Mini Project 2: Array operations in MATLAB (3 pages)

Rule 1. Submit an M-file named `mp2_ABC_WXYZ.m`, where `ABC` is your section number and `WXYZ` are the last 4 digits of your NJIT ID. Observe capitalization. Use an underscore _ instead of a dash - or MATLAB will get confused. **SUPPRESS output for all of your MATLAB commands.**
Rule 2. Send an email to the instructor `alexg+cs101@njit.edu`, to the grader `js87+cs101@njit.edu`, and if you want to make sure that everything went fine with the transmission of your mail, to yourself as well. Include the three words `mp2 ABC WXYZ` in the subject line, separated with space(s) if your id ends with `WXYZ` and you are enrolled in CS101-ABC.
Rule 3. It is imperative that you fully obey Rules 1 and 2. Testing will be done by running a MATLAB program (that will also be given to you afterwards). If you deviate from these rules you will be getting errors (and points deducted). Observe variable names and capitalization. Only MATLAB commands introduced and used in class may be used in this assignment as instructed in it.
Due Date: Before noon time of Friday April 11, 2014.

1 Part A: Preliminary (10 points)

You will create a text-based M-file named (Rule 1) `mp2_ABC_WXYZ.m` where `ABC` and `WXYZ` are as specified in Rule 1. **Use semicolons to suppress output**.
First line. The first line of the MATLAB file will contain in the form of a MATLAB comment the name of the file in question i.e. `mp2_ABC_WXYZ.m` with the first character, the em, separated by at least two spaces from the MATLAB comment symbol.
Second line. The second line would contain in the form of a MATLAB comment line your last name fully capitalized followed by your first or other given names in lower case (all characters). Then include the last four digits of your id, eg `WXYZ`. Use one or more space characters to separate names, and id.
Third line. The third line will be empty.

The remaining lines of the MATLAB file are described in the following questions. Pay attention to the details. Variables have names starting with a v followed by a number. Do not change names or capitalization. Grading will be done automatically for most problems. **Use brief MATLAB array operators rather than explicit listing of matrix or vector elements. Thus If you need to create a row vector `v` containing 1,2,3,4, a solution that is along the lines `v= 1:4` is a correct one; writing something like `v = [ 1 2 3 4]` will get you 0 points. Use only commands introduced in class. You have been warned!**

Fourth line. Create variable `last_name` and set its value to be your last name in lower case in the form of a string. Use one (separate) line to realize this.

Fifth line. Create variable `id_four` and assign to it the last four digits of your NJIT id in the form of a string. Use one (separate) line to realize this

2 Part B: Array operations and sums (35 points)

6th line. In the context of this problem, variable n below will be storing values that the grader will be using to test your code. Include as a sixth line the following line.

```
if (exist('n')~=1); n=10; else n=n*10; end
```

The effect of it is that it will create variable `n` and initialize it to 10. For this to work variable n should not exist. If it exists its value will be multiplied by 10 instead. Thus every time you decide to run this script for testing or debugging make sure that you run first a `clear` and then a `mp2_ABC_WXYZ`.
The sum S_1 below approximates π or more specifically $pi^2/6$. Thus, if we compute for some n sum S_1 and then multiply the result with 6 and obtain the square root of it, we get an approximation of π.

$$S_1 = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots + \frac{1}{n^2} = \sum_{k=1}^{n} \frac{1}{k^2} \approx \frac{\pi^2}{6}$$

7th line. Leave it empty.

8th-12th lines (15 points). Compute in variable `v1` the approximation of π induced by the sum of the n terms of S_1 for whatever the current value of n is. To do so first generate vector $1, 2, \ldots, n$ into variable `v1a`, then square it to get $1^2, 2^2, \ldots, n^2$ and store the results into `v1b` , then invert its elements into `v1c` and add up `v1c`'s terms to generate a scalar value into `v1d`. These four steps can be realized in four lines using only array operations. (You are not allowed to use for/while loops, BTW.) Then in a fifth line compute in `v1` the approximation to π obtained through this S_1 equation. We will check all `v1a, v1b, v1c, v1d, v1` variables and their values. If you use more than 4 lines we will not penalize you (as long as no for/while loop are used).

The sum S_2 below also approximates π.

$$S_2 = 1 + \frac{1}{2^4} + \frac{1}{3^4} + \ldots + \frac{1}{n^4} = \sum_{k=1}^{n} \frac{1}{k^4} \approx \frac{\pi^4}{90}$$

13th line. Leave it empty.

14th-l8th lines (15 points). Do the same as before for this sum S_2, i.e. compute using variables `v2a,v2b,v2c,v2d` and into `v2` the approximation of π induced by the sum S_2.

19th line. Leave it empty.

20th line (5 points). Use an fprintf statement to print the first approximation of pi, then the second and finally the value of the Matlab `pi` constant all in one line. Starting from the far left margin of the command window the following strings are to be printed inside fprintf in one line (even if they are shown below in 3 lines to showcase their equal length). Each one will be followed by the value of `v1, v2`, π as needed. Each one of the three values printed will be wide just enough to accommodate 6 decimal digits to the right of the decimal points and one space from its preceding string, and one space from its following string, if any. The three strings, each one 12 characters long are now shown in separate lines below for clarity

```
S1 approx is
S2 approx is
MATLAB pi is
```

The total output length should not be more than about 66 characters long. Terminate it with a newline as the 67th character so that the the cursor moves to the next line.

3 Part C: Euler's constant(15 points)

The sum

$$S_3 = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k} \approx \ln n + \gamma$$

is approximately equal to $\ln n$ up to a constant that is known as Euler's constant (γ). We are interested in finding Euler's constant γ. Towards this we can compute constant c to a number of terms n

$$c = \left(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}\right) - \ln n$$

The value of c would become the n-th approximation of γ. For different values of n such that $n = 10$, $n = 100$, and $n = 1000$ compute the corresponding value of c, named $c10, c100, c1000$ respectively. If you plan to use any other variables (other than n that is) name them `v3, v3a, v3b, v3c,` etc. (You don't need that many anyway.) The value of c should be computed through array operations (see for example the discussion of section 2). Yet we ask you not to hardwire the value of n for the three separate computations. Instead utilize line 6 and replicate it or modify it accordingly. (Line 6, checks whether n is defined; if it is, its value is multiplied by 10, otherwise it is set to 10.) The end result is that you can use it to generate the values $n = 100, n = 1000$ without explicitly typings $n = 100$ and $n = 1000$, but using a previous value and multiplying it accordingly.

This will give you 9 of the 15 points for each one of $c10, c100, c1000$ calculations. Feel free to use as many lines as you like. We are only interested in the values of the three variables only, computed correctly!

Eventually we want to see an output line worth 6 points that is printed by an fprintf statement and includes strings and inbetween them the corresponding value computed with 6 decimal digits and only one space after the equal sign or before the following c character of strings such as c100, etc. Each one of the strings below is 7 characters long starting with the c and ending with the equal sign. Other than that (names of the three strings) the details are those of line 20 of Part B.

```
c10   =
c100  =
c1000 =
```

Note that in the output these three strings along the properly formatted values of $c10, c100, c1000$ must appear in the same line. The total width of the fprintf strings won't be more than about 52 characters. Make sure that you move the cursor (aka MATLAB prompt) to the next line at the end of fprintf.