Matrix functions	
Function signature	**Description**
matrix(A,m,n) matrix([R]) matrix(A,[R])	matrix(A,m,n) creates the m × n null matrix with entries in the ring A. If M = matrix(A,m,n), M can be populated by expressions of the form M[j,k] = a, where a is an element of A. If R is a sequence of lists of the same length, matrix([R]) constructs the matrix whose rows are the elements of R. If A is domain, matrix(A, [R]) is like matrix([R]) followed by a projection of its elements to A. Examples: M = matrix(Zn(18),2,3) ⇒ [[0 0 0] [0 0 0]] :: Matrix[Z18] matrix(Zn(17),[[3, 2,35],[1,1,1]]) ⇒ [[321] [111]] :: Matrix[Z17]
null_matrix() null_matrix(K)	It returns a matrix of 0 elements. If a domain K is given, the null matrix will have domain K.
I_(n,K) I_(n)	Given an integer n and a domain K, it returns the identity matrix of order n over the domain K. The default value of K is Zn(2).
permutation_matrix(p) rd_permutation_matrix(n)	If p is a permutation of 0, 1, ..., n-1, this function creates the n × n matrix that has 0 everywhere, except at the entries (j,p[j]) that are set to 1. If p is an integer n, it agrees with rd_permutation_matrix(n) that selects p at random and returns permutation_matrix(p).
ncols(M) nrows(M) shape(M)	If M is an m × n matrix, we get m with nrows(M), n with ncols(M), and the pair (m,n) with shape(M).
submatrix (A,J) cosubmatrix(A,i,j)	If A is a matrix and J is any sequence in range(ncols(A)), submatrix(A,J) constructs the matrix whose colums are the columns A_j for j ∈ J. It is equivalent to A[:,J]. cosubmatrix(A,i,j) constructs the matrix obtained by deleting row i and column j of the matrix A.
cofactor(A,i,j)	If A is a square matrix, it yields (-1)**(i+j) * det(cosubmatrix(A,i,j))

<table>
<tr><th>Matrix functions</th><th></th></tr>
<tr><td></td><td></td></tr>
<tr><td>Function signature</td><td>Description</td></tr>
<tr><td>splice(A,B)
stack(A,B)</td><td>If A and B are matrices and nrows(A) = nrows(B), then splice(A,B) is the matrix A|B obtained by appending each row of B to the corresponding row of A. The expression stack(A,B) is defined when ncols(A) = ncols(B) and it appends the rows of B to the rows of A.

Examples:
A = I_(2,Z_); B = permutation_matrix([1,0])
splice(A,B) ⇒
  [[1 0 0 1]
   [0 1 1 0]] :: Matrix[ZZ]
stack(A,B) ⇒
  [[1 0]
   [0 1]
   [0 1]
   [1 0]] :: Matrix[ZZ]</td></tr>
<tr><td>hankel_matrix(s)</td><td>Given a list or vector S, this function constructs the square Hankel matrix H associated to S, which is defined by
H[j,k] = S[i+j] for 0 ≤ i, j < (l+1)//2, l = len(S).</td></tr>
<tr><td>parity_completion (G)
right_parity_completion (G)
left_parity_completion (G)</td><td>If G is a k × n matrix, it adds to the right of G the colum formed by the negatives of the sums of the rows of G. Thus the sum of every row of the matrix so obtained is 0. The form right_parity_completion (G) is an alias of parity_completion (G). left_parity_completion(G) works as parity_completion(G), but with the extra column placed on the left of G.</td></tr>
<tr><td>paley_matrix (F)</td><td>If F is a finite field of q elements, q odd, paley_matrix(F) yields the q × q matrix (legendre(x_i−x_j)), where x_0,...,x_{q−1} are the elements of F in their standard order.</td></tr>
<tr><td>vandermonde (a, r)
vandermonde_matrix = vandermonde</td><td>If a is a vector, vandermonde(a,r) produces the vandermonde matrix of r rows associated to a.</td></tr>
<tr><td>cyclic_matrix (g, n)
cyclic_generating_matrix(g,n)
cyclic_normalized_matrix (g, n)</td><td>If g is a monic univariate polynomial of degree n−k > 0 dividing X^n−1, or its vector of coefficents, cyclic_matrix(g,n) yields the standard generating matrix of the cyclic code C_g.
cyclic_generating_matrix(g,n) yields the standard generating matrix of the cyclic code C_g.
cyclic_normalized_matrix(g,n) works like cyclic_matrix(g,n), but the returned matrix is the normalized generating matrix of the cyclic code C_g.</td></tr>
<tr><td>cyclic_control_matrix (g,n)
cyclic_normalized_control_matrix (g, n)</td><td>If h is a monic univariate polynomial of degree k > 0 dividing X^n−1, or its vector of coefficents, cyclic_control_matrix(g,n) yields the standard control matrix of the cyclic code C_g, g = (X^n−1) / h.
cyclic_normalized_control_matrix(g,n) is the standard control matrix associated to cyclic_normalized_matrix(g,n).</td></tr>
<tr><td>components (x, F)
blow (h, F)</td><td>If F is a finite field and x an element of F, components(x,F) is the vector of components of x with respect to the standard basis of F over base(F).
If F is a finite field and h ∈ F^n, blow(h,F) delivers the vector obtained by replacing each element of h by the sequence of its components with respect to the standard basis of F over base(F).</td></tr>
<tr><td>blow (H, F)
prune (H, F)</td><td>If F is a finite field and H is an r×n matrix with entries in F, blow(H,F) constructs the matrix obtained by replacing each entry of H by the column of its components with respect to the standard basis of F over base(F).
If H is any matrix, prune(H) eliminates each row of H that is a linear combination of the preceding ones.</td></tr>
<tr><td>alternant_matrix (h, a, r)
alternant_matrix (P ,A)</td><td>The call alternant_matrix (h, a, r) provides the alternant control matrix of order r associated to the vectors h and a.
The call alternant_matrix(P,A), where P=[p_1, ..., p_r] is assumed to be a vector of univariate polynomials and A=[a_1, ..., a_n] a vector, constructs the matrix (p_i(a_j)).</td></tr>
</table>

Matrix functions	
Function signature	**Description**
scramble_matrix(A,k)	It returns a random kxk matrix M with elements in A with |det(M)| = 1
rd_GL(n) rd_GL(n,F)	It returns a random kxk invertible matrix M with elements in F. By default F is Zn(2).
hadamard_matrix_recursive(n)	It returns the hadamard matrix of order n computed recursively.
hadamard(r)	It returns the hadamard matrix of order n computed using the bdot function.
paley_matrix (F)	If F is a finite field of q elements, q odd, paley_matrix(F) yields the q × q matrix (legendre(xi−xj)), where x0,...,xq−1 are the elements of F in some order.
hadamard_matrix_paley(K)	It computes a hadamard matrix with elements in K using the Paley construction.
conference_matrix(K)	It computes a conference matrix over a domain K.
hadamard_matrix_finite_field(K)	It computes the hadamard matrix of the finite field K.
normalized_hamming_matrix(r,F) normalized_hamming_matrix(r)	It returns the normalized hamming matrix of rank r of elements in F. By default F is Zn(2)
transpose(M)	If M is a matrix, with this expresion we get the transpose of M.
subs_element(A,x,y)	Given a vector or matrix A, it changes all the entries equal to x by the elemnt y.
row_index(h,H) col_index(h',H) get_row=row_index get_col=col_index	Let H be an m × n matrix. If h is one of the rows of H, row_index(h,H) gives de index of h in H (H seen as a list of vectors). If h is not a row of H, the returned value is [ ]. The expression col_index(h',H) works similarly for a column h' of H.
rank(M)	This expression yields the rank of a matrix M with entries in a field. It agrees with the dimension of the space spanned by the rows of M.
det(M) trace(M)	If M is a square matrix, these expressions deliver the determinant and the trace of M, respectively.
GJ(S)	It computes the special Gauss-Jordan of the matrix S.
kernel(H) right_kernel(H) left_kernel(H)	Let H be an m × n matrix with entries in a field F. The expression kernel(H) returns an n × k matrix whose columns form a basis of the space of column vectors x of length n such that Hx = 0. This is also called the right_kernel(H). The space of row vectors x of length m such that xH = 0 is obtained by the function left_kernel(H), which produces an r × m matrix whose rows form a basis of left_kernel.
rd_linear_combination(G) rd_linear_combination(G,K)	If G is matrix and K is a field (or a ring), rd_linear_combination(G, K) returns a linear combination of the rows of G with coefficients randomly chosen in K.
rd_insert(A,r)	Auxiliary function for rd_GL: Given a list L, returns (e,r,a) where e = e_r, r is a random index of the list L and a is vector [0...0,rdnonzero(K), rd_vector(K,n-r-1)]
rd_extend(A)	Auxiliary function for rd_GL: Given a kxk matrix A of rank k, it returns a (k+1)x(k+1) matrix of rank k+1.

Matrix functions	
Function signature	**Description**
solve_linear_system(G,a) solve_pivot_matrix_system(G,a)	Given a matrix G and a vector a, it returns the vector x that fullfills Gx = a. solve_linear_system uses the LU descomposition and solve_pivot_matrix_system uses the Gauss pivoting algorithm.
tensor(A,B)	Given matrices A,B, it returns the matrix given by the Kronecker product of A and B.