Mini Project 2: Computing π (5 pages)

Rule 1. Read Handout 2; you submit an M-file `p2_WXYZ.m`, where `WXYZ` are the last 4 digits of your NJIT ID. Observe capitalization. Use an underscore _ instead of a dash -; **SUPPRESS output for all of your MATLAB commands.** If you submit a .mat file you get automatically a 0; if you submit a file with a different file-name than the one instructed, you will also receive a 0.

Rule 2. Send an email (with **subject line as specified in Handout 2**) to
(a) the instructor `alexg+cs101@njit.edu`,
(b) the grader whose email address was posted after Jan 31 on the course Web-page, and
(c) to yourself to make sure that everything went fine with the transmission of your mail.

Rule 3. It is imperative that you fully conform to Rules 1 and 2. If you deviate from these rules 0 points will get assigned or many points will be deducted. Observe variable names and capitalization.

Due Date: Before **noon time of Wed Apr 22, 2015**. (Penalties in Handout 1.)

1 Part 0: Primary function (18 points)

You will create a text-based M-file named (Rule 1) `p2_WXYZ.m` where `WXYZ` are as specified in Rule 1. **Always use semicolons to suppress output. 10 points may get deducted for every semicolon that is missing in the code.**

1. First 4 lines of M-file (7 points). The first and fourth line of this M-file are going to be empty MATLAB comment lines. The second line will contain in the form of a MATLAB comment the words `P2` (capitalized), the course prefix and number (capital-case), and the last four digits of your NJIT ID. The third line will contain in the form of a MATLAB comment, your first name (lower-case) and then your last name (capitalized). If you have more than one first/given names connect them with a dash(-), not an underscore _. Any number of 1 or 2 spaces may be used to delimit any two of the items described and/or the MATLAB comment symbol.

2. (Primary) Function and its definition (11 points for 11 lines). The fifth line would contain the definition of the primary function of this file. It returns no values and has only one parameter `NN`. You should be able to figure out what the name of this function should be. From this point one we REFER to the lines of the primary function in the items below, not to the lines of the M-file. The 11 points assigned to this item 2 are shared among these other lines that describe primary function code. Thus "first line" below in item 3 would refer to the first line of the primary function after its definition that is not empty: this might be the sixth line of the file. Empty lines (not empty comment lines) that you will be instructed to use won't be included into the count.

3. First line of (primary) function. It uses an fprintf statement to print your last name first fully capitalized in a box of width 20 right justified, followed by a single space, and then one (and only one) of your other given names in a box of width 15 right justified, followed by a single space, and then the last four digits of your NJIT ID in a box of width 7 as a decimal integer, left-justified. The fprintf statement should eventually move the cursor/prompt to the next line after it does what it is supposed to do. Make sure that the fprintf has 4 arguments. Leave an empty line after the fprintf statement before you go to the second line of the primary function.

2 Part A: $S_1(n)$ and π

4. Lines 2 and 3 of (primary) function. A MATLAB subfunction `p2_part_a` is to be called with input argument the value of `NN`; it accepts only one input argument and returns only one value. It is described in item 5 below. This function computes an approximation of π (not MATLAB's `pi`) by computing a finite number $(n+1)$ of terms of an infinite sum that converges to $\pi/4$. Sum $S_1(n)$ is:

$$S_1(n) = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4}$$

Subfunction `p2_part_a` would only compute the first $n+1$ terms of the sum, not an infinite number, and will thus compute an approximation to π of order n. The actions of the subfunction that you must implement are described below in item 5. The value returned by `p2_part_a` will be captured by a variable named `approx` of the primary function. This is the output argument of the subfunction invocation.

In the third line of the primary function immediately following the subfunction invocation, an fprintf statement is to be issued. The first three characters output are the last character of the subfunction's name (i.e. an a) followed by a colon followed by a single space. Then in a box of width 9, left-justified, the value of `NN` used as an argument to the subfunction is printed. After a single space is output, the approximation of π as captured in the value of variable `approx` is printed as a floating-point number of total width 10 with 6 decimal digits and right justified. Leave an empty line after the fprintf statement before you go to the fourth line of the primary function.

5. Subfunction `p2_part_a`. This subfunction is naturally defined inside the M-file as necessary. It has exactly one parameter that is called `n` (note the case) and returns one value, the value of a variable named `v1`. It works by performing a sequence of side effects. Variable `v1` first computes and stores the sum $S_1(n)$ for the necessary number of terms, and then the relevant approximation of π as derived from $S_1(n)$. The computation of $S_1(n)$ (i.e. `v1`) will be realized using array operations only. For this, generate in variables named `v1a, v1b, ...` (use as few as possible) the following.

5.1 A vector $1, 3, \ldots, 2 * n + 1$ into `v1a`,

5.2 then invert those elements to obtain the denominator of $S_1(n)$ in `v1a`

5.3 then generate the numerator $(-1)^n$ for those $n+1$ terms into vector `v1b`,

5.4 and then combine `v1a, v1b` using array operations, take the sum of the result to get $S_1(n)$ and store it into variable `v1`, and finally,

5.5 adjust appropriately `v1` to obtain an approximation to π into variable `v1`.

Subsequently the subfunction terminates its execution. (Again note that `v1` as soon as the execution of 5.4 completes does not hold an approximation to `pi` but of $S_1(n)$. It's your responsibility to compute an approximation of π from that value in 5.5. Minimize the number of additional variables used!

3 Part B: More to come and $S_2(n)$

In item 5 above we had stopped after the third line of the primary function, with empty lines not counted as 'lines'. We go on adding code to that primary function.

6. Lines 4 and 5 of (primary) function. These two lines are similar to lines 2 and 3. Line 4 calls a subfunction named `p2_part_b` that has syntax similar to `p2_part_a` and is described in item 7. This subfunction computes $S_2(n)$, another approximation to $\pi/4$, as before.

$$S_2(n) = \sum_{n=0}^{\infty} \left(\frac{1}{4n+1} - \frac{1}{4n+3} \right) = \frac{\pi}{4}$$

The mathematically curious of you might realize that $S_2(n)$ can be obtained by $S_1(n)$ if one pairs two terms of $S_1(n)$. In line 5 of the primary function an fprintf statement follows that works as before to print the approximation of π and `NN` along with other information related to this subfunction (i.e the first character printed now is not an a). An empty line follows line 5.

7. Subfunction `p2_part_b`. This subfunction is defined inside the M-file as necessary and after the complete definition of the previous subfunction. It has the same interface as the previous subfunction defined in item 5: one parameter n and one return value, the value of a variable `v1`. It works by performing a sequence of side effects. Variable `v1` first computes and stores the sum $S_2(n)$ for a number of terms, and then the relevant approximation of π as derived from $S_2(n)$. The computation of $S_2(n)$ will be realized using array operations only. For this generate in variables named `v1a, v1b, ...` the following.

7.1 a vector $1, 5, 9, \ldots, 4 * n + 1$ into `v1a`,

7.2 then invert those elements of `v1a` to obtain the left side of the general term,

7.3 a vector $3, 7, 11, \ldots, 4 * n + 3$ into `v1b`,

7.4 then invert those elements of `v1b` to obtain the right side of the general term,

7.5 find the difference of the two (7.2 minus 7.4),

7.6 and compute $S_2(n)$ as needed into the variable `v1`, and finally

7.7 compute an approximation to π into also `v1`.

Subsequently the subfunction terminates its execution.

4 Part C: More to come

In item 7 above we had stopped after the fifth line of the primary function. We go on adding code to that primary function.

8. Lines 6 and 7 of (primary) function. These two lines are similar to say lines 4 and 5. Line 6 calls a subfunction named `p2_part_c` that has syntax similar to `p2_part_b` and is described in item 9. This subfunction computes $S_3(n)$, another approximation to $\pi/4$. In line 7 an fprintf statement follows that works as before to print the approximation of π and `NN` along with other information related to this subfunction. An empty line follows line 7. The two terms of the general term of $S_2(n)$ are combined as follows in $S_3(n)$.

$$S_3(n) = S_2(n) = \sum_{n=0}^{\infty} \left(\frac{2}{(4n+1) * (4n+3)} \right) = \frac{\pi}{4}$$

9. Subfunction `p2_part_c`. This subfunction is defined inside the M-file as necessary and after the complete definition of the previously defined subfunction(s). It has the same interface as the previous subfunction(s) defined in item(s) 5, 7: one parameter n and one return value, the value of variable `v1`. It works by performing a sequence of side effects. Variable `v1` computes and stores the sum $S_3(n)$ for $n+1$ terms and then the relevant approximation of π as derived from $S_3(n)$. The computation of $S_3(n)$ will be realized using array operations only. For this generate in variables named `v1a, v1b, ...` the following.

9.1 a vector of $n+1$ terms $2/3, 2/35, \ldots$ into `v1a`,

9.2 and compute $S_3(n)$ as needed into variable `v1`, and finally

9.3 compute an approximation to π into `v1`.

Subsequently the subfunction terminates its execution.

5 Part D: Part A as a for loop

In item 9 above we had stopped after the seventh line of the primary function. We go on adding code to that primary function.

10. Lines 8 and 9 of (primary) function. These two lines are similar to lines 6 and 7. Line 8 calls subfunction named `p2_part_d` that has syntax similar to `p2_part_a` and is described in item 11. In line 9 an fprintf statement follows that works as before to print the approximation of π and `NN` along with other information related to the subfunction. An empty line follows line 9.

11. Subfunction `p2_part_d`. This subfunction is defined inside the M-file as necessary and after the complete definition of the previous subfunction. It has the same interface as the previous subfunction defined in item 5: one parameter n and one return value, the value of variable `v1`. It is a for loop implementation of $S_1(n)$ computed in `p2_part_a`.

6 Part E: Part A as a while loop

In item 11 we had stopped after the ninth line; we go on adding code to that primary function.

12. Lines 10 and 11 of (primary) function. These two lines are similarly (but not identical to say line 8 and 9, so pay more attention!).

Line 10 calls subfunction named `p2_part_e` WHOSE SYNTAX IS RADICALLY DIFFERENT THAN THAT OF THE PREVIOUS SUBFUNCTIONS. It does not accept any input argument, and it returns two values to the caller of this subfunction. The two return values are to be captured into two output arguments through variables `iter` and `approx` of the primary function. The latter has been used before, the former is introduced specifically to capture information of this subfunction. Subfunction `p2_part_e` is described in item 13. In line 11 of the primary function an fprintf statement follows that works as before to print the approximation of π; instead of printing `NN` the actual number of iterations of the while loop that has been captured in the code of the subfunction is printed through variable `iter`, along with other information related to the subfunction. No need for an empty line after line 11. The primary function appropriately terminates.

13. Subfunction `p2_part_e`. This subfunction is defined inside the M-file as necessary and after the complete definition of the previous subfunction. It does not require any parameter. The number of terms of $S_1(n)$ generated is not going to be fixed but variable and dependent on a terminating condition. The subfunction would return two values to the caller. The first of the two values returned and note that order is important and will be graded, would be the value of an iterative variable that you should name `ii` that keeps track of the number of iterations performed. Whether the number of iterations performed is `ii` or plus or minus one it is up to you to figure out correctly and print as needed in line 11 of the primary function. The second value returned by this subfunction is the approximation to π obtained and would be the value of variable `v1` of the subfunction.

In the while loop implementation of `p2_part_a` you generated fractions $1/(2n+1)$ for a variety of values $n = 0, 1, 2, \ldots$, that was determined by the input argument of the subfunction. Here you won't tell the subfunction how many terms to generate i.e. you won't provide it with an input argument n. The subfunction will generate as many as needed. The stopping (or terminating) condition would be whether $1/(2n+1)$ is less than or equal to 10^{-7} or not. Thus as long as $1/(2n+1)$ for arbitrary n is greater than 10^{-7} the corresponding term $(-1)^n/(2n+1)$ would be generated and used for the calculation of $S_1(n)$. If however $1/(2n+1)$ IS LESS THAN OR EQUAL TO 10^{-7}, adding or subtracting such a miniscule term into $S_1(n)$ won't make much difference. Thus we stop the execution of the while loop ignoring that and other subsequent contributions to $S_1(n)$.

7 Part F: Grading

Of the 85 points of the assigment

- 7 points for the first four lines of the M-file, and 11 points for the proper and correct formulation of the 11 lines of the primary function plus the other 'lines' inbetween,
- 6, 15, and 6 points respectively are assigned to the correct formulation of the fprintf statements in line 1, lines (3,5,7,9,11), and the correct value reported for the number of iterations of the while loop in line 11, and finally
- 8 points per correct implementation of a subfunction (5 subfunctions times 8 gives 40 points).

Note 1. If you do not suppress the output, or your code prints something else that is not specified in the fprintf statements of this assignment you will be losing 10pts for every piece of unspecified output or missing semi-colon!