Math 9830

Lab 1

Sean Ingimarson

1. Come up and write down the pseudo-code to compute the product of the transpose of the $n \times n$ sparse matrix A in CSR format with a vector x:

$$y = A^T x$$

Do not use the naive way by searching for all non-zero entries in column i. The number of operations performed in the algorithm shouild be $O(n)$ (assuming a constant number of entries by row).

We learned in class the pseudo-code for calculating $y = Ax$, which is given in algorithm 1.

```
for i = 0 : n − 1 do
    y[i] = 0;
    for idx = rowstart[i] : rowstart[i + 1] − 1 do
        y[i]+ = values[idx] · x[columns[idx]];
    end
end
```

Algorithm 1: Solving $y = Ax$ in CSR

We can think about this as running through the x vector starting off at the row vectors (where *idx* iterates through) and then indexing through the column vectors to find the x entry that multiplies the corresponding *values* entry.

Now to solve $y = A^T x$ while saving the exact same *rowstart*, *columns*, and *values* vectors, we would like to swap the location of the *columns* vector check for y and x. It's intuitive to think this way because the rows and columns are very literally being swapped. The pseudo-code is described in algorithm 2.

```
for i = 0 : n − 1 do
    y[i] = 0
end
for i = 0 : n − 1 do
    for idx = rowstart[i] : rowstart[i + 1] − 1 do
        y[columns[i]]+ = values[idx] · x[idx];
    end
end
```

Algorithm 2: Solving $y = A^T x$ in CSR

Note we also initialize the y vector as a row of zeros to start off with because we will call on other values of y throughout the new algorithm, so initialization is necessary. The output is visible in problem 3, figure 3.

2. Take a look at the `01_sparse_mat` source code from the class repo and implement your pseudocode in 1) in the function `mat_vec_transposed`. Submit your `main.cc` solution on Canvas.

The code verbatim is copied directly from Qt Creator on Linux:

```
void mat_vec_transposed(Vector &dest, const Vector &src)
{
    for (int i=0; i<n; ++i)
    {
        dest.values[i]=0.0;
    }
    for (int i=0; i<n; ++i)
      {
        for (int idx=row_start[i]; idx<row_start[i+1]; ++idx)

          dest.values[column_indices[idx]] += values[idx]*src.values[i];
      }
}
```

3. Take a look at the function `print_full` in the same program: notice that the output is incorrect (see the last intry in the second row). Try to fix this bug.

The code is copied directly from Qt Creator:

```
 void print_full()
{
  for (int r=0; r<n; ++r)
    {
      int idx = row_start[r];
      for (int c=0; c<n; ++c)
        {
          if (c > column_indices[idx])
            ++idx;
          if (c == column_indices[idx])
            std::cout << values[idx] << "\t";
          else
            std::cout << "-\t";
        }
      std::cout << std::endl;
    }
}
```

Upon inspection of the code and the output, this function outputs the following matrix:

$$\begin{bmatrix} 1.1 & - & - & -1.5 \\ 1.2 & 2 & - & 5.5 \\ - & - & - & 5.5 \\ - & - & 1.3 & - \end{bmatrix}$$

The issue here is that there is an extra 5.5 in the $(1,3)$ position. This tells us there is something wrong with the way the values are being called, in other words idx is not being iterated correctly. This should lead us right to the if statement where $c > col[idx]$ is being considered.

Coming from a theory perspective, we should notice that idx needs to stop iterating once it reaches it's iteration limit, which is `row_start[r+1]-1`. So we implement a check to make sure idx is not too large. We change the if statement to the following:

```
if (c > column_indices[idx] && idx < row_start[r+1]-1)
  ++idx;
```

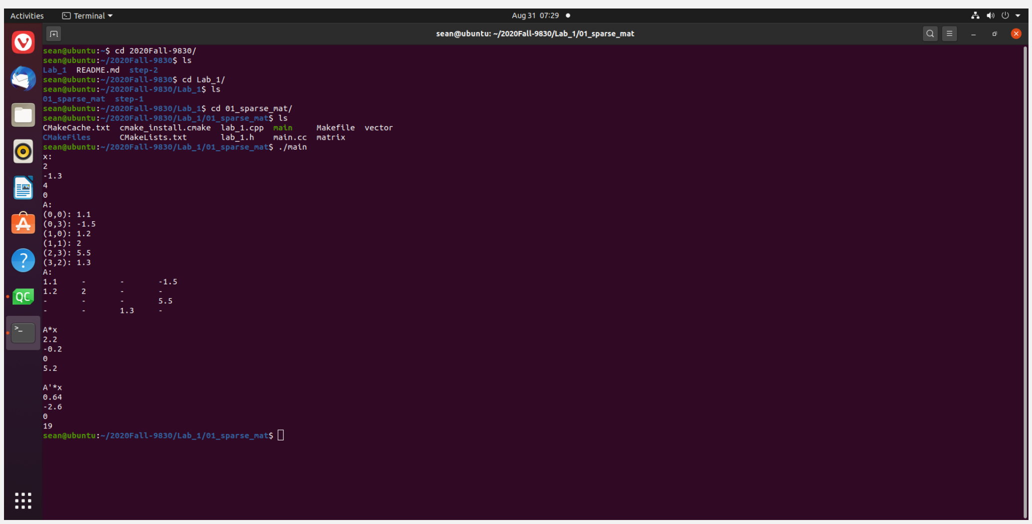

Figure 1: Screenshot of deal.II output

4. Install deal.II version 9.2.0 on your computer. See the lecture and Canvas for more information. Make sure you can run tutorial `step-1` from deal.II.

I've provided a screenshot of my deal.II output in figure 3, so that should be enough justification that I've installed it. Below I will provide the output from `step-1`:

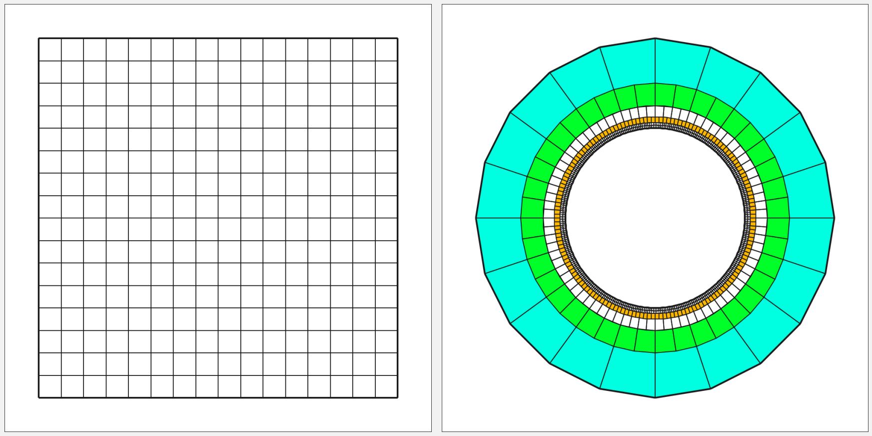

Figure 2: Figure 1 and 2 respectively from `step-1`