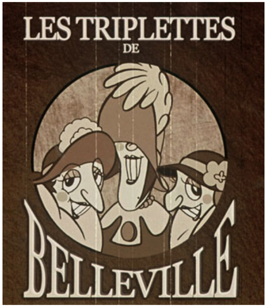
0. Annoated slides from Wednesday

CS319: Scientific Computing

Week 11: Sparse Matrices and templates

Dr Niall Madden

26 + 28 March, 2025



Slides and examples:
https://www.niallmadden.ie/2425-CS319

0. Outline

1. News and Updates

- Grades for Lab 4, Class Test, and Lab 6 will be posted soon (honest!).
- Presentations have been scheduled (switch to `https://www.niallmadden.ie/2425-CS319/2425-CS319-Projects.pdf`)

2. Sparse Matrices

Last week we designed a class for representing a matrix. Although we didn't discuss it at the time, the matrices represented are called "**dense**" or "**full**".

Today we want to see how to store **SPARSE MATRICES**: these are matrices that have so many zeros that it is worth our while exploiting the fact.

2. Sparse Matrices

There are numerous examples of sparse matrices. For example, they occur frequently when solving differential equations numerically.

But perhaps the most obvious example is when we use matrices to represents graphs and **networks**.

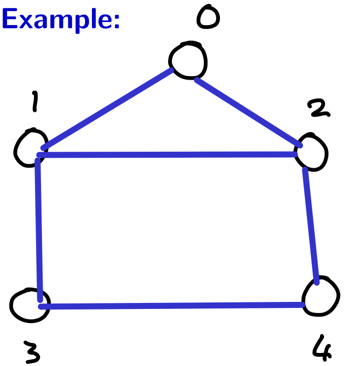
Most real world networks have far more nodes/vertices than they do connections/edges between those nodes.

In a computational setting, most graphs/networks are represented as a matrix, such as the **adjacency matrix**.

- If the graph has N vertices, the matrix, A, has N rows and columns: each corresponds to vertex.
- If (i, j) is an edge in the graph, then $a_{ij} = 1$. Otherwise, $a_{ij} = 0$.

2. Sparse Matrices

Example:



$$\begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 & 0 & 1 \\ 4 & 0 & 0 & 1 & 1 & 0 \end{array}$$

Note the matrix has 25 entries. which is quite a lot, given we've only 6 edges. This has NNZ: 12.

2. Sparse Matrices

Compared to the over-all number of entries in the matrix, the **number of non-zeros** (**NNZs**) is relatively small. So it does not make sense to store them all. Instead, one uses one of the following formats:

- **Triplet** (which we'll look at presently),
- **Compressed Sparse Row** storage (CSR) (after triplet)
- **Compressed Sparse Column** storage (CSC)

And the following formats for very specialised matrices, which we won't study in CS319:

- **Block Compressed Row/Column Storage**
- **Compressed Diagonal Storage**
- **Skyline**

What is a matrix sparse?

Although the representation and manipulation of sparse matrices is an major topic in Scientific Computing, there isn't a universally agreed definition of an (abstract) *sparse matrix*.

This is because, when coding, we should ask the question: **"When is it worth the effort to store a matrix in a sparse format, rather than in standard (dense) format?"**

The answer is often context-dependent. But roughly, use a sparse format when

- The memory required by the sparse format is less then the "dense" (or "full") one;
- The expense of updating the sparse format is not excessive;
- Computing a `MatVec` is faster for a sparse matrix.

2. Sparse Matrices

For **triplet** form we store a **sparse** matrix with NNZ non-zeros in three arrays:

- an unsigned integer arrays I[NNZ], with row indices
- an unsigned integer arrays J[NNZ],, with column indices.
- a double array X[NNZ].
- Then entry a_{ij} is stored as I[k]=i, J[k]=j, X[k]=a_{ij}, for some k.

This is also known as **Coordinate list (COO)**

NNZ = "number of non-zeros".

Example: write down the triplet form of the following matrix:

$$\begin{array}{c|cccc} 0 & 1 & 0 & 11 & 0 \\ 1 & 1 & 0 & 2 & 0 \\ 2 & 9 & 19 & 0 & 29 \\ 3 & 0 & 0 & 0 & 5 \\ \hline & 0 & 1 & 2 & 3 \end{array}$$

$NNZ = 8$

$$I = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \\ 3 \end{bmatrix}, \quad J = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 2 \\ 0 \\ 1 \\ 3 \\ 3 \end{bmatrix} \quad x = \begin{bmatrix} 1 \\ 11 \\ 1 \\ 2 \\ 9 \\ 19 \\ 29 \\ 5 \end{bmatrix}$$

In Compressed Row (see later) replace I with $R = [0, 2, 4, 7, 8]$ ↑ NNZ

:

2. Sparse Matrices

Example: Suppose A is a square $N \times N$ matrix, with $N = 1000$, three non-zero entries per row, and all data stored as `double`s.

(a) How many values need to be stored if the A is stored in the usual "dense" format? How much memory is required?

(b) How many values need to be stored if the A is stored in **triplet** format? How much memory is required?

We have $N^2 = 1{,}000{,}000$ values to store. Each requires 8 bytes. Total is 8,000,000 bytes, which is roughly 8 Mb.

NNZ is 3,000. So we store 9,000 values (3,000 for each of I, J, X). I, J store ints, which need 4 bytes each. They need 12,000 bytes. X needs 24,000. Total 36,00
≈ 36 Kb.

Our next goal is implement a triplet matrix as a `class`. The main tasks are:

- Decide what private data elements are needed.
- Decide what public methods are needed.
- Implement a matrix-vector multiplication algorithm.

Discussion...

DATA

Arrays for I, J, X.
N (size of matrix).
NNZ.

METHODS

- Same as matrix.
- Also:
  "where" – gives k value for i, j.
  Also get i, j, x for given k.

3. Coding triplet

Triplet.h

```
// Triplet.h:  For 2425-CS319 Week 11
// Author: Niall Madden

#ifndef _TRIPLET_H_INCLUDED
#define _TRIPLET_H_INCLUDED

#include "Vector10.h"
#include "Matrix11.h"
```

3. Coding triplet

Triplet.h

```
class Triplet {
  friend Triplet full2Triplet(Matrix &F, unsigned NNZ_MAX);
private:         int
  unsigned *I, *J;
  double *X;
  unsigned N;
  unsigned NNZ;    → "Current NNZ".
  unsigned NNZ_MAX;

public:
  Triplet (unsigned N, unsigned nnz_max); // Constructor
  Triplet (const Triplet &t); // Copy constructor
  ~Triplet(void);

  Triplet &operator=(const Triplet &B); // overload assignment
```

3. Coding triplet

Triplet.h

```
  unsigned size(void) {return (N);};
  int where(unsigned i, unsigned j); // negative return on error
  unsigned nnz(void) {return (NNZ);};
  unsigned nnz_max(void) {return (NNZ_MAX);};

  double getij (unsigned i, unsigned j);
  void setij (unsigned i, unsigned j, double x);

  unsigned getI (unsigned k) { return I[k];};
  unsigned getJ (unsigned k) { return J[k];};
  double getX (unsigned k) { return X[k];};

  Vector operator*(Vector u);
  void print(void);
};
#endif
```

$\rightarrow$ compute a Triplet-vector product.

3. Coding triplet

Triplet.cpp

```
// Triplet.cpp  for 2425-CS319 Week 11
//   What: Methods for the Triplet class
// Author: Niall Madden
#include <iostream>
#include <iomanip>
#include "Vector10.h"
#include "Matrix11.h"
#include "Triplet.h"
```

3. Coding triplet

Triplet.cpp (Constructor)

```
// Standard  constructor.
Triplet::Triplet (unsigned int N, unsigned nnz_max) {
  this->N = N;
  this->NNZ_MAX = nnz_max;
  this->NNZ = 0;

  X = new double [nnz_max];
  I = new unsigned [nnz_max];
  J = new unsigned [nnz_max];
  for (unsigned k=0; k<nnz_max; k++)  {
    I[k]=-1;
    J[k]=-1;
    X[k]=(double)NULL;
  }
}
```

When the Matrix is created it is empty.

Represent that by setting I[k] = J[k] = -1.

3. Coding triplet

When using a Triplet object to represent a matrix, T, we often need to find where in the array X, the value of $t_{i,j}$ is stored. That is done by the following function.

Triplet.cpp (where)

```
int Triplet::where(unsigned i, unsigned j)
{
  unsigned int k=0;
  do {
    if ( (I[k]==i) && (J[k]==j) )
      return(k);
    k++;
  } while (k<NNZ);
  return(-1);
}
```

[Finished here Friday]