CES 703 Assignment #1 Due date: Monday, January 21, 9:45am

Dr. Imre Pólik (30+10 points)

Electronic submission is preferred. Do not print anything that's generated electronically. Whatever can be submitted electronically, please do it that way. Please submit the source code and the output along with the solutions.

For the programming problems you can use your favourite language (C, Fortran, Matlab, Python, Java, etc.) provided that it can handle the problem. If you choose a compiled language please provide instructions for compilation (possibly a Makefile).

---

1. (15 points) Consider the following two storage schemes for sparse matrices:

   Matlab: The storage consists of:

   - the number of rows and columns in the matrix: m and n;
   - an array c of length $n+1$, where c_i is the number of nonzeros in the first $i-1$ columns, (therefore $c_1 = 0$, c_{n+1} is the number of nonzeros in the whole matrix and there are $c_{i+1} - c_i$ nonzeros in the i^{th} column);
   - an array r of length c_{n+1} storing the row indices of the nonzero elements in each column;
   - an array v of length c_{n+1} storing the nonzero elements in each column.

   For example the matrix

   $$\begin{pmatrix} 0 & 1 & 2 \\ 5 & 0 & 0 \\ 2 & 8 & 9 \\ 0 & 0 & 1 \end{pmatrix}$$

   is stored as

   $$\begin{aligned} m &= 4, \\ n &= 3, \\ c &= (0, 2, 4, 7), \\ r &= (2, 3, 1, 3, 1, 3, 4), \\ v &= (5, 2, 1, 8, 2, 9, 1). \end{aligned}$$

   Jiaping: Almost the same as the previous one, except that instead of c and r we have a combined linear index $(j-1)m + i$ for the (i, j) element.
   The matrix in the previous example would be stored as:

   $$\begin{aligned} m &= 4, \\ n &= 3, \\ cr &= (2, 3, 5, 7, 9, 11, 12), \\ v &= (5, 2, 1, 8, 2, 9, 1). \end{aligned}$$

Choose one of the two storage schemes. How much work does it take to compute the representation of the transpose of a sparse matrix stored that way? How does it depend on the size of the matrix and the number of nonzeros? Implement the algorithm and verify its complexity. Confirm your findings with some test runs. Also test how long it takes for Matlab to transpose a sparse matrix as a function of matrix size and number of nonzeros.

2. (10 points) Investigate the behaviour of your computer's cache architecture by analyzing the performance of different BLAS calls for vectors and matrices of increasing size. Plot the running time against the vector length. Can you notice some odd behaviour? Identify the maximum dimension that fits in your cache. Once you have the results for doubles, change the type to single and repeat the experiment. A standard double number takes up 8 bytes, while a single is only 4 bytes.

   To ensure that the timing is precise and reproducible disable dynamic processor speed changes, enabled by default on most laptops. Since we are dealing with the cache here, make sure no other program is using the CPU intensively.

   (Hint: You can try to use Matlab, but be careful. It won't always work for this experiment. A compiled language is preferred here. (Why?) If you use Matlab stick to a simple $x^T x$ or $x^T y$ call and don't do multiple runs.)

3. (5 points) The Hilbert matrix is notoriously ill-conditioned. Remember that the (i, j) element of a Hilbert matrix is $\frac{1}{i+j-1}$. Find an explicit formula for the inverse of a Hilbert matrix. Assume that instead of solving a system $Hx = b$ where H is a Hilbert matrix we use the inverse to express $x = H^{-1}b$. Present an example that slightly perturbing b changes x a lot and thus show that the system cannot be solved accurately by any algorithm. Explain why.

Bonus: (10 points) Solve the first question for both storage schemes. Demonstrate the cache size with the sparse transposition problem.