1, n_1], \ldots, [c_r, n_r]])$;	generate a Jordan matrix with Jordan blocks $(c_1, n_1), \ldots, (c_r, n_r)$
companion(p, x);	CompanionMatrix(p, x);	generate the $n \times n$ companion matrix associated to a monic polynomial $p(x)$ of degree n
5. Extracting parts of a matrix		
$a[i, j]$;	$a[i, j]$;	the (i, j)-th entry of matrix a
row(a, i);	$a[i, 1.. - 1]]$; or Row(a, i);	the i-th row of matrix a
col(a, j);	$a[1.. - 1, j]$ or Column(a, j);	the j-th column of matrix a
row$(a, i_1..i_2)$;	seq$(a[i, 1.. - 1], i = 1_1..i_2)$;	the list of rows i_1 to i_2 of a
col$(a, j_1..i_2)$;	seq$(a[1.. - 1, j], j = j_1..j_2)$;	the list of columns j_1 to j_2 of a
submatrix$(a, [i_1, ..., i_r], [j_1, ..., j_s])$;	$a[[i_1, \ldots, i_r], [j_1, \ldots, j_s]]$; or Submatrix$(a, [i_1, \ldots, i_r], [j_1, \ldots, j_s])$;	the $r \times s$ submatrix of a with row indices i_k and column indices j_k
submatrix$(a, i_0..i_1, j_0..j_1)$;	$a[i_0..i_1, j_0..j_1]$; or Submatrix$(a, i_0..i_1, j_0..j_1)$;	the submatrix of a having row and column indices from i_0 to i_1 and j_0 to j_1, respectively
6. Pasting and altering matrices		
stackmatrix$(a, b, \ldots)$;	$< a, b, \ldots >$;	join two (or more) matrices vertically
augment$(a, b, \ldots)$;	$< a \mid b \mid \ldots >$;	join two (or more) matrices horizontally
matrix(m, n, lis);	n/a	partition a list lis of mn elements into an $m \times n$ matrix
diag$(a_1, a_2, \ldots)$;	DiagonalMatrix$([a_1, a_2, \ldots])$;	construct a block diagonal matrix using the (square) matrices $a_1, a_2, \ldots$
extend(a, r, s, c);	Matrix$(m + r, n + s, [a]$,fill$= c)$;	enlarge the $m \times n$ matrix a by r additional rows and s additional columns with the value c
$b :=$ evalm(a);	$b :=$ copy(a);	copying the entries of matrix a to b
copyinto(a, b, i, j);	$b[i..i + m - 1, j..j + n - 1] := a$;	copy the entries of matrix a into matrix b at index position (i, j) (b is altered)
$a[i, j] := expr$;	$a[i, j] := expr$;	replace the (i, j)-th entry of matrix a by the expression $expr$

7. Row and column operations		
addrow(a, i_1, i_2, c);	RowOperation$(a, [i_2, i_1], c)$;	add c times row i_1 to row i_2 (thus, the result is put in row i_2)
addcol(a, j_1, j_2, c);	ColumnOperation$(a, [j_2, j_1], c)$;	add c times column j_1 to col. j_2
mulrow(a, i, c);	RowOperation(a, i, c);	multiply row i by c
mulcol(a, j, c);	ColumnOperation(a, j, c);	multiply column j by c
swaprow(a, i_1, i_2);	RowOperation$(a, [i_1, i_2])$;	interchange rows i_1 and i_2
swapcol(a, j_1, j_2);	ColumnOperation$(a, [j_1, j_2])$;	interchange columns j_1 and j_2
delrows$(a, i_1..i_2)$;	DeleteRow$(a, i_1..i_2)$;	delete rows i_1 to i_2 of a
delcols$(a, j_1..j_2)$;	DeleteColumn$(a, i_1..i_2)$;	delete columns j_1 to j_2 of a
8. Row reduction and solutions of linear systems		
linsolve(a, b);	LinearSolve(a, b);	find the (general) solution of the matrix equation $ax = b$
nullspace(a); or kernel(a);	NullSpace(a);	compute a basis for the nullspace of a matrix a
gausselim(a);	GaussianElimination(a);	find an upper triangular matrix row equivalent to a
backsub(a, b);	BackwardsSubstitution(a, b);	solve $ax = b$ by back-substitution (if a is upper triangular)
gaussjord(a); or rref(a);	ReducedRowEchelonForm(a);	compute the reduced row echelon form of a
pivot(a, i, j);	Pivot(a, i, j);	row-reduce a to make j^{th} column = 0, except for $(i, j)^{th}$ entry
ihermite(a);	HermiteForm(a);	row-reduce the integer matrix a to an upper-Δ integer matrix
ismith(a);	SmithForm(a);	row/column reduce the integer matrix a to a diagonal integer matrix
9. Eigenvalues, Eigenvectors		
charmat(a, x);	CharacteristicMatrix(a, x);	compute the characteristic matrix $Ix - a$ (x a variable)
charpoly(a, x);	CharacteristicPolynomial(a, x);	find the characteristic polynomial of a
minpoly(a, x);	MinimalPolynomial(a, x);	find the minimal polynomial of a
eigenvals(a);	Eigenvalues(a);	compute the eigenvalues of a
eigenvects(a);	Eigenvectors(a);	find the eigenvectors of a
jordan(a); or jordan$(a,$‘q‘$)$;	JordanForm(a); or JordanForm (a, output = ['J','Q']);	compute the Jordan canonical form J of a and the matrix q such that $a = q^{-1}Jq$
frobenius(a); or frobenius$(a,$‘p‘$)$;	FrobeniusForm(a); or Frobenius-Form(a, output = ['F', 'Q']);	compute the Frobenius (or rational) canonical form F of a; find p such that $a = pFp^{-1}$
issimilar(a, b); or issimilar$(a, b,$ ‘q‘$)$;	IsSimilar(a, b); or IsSimilar$(a, b,$ output = ['query', 'C']);	determine whether a is similar to b; if so, find the matrix q such that $a = q^{-1}bq$

10. Vector operations		
vector($[a_1, \ldots, a_n]$);	Vector($[a_1, \ldots, a_n]$); or $< a_1, \ldots, a_n >$;	define a (column) vector by the list $[a_1, \ldots, a_n]$
vector($[a_1, \ldots, a_n]$);	$< a_1 \mid \ldots \mid a_n >$;	define a (row) vector by the list $[a_1, \ldots, a_n]$
vectdim(v);	Dimension(v);	the dimension of a vector
type($expr$, vector);	type($expr$, Vector);	gives *true* if $expr$ has the form of a vector
vector([op(convert(v, list)), op(convert(w, list)))]);	convert($< v, w >$, Vector);	direct sum of two vectors v, w
evalm($v + w$); or matadd(v, w);	$v + w$;	add two vectors v, w
evalm($c * v$); or scalarmul(v, c);	$c * v$;	multiply the vector v by scalar c
multiply(a, v); or innerprod(a, v);	a . v;	multiply the matrix a by the (column) vector v (on the right)
multiply(v, a); or innerprod(v, a);	w . a;	multiply the matrix a by the (row) vector w (on the left)
dotprod(v, w);	v . w;	the dot (scalar) product of v and w
norm(v, 2);	Norm(v);	the length or norm $\|\|v\|\|$ of a vector v
normalize(v);	Normalize(v);	divide the vector v by its length
angle(v, w);	VectorAngle(v, w);	the angle between the vectors v and w
11. Vector spaces		
basis(lis);	Basis(lis);	find a basis of the vector space spanned by list lis of vectors
intbasis($lis_1, lis_2, \ldots$);	IntersectionBasis($[lis_1, lis_2 \ldots]$);	find a basis of the intersection of the spaces spanned by the lists $lis_1, lis_2, \ldots$
sumbasis($lis_1, lis_2, \ldots$);	SumBasis($[lis_1, lis_2 \ldots]$);	find a basis of the sum/union of the spaces spanned by the lists $lis_1, lis_2, \ldots$
rowspace(a); colspace(a);	RowSpace(a); ColumnSpace(a);	find a basis for the row/column space of the matrix a
rowspan(a); colspan(a);		find a spanning set for the row space (column space) of a, where a has polynomial entries
nullspace(a); or kernel(a);	NullSpace(a);	compute a basis for the nullspace of a matrix a
GramSchmidt($[v_1, \ldots, v_n]$);	GramSchmidt($[v_1, \ldots, v_n]$);	apply the Gram-Schmidt procedure to the vectors $v_1, \ldots, v_n$
12. Miscellaneous		
orthog(a);	IsOrthogonal(a);	test whether a is an orthogonal matrix
leastsqrs(a, b);	LeastSquares(a, b);	find x such that $\| ax - b \|$ is minimal
equal(a, b);	Equal(a, b);	test whether matrices a and b are equal
evalm(subs($x = a, f$));	convert(evalm(subs($x = a, f$)), Matrix);	evaluate the matrix polynomial $f(a)$ (where $f(x)$ is a polynomial)