Homework set 5 — APPM4720/5720, Spring 2016

Problem 1: The objective of this problem is to computationally investigate the error incurred by truncating multipole expansions. Consider the following geometry: Let Ω_τ and Ω_σ be two well-separated boxes with centers $\mathbf{c}_\tau$ and $\mathbf{c}_\sigma$. Let $\mathbf{x} \in \Omega_\sigma$ be a source point and let $\mathbf{y} \in \Omega_\tau$ be a target point. Consider the error function

$$e(P) = \sup\Big\{\Big|\log|\mathbf{x}-\mathbf{y}| - \mathbf{B}_P(\mathbf{x},\,\mathbf{c}_\tau)\,\mathbf{Z}_P(\mathbf{c}_\tau,\,\mathbf{c}_\sigma)\,\mathbf{C}_P(\mathbf{c}_\sigma,\,\mathbf{y})\Big|\,:\,\mathbf{x}\in\Omega_\sigma\;\mathbf{y}\in\Omega_\tau\Big\}$$

where P is the length of the multipole expansion, and where

$\mathbf{C}_P(\mathbf{c}_\sigma,\,\mathbf{y}) \in \mathbb{C}^{(P+1)\times 1}$	maps a source to an outgoing expansion
$\mathbf{Z}_P(\mathbf{c}_\tau,\,\mathbf{c}_\sigma) \in \mathbb{C}^{(P+1)\times(P+1)}$	maps an outgoing expansion to an incoming expansion
$\mathbf{B}_P(\mathbf{x},\,\mathbf{c}_\tau) \in \mathbb{C}^{1\times(P+1)}$	maps an incoming expansion to a target

(a) Estimate $e(P)$ experimentally for the geometry:

$$\Omega_\sigma = [-1,1]\times[-1,1], \qquad \Omega_\tau = [3,5]\times[-1,1].$$

(b) Fit the function you determined in (a) to a curve $e(P) \sim c\cdot\alpha^P$. What is α?

(c) Is the supremum for a given P attained for any specific pair $\{\mathbf{x},\,\mathbf{y}\}$?
If so, find (experimentally) the pair. Does the choice depend on P?

(d) Repeat questions (a), (b), (c) for a different geometry of your choice. (Provide a picture.)

(e) (Optional) Can you support your observations with analysis?

Hint: The file `main_T_ops_are_fun.m` (provided inside `tutorialfmm.zip`) might be useful.

Problem 2: The objective of this exercise is to familiarize yourself with the provided prototype FMM. The questions below refer to the basic FMM provided in the file `main_fmm.m` when executed on a uniform particle distribution. For this case, precompute only the translation operators $\mathbf{T}^{(\mathrm{ofo})}$, $\mathbf{T}^{(\mathrm{ifo})}$, and $\mathbf{T}^{(\mathrm{ifi})}$ (i.e. set `flag_precomp=0`).

(a) Estimate and plot the execution time of the FMM for the choices

$$N_{\mathrm{tot}} = 1\,000,\ 2\,000,\ 4\,000,\ 8\,000,\ 16\,000,\ 32\,000,\ 64\,000,\ 128\,000,\ 256\,000.$$

(If you have difficulties running the larger problems, then you can skip them.) Set `nmax=50`. Provide plots that track the following costs:

t_{tot}	total execution time, including initialization.
t_{init}	cost of initialization (computing the tree, the object `T_OPS`, etc.).
t_{ofs}	cost of applying $\mathbf{T}^{(\mathrm{ofs})}$.
t_{ofo}	cost of applying $\mathbf{T}^{(\mathrm{ofo})}$.
t_{ifo}	cost of applying $\mathbf{T}^{(\mathrm{ifo})}$.
t_{ifi}	cost of applying $\mathbf{T}^{(\mathrm{ifi})}$.
t_{tfi}	cost of applying $\mathbf{T}^{(\mathrm{tfi})}$.
t_{close}	cost of directly evaluating close range interactions.

(b) Repeat exercise (a) but now for a few different choices of `nmax`. Which `nmax` leads to the smallest total execution time t_{tot}? Provide a new plot of the times required for this optimal choice.

Problem 3: In this exercise, you will numerically estimate the probability density function for the errors produced by two randomized algorithms, which are both implemented in the script `HW05.m`. The two algorithms are:

(i) A Monte Carlo algorithm for estimating the value of π. The idea is to draw n points from a uniform distribution on $\Omega = [-1,1]^2$. Let n_{in} denote the number of points that fall inside the circle of radius one centered at the origin. Since the area of this circle is π, and the area of the box is 4, we expect that the ratio n_{in}/n should approach $\pi/4$ as n increases. The error is $e = 4\,n_{\text{in}}/n - \pi$.

(ii) The Randomized SVD describe in Figure 2.1 in the course notes. The objective is to compute a a rank-k approximate SVD. The error is $e = \|\mathbf{A} - \mathbf{UDV}^*\|$.

(a) Consider the Monte Carlo algorithm for $n = 1\,000$. Write a script that executes the algorithm a m times, to produce a vector $\{e_j\}_{j=1}^m$ of measured errors. After you are done, use your vector of measured errors create an approximate plot of the probability density function of the error. The Matlab command `histogram` might be useful.

(b) Now estimate the variance of the error as a function of n. Consider the values $n = 10,\ 10^2,\ 10^3,\ 10^4,\ 10^5$. For each value of n, run the MC algorithm m times, to get a vector $\{e_j\}_{j=1}^m$ of errors. The estimate of the variance is then $s = s(n) = (1/(m-1))\sum_{j=1}^m e_j^2$. Create a plot that investigates whether $\sqrt{s}$ scales as $1/\sqrt{n}$, as claimed.

(c) Repeat exercise (a), but now do it the RSVD algorithm. Do not use the power method here, so $q = 0$. Consider the choices of parameter $p = 0$, $p = 10$, $p = 20$, $p = 30$. For each value of p, create a plot of the estimated probability density function for the error. Hand in plots of the four probability density functions.

(d) Repeat exercise (c), but now use the power method with $q = 2$. Hand in plots of the four probability density functions.

In each exercise, choose m to be as large as you have patience for! The larger m is, the better your estimate will be, but of course the execution time will take longer. As always when coding, save any big runs for after your code works well. In other words, while developing, use moderate values of m.

Note: The distribution of the error in the Monte Carlo experiments is known analytically. Observe that each "draw" of a point can be interpreted as a coin toss with a biased coin (probability of, say, heads being $\pi/4$ and the probability of tail $1-\pi/4$). The number of heads out of n tosses follows the so called "binomial distribution." You can look up the variance for this distribution in any standard reference on probability (or Wikipedia). It might be fun to plot the exact pdf along with your estimate of the pdf.