Assignment 2 — 2.086/2.090 Spring 2013

Released: *Friday, 22 February, at 5 PM.*

Due: *Friday, 15 March, at 5 PM.*

Upload your solution as a zip file "`YOURNAME_ASSIGNMENT_2`" which includes the script for each question *as well as* all MATLAB functions (of your own creation) called by your scripts; both scripts and functions must conform to the formats described in **Instructions** and **Questions** below. You should also include in your folder all the `grade_o_matic .p` files for Assignment 2.

Instructions

Before embarking on this assignment you should

(1) Complete the Textbook reading for Unit II

(2) Execute ("cell-by-cell") two MATLAB Tutorials

a tutorial on MATLAB Functions (Rec4) ;
a tutorial on MATLAB Double Index Arrays (Rec 5).

(3) Download the `Assignment_2_Materials` folder. This folder contains a template for the script associated with each question (`A2Qy_Template` for Question `y`), as well as a template for each function which we ask you to create (`func_Template` for a function `func`). The `Assignment_2_Materials` folder also contains the `grade_o_matic` codes needed for Assignment 2. (Please see Assignment 1 for a description of `grade_o_matic`.)

We indicate here several general format and performance requirements:

(a.) Your script for Question `y` of Assignment `x` *must* be a proper MATLAB ".m" script file and *must* be named `AxQy.m`. In some cases the script will be trivial and you may submit the template "as is" — just remove the `_Template` — in your "`YOURNAME_ASSIGNMENT_2` folder. But note that you still must submit a proper `AxQy.m` script or `grade_o_matic_A2` will not perform correctly.

(b.) In this assignment, for each question `y`, we will specify inputs and outputs both for the script `A2Qy` and (as is more traditional) any requested MATLAB functions; we shall denote the former as script inputs and script outputs and the latter as function inputs and function outputs. For each question and hence each script, and also each function, we will identify *allowable instances* for the inputs — the parameter values or "parameter domains" for which the codes must work.

(c.) Recall that for scripts, input variables must be assigned *outside* your script (of course before the script is executed) — *not* inside your script — in the workspace; all other variables required by the script must be defined *inside* the script. Hence you should test your scripts in the following fashion: `clear` the workspace; assign the input variables in the workspace; run your script. Note for MATLAB functions you need not take such precautions: all inputs and outputs are passed through the input and output argument lists; a function enjoys a private workspace.

(d.) We ask that in the submitted version of your scripts and functions you suppress all display by placing a "`;`" at the end of each line of code. (Of course during debugging you will often choose to display many intermediate and final results.) We also require that **before** you upload your solution you should run `grade_o_matic_A2` (from your `YOURNAME_ASSIGNMENT_2` folder) for final confirmation that all is in order.

Questions

1. (10 points) Write a script which, given a 20×40 array `M`, performs the following operations (in sequence):

   (*i*) creates a new 20×40 array, `D`, which is all zeros except `D(i,i) = 1` for $1 \leq$ `i` ≤ 20 and `D(i,i+20) = 2` for $1 \leq$ `i` ≤ 20;

   (*ii*) creates a new 20×40 matrix `A` as the sum of the corresponding entries of arrays `M` and `D` — for example, `A(1,2) = M(1,2) + D(1,2)`;

   (*iii*) creates a new 20×40 array, `B`, which is the same as array `A` except row 11 for which `B(11,j) = 1/j`, for $1 \leq$ `j` ≤ 40;

   (*iv*) creates a new 20×41 array, `C`, which is the same as array `B` for columns 1 through 40 but also includes a column 41 in which all elements are assigned the value 3;

   (*v*) creates a new 20×41 array, `P`, which is the same as array `C` except the first ten entries on the main diagonal for which `P(i,i) = .1 * i * C(i,i)`, for $1 \leq$ `i` ≤ 10;

   (*vi*) creates a new 20×41 array, `Q`, which is the same as array `P` except the `(1,2)` entry for which `Q(1,2)` is assigned the value 7;

   (*vii*) creates a new 20×41 array, `R`, in which each element is the square of the corresponding element in array `Q` — for example, since `Q(1,2)` $= 7$ then `R(1,2)` $= 49$;

   (*viii*) creates a scalar `bigsum` which is the sum of all the elements (820 in total) of the array `R`.

   No functions are required for this question.

   The script takes a single script input: the input is a 20×40 array `M` and must correspond in your script to MATLAB variable `M`; the allowable instances, or input parameter domain, is given by `abs(M(i,j))` $\leq 10, 1 \leq$ `i,j` ≤ 10. The script yields a single script output: our output is the scalar `bigsum` and must correspond in your script to MATLAB (scalar) variable `bigsum`. The template is provided in `A2Q1_Template`.

2. (12 points) A manufacturer installs a quality control sensor system on an assembly line in order to reduce the number of flawed parts shipped to customers. We denote by DECISION a random "variable" which represents the sensor system action. The random variable DECISION can take on two "values": DECISION = ACCEPT — accept the part and ship to customer; DECISION = REJECT — reject the part and recycle the material (i.e, do not ship to customer). We denote by PART a random "variable" which represents the quality of the part. The random variable PART can take on two "values": PART = UNFLAWED — the part meets the necessary quality standards, and is thus unflawed (i.e., without flaws); PART = FLAWED — the part does not meet the necessary quality standards, and is thus flawed.

Although in principle we might hope that the sensor system will accept *all* parts which are unflawed and furthermore accept *only* parts which are unflawed, in actual practice the sensor system will not be perfect: it may not accept some parts which are unflawed, and it may accept some parts which are flawed. The developer of the sensor system, based on simulation results, provides the following conditional probabilities: P(DECISION = ACCEPT | PART = UNFLAWED) = 0.95; P(DECISION = ACCEPT | PART = FLAWED) = 0.02. Furthermore, the manufacturer of the part, based on historical data, provides the marginal probability P(PART = FLAWED) = 0.03.

(*i*) (4 points) The marginal probability P(DECISION = ACCEPT) is

(*a*) 0.9700
(*b*) 0.9221
(*c*) 0.9500
(*d*) 0.9800

Hint: Express the marginal probability as a sum of joint probabilities; express the joint probabilities in terms of conditional and (other) marginal probabilities.

(*ii*) (4 points) The conditional probability P(PART = FLAWED | DECISION = ACCEPT) is

(*a*) 0.05000
(*b*) 0.00060
(*c*) 0.00065
(*d*) 0.02000

Hint: Consider Bayes' Theorem.

(*iii*) (4 points) In one month 1,000,000 parts are made on the assembly line. How many flawed parts will be (accepted by the sensor and hence) shipped to customers?

(*a*) 600
(*b*) 30,000
(*c*) 650
(*d*) 28

You should choose the answer which is most likely. Hint: Consider the appropriate *joint* probability and then apply the frequentist interpretation of probability.

The template `A2Q2_Template.m` contains the multiple-choice format required by `grade_o_matic_A2`.

3. (10 points) In this question we consider a discrete random variable X which can take on three values, 0 or 1 or 2, according to the probability mass function $f_X(x)$ given by

$$f_X(x) = \begin{cases} 2/5 & x = 0 \\ 1/2 & x = 1 \\ 1/10 & x = 2 \end{cases} . \tag{1}$$

We recall that $f_X(x)$ is the probability that the random variable X takes on the value x. We also consider a second random variable Y which is a function of X such that

$$Y = \begin{cases} 0 & \text{if } X = 0 \text{ or } X = 2 \\ 1 & \text{if } X = 1 \end{cases} . \tag{2}$$

Note that Y can take on only two values, either 0 or 1.

(i) (2.5 points) The mean of X is

(a) 1/4
(b) 1/2
(c) 7/30
(d) 7/10
(e) 1

(ii) (2.5 points) The standard deviation of X is

(a) 0.9487
(b) 0.6403
(c) 2.0000
(d) 1.0000
(e) 0.5000

Note we are asking for the standard deviation, *not* the variance.

(iii) (2.5 points) The mean of Y is

(a) 1/4
(b) 1/2
(c) 7/10
(d) 7/30
(e) 1

(iv) (2.5 points) The standard deviation of Y is

(a) 0.9487
(b) 0.6403
(c) 2.0000
(d) 1.0000
(e) 0.5000

Note we are asking for the standard deviation, *not* the variance.

The template `A2Q3_Template.m` contains the multiple-choice format required by `grade_o_matic_A2`.

4. (10 points) Write a function with "signature"

```
function [variates_vec] = prob_mass_Q3(n)
```

which returns a row vector `variates_vec` of `n` independent random variables (more precisely, realizations of independent random variables — random variates) drawn from the probability mass function $f_X(x)$ of Question 3 as given in Equation (1).

The function must be named `prob_mass_Q3` and furthermore must be stored in a file named `prob_mass_Q3.m`. The function takes a single function input: the input `n` is a scalar; the allowable instances, or parameter domain, is $10 \leq$ `n` $\leq 100,000$. The function yields a single function output: the output is the $1\times$ `n` row vector `variates_vec`. Note that `variates_vec` must be random not just in terms of the frequencies of the outcomes ($x = 0, x = 1$, and $x = 2$) but also in terms of the order — such that (say) the first `n/2` elements of `variates_vec` should also constitute a sample of i.i.d. random variates from f_X.

The script for this question is provided in `A2Q4_Template`; no modifications are required (except to remove the `_Template` from the name).

5. (4 points) Write a function with "signature"

```
function [zpts] = unif_over_sym_int(c,n)
```

which provides the values `zpts(i)`, $1 \leq$ `i` $\leq$ `n`, of `n` independent ("i.i.d.") random variates of the univariate uniform distribution over the symmetric (about zero) interval `-c` $\leq z \leq$ `c`.

The function must be named `unif_over_sym_int` and furthermore must be stored in a file named `unif_over_sym_int.m`. The function takes two function inputs. The first input is the scalar `c`; the allowable instances, or parameter domain, is $1 \leq$ `c` ≤ 10. The second input is the scalar `n`; the allowable instances, or parameter domain, is $10 \leq$ `n` $\leq 100,000$. The function yields a single function output: the output is the $1\times$ `n` row vector `zpts`. Note that `zpts` must be random not just in terms of the frequencies of the outcomes but also in terms of the order — such that (say) the first `n/2` elements of `zpts` should also constitute a sample of i.i.d. random variates from the uniform density over `-c` $\leq z \leq$ `c`.

The script for this question is provided in `A2Q5_Template`; no modifications are required (except to remove the `_Template` from the name).

6. (10 points) The (univariate) normal density is often a good and also convenient description of the outcome of a random phenomenon. However, in the case in which the outcome must be positive (on physical grounds, for example a spring constant), a normal random variable — which can in principle take on all values negative and positive — can create difficulties. In the case in which negative values are very rare, we can justifiably consider a "rectified" normal random variable: a zero or negative value is rejected and we draw again from the desired normal population until we obtain a positive value. *Note this procedure creates a density which (is zero for negative values and) has the same* shape *as the normal density for positive values but is uniformly amplified to integrate to unity over the positive real numbers.*

Write a function with signature

```
function [xpts_pos] = positive_normal(mu,sigma,n)
```

which provides, based on the procedure described above, a vector of (realizations of) `n` i.i.d. (independent, identically distributed) rectified normal random variables drawn from a normal population with mean `mu` and standard deviation `sigma` (note the mean and standard deviation are defined for the normal random variables *before* rectification).

The function must be named `positive_normal` and furthermore must be stored in a file named `positive_normal.m`. The function takes three function inputs. The first two inputs, respectively `mu` and `sigma`, are scalars; the allowable instances, or parameter domain, is $.2 \leq$ `mu` ≤ 2 and $.05 \leq$ `sigma` $\leq$ `2*mu`. The input `n` is a scalar; the allowable instances, or parameter domain, is $10 \leq$ `n` $\leq 100,000$. The function yields a single function output: the output is the $1\times$ `n` row vector `xpts_pos`. Note that `xpts_pos` must be random not just in terms of the frequencies of the outcomes but also in terms of the order — such that (say) the first `n/2` elements of `xpts_pos` should also constitute a sample of i.i.d. random variates from the rectified normal density.

The script for this question is provided in `A2Q6_Template`; no modifications are required (except to remove the `_Template` from the name).

7. (12 points) A spring for a micro-robot suspension is required to have a spring constant k between $k_{\text{lower}} = 2000$ N/m and $k_{\text{upper}} = 2500$ N/m in order to provide the right balance between isolation (of the cargo) and control for navigation. The robot manufacturer receives a batch of springs from the spring supplier and proceeds to measure the spring stiffness of $n = 1000$ randomly chosen springs from the batch. It is found that, of these 1000 springs, 987 of the springs do indeed have a spring constant within the desired range — between 2000 N/m and 2500 N/m; the remaining 13 springs have a spring constant outside the desired range.

   To model this situation we introduce a Bernoulli random variable B: a spring constant outside the desirable range — $K > k_{\text{upper}}$ or $K < k_{\text{lower}}$ — is encoded as $B = 0$ and occurs with probability $1 - \theta$; a spring constant inside the desirable range — $k_{\text{lower}} \leq K \leq k_{\text{upper}}$ — is encoded as $B = 1$ and occurs with probability θ. Here K is the random variable which represents the spring constant and which in turn defines the Bernoulli random variable. We wish to determine the parameter θ from our sample of 1000 randomly chosen springs.

   (*i*) (4 points) Based on the experimental data from the sample of $n = 1000$ springs, the sample-mean estimate for θ is

   (*a*) 987
   (*b*) 0.9870
   (*c*) 0.9500
   (*d*) 0.0130

   (*ii*) (4 points) Based on the experimental data from the sample of $n = 1000$ springs, the (two-sided) normal-approximation confidence interval for θ at confidence level $\gamma = 0.95$ is

   (*a*) $[0.9800, 0.9940]$
   (*b*) $[0.9834, 0.9906]$
   (*c*) $[0.7650, 1.2090]$
   (*d*) can not be evaluated as the normal-approximation criteria (page 178 of textbook) are not satisfied

   (*iii*) (4 points) A value for θ less than 0.97 requires the spring supplier to pay the robot manufacturer a penalty, whereas a value for θ greater than 0.99 requires the robot

manufacturer to pay the spring supplier a premium. From your result of part (*ii*) can you conclude with confidence level 0.95 that $0.97 < \theta < 0.99$?

(*a*) Yes

(*b*) No

Note you may assume here that our random model for the spring constant K and hence Bernoulli variable B is valid (as only in this case can you make rigorous statistical inferences).

The template `A2Q7_Template.m` contains the multiple-choice format required by `grade_o_matic_A2`.

8. (12 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.*x + 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) (4 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) $-1 \leq$ `x1(i)` ≤ 1, $-1 \leq$ `x2(i)` ≤ 1
(b) $-1 \leq$ `x1(i)` ≤ 1, $\quad 0 \leq$ `x2(i)` ≤ 1
(c) $\quad 0 \leq$ `x1(i)` ≤ 1, $-1 \leq$ `x2(i)` ≤ 1
(d) $\quad 0 \leq$ `x1(i)` ≤ 1, $\quad 0 \leq$ `x2(i)` ≤ 2

(ii) (4 points) The area A_D is given by

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

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` conditional in BLOCK; finally, recall the area interpretation of the definite integral to determine A_D.

(iii) (4 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`.

The template `A2Q8_Template.m` contains the multiple-choice format required by `grade_o_matic_A2`.

9. (12 points) Write a function with signature

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

which computes, based on the method of Section 12.1.1–12.1.4 (of the textbook), 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

$$D = \{x_1^2 + x_2^2 \leq 0.75\} \cup \{(x_1 - \alpha)^2 + x_2^2 \leq 0.75\} \; . \tag{3}$$

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.

The function must be named `MC_area` and furthermore must be stored in a file named `MC_area.m`. The function takes five function inputs. The first input, scalar `alpha`, is the center shift α; the allowable instances, or parameter domain, is $0 \leq$ `alpha` $\leq .2$. The second input, the scalar `c`, defines the bounding rectangle (in fact, here a 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. The third input is scalar `n`, the number of random darts; the allowable instances, or parameter domain, is $10 \leq$ `n` $\leq 100,000$. The fourth and fifth inputs are the coordinates of the `n` random darts, (`x1pts(i),x2pts(i)`), $1 \leq$ `i` $\leq$ `n`, thrown at the square R: `x1pts` and `x2pts` are each $1\times$ `n` array of independent ("i.i.d.") random uniform variates over the interval $-c \leq x_1 \leq c$; there are no restrictions on allowable instances (as the number of darts is already controlled by `n`). The function yields two function outputs: the first output is a scalar `area_est` which is the Monte-Carlo estimate for the area, $(\hat{A}_D)_n$; the second output is the 1×2 row vector `area_conf_int` such that `area_conf_int(1)` and `area_conf_int(2)` are respectively the lower and upper limits of the confidence interval $[\mathrm{ci}_{A_D}]_n$.

Two further points: First, in the event that the normal-approximation criteria for the construction of the confidence interval (see page 178 of the textbook) are not satisfied *your code should return* `[-1,-1]` for `area_conf_int`. Second, you should use 1.96 for the value of z_γ for $\gamma = 0.95$.

In order to test your function `MC_area` for any desired (`alpha` and) `c` and `n` you must first generate the random darts `x1pts,x2pts`. Note, however, that for purposes of grading, `grade_o_matic_A2` will automatically generate appropriate inputs `x1pts,x2pts` for your function — no action needed on your part.

The script for this question is provided in `A2Q9_Template`; no modifications are required (except to remove the `_Template` from the name).

10. (8 points) We ask you here to reconsider from a theoretical perspective some aspects of earlier more computational questions. Note that although we intend you to derive your answers from analysis rather than computation you may certainly take advantage of your codes to confirm your theoretical predictions.

    (i) (4 points) In Question 6 the cumulative distribution function of the rectified normal random variable X is given by

    (a) $(\Phi(\frac{x-\mu}{\sigma}))^2,\ x > 0$;

    (b) $\Phi(\frac{x-\mu}{\sigma})/(1-\Phi(\frac{-\mu}{\sigma})),\ x > 0$;

    (c) $\Phi(\frac{x-\mu}{\sigma})/\Phi(\frac{-\mu}{\sigma}),\ x > 0$;

    (d) $\Phi(\frac{x-\mu}{\sigma}),\ x > 0$;

    here μ and σ refer here to the mean and standard deviation of the unrectified normal random variable from which the rectified normal variable X is derived. Note that Φ is the cumulative distribution function of the standard normal distribution such that $F^{\mathrm{normal}}(x;\mu;\sigma^2) = \Phi(\frac{x-\mu}{\sigma})$. Hint: Recall that for $x > 0$ the probability density for the rectified normal random variable has the same shape (but not necessarily the same amplitude) as the probability density for the unrectified normal random variable.

(*ii*) (4 points) In Question 9 the half-length of the confidence interval — given by the expression `(area_conf_int(2) - area_conf_int(1))/2` — will scale (for fixed `n`) as

(*a*) $\texttt{c}^1$

(*b*) $\texttt{c}^{1/2}$

(*c*) $\texttt{c}^0$ (i.e., independent of `c`)

(*d*) $\texttt{c}^{-1/2}$

as `c` increases. (Of course `area_conf_int` will fluctuate; we are interested here in the average behavior.) For the purposes of this theoretical question we need not restrict the "instances" of `c` from above: we may consider any `c` greater than unity. Your answer should of course suggest the *best* value of `c` if we wish to obtain the most accurate estimate for the area.

The template `A2Q10_Template.m` contains the multiple-choice format required by `grade_o_matic_A2`.

CHALLENGE: Is the birthmonth data collected in lecture on 21 February "consistent" — within some reasonable statistical fluctuation — with the discrete uniform probability mass function: $f_X(x_j \equiv (\text{birthmonth})\, j) = 1/12, 1 \leq j \leq 12$? The data is found in the `Assignment_2_Materials` folder in `birthmonth.mat` as array `birthmonth_frequency` in which `birthmonth_frequency(j) =` number of members of the sample with birthday in month `j` (with the usual convention of January as month 1 and December as month 12). You might consider the statistic

$$Y = \Big(\frac{1}{12} \sum_{\texttt{j}=1}^{12} \big(\texttt{birthmonth\_frequency\_hypothesis(j)} - \frac{1}{12}\big)^2 \Big)^{1/2} , \tag{4}$$

in which `birthmonth_frequency_hypothesis(j)` is equal to the number of members with birthday in month `j` in a sample of size `sum(birthmonth_frequency)` drawn from the uniform mass function f_X. In particular, you can create, by Monte Carlo methods, the (approximate) probability mass function for Y — note each experiment to generate a realization of the 1×12 array `birthmonth_frequency_hypothesis` is derived from a sample of `sum(birthmonth_frequency)` i.i.d. random variates from f_X — and then assess whether the deviation from uniformity actually observed in our 2.086 sample is a reasonable statistical fluctuation (or not).

MIT OpenCourseWare
http://ocw.mit.edu

2.086 Numerical Computation for Mechanical Engineers

Spring 2013

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.