MATLAB Projects

*Solutions to the projects marked with an asterisk * are in the DIPUM3E Student Support Package (consult the book web site). All your code must be documented so that typing* `help` *at the prompt, followed by the script or function name, gives enough detail for a user to be able to run it. Test the functionality of all your code thoroughly.*

2.1 The following scripts and functions will improve your productivity by reducing the amount of typing you have to do to execute common commands during code development and testing.

(a)* Write a script called `ca` for closing all open figure windows.

(b) Write a script called `xx` for clearing the **Workspace**, **Command Window**, and closing all open figure windows.

(c)* Write a function `fs` with input image `f` for performing the command: `>> figure, imshow(f)`.

(d) Write a function `fss` with input image `f` for performing the command: `>> figure, imshow(f,[])`.

2.2 Do the following.

(a)* MATLAB does not have a function for determining which elements of an array are even numbers. Write a function with the following specifications for this purpose. (*Hint:* Consider using one of the functions `floor`, `rem`, or `mod`.)

```
function [E,S] = isevenTest(A)
%ISEVENTEST Determines which elements of an array are even numbers.
%   [E,S] = ISEVENTEST(A) returns a logical array, E, of the same size as
%   A, with 1s (TRUE) in the locations corresponding to even numbers
%   (i.e., . . . -4, -2, 0, 2, 4 . . . ) in A, and 0s (FALSE) elsewhere.
%   Scalar S is 1 (TRUE) if ALL elements of A are even; otherwise S is 0
%   (FALSE). A must be a real, numeric array. S is used in cases where
%   testing the entire array as a unit is required.
```

(b) MATLAB does not have a function for determining which elements of an array are odd numbers. Write a function with the following specification for this purpose.

```
function [E,S] = isoddTest(A)
%ISODDTEST Determines which elements of an array are odd numbers.
%   [D,S] = ISODDTEST(A) returns a logical array, D, of the same size as
%   A, with 1s (TRUE) in the locations corresponding to odd numbers
%   (i.e., . . . -3, -1, 1, 3, . . . ) in A, and 0s (FALSE) elsewhere.
%   Scalar S is 1 (TRUE) if ALL elements of A are odd; otherwise S is 0
%   (FALSE). A must be a real, numeric array. S is used in cases where
%   testing the entire array as a unit is required.
```

2.3 As indicated in Table 2.13, MATLAB has a function called `isinteger` that checks if the elements of an array are of a MATLAB *integer class* (see Table 2.5). However, we are often interested in finding the elements of an array that are *integers* in the "traditional" sense; that is, elements that have no fractional parts. Such numbers are also called *whole numbers*. In this project you are asked to write a function for finding such numbers, according to the following specification. (*Hint:* Consider using function `floor`).

```
function [W,S] = iswholeTest(A)
%ISWHOLETEST Elements of an array that are whole numbers (integers)
%   [W,S] = ISWHOLETEST(A) returns a logical array, W, of the same size
%   as A, with 1s (TRUE) in the locations corresponding to whole numbers
%   in A, and 0s (FALSE) elsewhere. Scalar S is 1 (TRUE) if ALL elements
%   of A are whole numbers; otherwise S is 0 (FALSE). A must be a real,
```

```
%   numeric array. S is used in cases where testing the entire as a
%   unit is required.
```

2.4 You may need a utility program in various parts of the book for masking regions in an image.

(a)* Write a function with the following specifications to be used for this purpose (do not use loops).

```
function R = mask(M,N,r,c,m,n)
%MASK Creates a binary image mask.
%   R = mask(M,N,r,c,m,n) creates an M-by-N binary image mask with zeros
%   in the background and 1s in a rectangular m-by-n region with top,
%   left corner at coordinates (r,c).
```

(b) Read the image `rose512.tif` and use `mask` to generate an image the same size as the original, but containing only the rose without the stem. (*Hint:* Consider using the Toolbox function `impixelinfo` to interactively determine the values needed for `mask`.)

2.5 The purpose of this project is to compare the performance of loop-based and vectorized code. In the following, $M = N = 1000$, $r = c = 100$, and $m = n = 500$.

(a) Modify the function `mask` from Project 2.4 using `for` loops with space allocation. Call the new function `maskLoopsSA`.

(b) Modify `maskLoopsSA` by removing the code used for allocation. Call this function `maskLoopsNSA`.

(c) Use the function `timeit` to time the functions `mask`, `maskLoopsSA`, and `maskLoopsNSA` using the parameters shown above. List the individual times for these three functions. List the ratio of each of the times of the two loop function with respect to the time for `mask`.

2.6 Do the following:

(a) Write a function for generating grayscale test images, based on the following specifications. (*Hint:* Consider using function `iswholeTest` from Project 2.3, and MATLAB functions `checkerboard`, `rand`, and `randn`.)

```
function f = testImage(type,param)
%TESTIMAGE Generates gray scale test images.
%   F = TESTIMAGE(TYPE,PARAM) generates a grayscale image of size M-by-N
%   and class double. TYPE is a string, and PARAM is a vector of
%   parameters, as described below:
%
%       TYPE         PARAM                    DESCRIPTION
%   'checkerboard'  [M,N,n]        Checkerboard image of
%                                  size M-by-N, with each square of size
%                                  n-by-n pixels. Note: Both M N must be
%                                  divisible by 2n; otherwise the
%                                  checkerboard will not make sense.
%   'unoise'        [M,N]          Image of size M-by-N whose pixels
%                                  are uniformly distributed random
%                                  numbers in the range [0,1].
%   'gnoise'        [M,N]          Image of size M-by-N whose pixels
%                                  are random numbers drawn from a
%                                  standard normal distribution (i.e., a
%                                  normal(Gaussian) distribution with
%                                  mean of 0 and standard deviation
%                                  equal to 1).
%   'vstripe'      [M,N,n]         M-by-N black image with a white,
%                                  centered, vert stripe n pixels wide.
```

```
%                                   n cannot exceed N.
%   'hstripe'      [M,N,n]          M-by-N black image with a white,
%                                   centered, horiz stripe n pixels wide.
%                                   n cannot exceed M.
```

(b)* Generate and display each image type for $M = N = 512$ using parameters of your choice.

2.7* As we discussed in Section 2.4, function `imfinfo` can be used to obtain information about an image. However, this function expects its input to be a file name or URL location. Often we are interested in numerical information about an image being processed.

(a) Write a function with the following specifications for obtaining such information. Use structures, and keep in mind that some fields of the structure can have more than one value:

```
function info = myimageinfo(f)
%MYIMAGEINFO Numerical information about an image.
%   INFO = MYIMMAGEINFO(F) extracts numerical information from image F
%   and outputs it using the following structure forms. In general, F
%   can be of size M-by-N-by-P, with P >= 1 (for example, for RGB
%   images, P = 3).
%
%   info.rows = number of rows in the image.
%   info.cols = number of columns.
%   info.planes = the number of image planes.
%   info.max = maximum value of each image plane as a 1xP vector.
%   info.min = minimum value of each image plane as a 1xP vector.
```

(b) Read the image `ballerina-rgb.tif`. Apply function `myimaginfo(f)` to it and print out the results.

2.8 Repeat Project 2.7 using cell arrays.

(a) Your function should be called `myimageinfoC(f)` and have the following specifications.

```
function infoC = myimageinfoC(f)
%MYIMAGEINFOC Numerical information about an image.
%  INFOC = MYIMAGEINFOC(F) extracts numerical information from image F
%  and outputs it using the following cell array forms. In general, F
%  can be of size M-by-N-by-P, with P >= 1 (for example, for RGB images,
%  P = 3).
%
%  infoC{1} = number of rows in the image.
%  infoC{2} = number of columns.
%  infoC{3} = the number of image planes.
%  infoC{4}(I) = maximum value of each image plane, I = 1:P.
%  infoC{5}(I) = minimum value each image plane, I = 1:P.
```

(b) Read the image `ballerina.tif` from the DIPUM3E Images folder. Apply function `myimageinfoC(f)` to it and print out the results.

2.9 Two-dimensional plots. Part of this project is intended to give you training in looking up MATLAB documentation topics with which you may not be familiar.

(a)* Plot the 1-D function $g_1(x) = \sin(x) + \sin^3(x)$ from 0 to 2π. Your plot should have the following characteristics: (1) 200 points; (2) a solid red line with a weight of 2 points; (3) the axes must be tight; (4) the horizontal axis should be labeled x (radians); (5) the vertical axis should be labeled sin(x)^2 + sin(x)^3; (6) the font should be Arial, of size 16 points; and (7) the title of the plot should be "A Plot Within a Plot".

(b)* Now plot the function $g_2(x) = \cos^2(x) + \cos^3(x)$ from 0 to 2π. This plot should be placed in the upper right quadrant of the first plot. The plot line should be blue, with a weight of 1 point. With the exception of the title, the embedded plot must be labeled as the first one, but using Arial font of size 10.

(c) Read the image `lunar-shadows.tif`. The objective of this part of the project is to compute the image histogram (normalize it by dividing it by the number of pixels in the image) and plot it as a bar graph in the lower right quadrant of the image. Your plot should have the following specifications: (1) the plot should contain 10 red bars; (2) the horizontal axis should range from 0 to 255, with tick marks at 0, 63, 127, 191, and 255; (3) the vertical axis should range from 0 to 0.5, with tick marks in increments of 0.1; (4) the axis numbers should be Arial, yellow, bold, 12 pt; (5) annotate the bar graph with the text "Histogram" (in blue, bold), with a font size of 16, and appearing on the top left quadrant of the bar graph window (not the image); (6) the horizontal axis should be labeled Intensity, with bold, green text of size 14 pt; (7) the vertical axis should be labeled "Histogram Values", using the same font attributes.

2.10 Three-dimensional plots. Part of this project is intended to give you training in looking up MATLAB documentation topics with which you may not be familiar.

(a)* Generate a mesh plot of the function $f(x,y) = xe^{-(x^2 + y^2)}$ for $-3 \leq x \leq 3$ and $-2 \leq y \leq 2$ in increments of 0.1. The axis values in your plot should span the exact ranges of x and y.

(b)* Change plot colors to black and white. Then change the line width to 1.0 pt and the surface face color to yellow.

(c) Change the axes font to Arial, size 8.

(d) Change the color of all three axes to blue.

(e) Label the x, y, and z axes with the labels "x-axis", "y-axis" and "z-axis", respectively. The font should be 16 pt, bold, red.

(f)* Change the z-axis limits to the range –0.5 to 0.5, and place ticks at locations –0.50, –0.25, 0, 0.25, and 0.50. The x- and y-axis should show values in ranges $[-3,3]$ and $[-2,2]$, respectively, with a minimum of five ticks each. Each tick should be labeled to show its value.

(g) Enclose the entire plot with an axis box.

(h) Annotate the figure with the text "3-D Plot with a Box" using Arial font of size 12 pt. Place the text at location $(x, y, z) = (30, 30, 0.5)$.