Project: Sparse Matrix Vectorization

Pedram Pasandide

Due Date: 13 August

Download the rest of the files from Avenue. Read the file `MatrixMarket.pdf` and GSL to understand what a matrix market format is (saving a matrix that has many zeros and skipping saving zeros to lower the memory amount taken by matrix) and how we save it in Compressed Sparse Row (CSR) format. Check out the small example of a `.mtx` named `b1_ss.mtx`. Open this file and see the values. You should see this:

```
%%MatrixMarket matrix coordinate real general
%-------------------------------------------------------------
% UF Sparse Matrix Collection, Tim Davis
% http://www.cise.ufl.edu/research/sparse/matrices/Grund/b1_ss
% name: Grund/b1_ss
% [Unsymmetric Matrix b1_ss, F. Grund, Dec 1994.]
% id: 449
% date: 1997
% author: F. Grund
% ed: F. Grund
% fields: title A b name id date author ed kind
% kind: chemical process simulation problem
%--------------------------------------------------------------
7 7 15
5 1 -.03599942
6 1 -.0176371
7 1 -.007721779
1 2 1
2 2 -1
1 3 1
3 3 -1
1 4 1
4 4 -1
2 5 .45
5 5 1
3 6 .1
6 6 1
4 7 .45
7 7 1
```

Any lines starts with `%` is a comment. the first actual row, `7 7 15`, always shows the number of rows, columns and none-zero values. a `.mtx` file has always three columns. The first column is the row indexes, and the second column is column indexes, and the third column is values corresponding to its rows and columns. Don't forget this is still Matrix Format, and the purpose of `read_csr.c` is to to read it and save it in CSR format, and you call this function in `main.c`. The file `read_csr.c` reads the `.mtx` file using GSL library.

A. [**Total 10 points**] To read the data in CSR format, the GSL library has been used. Install GSL library:

1. Update the package list: Before installing new packages, it's a good idea to update the package list to ensure you get the latest version of GSL. Run the following command:

   ```
   sudo apt update
   ```

   ```
   sudo apt upgrade
   ```

2. Install GSL: Now, you can install the GSL library using the following command:

   ```
   sudo apt install libgsl-dev
   ```

3. to check if you have installed it use

   ```
   gsl-config --version
   ```

4. check where you have installed GSL you can use:

   ```
   pkg-config --variable=prefix gsl
   ```

You might need to set some environment variables to compile and link your code properly. GSL provides a tool called `gsl-config`, which can help you get the necessary flags for compiling and linking with the GSL library. If you want these flags to be set automatically every time you start a new terminal session, you can add these lines to your ` /.bashrc` file. Here's how you can do it using the `nano` editor:

1. Open ` /.bashrc` in the terminal using `nano`:

   ```
   nano  /.bashrc
   ```

2. Add the following lines at the end of the file:

   ```
   export CFLAGS=$(gsl-config --cflags)
   ```

   ```
   export LDFLAGS=$(gsl-config --libs)
   ```

3. Save the changes and exit nano by pressing Ctrl + X, then Y (to confirm saving), and finally Enter.

4. To apply the changes to the current terminal session, run:

   ```
   source  /.bashrc
   ```

Now, whenever you compile your C/C++ programs that use GSL, the necessary compiler and linker flags will be automatically included, making it easier to build GSL-based applications.

To compile a C/C++ code that includes the GSL library using the GCC compiler, you need to include the necessary compiler and linker flags to link the GSL library properly. As mentioned earlier, you can use the gsl-config tool to get the required flags.

Here's the general command to compile a C code with GSL library:

```
gcc -o main main.c $(gsl-config --cflags) $(gsl-config --libs)
```

Warning! If your any reason you have installed GSL library manually instead of **step 2** you must add:

```
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

Here `/usr/local/lib`: This is a directory path that is being added to the `LD_LIBRARY_PATH` variable. It indicates the location where additional shared libraries are stored. and to compile the code you need to manually give include `-lgsl -lgslcblas -lm` flags. So in this case you must compile your code with:

```
gcc -o main main.c -lgsl -lgslcblas -lm
```

Now you should be able to compile and run your code by including `#include <gsl/gsl_spmatrix.h>` library at the top of your `main.c`. Try this for the small example `b1_ss.mtx` and use GDB or LLDB to set some break points to make sure it works. To call the read function you need to add the following code in your `main.c`:

```
const char *filename = "af_shell10.mtx";
gsl_spmatrix *Acsr = ReadMMtoCSR(filename);
```

- The pointer to rows can be called by `ia[i]=Acsr->p[i]`, where i is the counter
- The column indexes can be accessed by `ja[i]=Acsr->i[i]`
- And the none zeros values, the third column in the file, can be accessed by `value[i]=Acsr->data[i]`

You don't need to change anything in `read_csr.c` file, you just need to use it. Print out the value of `Acsr` for `b1_ss.mtx` example, including row pointer, column index and value in and include them in your report `ReadMe.tex` file. I should be able to get some thing like this:

```
Row Pointer (ia): 0 3 5 7 9  <and the rest!>
Column Index (ja): 1 2 3 1 4 <and the rest!>
Values: 1.0000 1.0000 1.0000 -1.0000 0.4500 <and the rest!>
```

B. [**Total 20 points**] Make a matrix (name it b) with the same number of rows the sparse matrix you are reading, and one column. Initialize all values with 1 (`b[i]=1`) and compute $A \times b$ by calling a function with name `Spmv()`. The implementation of Spmv function must be in `Spmv.c`, and the function must be called in `main.c`. The same thing we did in the second assignment. This is very important that the `Acsr` matrix is in CSR format and you must use the row pointers and columns indexes to get the access to values. This is how Spmv is done:

```
for (int i = 0; i <numnber of rows>; i++)
{
 <your code>
 for (int j = ia[i]; j < ia[i + 1]; j++)
 {
  <your code>
 }
 <your code>
}
```

Call the `Spmv` function in `main.c` by the following format:

```
for (int i = 0; i < repeat; i++)
{
 <call the smpv>
}
```

This code does the same calculation (Spmv) for the `repeat` times. Define this value at the top of your code with `#define repeat 100`. I strongly suggest you start with the small matrix example. Use `#include <time.h>` library to evaluate the CPU time taken to do $res = A \times b$ for `repeat = 100` times. When you are sure that the results are correct, produce the following table for the other three matrices (`af_shell10.mtx` and `nlpkkt80.mtx` and `StocF-1465.mtx`):

Table 1: CPU time with repeat 100 times

Matrix	CPU time [second]
af_shell10	
nlpkkt80	
StocF-1465	

Use -03 flag to produce the same results for three matrices ... And include a short description of why with this flag the runtime is lower?

Table 2: CPU time with `-03` flag repeat 100 times

Matrix	CPU time [second]
af_shell10	
nlpkkt80	
StocF-1465	

Include a description why with this flag it works faster? In the next step produce the following table and mention why it is taking more time than the previous table.

Table 3: CPU time with `-03` flag repeat 500 times

Matrix	CPU time [second]
af_shell10	
nlpkkt80	
StocF-1465	

C. [**Total 15 points**] Currently all the $A \times b$ computation is done in double precision inside Spmv function. Make another function called `Spmv_float` and do all the procedure in `float` precision by type casting and produce the following results. Compare the CPU time here and CPU time in double precision. Which one is faster and why?

Table 4: CPU time with repeat 100 times, without `-03` flag, with `float` precision

Matrix	CPU time [second]
af_shell10	
nlpkkt80	
StocF-1465	

Submit On Avenue to Learn following files:

1. `main.c`
2. `utility.h` including the declaration of all function like `Spmv()` and `Spmv_float()` and others!
3. `Spmv.c`
4. `Spmv_float.h`
5. `Makefile` make sure you include this! I won't run you code this without makefile.
6. `ReadME.tex` including the description. Make sure this is a LaTeX format file. Descriptions in other formats will not be accepted.