Assignment 9 — 2.086 Fall 2015

Due: *Friday, December 4 at 5 PM.*

Upload your solution to Stellar as a zip file "`YOURNAME_ASSIGNMENT_9.zip`" which includes the script for each question in the proper format *as well as* any MATLAB functions (of your own creation) your scripts may call. Remember to verify your upload before the assignment due time by downloading your solution folder yourself and re-running `grade_o_matic`. Full instructions for assignment preparation are given in the document "Preparing assignments for `grade_o_matic`" available on the 2.086 website.
In this assignment, the student mode of `grade_o_matic` will only provide partial credit for questions 1 and 5. No partial credit will be provided for the rest of the questions. This means that `grade_o_matic` will **not** check the MATLAB syntax of your scripts for questions 2 and 6, hence you are responsible for ensuring that your scripts have no syntax errors and that they perform correctly for any set of input data. *In fact, for all questions you should yourself devise several test cases for which you can anticipate the correct answers and hence test your script.*
For question 3, please submit your answer in pdf format. Your written responses may be typed in your favorite software, but scans of handwritten material will not be accepted.

Questions

1. (25 points) We would like you to write a function which computes a Monte Carlo estimate $(\hat{A}_D)_n$, and associated 95% confidence-level (normal-approximation two-sided) confidence interval $[\mathrm{ci}_{A_D}]_n$, for the area of the region

$$D = \{x_1^2 + x_2^2 \leq 0.75\} \cup \{(x_1 - \alpha)^2 + x_2^2 \leq 0.75\} \; . \tag{1}$$

Note that D is the union of two circles in which the center of the second circle is shifted by α (in x_1) from the center of the first circle. We choose for our background rectangle R the square $R = [-c, c]^2$ for c sufficiently large to enclose D.

Your function should have signature

```
function [area_est,area_conf_int] = MC_area(alpha,c,n,x1pts,x2pts)
```

with five function inputs and two function outputs. The first function input is a scalar `alpha`, which corresponds to the center shift of the second circle α; the allowable instances, or parameter domain, is $0 \leq \alpha \leq 0.1$. The second function input is a scalar `c` which corresponds to c that defines the bounding square R: the lower left corner of R is $(x_1, x_2) =$ (`-c,-c`) and the upper right corner of R is $(x_1, x_2) =$ (`c,c`). The allowable instances are $1 \leq$ `c` ≤ 10. (Note that for these instances of `c` the domain D is indeed included in R for all allowable instances of `alpha`.) The third function input is a scalar integer `n`, which corresponds to the number of random darts; the allowable instances, or parameter domain, is `n` $\in \{10, 11, ..., 10^6\}$. The fourth and fifth function inputs are the `n`$\times 1$ vectors `x1pts` and `x2pts`, which correspond to the coordinates of the `n` random darts, (`x1pts(i),x2pts(i)`), `i` $\in \{1, 2, \ldots$ `n`$\}$, thrown at the square

R — on the basis of which you will calculate your area estimate and associated confidence interval: we emphasize that `x1pts` and `x2pts` correspond to arrays of i.i.d. pseudo-random variates from the (univariate) continuous uniform distribution over the interval $(-c, c)$.

The first function output is a scalar `area_est` which corresponds to the Monte-Carlo estimate for the area, $(\hat{A}_D)_n$. The second function output is the 1×2 row vector `area_conf_int` which corresponds to the confidence interval $[\mathrm{ci}_{A_D}]_n$ such that the entries `area_conf_int(1)` and `area_conf_int(2)` are respectively the lower and upper limits of the confidence interval $[\mathrm{ci}_{A_D}]_n$.

Three further points: first, in the event that the normal-approximation criterion[1], $n\theta > \phi$ **and** $n(1-\theta) > \phi$ with $\phi = 10$, for the construction of the confidence interval is not satisfied *your code should return* `[-1,1]` for `area_conf_int`. Second, you should set $z_\gamma = 1.96$ which corresponds to confidence level $\gamma = 95\%$. Third, in order to test your function `MC_area` for any desired `alpha`, `c` and `n`, you must first generate the random darts `x1pts` and `x2pts` *outside* your function. Note, however, that for purposes of grading, `grade_o_matic` will automatically generate appropriate inputs `x1pts,x2pts` for your function — no action needed on your part.

The script template for this question is provided in `A9Q1_Template`; no modifications are required (except to remove the `_Template` from the name). The function template is provided in `MC_area_Template.m`; note for function templates (as well as script templates) you should first save as with `_Template` removed from the name.

2. (20 points) We would like you to write a function which computes, based on the sample-mean method, a Monte Carlo estimate $\hat{I}_n$ for the integral

$$I = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} |\vec{x} - \vec{y}| \exp\left(-\frac{x^2}{2}\right) \exp\left(-\frac{y^2}{2}\right) d\vec{x} d\vec{y} \tag{2}$$

Here $\vec{x}$ denotes the vector (x_1, x_2, x_3) with magnitude $x = \sqrt{x_1^2 + x_2^2 + x_3^2}$; similar definitions hold for $\vec{y}$. It follows that $d\vec{x}$ is the differential element $dx_1 dx_2 dx_3$; similarly for $d\vec{y}$. As a result, the domain of integration is six-dimensional.

The domain of integration in equation (2) is infinite. To avoid the difficulty associated with discretizing an infinite domain, we will truncate it to the domain $-5 \leq x_1, x_2, x_3, y_1, y_2, y_3 \leq 5$, since the exponential function in the integrand makes the contribution of points beyond this truncated domain negligible. In other words, we will approximate (2) by

$$I \approx \int_{-5}^{5} \int_{-5}^{5} \int_{-5}^{5} \int_{-5}^{5} \int_{-5}^{5} \int_{-5}^{5} |\vec{x} - \vec{y}| \exp\left(-\frac{x^2}{2}\right) \exp\left(-\frac{y^2}{2}\right) dx_1 dx_2 dx_3 dy_1 dy_2 dy_3 \tag{3}$$

Write a function with signature

```
function [int_est] = sample_mean_integral(n,upts)
```

which takes two function inputs and yields one function output. The first function input is the scalar integer `n`, which is related to the size of the random sample; the allowable instances,

[1] In general, the larger the value of ϕ the better the approximation becomes; $\phi \approx 5$ ensures errors of a few percent, while the value chosen here, $\phi = 10$, ensures errors on the order of 1% or less

or parameter domain, is `n` $\in \{10, 11, ..., 10^5\}$. The second function input is the `n`×6 array `upts`, which corresponds to i.i.d. pseudo-random variates from the (univariate) continuous uniform distribution over the interval $(0, 1)$. Every column of `upts` contains `n` pseudo-random variates; columns $1 - 6$ in `upts` will be related to x_1, x_2, x_3, y_1, y_2 and y_3, respectively, after the transformations required for the integral estimation. The function yields a single function output: the scalar `int_est`, which is the sample-mean estimate for I, $\hat{I}_n$, based on (3).

One further point: in order to test your function `sample_mean_integral` you must first generate the random sample `upts` **outside** your function. Note, however, that for purposes of grading, `grade_o_matic` will automatically generate appropriate inputs `upts` for your function — no action needed on your part. **Attention**: `upts` is drawn from the continuous uniform distribution over the interval $(0, 1)$; inside your function you must transform these pseudo-random variates to the correct interval as required in the integral (3).

The script template for this question is provided in `A9Q2_Template`; no modifications are required (except to remove the `_Template` from the name). The function template is provided in `sample_mean_integral_Template.m`.

3. (15 points) As you very well know, your estimate of the integral in question 2, `int_est`, is itself a random variable since it is a sum of random variables. Using your function from question 2 generate $m = 5000$ realizations of this random variable and study its distribution for $n = 10000$ and $n = 50000$. If your computer has multiple cores you may want to take advantage of the command `parfor` to reduce the computational cost of this exercise.

   Your solution will not be graded by `grade_o_matic`; as a result, there will be no student instances or partial credit. Please submit your answer in a pdf document named A9Q3.pdf; this file should include two histograms of the distribution of the variable `int_est` (one for each of the cases $n = 10000$ and $n = 50000$), along with a discussion of your study of these two distributions (e.g. What is the observed distribution of `int_est`? Is it what you expected from theory? How does the standard deviation of `int_est` scale with n? Is this what you expected?).

4. (15 points) Consider the following MATLAB script

```
% begin script

clear
% note we "clear" the workspace

n = 10000; % number of random darts

u1 = rand(1,n);
u2 = rand(1,n);

x1 = u1;
x2 = 2.*u2;

addfac = 1.0;

% BEGIN BLOCK
```

```
numinside = 0;
for i = 1:n
   if( x2(i) <= my_func(x1(i),addfac) )
       numinside = numinside + 1;
   end
end
% END BLOCK

area_estimate = (numinside/n)*2.0;

% end script
```

and function `my_func`

```
function [value] = my_func(x,yplus)
value = x.^3 + yplus;
return
end
```

for approximating an area A_D of a domain D by the Monte Carlo method. In the limit that `n` tends to infinity, `area_estimate` will approach A_D.

(i) (5 points) Each of the bivariate uniform random variates (realizations of random variables from the bivariate uniform density) `(x1(i),x2(i))`, $i = 1, \ldots,$ `n`, will reside in the rectangle:

(a) $-1 <$ `x1(i)` < 1, $-1 <$ `x2(i)` < 1
(b) $-1 <$ `x1(i)` < 1, $\quad 0 <$ `x2(i)` < 1
(c) $\quad 0 <$ `x1(i)` < 1, $-1 <$ `x2(i)` < 1
(d) $\quad 0 <$ `x1(i)` < 1, $\quad 0 <$ `x2(i)` < 2

(ii) (5 points) The area A_D is given by:

(a) $2/3$
(b) $4/3$
(c) 2
(d) $5/4$

Hint: draw a sketch in which you indicate the rectangle over which `x1,x2` is defined (i.e., at which you throw your darts); then include in your sketch the domain D defined by the `if` condition in BLOCK; finally, recall the area interpretation of the definite integral to determine A_D.

(iii) (5 points) Which single line of MATLAB code below is equivalent to (and also much more efficient than) the entire BLOCK of code in the script above:

(a) `numinside = sum( find( x2 <= my_func(x1,addfac) ) );`
(b) `numinside = sum( x2 <= my_func(x1,addfac) );`

(*c*) `numinside = length( x2 <= my_func(x1,addfac) );`

(*d*) `numinside = length( my_func(x1,addfac) );`

Note by the entire BLOCK of code we refer to the lines of code between the comments `BEGIN BLOCK` and `END BLOCK`.

We provide you with the template `A9Q4_Template.m` file that contains the multiple-choice format required by `grade_o_matic`.

5. (10 points) In Assignment 8, question 4, we asked you to create a function `positive_normal` which generates pseudo-random variates from a rectifed normal distribution with mean `mu` and standard deviation `sigma`.

   In this question, we ask you to modify `positive_normal` from Assignment 8 so that it generates random variates from a "rectified and bounded normal" distribution for which the probability of observing values below 0 and above 100 is zero. In other words, create a new function with signature:

   ```
   function [xpts_r_and_b] = r_and_b_normal(mu,sigma,n)
   ```

   which provides a row vector of `n` rectified and bounded normal random variates (which take values between 0 and 100, as explained above) originating from a normal population with mean `mu` and standard deviation `sigma` (note that the mean and standard deviation are defined before the bounding process).

   The function takes three inputs. The first two inputs, respectively `mu` and `sigma`, are scalars; the allowable instances, or parameter domain, is $0 \leq$`mu`≤ 100 and $1 \leq$`sigma`≤ 30. The input `n` is a scalar integer; the allowable instances, or parameter domain, is `n` $\in \{10, 11, ..., 10^5\}$. The function yields a single function output: the output is the $1\times$`n` row vector `xpts_r_and_b`, which is our pseudo-random sample realization. Note that `xpts_r_and_b` must be random not just in terms of the outcome frequencies but also in terms of the order - such that (say) the first `n`$/2$ elements of `xpts_r_and_b`, or the last `n`$/2$ elements of `xpts_r_and_b`, or the "middle" `n`$/2$ elements of `xpts_r_and_b`, should also yield (approximately) the correct outcome frequencies.

   The script for this question is provided in `A9Q5_Template`; no modifications are required (except to remove the `_Template` from the name). The function template is provided in `r_and_b_normal_Template.m`.

6. (15 points)

   A 2.086 student hears in lab that Monte Carlo methods can be used for a lot more than just evaluating integrals. Having been preoccupied with their grade in 2.086, they decide to use Monte Carlo *simulation* to estimate the probability of getting an "A" in class. Their plan is as follows: they know that they have received already X credit based on 7 assignments and the midterm. Their final score will thus be

   $$S = X + w_{A8}A8 + w_{A9}A9 + w_{IE}IE + w_{FE}FE \tag{4}$$

   where $A8$, $A9$, IE and FE refer to the (yet unknown) scores in Assignment 8, Assignment 9, Instructor Evaluation and the Final Exam, and w_{A8}, w_{A9}, w_{IE} and w_{FE} to the corresponding weights, respectively. Note that here scores, including X, will be reported on the scale $[0, 100]$.

Given reliable stochastic data for these last 4 variables, one can generate stochastic data for their score in the class, S, which can be used to estimate the probability of getting any grade.

In order to see how well this idea works, we ask you to create a function with signature

```
function [probs] = grade_distribution(r_and_b_grades,cutoffs,w,X)
```

which will return the 5×1 vector `probs` containing the probabilities of the student obtaining the grades A, B, C, D and F (in that order) expressed as a fraction of unity (i.e. 0.95 for a probability of 95%). The inputs to this function are:

- The 4×`n` array of bounded and rectified normal random variates `r_and_b_grades`. Each row of this array contains `n` random variates corresponding to `n` realizations of the student performance in Assignment 8, Assignment 9, Instructor Evaluation and the Final Exam, respectively.
- The 4×1 vector `cutoffs` containing the *lower* cutoff for each of the grades A, B, C and D (in that order) expressed on the scale $[0, 100]$.
- The 4×1 vector `w` containing the weights w_{A8}, w_{A9}, w_{IE} and w_{FE} (in that order) expressed as a fraction of unity (i.e. 0.05 for a weight of 5%).
- The scalar `X`, which is the credit received from already graded assignments and the midterm exam, expressed as a score (on the scale $[0, 100 * (1 - w_{A8} - w_{A9} - w_{IE} - w_{FE})]$).

The deliverable for this question is the function described above. Please note that in this question, `grade_o_matic` will not provide any partial credit. Therefore, as always, you should test your function using your own data and instances. In fact, we have asked you to develop the function `r_and_b_normal` in question 5 for this specific reason.

The script template for this question is provided in `A9Q6_Template`; no modifications are required (except to remove the `_Template` from the name). The function template is provided in `grade_distribution_Template.m`.