Linear Algebra Commands

In MAPLE and MATHEMATICA

MAPLE V, Release 5	MATHEMATICA	Description
1. General		
with(linalg); or with(linalg):	n/a	read in the linear algebra package (mandatory for MAPLE)
evalm(a);	MatrixForm[a]	display the matrix a in matrix form (on the screen)
type(*expr*,matrix);	MatrixQ[*expr*]	gives TRUE if *expr* has the form of a matrix
rowdim(a);	Dimensions[a][[1]] or Length[a]	the number of rows of matrix a
coldim(a);	Dimensions[a][[2]]	the number of columns of a
2. Basic matrix operations		
$a+b$; or matadd(a,b);	$a+b$	the sum of the matrices a and b
$c*a$; or scalarmul(a,c);	$c\ a$	multiply the matrix a by a scalar c
a &* b; or multiply(a,b);	$a.b$	matrix multiplication
a^n;	MatrixPower[a,n]	the n-th power of a matrix
evalm(subs($x=a,f$));		evaluate the matrix polynomial $f(a)$, where f is a polynomial in x
inverse(a); or $1/a$; or a^(-1);	Inverse[a]	the inverse of a matrix
map(f,a);	$f[a]$	the matrix obtained by applying the function f to each entry of a
transpose(a);	Transpose[a]	the transpose of a
det(a);	Det[a]	the determinant of a
trace(a);	Sum[a[[i,i]], $\{i$, Length[a]$\}$]	the trace of a
rank(a);	Dimensions[a][[2]]− Length[NullSpace[a]]	the rank of a
adjoint(a); or adj(a);	Minors[a]	the adjoint matrix (matrix of minors) of a
3. Constructing matrices		
matrix([[$a_{11},\ldots,a_{1m}$],...]);	$\{\{a_{11},\ldots,a_{1m}\},\ldots\}$	build an $m\times n$ matrix by listing its elements
matrix(m,n,f);	Table[$f,\{i,m\},\{j,n\}$]	build an $m\times n$ matrix whose ij-th entry is $f(i,j)$
matrix$(m,n,(i,j)\rightarrow$ *expr*);	Table[*expr*, $\{i,m\},\{j,n\}$]	build an $m\times n$ matrix with ij-th entry *expr*(i,j)
matrix(*lis*);	*lis*	build a matrix whose i-th row is the i-th entry in the list *lis*
matrix$(m,n,$ *lis*);	Partition[*lis*, n]	partition a list *lis* of mn elements into an $m\times n$ matrix

4. Special matrices		
matrix$(m, n, 0)$;	Table$[0, \{m\}, \{n\}]$	the $m \times n$ zero matrix
diag(lis);	DiagonalMatrix$[lis]$	generate a diagonal matrix with list lis as the diagonal entries
band$([1], n)$;	IdentityMatrix$[n]$	generate an $n \times n$ identity matrix
vandermonde(lis);	Table$[lis[[i]]\hat{}(j-1), \{i, \text{Length}[lis]\}, \{j, \text{Length}[lis]\}$	generate a Vandermonde matrix with 2nd column the list lis
band$(list, n)$	Table$[\text{Switch}[i-j, -1, a[[i]], 0, b[[i]], 1, c[[i-1]], _, 0], \{i, m\}\{j, n\}]$	create a tri–diagonal matrix (or an arbitrary band matrix)
JordanBlock(c, n);	Table$[\text{Switch}[i-j, -1, 1, 0, c, _, 0], \{i, m\}, \{j, n\}]$	generate an $n \times n$ Jordan block with eigenvalue c
companion(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);	$a[[i]]$	the i-th row of matrix a
col(a, j);	Transpose$[a][[j]]$	the j-th column of matrix a
row$(a, i_1..i_2)$;	Take$[a, \{i_1, i_2\}]$	the list of rows i_1 to i_2 of a
col$(a, j_1..i_2)$;	Transpose[Take[Transpose$[a]$, $\{j_1, j_2\}$]]	the list of columns j_1 to j_2 of a
submatrix$(a, [i_1, ..., i_r], [j_1, ..., j_s])$;	$a[[\{i_1, \ldots, i_r\}, \{j_1, \ldots, j_s\}]]$	the $r \times s$ submatrix of a with row indices i_k and column indices j_k
submatrix$(a, i_0..i_1, j_0..j_1)$;	$a[[\text{Range}[i_0, i_1], \text{Range}[j_0, j_1]]$	the submatrix of a having row and column indices from i_0 to i_1 and j_0 to j_1, respectively
minor(a, i, j);		submatrix of a obtained by deleting the i^{th} row and j^{th} column
6. Pasting matrices		
stackmatrix$(a, b, \ldots)$;	Join$[a, b, \ldots]$	join together two (or more) matrices vertically
augment$(a, b, \ldots)$; or concat(.)	Transpose[Join[Transpose$[a]$, Transpose$[b], \ldots$]]	join together two (or more) matrices horizontally
diag$(a_1, a_2, \ldots)$;		construct a block diagonal matrix using the matrices $a_1, a_2, \ldots$
blockmatrix(m, n, lis);		construct a matrix of $m \times n$ blocks by using the list lis consisting of $m \cdot n$ matrices
extend(a, m, n, c);		enlarge the matrix a by m additional rows and n additional columns with the value c
copyinto(a, b, i, j);	(use Table[If$[\ldots], \{\ldots\}, \{\ldots\}$])	copy the entries of matrix a into matrix b at index position (i, j)
stackmatrix(row$(a, 1..i-1)$, lis, row$(a, i..\text{rowdim}(a))$);	Insert$[a, lis, i]$	insert the list lis into the matrix a at position (row) i

7. Row and column operations		
addrow(a, i_1, i_2, c);	$a[[i_2]] = a[[i_2]] + c * a[[i_1]]$	add c times row i_1 to row i_2
addcol(a, j_1, j_2, c);	(use Transpose)	add c times column j_1 to col. j_2
mulrow(a, i, c);	$a[[i]] = c * a[[i]]$	multiply row i by c
mulcol(a, j, c);	(use Transpose)	multiply column j by c
swaprow(a, i_1, i_2);		interchange rows i_1 and i_2
swapcol(a, j_1, j_2);		interchange columns j_1 and j_2
delrows($a, i_1..i_2$);	Drop[a, i_1, i_2]	delete rows i_1 to i_2 of a
delcols($a, j_1..j_2$);	Transpose[Drop[Transpose[a], $\{i_1, i_2\}$]]	delete columns j_1 to j_2 of a
pivot(a, i, j);		row-reduce a to make j^{th} column $= 0$, except for $(i, j)^{th}$ entry
ihermite(a);		row-reduce the integer matrix a to an upper-Δ integer matrix
ismith(a);		row/column reduce the int. matrix a to a diagonal int. matrix
8. Solving linear systems		
gausselim(a);	RowReduce[a]	find an upper triangular matrix row equivalent to a
ffgausselim(a);		fraction-free Gauss elimination
backsub(a, b); or backsub(a)		solve $ax = b$ by back-substitution
gaussjord(a); or rref(a);		compute the reduced row echelon form of a
nullspace(a); or kernel(a);	NullSpace[a]	compute a basis for the nullspace of a matrix a
linsolve(a, b);	LinearSolve[a, b]	find the (general) solution of the matrix equation $ax = b$
9. Eigenvalues, eigenvectors		
charmat(a, x);	IdentityMatrix[Length[a]]$x - a$	compute the characteristic matrix $Ix - a$
charpoly(a, x);	Det[IdentityMatrix[Length[a]]$x - a$]	find the characteristic polynomial of a
minpoly(a, x);		find the minimal polynomial of a
eigenvals(a); or eigenvalues(a);	Eigenvalues[a]	compute the eigenvalues of a
eigenvects(a); or eigenvectors(a);	Eigenvectors[a]	find the eigenvectors of a
jordan(a); or jordan(a,'q');		compute the Jordan canonical form J of a and the matrix q such that $a = q^{-1}Jq$
issimilar(a, b); or issimilar(a, b, 'q');		determine whether a is similar to b; if so, find the matrix q such that $a = q^{-1}bq$
frobenius(a); or frobenius(a,'p');		compute the Frobenius (or rational) canonical form F of a; find p such that $a = pFp^{-1}$

10. Vector operations		
vector(*lis*);	*lis*	define a vector by list *lis*
vectdim(v);	Dimensions[v]	the dimension of a vector
type(*expr* ,vector);	VectorQ[*expr*]	gives TRUE if *expr* has the form of a vector
vector([op(convert(v, list)), op(convert(w, list)))]);	Join[v, w]	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);	cv	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);	$v.a$	multiply the matrix a by the (row) vector v (on the left)
dotprod(v, w);	$v.w$	the dot (scalar) product of v and w
norm($v, 2$);	Sqrt[Sum[v[[i]]^2, {i, 1, Length[v]}]]	the length or norm $\|\|v\|\|$ of a vector v
normalize(v);		divide the vector v by its length
angle(v, w);		the angle between the vectors v and w
11. Vector spaces		
basis(*lis*);		find a basis of the vector space spanned by list *lis* of vectors
intbasis($lis_1, lis_2, \ldots$);		find a basis of the intersection of the spaces spanned by the lists $lis_1, lis_2, \ldots$
sumbasis($lis_1, lis_2, \ldots$);		find a basis of the sum/union of the spaces spanned by the lists $lis_1, lis_2, \ldots$
rowspace(a); colspace(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($v_1, \ldots, v_n$);		apply the Gram-Schmidt procedure to the vectors $v_1, \ldots, v_n$
12. Miscellaneous		
hadamard(a);		compute the 'Hadamard norm'
orthog(a);		test whether a is orthogonal
equal(a, b);		test whether $a = b$ (as matrices)
leastsqrs(a, b);		find x such that $\|\| ax - b \|\|$ is minimal