Sparse matrices in PARI/GP

Work in progress

A. Page

Inria/Université de Bordeaux/IMB

15/01/2026

What is a sparse matrix?

A matrix is **sparse** if it has few nonzero entries. This is usually measured by the proportion of nonzero entries overall, or by row or column.

Manipulating a sparse matrix with the usual (dense) representation of matrices is very wasteful (memory and time). A **sparse representation** of matrices is one where we only store nonzero coefficients.

Why sparse matrices in GP?

Usually, if you care a lot about efficiency, you should use the pari library instead of GP. However, for sparse matrices, even if you only want to call one function you cannot even write the input without a sparse representation.
I am proposing a **GP interface** (open to discussion).

In addition, for a project with Jean Raimbault I wrote an experimental implementation of an algorithm for **homology computations** with sparse matrices, which other people have found useful. I would like to make it available and get feedback.

Existing libpari format: `zMs`

The existing format for sparse matrices in libpari is `zMs`: sparse matrices with integer coefficients that fit in a C `long`.

- Sparse column vector: a `zCs` is a `t_COL` with two components $[C, E]$ where C and E are `t_VECSMALL`: C is the vector of **indices** and E is the vector of **coefficients**.
- Sparse matrix: a `zMs` is a `t_VEC` of `zCs` representing the **columns** of the matrix.

Remark: the representation does not contain the number of rows of the matrix.

Conversion to/from dense matrices

```
? m = [1,0,2;0,3,0;0,0,-1]
% =
[1 0  2]
[0 3  0]
[0 0 -1]
? ms = tosmat(m)
% = [[Vecsmall([1]), Vecsmall([1])], [Vecsmall([2])
Vecsmall([3])], [Vecsmall([1, 3]), Vecsmall([2, -1]
? smattomat(ms,4) \\provide number of rows
% =
[1 0  2]
[0 3  0]
[0 0 -1]
[0 0  0]
```

Conversion to/from vectors of coefficients

Provide a vector of triples (i, j, c) meaning $M_{i,j} = c$.

```
? mv = [[4,1,-1],[2,1,10],[3,2,-5]];
? smattomat(tosmat(mv),4)
% =
[ 0  0]
[10  0]
[ 0 -5]
[-1  0]
? smattovec(ms)
% = [[1, 1, 1], [2, 2, 3], [1, 3, 2], [3, 3, -1]]
```

Conversion to/from maps

Provide a map $(i,j) \mapsto c$. Since maps can be dynamically updated, this is more flexible than vectors.

```
? mm = Map();
? mapput(~mm,[3,2],10);
? mapput(~mm,[1,3],20);
? mapput(~mm,[2,1],30);
? smattomat(tosmat(mm),3)
% =
[ 0  0 20]
[30  0  0]
[ 0 10  0]
```

Write/load SMS files

It is useful to be able to save and load sparse matrices in files. The SMS format is a simple format for integer sparse matrices, used by other software such as Linbox and SPASM (see for instance `https://hpac.imag.fr/`).

```
? smatexport("test.sms",ms,3);
? [nr,nc,ms2] = smatimport("test.sms");
? nr
% = 3
? nc
% = 3
? ms == ms2
% = 1
```

Maybe we want to use the transpose of the SMS convention.

Homology of complexes

A **complex** of abelian groups is a sequence

$$C_0 \xleftarrow{d_1} C_1 \xleftarrow{d_2} C_2 \xleftarrow{d_3} \dots \xleftarrow{d_n} C_n$$

such that $d_{i-1}d_i = 0$ for all $i > 1$, i.e. $\operatorname{im}(d_i) \subset \ker(d_{i-1})$.
Its **homology groups** are

$$H_i(C) = \ker(d_{i-1}) / \operatorname{im}(d_i).$$

Example: for any morphism $f\colon A \to B$ we get the complex

$$0 \longleftarrow B \xleftarrow{f} A \longleftarrow 0,$$

whose homology is $H_1(C) = \operatorname{coker}(f)$ and $H_2(C) = \ker(f)$.

Algebraic Morse theory

Leon Lampret proposed an algorithm to **simplify a complex** into one with the same homology: algebraic Morse theory.

Ideas:

- Construct a certain matching on the corresponding graph in linear time. An edge selected in the matching is analogous to a pivot in pivoting algorithms.
- Construct the simplified complex by a simple graph traversal.

Implementation

I wrote a toy implementation over $\mathbb{Z}$ for small coefficients.
If you compiled from source: in the pari folder, type

```
make test-amt
```

The function is `KzMs_simplify`.
A `KzMs` is a `t_VEC [dims,diffs]` where

- `dims` is a `t_VECSMALL` of dimensions $k_0, \dots, k_n$
- `diffs` is a `t_VEC` of `KzMs`, the differentials $d_1, \dots, d_n$.

$$d_i \colon \mathbb{Z}^{k_i} \longrightarrow \mathbb{Z}^{k_{i-1}}.$$

Example: random sparse SNF

We construct random sparse matrices simply from a map.

```
? randomsmat(nrows,ncols,nbpercol) =
{
  my(M,i);
  M = Map();
  for(j=1, ncols,
    for(c=1,nbpercol,
      i = random([1,nrows]);
      mapput(~M, [i,j], (-1)^random(2));
    );
  );
  tosmat(M)
};
```

Example: random sparse SNF

```
? nrows = 1600;
? ncols = 1605;
? dims = Vecsmall([0,nrows,ncols,0]);
? d1 = tosmat(matrix(0,nrows));
? d2 = randomsmat(nrows,ncols,5);
? d3 = tosmat(matrix(ncols,0));
? diffs = [d1,d2,d3];
? C = [dims,diffs];
? homology(C,0,2)
cpu time = 54,976 ms, real time = 55,748 ms.
% = [[11, []], [16, []]]
? densehomology(densecomplex(C),0,2)
cpu time = 3min, 42,757 ms, real time = 3min, 54,30
% = [[11, []], [16, []]]
```

Example: homology of a hyperbolic 5-orbifold

Here is the kind of example I wrote this code for.

```
? data = vector(7,i,smatimport(strprintf("D%d-19
  .sms",i-1)));
? dims = Vecsmall(concat([data[1][1]],[dat[2] |
  dat <- data]));
? diffs = [dat[3] | dat <- data];
? C = [dims,diffs];
? dims
% = Vecsmall([0, 589, 9041, 37860, 65608, 50680,
  14480, 0])
? homology(C,0,6)
cpu time = 365 ms, real time = 382 ms.
% = [[1, []], [0, [3]], [2, []], [2, [3]], [0, []],
  [1, []]]
```

More timings

On my laptop, with less than 10GB of memory.

total dim	max dim	# coeffs	time
$1.8 \cdot 10^5$	$6.5 \cdot 10^4$	$3.7 \cdot 10^5$	0.4s
$6.2 \cdot 10^5$	$2.3 \cdot 10^5$	$2.9 \cdot 10^6$	2s
$1.7 \cdot 10^6$	$6.4 \cdot 10^5$	$8.0 \cdot 10^6$	16s
$5.1 \cdot 10^6$	$1.9 \cdot 10^6$	$2.4 \cdot 10^7$	3min 21s
$8.8 \cdot 10^6$	$3.3 \cdot 10^6$	$4.1 \cdot 10^7$	30min 55s

Conclusion

- Please try this algorithm on your matrices and give me feedback!
- Caveats: only uses invertible coefficients, depends heavily on ordering of coordinates. Not a silver bullet!
- TODO: better matchings, homotopy inverses, other rings.

Thank you!