ACM 11: Homework 1

Assigned Monday, Sept 29 2008. Due at beginning of class on Monday, October 6. 50 points.

1. Go to `http://www.acm.caltech.edu/~acm11/2008/FALL/hw1data.html` and, using copy-and-paste, save the data as a MATLAB vector. The data is the encrypted version of a short message, and the code used was a simple cyclical rotation by a fixed amount k (e.g. if $k = 2$, then "a" becomes "c", "b" becomes "d", ...) of the ASCII characters (i.e. those characters represented by integers 0 through 127).

   Using MATLAB, decode the message and determine which value of k was used. Hint: the original message contains the word "work". Use the command `mod` to make everything cyclical. You will need to make the data vector into a row-vector if you wish to perform string operators. Turn in your code. 15 points.

2. Vectorization. Let a and b be two vectors of length 10^7 with independent entries, all from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$. To generate these in MATLAB, use `randn`, but first "seed" the random number generator with the seed "2008", so that your calculations are reproducible. When you seed the random number generator, make sure to use the Ziggurat algorithm.

   Write code that computes the dot product between a and b using a `for` loop, and record the time it takes the code to run (use `tic` and `toc` for timing).

   Then, use MATLAB's builtin dot product (using either `dot`, or, for faster code, use just `a'*b`), and record the time. How much slower was your code than MATLAB's, or was it faster? Turn in your code and include the value of the dot product. 10 points.

3. Given a square matrix A, write the MATLAB commands that would form a diagonal matrix B that has the same diagonal as A. 5 points.

4. Linear systems. MATLAB has a powerful linear system solver, and you can call it with either the backslash command (`A\b`) or the function `linsolve`.

   Let A be a Hilbert matrix of size 11 x 11. The Hilbert matrix is a notoriously ill-conditioned matrix, often used for test examples. Let x be an 11-dimensional vector of all ones, and let $b = Ax$. If x were unknown, it is possible to solve for it using A and b, since A has full rank. For example, $\hat{x} = A^{-1}b$. However, because A is ill-conditioned, the answer is hard to compute numerically, and using A^{-1} is not the best method.

   (a) Solve for $\hat{x}$ using the inverse of A as returned by the `inv` command, and report both the absolute error in x (i.e. $\|x - \hat{x}\|_2$) and the size of the residual (i.e. $\|b - A\hat{x}\|_2$; use `norm` for $\|\cdot\|$).

   (b) Repeat the above, but use the command `invhilb` to compute the inverse of A.

   (c) Repeat, but use MATLAB's linear system solver.

   To generate A, use the command `hilb`, and to generate x, use `ones`. 10 points.

5. Linear systems again. Now, we'll compare the speed of using `inv` to solve a system of linear equations, compared to the builtin linear system solver. Let A be a Hadamard matrix, which is a special kind of matrix with $+1$ and -1 entries, and is very well-conditioned. Let A have dimensions 2048 x 2048, and let x be a 2048-dimensional vector of all ones. Generate $b = Ax$, and solve for $\hat{x} = A^{-1}b$ using (a) the `inv` operator, and (b) MATLAB's builtin linear system solver. For each method, report the absolute error in x, the residual, and the time taken for the command to run. To generate A, use the command `hadamard`. 10 points.