The goal of this week's assignment is to get you into multiple dimensions, and to use the "compress" operator that was introduced in class, by extending the new fast Gauss transform that was developed in class and in the previous assignment to two dimensions.

1 Problem (Homework 3)

Compute the matrix-vector product

$$\mathbf{v} = \mathbf{\Phi u}, \tag{1}$$

or

$$v_j = \sum_{i=1}^{N} \Phi_{ji} u_i, \quad j = 1, ..., M, \tag{2}$$

with absolute error $\epsilon < 10^{-5}$.Here

$$\mathbf{\Phi} = \begin{pmatrix} \Phi_{11} & \Phi_{12} & ... & \Phi_{1N} \\ \Phi_{21} & \Phi_{22} & ... & \Phi_{2N} \\ ... & ... & ... & ... \\ \Phi_{M1} & \Phi_{M2} & ... & \Phi_{MN} \end{pmatrix}, \quad \mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ ... \\ u_N \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ ... \\ v_M \end{pmatrix}, \tag{3}$$

$$\Phi_{ji} = e^{-|\mathbf{y}_j - \mathbf{x}_i|^2}, \quad i = 1, ..., N, \quad j = 1, ..., M.$$

and $\mathbf{x}_1, ..., \mathbf{x}_N$, $\mathbf{y}_1, ..., \mathbf{y}_M$ are random points in two dimensions that are uniformly distributed inside a unit square with center that has coordinates $\mathbf{x}_* = (5, 5)$. $u_1, ..., u_N$, are random numbers uniformly distributed on $[0, 1]$. The matrix size, $N, M > 0$ are given (fixed) positive integers.

1. Using examples of Lecture #3 and Lecture #5 and the residual term evaluation for Taylor series evaluate the truncation number, p, for the infinite series to achieve the required accuracy and provide a formula that can be used for the "Fast" ($O(N + M)$) method at $N \leqslant 10^3$.

2. Write a program that implements both straightforward computation based on Eq. (2) and the "Fast" method.

3. Provide a graph of the absolute maximum error between the straightforward and "Fast" method for $N = 10^3$ and $M = 2N$ and p varying between 1 and 12. Compare the results with your evaluations of the accuracy.

4. Provide a graph that compares the CPU time required by the straightforward and the "Fast" method for N varying between 10^2 and 10^3 for straightforward and N varying between 10^2 and 10^4 for the "Fast" method. Take $M = 2N$ and the theoretical value of the truncation number that ensures that the required accuracy is achieved.

5. Find the "breakeven" point (i.e. N at which the "Fast" method requires the same CPU time as the straightforward method).

6. Provide a graph of actual error (between the standard and the fast methods) for N varying between 10^2 and 10^3, $M = 2N$ and the truncation numbers used.

Hints

1. The following property of the scalar product is useful for evaluations

$$|(\mathbf{x} - \mathbf{x}_*) \cdot (\mathbf{y} - \mathbf{x}_*)| \leqslant |\mathbf{x} - \mathbf{x}_*| \, |\mathbf{y} - \mathbf{x}_*| \, . \tag{4}$$

   Use the "compression" operator defined in class to reduce the length of function expansion.

2. Use Matlab. Implement the compression operator. Matlab function $nchoosek(n, m)$ returns the binomial coefficients $\left(\begin{array}{c} n \\ m \end{array} \right)$.

3. You may keep the truncation number constant (evaluated for $N \leqslant 10^4$) or vary it with N according the theory. In this case a small program for $p(\epsilon, N)$ will be helpful.

4. You may find this point approximately from the graph of CPU time (as intersection of two lines approximating the CPU time of each method). This point of course depends on your implementation. If it appears that this point is not within the range of your computed data, evaluate it assuming that the straightforward method has complexity $O(N^2)$, while the "Fast" method is $O(N)$ method.

5. To speed up your implementation you should follow good programming style and *vectorize* your program. You may want to look at
`http://www.mathworks.com/access/helpdesk/help/techdoc/matlab_prog/ch7_per3.shtml`
or the news group `comp.soft-sys.matlab` (accessed conveniently via `http://groups.google.com`) or find your own resource on the web. If you find a particularly good site on Matlab code acceleration, let the class know by sending a message to the class mailing list.