Mathematics is the art of reducing any problem to linear algebra.

William Stein

8 Linear Algebra

This chapter deals with exact and symbolic linear algebra, i.e., linear algebra over rings specific to computer algebra, such as $\mathbb{Z}$, finite fields, or polynomial rings. Numerical linear algebra, based on fixed precision approximate arithmetic, is presented in Chapter 13. We first present constructions on matrices and their vector spaces together with basic operations (§8.1), then various computations on these matrices, gathered in two groups: operations related to Gaussian elimination and left equivalence transformations (§8.2.1-§8.2.2), and computations related to eigenvalues, eigenspaces and similarity transformations (§8.2.3).

The reader may refer to the books of Gantmacher [Gan90], von zur Gathen and Gerhard [vzGG03], and the Ph. D. thesis of Storjohann [Sto00] for further details on the notions presented in this chapter.

8.1 Elementary Constructs And Manipulations

8.1.1 Spaces of Vectors And Matrices

Similarly as for polynomials, vectors and matrices are handled as algebraic objects belonging to a space. This is a vector space when the coefficients are elements of a field, or a free K-module when the coefficients are elements of a ring.

The space $\mathcal{M}_{2,3}(\mathbb{Z})$ and the vector space $(\mathbb{F}_{3^2})^3$ are constructed by:

```
sage: MS = MatrixSpace(ZZ,2,3); MS
Full MatrixSpace of 2 by 3 dense matrices over Integer Ring
sage: VS = VectorSpace(GF(3^2,'x'),3); VS
Vector space of dimension 3 over Finite Field in x of size 3^2
```

Matrix space	
construct	`MS = MatrixSpace(K, nrows, ncols)`
construct (sparse matrix)	`MS = MatrixSpace(K, nrows, ncols, sparse = True)`
base ring K	`MS.base_ring()`
extending the ring	`MS.base_extend(B)`
changing the ring	`MS.change_ring(B)`
group generated	`MatrixGroup([A,B])`
basis of the vector space	`MS.basis()` *or* `MS.gens()`

Matrix constructs	
zero matrix	`MS()` *or* `MS.zero()` *or* `matrix(K,nrows,ncols)`
matrix from coefficients	`MS([1,2,3,4])` *or* `matrix(K,2,2,[1,2,3,4])` *or* `matrix(K,[[1,2],[3,4]])`
identity matrix	`MS.one()` *or* `MS.identity_matrix()` *or* `identity_matrix(K,n)`
random matrix	`MS.random_element()` *or* `random_matrix(K,nrows,ncols)`
Jordan block	`jordan_block(x,n)`
block matrix	`block_matrix([A,1,B,0])` *or* `block_diagonal_matrix(A,B)`

Elementary manipulations	
accessing a coefficient	`A[2,3]`
accessing the last row, the third column	`A[-1,:]`, `A[:,2]`
accessing the first four even columns	`A[:,0:8:2]`

submatrices	`A[3:4,2:5]`, `A[:,2:5]`, `A[:4,2:5]`
	`A.matrix_from_rows([1,3])`
	`A.matrix_from_columns([2,5])`
	`A.matrix_from_rows_and_columns([1,3],[2,5])`
	`A.submatrix(i,j,nrows,ncols)`

row concatenation	`A.stack(B)`
column concatenation	`A.augment(B)`

TABLE 8.1 – Constructs for matrices and their spaces.

A generating family for these spaces, namely the canonical basis, is obtained by the methods `MS.gens()` or `MS.basis()`.

```
sage: B = MatrixSpace(ZZ,2,3).basis()
sage: list(B)
```

$$\left[\begin{pmatrix}1&0&0\\0&0&0\end{pmatrix},\begin{pmatrix}0&1&0\\0&0&0\end{pmatrix},\begin{pmatrix}0&0&1\\0&0&0\end{pmatrix},\begin{pmatrix}0&0&0\\1&0&0\end{pmatrix},\begin{pmatrix}0&0&0\\0&1&0\end{pmatrix},\begin{pmatrix}0&0&0\\0&0&1\end{pmatrix}\right]$$

One can conveniently access its elements by row and column number:

```
sage: B[1,2]
```

$$\begin{pmatrix}0&0&1\\0&0&0\end{pmatrix}$$

Matrix Groups. One can also define groups and subgroups in the space of matrices. The general linear group of degree n over a field K, denoted by $\mathrm{GL}_n(K)$, is the group composed by all invertible $n \times n$ matrices in $\mathcal{M}_{n,n}(K)$. It is

constructed in Sage with the command `GL(n,K)`. The special linear group $\mathrm{SL}_n(K)$, composed by the elements of $\mathrm{GL}_n(K)$ with determinant one, is constructed with the command `SL(n,K)`.

The construction `MatrixGroup([A,B,...])` returns the group generated by the matrices in the list argument, all of which need to be invertible.

```
sage: A = matrix(GF(11), 2, 2, [1,0,0,2])
sage: B = matrix(GF(11), 2, 2, [0,1,1,0])
sage: MG = MatrixGroup([A,B])
sage: MG.cardinality()
200
sage: identity_matrix(GF(11),2) in MG
True
```

8.1.2 Vector And Matrix Construction

Matrices and vectors can naturally be generated as elements of their space, by providing the list of their coefficients. For matrices, they are listed in a row major mode:

```
sage: MS = MatrixSpace(ZZ,2,3); A = MS([1,2,3,4,5,6]); A
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

The empty constructor `MS()` returns the zero matrix, and so does the method `MS.zero()`. Several specialized constructors produce the most common matrices, as for example `random_matrix`, `identity_matrix`, `jordan_block` (see Table 8.1). In particular, one can construct matrices and vectors using the `matrix` and `vector` constructors, without having to construct the related space beforehand. By default, a matrix is build over the ring of integers $\mathbb{Z}$ and has dimension 0×0.

```
sage: a = matrix(); a.parent()
Full MatrixSpace of 0 by 0 dense matrices over Integer Ring
```

Of course, one can also specify the coefficient domain and the dimensions, to form a zero matrix or a matrix with prescribed coefficients provided in a list.

```
sage: a = matrix(GF(8,'x'),3,4); a.parent()
Full MatrixSpace of 3 by 4 dense matrices over Finite Field
in x of size 2^3
```

The constructor `matrix` also accepts as argument objects that have a natural transformation into a matrix. For instance, it can be used to generate the adjacency matrix of a graph, with coefficients in $\mathbb{Z}$.

```
sage: g = graphs.PetersenGraph()
sage: m = matrix(g); m; m.parent()
[0 1 0 0 1 1 0 0 0 0]
[1 0 1 0 0 0 1 0 0 0]
[0 1 0 1 0 0 0 1 0 0]
[0 0 1 0 1 0 0 0 1 0]
[0 0 1 0 1 0 0 0 1 0]
```

```
[1 0 0 1 0 0 0 0 0 1]
[1 0 0 0 0 0 0 1 1 0]
[0 1 0 0 0 0 0 0 1 1]
[0 0 1 0 0 1 0 0 0 1]
[0 0 0 1 0 1 1 0 0 0]
[0 0 0 0 1 0 1 1 0 0]
Full MatrixSpace of 10 by 10 dense matrices over Integer Ring
```

Block Matrices. The function `block_matrix` allows to define a matrix by blocks from several submatrices.

```
sage: A = matrix([[1,2],[3,4]])
sage: block_matrix([[A,-A],[2*A, A^2]])
```

$$\left(\begin{array}{cc|cc} 1 & 2 & -1 & -2 \\ 3 & 4 & -3 & -4 \\ \hline 2 & 4 & 7 & 10 \\ 6 & 8 & 15 & 22 \end{array}\right)$$

By default, this structure is square by blocks but the number of block rows or block columns can be specified by the optional arguments `ncols` and `nrows`. Whenever it makes sense, a scalar coefficient, such as `0` or `1`, is interpreted as a block, namely a zero block or the identity block, with conforming dimensions.

```
sage: A = matrix([[1,2,3],[4,5,6]])
sage: block_matrix([1,A,0,0,-A,2], ncols=3)
```

$$\left(\begin{array}{cc|ccc|cc} 1 & 0 & 1 & 2 & 3 & 0 & 0 \\ 0 & 1 & 4 & 5 & 6 & 0 & 0 \\ \hline 0 & 0 & -1 & -2 & -3 & 2 & 0 \\ 0 & 0 & -4 & -5 & -6 & 0 & 2 \end{array}\right)$$

In the special case of block diagonal matrices, the list of the diagonal blocks is simply passed to the constructor `block_diagonal_matrix`.

```
sage: A = matrix([[1,2,3],[0,1,0]])
sage: block_diagonal_matrix(A, A.transpose())
```

$$\left(\begin{array}{ccc|cc} 1 & 2 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 3 & 0 \end{array}\right)$$

The block structure is only a display feature and Sage treats the matrix as any other matrix. This display mode can be disabled by providing the argument `subdivide=False` to the `block_matrix` constructor.

8.1.3 Basic Manipulations And Arithmetic on Matrices

Indexing And Accessing Coefficients. Coefficients and submatrices are accessed in a unified way through the square bracket operator `A[i,j]`, following

the usual Python conventions. Row and column indices i and j can be integers (in order to access a coefficient) or intervals of the form `1:3` (recall that indices are zero based in Python and intervals are always inclusive on their lower end and exclusive on their upper end). The interval ":" without bounds corresponds to the entirety of the possible indices in the dimension considered. Notation `a:b:k` lists all indices between a and $b-1$ by steps of k. Lastly, negative indices are also valid and allow one to iterate from the end of the index space. Thus `A[-2,:]` refers to the second to last row of matrix `A`. These access patterns to submatrices are available for both read and write operations. For instance, a given column can be modified as follows:

```
sage: A = matrix(3,3,range(9))
sage: A[:,1] = vector([1,1,1]); A
```

$$\begin{pmatrix} 0 & 1 & 2 \\ 3 & 1 & 5 \\ 6 & 1 & 8 \end{pmatrix}$$

The step increment k can also be negative, in order to iterate in decreasing order.

```
sage: A[::-1], A[:,::-1], A[::2,-1]
```

$$\left(\begin{pmatrix} 6 & 1 & 8 \\ 3 & 1 & 5 \\ 0 & 1 & 2 \end{pmatrix}, \begin{pmatrix} 2 & 1 & 0 \\ 5 & 1 & 3 \\ 8 & 1 & 6 \end{pmatrix}, \begin{pmatrix} 2 \\ 8 \end{pmatrix}\right)$$

Extracting a Submatrix. In order to extract a submatrix from a list of rows or column indices, not necessarily contiguous, one can use the methods `A.matrix_from_rows`, `A.matrix_from_columns` or in the more general setting the method `A.matrix_from_rows_and_columns`.

```
sage: A = matrix(ZZ,4,4,range(16)); A
```

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{pmatrix}$$

```
sage: A.matrix_from_rows_and_columns([0,2,3],[1,2])
```

$$\begin{pmatrix} 1 & 2 \\ 9 & 10 \\ 13 & 14 \end{pmatrix}$$

Alternatively, when the row and column indices are contiguous, one can also use the method `A.submatrix(i,j,m,n)` forming the submatrix of dimension $m \times n$ whose upper left coefficient is at position (i, j) in `A`.

Basic operations	
transpose, conjugate	`A.tranpose()` , `A.conjugate()`
scalar product	`a*A`
sum, product, k-th power, inverse	`A + B`, `A * B`, `A^k`, `A^-1` *or* `~A`

TABLE 8.2 – Basic operations and matrix arithmetic.

Embedding And Extension. The method `base_extend` of a matrix space makes it possible to embed a matrix space into another matrix space with the same dimensions but over an extension of the base ring. This operation is however only valid for a field or a ring extension. In order to change the ring of a matrix space, following a ring morphism (when it exists), one can rather use the method `change_ring`.

```
sage: MS = MatrixSpace(GF(3),2,3)
sage: MS.base_extend(GF(9,'x'))
Full MatrixSpace of 2 by 3 dense matrices over Finite Field
in x of size 3^2
sage: MS = MatrixSpace(ZZ,2,3)
sage: MS.change_ring(GF(3))
Full MatrixSpace of 2 by 3 dense matrices over Finite Field of size 3
```

Mutability And Caching. By default, matrix objects are mutable, which means that one can modify their members (namely their coefficients) after their construction. In order to protect the matrix against modification, one can set it immutable with the function `A.set_immutable()`. It is then still possible to create mutable copies of this matrix with the function `copy(A)`. Remark that the caching mechanism for the computed results, such as the rank, the determinant, etc, is always active, regardless of the mutability status.

8.1.4 Basic Operations on Matrices

Arithmetic operations on matrices are done with the usual operators `+,-,*,^`. The inverse of a matrix `A` is obtained equivalently by `A^-1` or `~A`. For a scalar `a` and a matrix `A`, the operation `a*A` corresponds to the scalar multiplication of the matrix space. For any other operation where a scalar `a` is provided in place of a matrix (as for instance in the operation `a+A`), this scalar is interpreted as the corresponding scalar matrix aI_n if $a \neq 0$ if dimensions permit it. Elementwise product of two matrices is achieved by the method `elementwise_product`.

8.2 Matrix Computations

In linear algebra, matrices are typically used to represent families of vectors, systems of linear equations, linear applications, or sub-vectorspaces. Consequently, computing properties such as the rank of a family of vectors, the solution to

a linear system, the eigenspaces of a linear application or the dimension of a subspace all boil down to transformations on the corresponding matrices which will reveal the property.

These transformations most often correspond to changes of basis, which from the matrix point of view translate into equivalence transformations: $B = PAQ^{-1}$, where P and Q are invertible matrices. Two matrices are equivalent if such a transformation from one to another exists. One can then form classes of equivalence for this relation and define normal forms that characterize each equivalence class in a unique manner. In the following, we will present most matrix computations in Sage, from the viewpoint of two instances of these transformations:

- The left equivalence transformations, of the form $B = UA$, revealing characteristic properties for families of vectors, such as their rank (the number of linearly independent vectors), the determinant (the volume of the parallelepiped formed by the family of vectors), the rank profile (the first set of vectors forming a basis of the space spanned by the family)... Gaussian elimination is the key tool for these transformations and the reduced echelon form is the corresponding normal form (or the Hermite form over $\mathbb{Z}$).

- Similarity transformations, of the form $B = UAU^{-1}$, which reveal characteristic properties of the matrices representing endomorphisms, like eigenvalues, eigenspaces, minimal and characteristic polynomials... The Jordan or Frobenius form, according to the underlying domain, will be normal forms for these transformations.

The Gram-Schmidt orthogonalisation process leads to another decomposition based on left equilavence transformations, changing a matrix into a set of orthogonal vectors.

8.2.1 Gaussian Elimination, Echelon Form

Gaussian Elimination And Left Equivalence. Gaussian elimination is a building block operation in computational linear algebra, as it gives access to a matrix representation, a product of triangular factors, which is both better suited for computations, e.g., solving linear systems, and which reveals fundamental properties such as the rank, the rank profile, the determinant, etc. The basic operations used to define Gaussian elimination are the elementary row operations:

- permuting two rows: $L_i \leftrightarrow L_j$,

- adding a multiple of a row to another: $L_i \leftarrow L_i + sL_j$.

From a matrix point of view, these transformations correspond to the left multiplication by respectively a transposition matrix $T_{i,j}$ and by a transvection

Gaussian elimination and applications	
row transvection	`add_multiple_of_row(i,j,s)`
column transvection	`add_multiple_of_column(i,j,s)`
row, column transposition	`swap_rows(i1,i2)`, `swap_columns(j1,j2)`
reduced row echelon form, immutable	`echelon_form`
reduced row echelon form, in-place	`echelonize`
invariant factors	`elementary_divisors`
Smith normal form	`smith_form`
determinant, rank	`det`, `rank`
minors of order k	`minors(k)`
column, row rank profile	`pivots`, `pivot_rows`
left-hand side system solving ($x^tA = b^t$)	`b/A` *or* `A.solve_left(b)`
right-hand side system solving ($Ax = b$)	`A\b` *or* `A.solve_right(b)`
image space	`image`
left kernel	`kernel` *or* `left_kernel`
right kernel	`right_kernel`
kernel in the base ring	`integer_kernel`

Spectral decomposition	
minimal polynomial	`minimal_polynomial` *or* `minpoly`
characteristic polynomial	`characteristic_polynomial` *or* `charpoly`
Krylov iterates on the left-hand side	`maxspin(v)`
eigenvalues	`eigenvalues`
left, right eigenvectors	`eigenvectors_left`, `eigenvectors_right`
left, right eigenspaces	`eigenspaces_left`, `eigenspaces_right`
diagonalisation	`eigenmatrix_left`, `eigenmatrix_right`
Jordan block $J_{a,k}$	`jordan_block(a,k)`

TABLE 8.3 – Matrix computations.

matrix $C_{i,j,s}$ defined by:

$$T_{i,j} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccc} & & & i & & j & \\ \end{array} \\ \left[\begin{array}{ccccccc} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{array}\right] \end{array}, C_{i,j,s} = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \left[\begin{array}{ccccccc} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & s & & \\ & & & \ddots & & & \\ & & & & 1 & & \\ & & & & & \ddots & \end{array}\right] \\ \end{array} \begin{array}{c} \\ \\ \\ \\ i \\ \\ j \end{array}.$$

These matrices all have determinant 1 or -1. As a consequence, multiplying on the left by any product of these matrices, is a volume preserving change of basis, namely preserving the determinant (up to the sign). In Sage, a transvection is achieved by the method `add_multiple_of_row(i,j,s)`, and a transposition by the method `swap_rows(i,j)`.

For a given column vector $x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ whose k-th coefficient is invertible, the Gauss transform is the composition of the transvections C_{i,k,ℓ_i} for $i = k+1 \dots m$, with $\ell_i = -\frac{x_i}{x_k}$ (the order is irrelevant, since they all commute one with the other). The corresponding matrix is the following:

$$G_{x,k} = C_{k+1,k,\ell_{k+1}} \times \cdots \times C_{m,k,\ell_m} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & \ell_{k+1} & \ddots & & & \\ & & \ell_{k+2} & & \ddots & & \\ & & \vdots & & & \ddots & \\ & & \ell_m & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ k \\ \\ \\ \\ \\ \end{matrix} .$$

The effect of a Gauss transform $G_{x,k}$ is to eliminate the coefficients of the vector below the pivot x_k.

$$G_{x,k} \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \\ 0 \\ \vdots \\ 0 \end{bmatrix} .$$

For an $m \times n$ matrix $A = [a_{i,j}]$, the Gaussian elimination algorithm then proceeds iteratively, from the leftmost column to the rightmost column. Assuming that the $k-1$ first columns have already been processed, generating $p \leq k-1$ pivots, the k-th column is then treated as follows:

- find the first invertible coefficient $a_{i,k}$ in the column C_k on a row $i > p$. It is the pivot.
- If no pivot can be found, move on to the next column.
- Apply the transposition $T_{i,p+1}$ on the rows of the matrix, to place the pivot at position $(p+1, k)$.
- Apply the Gauss transform $G_{x,p+1}$, where x is the new k-th column C_k.

This algorithm transforms the matrix A into an upper triangular matrix. More precisely, it will have an echelon form: the leading coefficient of each non-zero row is to the right of that of the preceding row, and all zero rows are on the bottom part of the matrix. The following example traces the execution of this algorithm on a 4×3 matrix.

```
sage: a = matrix(GF(7),4,3,[6,2,2,5,4,4,6,4,5,5,1,3]); a
```

$$\begin{pmatrix} 6 & 2 & 2 \\ 5 & 4 & 4 \\ 6 & 4 & 5 \\ 5 & 1 & 3 \end{pmatrix}$$

```
sage: u = copy(identity_matrix(GF(7),4)); u[1:,0] = -a[1:,0]/a[0,0]
sage: u, u*a
```

$$\left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 \\ 6 & 0 & 1 & 0 \\ 5 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 6 & 2 & 2 \\ 0 & 0 & 0 \\ 0 & 2 & 3 \\ 0 & 4 & 6 \end{pmatrix}\right)$$

```
sage: v = copy(identity_matrix(GF(7),4)); v.swap_rows(1,2)
sage: b = v*u*a; v, b
```

$$\left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 6 & 2 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 4 & 6 \end{pmatrix}\right)$$

```
sage: w = copy(identity_matrix(GF(7),4))
sage: w[2:,1] = -b[2:,1]/b[1,1]; w, w*b
```

$$\left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 5 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 6 & 2 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\right)$$

Gauss-Jordan Elimination. The Gauss-Jordan transformation is similar to the Gauss transformation, simply adding to $G_{x,k}$ the transvections corresponding to the rows of index $i < k$; this has the effect of eliminating all coefficients, above and below the pivot. If in addition each row is divided by its pivot, this leads to the so called reduced echelon form or Gauss-Jordan form. It is a normal form: for every equivalence class, there is a unique such reduced echelon form.

DEFINITION. A matrix is in reduced echelon form if:

- all zero rows are at the bottom,
- the leading coefficient of every non-zero row, called a pivot, is a 1 and is to the right of the pivot of the row above,
- pivots are the only non-zero elements in their column.

THEOREM. For every $m \times n$ matrix A over a field, there is a unique $m \times n$ matrix R in reduced echelon form and an invertible $m \times m$ matrix U such that $UA = R$.

In Sage, the reduced echelon form is obtained by the methods `echelonize` and `echelon_form`. The former replaces the input matrix by its reduced echelon form, while the latter returns an immutable matrix without modifying the input matrix.

```
sage: A = matrix(GF(7),4,5,[4,4,0,2,4,5,1,6,5,4,1,1,0,1,0,5,1,6,6,2])
sage: A, A.echelon_form()
```

$$\left(\begin{pmatrix} 4 & 4 & 0 & 2 & 4 \\ 5 & 1 & 6 & 5 & 4 \\ 1 & 1 & 0 & 1 & 0 \\ 5 & 1 & 6 & 6 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 5 & 0 & 3 \\ 0 & 1 & 2 & 0 & 6 \\ 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}\right)$$

Most variants of the Gaussian elimination algorithm yield a matrix decomposition of great interest for computations: decompositions of the form $A = LU$ for generic matrices, $A = LUP$ for regular matrices, $A = LSP, LQUP, PLUQ$ for matrices with arbitrary rank. In these decompositions, the L matrices are lower triangular (with zeros above the main diagonal), with a diagonal of ones, the U matrices are upper triangular (with zeros below the main diagonal) with a diagonal of invertible elements, and P and Q are permutation matrices. Although these decompositions are less expensive to compute than the reduced echelon form (nearly $\frac{2}{3}n^3$ against $2n^3$ for an $n \times n$ full rank matrix), they do not produce a normal form.

Echelon Form Over an Euclidean Ring. Over an Euclidean ring, nonzero coefficients are not necessarily invertible while only the invertible ones can be chosen as pivots in the course of Gaussian elimination. Hence some nonzero columns may not contain any pivot, and elimination would no longer be possible. It is however still possible to define a unimodular transformation (whose determinant is invertible) eliminating the leading coefficient in a row with that of another row, thanks to the extended Euclidean algorithm. Let $A = \begin{bmatrix} a & * \\ b & * \end{bmatrix}$ and $g = \gcd(a, b)$. Let u and v be the Bézout coefficients computed with the extended Euclidean algorithm applied to a and b (such that $g = ua + vb$), and let $s = -b/g, t = a/g$ such that

$$\begin{bmatrix} u & v \\ s & t \end{bmatrix} \begin{bmatrix} a & * \\ b & * \end{bmatrix} = \begin{bmatrix} g & * \\ 0 & * \end{bmatrix}.$$

This transformation is unimodular since $\det \begin{pmatrix} u & v \\ s & t \end{pmatrix} = 1$.

Morevover, as in the Gauss-Jordan elimination, it is also always possible to add multiples of the pivot row to the rows above it in order to reduce the coefficients in the pivot column modulo the pivot g. When iterated over all columns of the matrix, this operation produces the Hermite normal form.

Definition. A matrix is in Hermite normal form if:

- its zero rows are at the bottom,

- the leading coefficient of each non-zero row, called the pivot, is to the right of the pivot of the preceding row,
- all coefficients above a pivot are reduced modulo the pivot.

THEOREM. For any $m \times n$ matrix A over an Euclidean ring, there is a unique $m \times n$ matrix H in Hermite form and an $m \times m$ unimodular matrix U, such that $UA = H$.

Over a field, the Hermite form coincides with the reduced echelon form. Indeed all pivots are then invertible, each non-zero row can be divided by its pivot, making this pivot equal to one. Then reducing the coefficients above each pivot modulo one, means setting them to zero, which produces a reduced echelon form. In Sage, there is therefore a unique method, `echelon_form`, which either returns the Hermite form or the reduced echelon form depending whether the coefficient domain is an Euclidean ring or a field.

For instance, a matrix with integer coefficients yields the following two distinct reduced echelon forms depending on whether the base ring is $\mathbb{Z}$ or $\mathbb{Q}$.

```
sage: a = matrix(ZZ, 4, 6, [2,1,2,2,2,-1,1,2,-1,2,1,-1,2,1,-1,\
....:                       -1,2,2,2,1,1,-1,-1,-1]); a.echelon_form()
```

$$\begin{pmatrix} 1 & 2 & 0 & 5 & 4 & -1 \\ 0 & 3 & 0 & 2 & -6 & -7 \\ 0 & 0 & 1 & 3 & 3 & 0 \\ 0 & 0 & 0 & 6 & 9 & 3 \end{pmatrix}$$

```
sage: a.base_extend(QQ).echelon_form()
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \frac{5}{2} & \frac{11}{6} \\ 0 & 1 & 0 & 0 & -3 & -\frac{8}{3} \\ 0 & 0 & 1 & 0 & -\frac{3}{2} & -\frac{3}{2} \\ 0 & 0 & 0 & 1 & \frac{3}{2} & \frac{1}{2} \end{pmatrix}$$

For matrices over $\mathbb{Z}$, the Hermite form can also be obtained with the method `hermite_form`. With both methods, the transformation matrix U, such that $UA = H$ is also returned when the option `transformation=True` is being passed.

```
sage: A = matrix(ZZ,4,5,[4,4,0,2,4,5,1,6,5,4,1,1,0,1,0,5,1,6,6,2])
sage: H, U = A.echelon_form(transformation=True); H, U
```

$$\left(\begin{pmatrix} 1 & 1 & 0 & 0 & 2 \\ 0 & 4 & -6 & 0 & -4 \\ 0 & 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 1 & -1 \\ 0 & -1 & 5 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & -2 & -4 & 2 \end{pmatrix} \right)$$

Invariant Factors And The Smith Normal Form. When eliminating further the Hermite normal form, using unimodular right transformations (acting on columns) one can then reach a diagonal canonical form, named the Smith normal form. Its diagonal coefficients are the elementary divisors of the matrix. They are totally ordered under the divisibility relation: $s_i \mid s_{i+1}$.

THEOREM. For any $m \times n$ matrix A with coefficients over a principal ideal ring, there exist unimodular matrices U and V of dimensions $m \times m$ and $n \times n$ respectively, and a unique $m \times n$ diagonal matrix S such that $S = UAV$. The coefficients $s_i = S_{i,i}$ for $i \in \{1, \ldots, \min(m,n)\}$ are the elementary divisors of A and satisfy $s_i \mid s_{i+1}, \forall i < \min(m,n)$.

In Sage, the method `elementary_divisors` returns the list of elementary divisors. One can also compute directly the Smith normal form, together with the transformation matrices U and V, with the `smith_form` method.

```
sage: A = matrix(ZZ, 4, 5,\
....:            [-1,-1,-1,-2,-2,-2,1,1,-1,2,2,2,2,2,-1,2,2,2,2,2])
sage: S,U,V = A.smith_form(); S,U,V
```

$$\left(\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -2 & 1 & 0 & 0 \\ 0 & 0 & -2 & -1 \end{pmatrix}, \begin{pmatrix} 0 & -2 & -1 & -5 & 0 \\ 1 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ -1 & 2 & 0 & 5 & 0 \\ 0 & -1 & 0 & -2 & 0 \end{pmatrix}\right)$$

```
sage: A.elementary_divisors()
[1, 1, 3, 6]
sage: S == U*A*V
True
```

Rank, Rank Profile And Pivots. Gaussian elimination reveals many matrix invariants, such as the rank, the determinant (computed as the product of the pivots). These values are accessible with the methods `det` and `rank`. They are cached and therefore are not recomputed when the method is called once again.

More generally, the notion of rank profile is of interest when considering the matrix as a sequence of vectors.

DEFINITION. The column rank profile of an $m \times n$ matrix A of rank r is the lexicographically minimal sequence of r indices of linearly independent columns in A.

The column rank profile reads off directly on the reduced row echelon form, as the sequence of column indices of the pivots. It is obtained by the method `pivots`. When the reduced row echelon form has already been computed, the column rank profile is stored in cache and can be obtained with no additional computation.

The row rank profile is defined similarly, considering the matrix as a sequence of m row vectors. It is obtained by the method `pivot_rows`. It is equivalent to the column rank profile of the transposed matrix.

```
sage: B = matrix(GF(7),5,4,[4,5,1,5,4,1,1,1,0,6,0,6,2,5,1,6,4,4,0,2])
sage: B.transpose().echelon_form()
```

$$\begin{pmatrix} 1 & 0 & 5 & 0 & 3 \\ 0 & 1 & 2 & 0 & 6 \\ 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
sage: B.pivot_rows()
(0, 1, 3)
sage: B.transpose().pivots() == B.pivot_rows()
True
```

8.2.2 Linear System Solving, Image And Nullspace Basis

Linear System Solving. A linear system of equations can be represented by a matrix A and a right-hand side or a left-hand side vector b for systems of the form $Ax = b$ or $x^tA = b^t$ respectively. The methods `solve_right` and `solve_left` solve these systems. One can alternatively use the operators `A\b` and `b/A`. When the system is given by a matrix over a ring, the resolution is systematically performed over the fraction field of this ring (e.g., $\mathbb{Q}$ for the ring $\mathbb{Z}$ or $K(X)$ for $K[X]$). We will see later on how to solve the system over the base ring. The right-hand side in the system equality can be indifferently a vector or a matrix (which corresponds to the resolution of several systems with the same matrix).

The system's matrix can be rectangular and the system may have a unique solution, no solution or an infinite number of solutions. In the latter case, methods `solve` return one of these solutions, zeroing out the coefficients corresponding to linearly dependent columns in the system.

```
sage: R.<x> = PolynomialRing(GF(5),'x')
sage: A = random_matrix(R,2,3); A
```

$$\begin{pmatrix} 3x^2+x & x^2+2x & 2x^2+2 \\ x^2+x+2 & 2x^2+4x+3 & x^2+4x+3 \end{pmatrix}$$

```
sage: b = random_matrix(R,2,1); b
```

$$\begin{pmatrix} 4x^2+1 \\ 3x^2+2x \end{pmatrix}$$

```
sage: A.solve_right(b)
```

$$\begin{pmatrix} \frac{4x^3+2x+4}{3x^3+2x^2+2x} \\ \frac{3x^2+4x+3}{x^3+4x^2+4x} \\ 0 \end{pmatrix}$$

```
sage: A.solve_right(b) == A\b
True
```

Image And Kernel. Viewed as a linear application Φ, an $m \times n$ matrix A defines two subspaces of K^m and K^n, respectively the image and the kernel of Φ.

The image is the set of all vectors in K^m which are a linear combination of the columns of A. It is given by the method `image`, returning a vector space, with a basis in reduced echelon form.

The kernel is the subspace of K^n formed by all vectors x satisfying $Ax = 0$. A basis of this subspace is useful to describe the set of solutions of an underdetermined linear system, having an infinite number of solutions: if $\overline{x}$ is a solution of $Ax = b$ and V is the kernel of A, then $\overline{x} + V$ is the set of all solutions to the system. This set is computed by the method `right_kernel` returning a vector space with a basis in reduced echelon form. The left kernel is defined similarly as the set of vectors $x \in K^m$ such that $x^t A = 0$, which is also the right kernel of the transposed matrix of A. It is returned by the method `left_kernel`. By convention the method `kernel` returns the left kernel and the bases are given as matrices of row vectors.

```
sage: a = matrix(QQ,3,5,[2,2,-1,-2,-1,2,-1,1,2,-1/2,2,-2,-1,2,-1/2])
sage: a.image()
Vector space of degree 5 and dimension 3 over Rational Field
Basis matrix:
[     1      0      0    1/4 -11/32]
[     0      1      0     -1   -1/8]
[     0      0      1    1/2   1/16]
sage: a.right_kernel()
Vector space of degree 5 and dimension 2 over Rational Field
Basis matrix:
[    1     0     0  -1/3   8/3]
[    0     1  -1/2 11/12   2/3]
```

The notion of kernel extends naturally to the case where coefficients no longer belong to a field, but a ring; it then has the structure of a free module. In particular, for a matrix defined over the fraction field of an integral ring, the kernel in the base ring is obtained with the method `integer_kernel`. For instance, the kernel of a matrix over $\mathbb{Z}$, embedded in the vector space over the field $\mathbb{Q}$ can either be the $\mathbb{Q}$-subvector space of $\mathbb{Q}^m$ or a $\mathbb{Z}$ free module of $\mathbb{Z}^m$.

```
sage: a = matrix(ZZ,5,3,[1,1,122,-1,-2,1,-188,2,1,1,-10,1,-1,-1,-1])
sage: a.kernel()
Free module of degree 5 and rank 2 over Integer Ring
Echelon basis matrix:
[   1  979  -11 -279  811]
[   0 2079  -22 -569 1488]
sage: b = a.base_extend(QQ)
sage: b.kernel()
Vector space of degree 5 and dimension 2 over Rational Field
Basis matrix:
[        1         0  -121/189 -2090/189   6949/63]
[        0         1    -2/189 -569/2079   496/693]
sage: b.integer_kernel()
Free module of degree 5 and rank 2 over Integer Ring
Echelon basis matrix:
[   1  979  -11 -279  811]
[   0 2079  -22 -569 1488]
```

8.2.3 Eigenvalues, Jordan Form And Similarity Transformation

A square matrix A is the representation of a linear operator, an endomorphism, in a given basis. Any change of basis corresponds to a similarity transformation of the form $B = U^{-1}AU$. The matrix B represents the linear operator in the new basis and the two matrices A and B are then *similar*. The properties of the linear operator, which are independent from the basis of representation, are thus revealed by the study of the similarity invariants of the matrix, namely its properties that remain invariant by similarity transformation.

Among these invariants, the most elementary are the rank and the determinant. Indeed, since the matrices U and U^{-1} are invertible, the rank of $U^{-1}AU$ is the rank of A. Moreover, $\det(U^{-1}AU) = \det(U^{-1})\det(A)\det(U) = \det(U^{-1}U)\det(A) = \det(A)$. Similarly, the characteristic polynomial of the matrix A, defined as $\chi_A(x) = \det(x\mathrm{Id} - A)$ is also invariant by similarity transformation:

$$\det(x\mathrm{Id} - U^{-1}AU) = \det(U^{-1}(x\mathrm{Id} - A)U) = \det(x\mathrm{Id} - A).$$

Consequently, the characteristic values of a matrix, defined as the roots of its characteristic polynomial in its splitting field, are thus also similarity invariants.

By definition, a scalar λ is an eigenvalue of a matrix A if there exists a non-zero vector u such that $Au = \lambda u$. The eigenspace associated with an eigenvalue λ is the set of all such vectors u verifying $Au = \lambda u$. It is a linear subspace defined by $E_\lambda = \mathrm{Ker}(\lambda\mathrm{Id} - A)$.

Eigenvalues coincide with chacteristic values:

$$\det(\lambda\mathrm{Id} - A) = 0 \Leftrightarrow \dim(\mathrm{Ker}(\lambda\mathrm{Id} - A)) \geq 1 \Leftrightarrow \exists u \neq 0, \lambda u - Au = 0.$$

These two points of view respectively correspond to the algebraic and the geometric approach to eigenvalues. The geometric viewpoint considers the action of the linear operator A on vectors in the ambiant space with more precision than in the algebraic viewpoint. In particular, they differ with the notion of multiplicity of an eigenvalue: the algebraic multiplicity is the multiplicity of the root in the characteristic polynomial while the geometric multiplicity is the dimension of the eigenspace associated to the eigenvalue. When the matrix is diagonalisable, these notions are equivalent, but otherwise the geometric multiplicity is less than or equal to the algebraic multiplicity.

The geometric point of view gives finer details on the structure of the matrix. It also helps designing efficient algorithms to compute eigenvalues, eigenvectors and the characteristic and minimal polynomials.

Cyclic Invariant Subspace And Frobenius Normal Form. Let A be an $n \times n$ matrix over a field K and $u \in K^n$ a vector. The vectors $v, Au, A^2u, \ldots A^nu$, called the Krylov sequence, are linearly dependent (as it is a set of $n+1$ vectors of dimension n). Let d be the first index such that A^du is linearly dependent with

its predecessors $u, Au, \ldots, A^{d-1}u$. We can write this linear dependence relation as

$$A^d u = \sum_{i=0}^{d-1} \alpha_i A^i u.$$

The polynomial $\varphi_{A,u}(x) = x^d - \sum_{i=0}^{d-1} \alpha_i x^i$, satisfying the relation $\varphi_{A,u}(A)u = 0$ is therefore a unit polynomial annihilating the Krylov sequence, of minimal degree. It is named the *minimal polynomial* of the vector u (with respect to the matrix A). The set of all annihilating polynomials of u forms an ideal of $K[X]$, generated by $\varphi_{A,u}$.

The minimal polynomial of the matrix A is defined as the least degree unit polynomial annihilating the matrix A: $\varphi_A(A) = 0$. In particular, applying the vector u to the right in this equation shows that φ_A is annihilating the Krylov sequence for u. It is therefore necessarily a multiple of the minimal polynomial of u. In addition, one can prove (see Exercice 31) that there always exists a vector $\overline{u}$ such that

$$\varphi_{A,\overline{u}} = \varphi_A. \tag{8.1}$$

When the vector u is chosen at random, the probability that it satisfies Equation (8.1) increases with the size of the field (one shows that it is at least $1 - \frac{n}{|K|}$).

Exercise 31. We will show that there always exists a vector $\overline{u}$ whose minimal polynomial coincides with the minimal polynomial of the matrix.

1. Let $(e_1, \ldots, e_n)$ be a basis of the vector space. Show that φ_A is equal to the least common multiple of the φ_{A,e_i} for all $1 \leq i \leq n$.
2. In the case where φ_A is an irreducible polynomial raised to some power, show that there is an index i_0 such that $\varphi_A = \varphi_{A,e_{i_0}}$.
3. Show that if the minimal polynomials $\varphi_i = \varphi_{A,e_i}$ and $\varphi_j = \varphi_{A,e_j}$ of the vectors e_i and e_j are coprime, then $\varphi_{A,e_i+e_j} = \varphi_i \varphi_j$.
4. Show that if $\varphi_A = P_1 P_2$ where P_1 and P_2 are coprime, then there exist two vectors $x_1 \neq 0$ and $x_2 \neq 0$ such that P_i is the minimal polynomial of x_i for $i = 1, 2$.
5. Conclude using the factorization of φ_A in irreducible factors $\varphi_A = \varphi_1^{m_1} \ldots \varphi_k^{m_k}$.
6. Illustration: let $A = \begin{bmatrix} 0 & 0 & 3 & 0 & 0 \\ 1 & 0 & 6 & 0 & 0 \\ 0 & 1 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 5 \end{bmatrix}$ be a matrix over GF(7). Compute the degrees of the minimal polynomial of A, of the minimal polynomials of the vectors $u = e_1$ and $v = e_4$ of the canonical basis, and of the vector $u + v$. One can use the method `maxspin(u)` applied to the transposed of the matrix A, returning the maximal sequence of linearly independent Krylov iterates of the vector u.

Let $P = x^k + \sum_{i=0}^{k-1} \alpha_i x^i$ be a unit polynomial of degree k. The companion matrix associated with the polynomial P is the $k \times k$ matrix C_P defined as

$$C_P = \begin{bmatrix} 0 & & & -\alpha_0 \\ 1 & & & -\alpha_1 \\ & \ddots & & \vdots \\ & & 1 & -\alpha_{k-1} \end{bmatrix}.$$

This matrix has the property that P equals its minimal polynomial and its characteristic polynomial.

PROPOSITION. Let K_u be the matrix formed by the d first Krylov iterates of a vector u. Then

$$AK_u = K_u C_{\varphi_{A,u}}.$$

Hence, when $d = n$, the matrix K_u is square and invertible. Therefore it defines a similarity transformation $K_u^{-1} A K_u = C_{\varphi_{A,u}}$ reducing A to a companion matrix. Now this transformation preserves the determinant and thus also the characteristic polynomial. The coefficients of the characteristic and minimal polynomial (which are identical in this case) can therefore be read off directly on the last column of the companion matrix.

```
sage: A = matrix(GF(97), 4, 4,\
....:            [86,1,6,68,34,24,8,35,15,36,68,42,27,1,78,26])
sage: e1 = identity_matrix(GF(97),4)[0]
sage: U = matrix(A.transpose().maxspin(e1)).transpose()
sage: F = U^-1*A*U; F
```

$$\begin{pmatrix} 0 & 0 & 0 & 83 \\ 1 & 0 & 0 & 77 \\ 0 & 1 & 0 & 20 \\ 0 & 0 & 1 & 10 \end{pmatrix}$$

```
sage: K.<x> = GF(97)[]
sage: P = x^4-sum(F[i,3]*x^i for i in range(4)); P
```

$$x^4 + 87x^3 + 77x^2 + 20x + 14$$

```
sage: P == A.charpoly()
True
```

In the general case ($d \leq n$) the iterates $u, \ldots, A^{d-1}u$ form a basis of a linear subspace I invariant under the action of the matrix A (i.e., such that $AI \subseteq I$). This subspace is also called cyclic invariant subspace, as these vectors are obtained cyclicly by applying the matrix A to the preceding vector. The dimension of this subspace is the degree of the minimal polynomial of u and is therefore bounded by the degree of the minimal polynomial of the matrix A. When the dimension is maximal, the space is generated by the Krylov iterates of the vector constructed in Exercise 31, which we will denote by u_1^*. It is called the first invariant subspace. Let V be the supplementary subspace of this first invariant subspace. Computing *modulo* the first invariant subspace, i.e., by considering that two vectors are equal whenever their difference belongs to the first invariant subspace, one can define a second invariant subspace for vectors in this supplementary subspace, as well as a minimal polynomial which is called the second similarity invariant. In this case we have a relation of the form:

$$A \begin{bmatrix} K_{u_1^*} & K_{u_2^*} \end{bmatrix} = \begin{bmatrix} K_{u_1^*} & K_{u_2^*} \end{bmatrix} \begin{bmatrix} C_{\varphi_1} & \\ & C_{\varphi_2} \end{bmatrix},$$

where φ_1, φ_2 are the first two similarity invariants and $K_{u_1^*}, K_{u_2^*}$ are the Krylov matrices corresponding to the two cyclic subspaces generated by the vectors u_1^* and u_2^*.

Iteratively, one can build a matrix $K = [K_{u_1^*} \quad \dots \quad K_{u_k^*}]$ square and invertible and such that

$$K^{-1}AK = \begin{bmatrix} C_{\varphi_1} & & \\ & \ddots & \\ & & C_{\varphi_k} \end{bmatrix}. \tag{8.2}$$

As each u_i^* is annihilated by the φ_j for $j \leq i$, we have that $\varphi_i \mid \varphi_{i-1}$ for any $2 \leq i \leq k$. Equivalently, the sequence of the φ_i is totally ordered for the division. One shows that for every matrix there exists a unique sequence of similarity invariants $\varphi_1, \dots, \varphi_k$. Therefore the block diagonal matrix $\mathrm{Diag}(C_{\varphi_1}, \dots, C_{\varphi_k})$, similar to the matrix A and revealing these polynomials, is a normal form, called the rational canonical form or the Frobenius normal form.

THEOREM (Frobenius normal form). For every matrix A over a field, there is a unique matrix $F = \begin{bmatrix} C_{\varphi_1} & & \\ & \ddots & \\ & & C_{\varphi_k} \end{bmatrix}$, with $\varphi_{i+1} \mid \varphi_i$ for all $i < k$, similar to A.

Equation (8.2) shows that one can read off the bases of the invariant subspaces on the transformation matrix K.

REMARK.

$$\begin{aligned} \chi_A(x) &= \det(x\mathrm{Id} - A) = \det(K)\det(x\mathrm{Id} - F)\det(K^{-1}) \\ &= \prod_{i=1}^{k} \det(x\mathrm{Id} - C_{\varphi_i}) = \prod_{i=1}^{k} \varphi_i(x). \end{aligned}$$

Hence, the minimal polynomial φ_1 is a divisor of the characteristic polynomial, which is therefore annihilating the matrix A.

In Sage, one can compute the Frobenius normal form over $\mathbb{Q}$ of matrices with coefficients over $\mathbb{Z}$ with the method `frobenius`[1]:

```
sage: A = matrix(ZZ,8,[[6,0,-2,4,0,0,0,-2],[14,-1,0,6,0,-1,-1,1],\
....:                  [2,2,0,1,0,0,1,0],[-12,0,5,-8,0,0,0,4],\
....:                  [0,4,0,0,0,0,4,0],[0,0,0,0,1,0,0,0],\
....:                  [-14,2,0,-6,0,2,2,-1],[-4,0,2,-4,0,0,0,4]])
sage: A.frobenius()
```

$$\begin{pmatrix} 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

One can also obtain the list of similarity invariants, by passing `1` as argument. In order to obtain information on the associated invariant subspaces, one passes

[1] It is a slight abuse in the interface of the software: although the Frobenius normal form is defined for any matrix over a field, Sage only allows to compute it with integer matrices, implicitely embedding them over $\mathbb{Q}$.

2 as argument, which will produce the transformation matrix K. It is a basis of the whole space, decomposed into the direct sum of the invariant subspaces.

```
sage: A.frobenius(1)
```

$$\left[x^4 - x^2 - 4x - 4, x^3 - x^2 - 4, x - 2\right]$$

```
sage: F,K = A.frobenius(2)
sage: K
```

$$\begin{pmatrix} 1 & -\frac{15}{56} & \frac{17}{224} & \frac{15}{56} & -\frac{17}{896} & 0 & -\frac{15}{112} & \frac{17}{64} \\ 0 & \frac{29}{224} & -\frac{13}{224} & -\frac{23}{448} & -\frac{17}{896} & -\frac{17}{896} & \frac{29}{448} & \frac{13}{128} \\ 0 & -\frac{75}{896} & \frac{75}{896} & -\frac{47}{896} & 0 & -\frac{17}{896} & -\frac{23}{448} & \frac{11}{128} \\ 0 & \frac{17}{896} & -\frac{29}{896} & \frac{15}{896} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -\frac{4}{21} & -\frac{4}{21} & -\frac{10}{21} & 0 & 0 & -\frac{2}{21} & 1 \end{pmatrix}$$

```
sage: K^-1*F*K == A
True
```

These results implicitely assume that the matrix A is embbeded in the fraction field $\mathbb{Q}$. In order to study the action of the matrix A on the free module $\mathbb{Z}^n$, and explicit the corresponding decomposition of the module, the method `decomposition` can be used. However, further explanations on this method would go beyond the scope of this book.

Invariant Factors And Similarity Invariants. There is a fundamental property relating the similarity invariants and the invariant factors, mentioned in Section 8.2.1.

THEOREM. The similarity invariants of a matrix A over a field F correspond to the invariant factors of its characteristic matrix $x\mathrm{Id} - A$ over the ring $F[x]$.

The proof of this theorem goes well beyond the scope of this book and we will only illustrate it on the previous example.

```
sage: S.<x> = QQ[]
sage: B = x*identity_matrix(8) - A
sage: B.elementary_divisors()
```

$$\left[1, 1, 1, 1, 1, x - 2, x^3 - x^2 - 4, x^4 - x^2 - 4x - 4\right]$$

```
sage: A.frobenius(1)
```

$$\left[x^4 - x^2 - 4x - 4, x^3 - x^2 - 4, x - 2\right]$$

Eigenvalues, Eigenvectors. Considering the decomposition of the minimal polynomial in irreducible factors, $\varphi_1 = \psi_1^{m_1} \dots \psi_s^{m_s}$, then every invariant factor can be written in the form $\varphi_i = \psi_1^{m_{i,1}} \dots \psi_s^{m_{i,s}}$, with multiplicities $m_{i,j} \leq m_j$. One can show that there is a similarity transformation, replacing each companion block C_{φ_i} in the Frobenius normal form into a diagonal block $\mathrm{Diag}(C_{\psi_1^{m_{i,1}}}, \dots, C_{\psi_s^{m_{i,s}}})$. This variant of the Frobenius normal form, called intermediary form, is still composed by companion blocks but each of which now corresponds to an irreducible polynomial raised to some power.

$$F = \begin{bmatrix} \boxed{\begin{matrix} C_{\psi_1^{m_{1,1}}} & & \\ & \ddots & \\ & & C_{\psi_s^{m_{1,s}}} \end{matrix}} & & & \\ & \boxed{\begin{matrix} C_{\psi_1^{m_{2,1}}} & \\ & \ddots \end{matrix}} & & \\ & & \ddots & \\ & & & \boxed{\begin{matrix} C_{\psi_1^{m_{k,1}}} & \\ & \ddots \end{matrix}} \end{bmatrix} \tag{8.3}$$

When an irreducible factor ψ_i has degree 1 and multiplicity 1, its companion block is a 1×1 matrix on the main diagonal and thus corresponds to an eigenvalue. When the minimal polynomial splits and is squarefree, the matrix is then diagonalisable.

The eigenvalues are obtained with the method `eigenvalues`. The methods `eigenvectors_right` and `eigenvectors_left` list for each eigenvalue, the right (respectively left) eigenvectors associated together with the multiplicity of the eigenvalue. Lastly, the eigenspaces together with a basis of eigenvectors are returned by the methods `eigenspaces_right` and `eigenspaces_left`.

```
sage: A = matrix(GF(7),4,[5,5,4,3,0,3,3,4,0,1,5,4,6,0,6,3])
sage: A.eigenvalues()
[4, 1, 2, 2]
sage: A.eigenvectors_right()
[(4, [
(1, 5, 5, 1)
], 1), (1, [
(0, 1, 1, 4)
], 1), (2, [
(1, 3, 0, 1),
(0, 0, 1, 1)
], 2)]
sage: A.eigenspaces_right()
[
(4, Vector space of degree 4 and dimension 1 over Finite Field
of size 7
User basis matrix:
[1 5 5 1]),
```

```
(1, Vector space of degree 4 and dimension 1 over Finite Field
of size 7
User basis matrix:
[0 1 1 4]),
(2, Vector space of degree 4 and dimension 2 over Finite Field
of size 7
User basis matrix:
[1 3 0 1]
[0 0 1 1])
]
```

More concisely, the method `eigenmatrix_right` returns the tuple of the diagonalised matrix and the matrix of the corresponding right eigenvectors. The `eigenmatrix_left` does similarly with the left eigenvectors.

```
sage: A.eigenmatrix_right()
```

$$\left(\begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 \\ 5 & 1 & 3 & 0 \\ 5 & 1 & 0 & 1 \\ 1 & 4 & 1 & 1 \end{pmatrix}\right)$$

Jordan Normal Form. When the minimal polynomial splits over the base field, but has factors with multiplicity greater than 1, the intermediary form (8.3) is not diagonal. One can show that there is no similarity transformation making it diagonal, hence the matrix is not diagonalisable. However it can be trigonalised, which means upper triangular, with eigenvalues on the main diagonal. Among all possible such upper triangular matrices, the most reduced one is the Jordan normal form. A Jordan block $J_{\lambda,k}$, associated with an eigenvalue λ and of order k, is the $k \times k$ matrix $J_{\lambda,k}$ given by

$$J_{\lambda,k} = \begin{bmatrix} \lambda & 1 & & \\ & \ddots & \ddots & \\ & & \lambda & 1 \\ & & & \lambda \end{bmatrix}.$$

This matrix plays a similar role as the companion blocks, revealing more precisely the multiplicity of an eigenvalue. Indeed, its characteristic polynomial is $\chi_{J_{\lambda,k}} = (X-\lambda)^k$. Moreover, its minimal polynomial also equals $\varphi_{J_{\lambda,k}} = (X-\lambda)^k$: it is necessarily a multiple of $P = X - \lambda$, now the matrix

$$P(J_{\lambda,k}) = \begin{bmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ & & & 0 \end{bmatrix}$$

is nilpotent of order k, hence $\varphi_{J_{\lambda,k}} = \chi_{J_{\lambda,k}} = (X-\lambda)^k$. The Jordan normal form corresponds to the intermediary form (8.3), where the companion blocks of the $\psi_j^{m_{i,j}}$ have been replaced by the Jordan blocks $J_{\lambda_j,m_{i,j}}$ (recall that since the minimal polynomial splits, the ψ_j are of the form $X - \lambda_j$). As a consequence,

every matrix whose minimal polynomial splits is similar to a Jordan matrix of the form

$$J = \begin{bmatrix} \boxed{\begin{matrix} J_{\lambda_1,m_{1,1}} & & \\ & \ddots & \\ & & J_{\lambda_s,m_{1,s}} \end{matrix}} & & & \\ & \boxed{\begin{matrix} J_{\lambda_1,m_{2,1}} & \\ & \ddots \end{matrix}} & & \\ & & \ddots & \\ & & & \boxed{\begin{matrix} J_{\lambda_1,m_{k,1}} & \\ & \ddots \end{matrix}} \end{bmatrix}. \tag{8.4}$$

In particular, in every algebraically closed field, such as $\mathbb{C}$, the Jordan normal form always exists. In Sage, the constructor `jordan_block(a,k)` produces the Jordan block $J_{a,k}$. The Jordan normal form is obtained by the method `jordan_form`. The option `transformation=True` makes the method also return the transformation matrix U such that $U^{-1}AU$ is in Jordan normal form.

```
sage: A = matrix(ZZ,4,[3,-1,0,-1,0,2,0,-1,1,-1,2,0,1,-1,-1,3])
sage: A.jordan_form()
```

$$\left(\begin{array}{c|c|cc} 3 & 0 & 0 & 0 \\ \hline 0 & 3 & 0 & 0 \\ \hline 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{array}\right)$$

```
sage: J,U = A.jordan_form(transformation=True)
sage: U^-1*A*U == J
True
```

The Jordan normal form is unique up to a permutation of the Jordan blocks. Depending on the bibliographic references, one sometimes imposes that their order respects the order of the similarity invariants, as in equation (8.4). Remark, from the above example, that Sage does not respect this order, since the first similarity invariant (the minimal polynomial) is the polynomial $(X-3)(X-2)^2$.

Primary Normal Form. For the sake of completeness, we should mention a last normal form, generalizing the Jordan form in the case where the minimal polynomial does not split. For an irreducible polynomial P of degree k, one defines the Jordan block of multiplicity m as the $km \times km$ matrix

$$J_{P,m} = \begin{bmatrix} C_P & B & & \\ & \ddots & \ddots & \\ & & C_P & B \\ & & & C_P \end{bmatrix}$$

where B is the $k \times k$ matrix whose coefficients are all zero except $B_{k,1} = 1$, and where C_P is the companion matrix associated to the polynomial P (§8.2.3). Note

that when $P = X - \lambda$, this definition coincides with the notion of Jordan block associated with the eigenvalue λ. One shows similarly that the minimal and characteristic polynomials of this matrix are

$$\chi_{J_{P,m}} = \varphi_{J_{P,m}} = P^m.$$

As a consequence, there exists a similarity transformation replacing each companion block $C_{\psi_j^{m_{i,j}}}$ in the intermediary form (8.3) with a Jordan block $J_{\psi_j, m_{i,j}}$. The resulting matrix is called the primary form or also the second Frobenius form. It is again a normal form, unique up to a permutation of the diagonal blocks.

The uniqueness of these normal forms is used for instance to test whether two matrices are similar, and in such a case, to produce a similarity transformation from one to the other.

Exercise 32. Write a program testing whether two input matrices A and B are similar and returning the transformation matrix U such that $A = U^{-1}BU$ (one can return `None` instead in the case where the two matrices are not similar).