Math 365 **Homework 2** **Due Sep. 21, 2016**
Grady Wright

1. (**Computing π**, 25pts) The history of computing π is a small, but important part of the history of mathematics that involves many of the greatest mathematicians the world has known. There are many amazing formulas for computing π including

$$\frac{\pi}{4} = \sum_{j=0}^{\infty} \frac{(-1)^j}{2j+1}, \tag{1}$$

which was used by James Gregory in 1671 to compute many of π's digits. Another interesting infinite series involving π is

$$(\log 2)^2 + \sum_{k=1}^{\infty} \frac{1}{k^2 2^{k-1}} = \frac{\pi^2}{6}, \tag{2}$$

which Leonhard Euler (who was the closest thing this world may have seen to a human calculating machine) proved around 1735.

None of the above series expansions for π, however, compare to the following one discovered by John Machin in 1706:

$$\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \left[4\left(\frac{1}{5}\right)^{2k+1} - \left(\frac{1}{239}\right)^{2k+1} \right] = \frac{\pi}{4}. \tag{3}$$

This formula is particularly nice since it only involves rational numbers and the second term in the sum decays to zero very rapidly. With this formula Machin was able to compute π to 100 digits (the most ever computed at the time).

Write a MATLAB function for approximating π using Machin's series (3). Your function should take as input the total number of terms to sum in the series series and output an approximation to π. Using this function do the following:

   (a) Compute an approximation to π using $N = 1, 2, \ldots, 16$ and report the results in a table.

   (b) For each value of N, compute the absolute error in the approximation using the built-in constant for π from MATLAB and report the results in the same table.

   (c) Plot the absolute error in the approximations as a function of the number of terms used in the sum N. Use a log scale on y axis in your plot to obtain a nice picture of the error curve. Label your plot and include it in your assignment.

2. (**Continued fractions**, 25 pts) Continued fractions take the following form

$$x = a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4 + \cfrac{1}{a_5 + \cfrac{1}{\ddots + \cfrac{1}{a_n}}}}}}, \tag{4}$$

where a_1 is an integer and $a_2, a_3, \ldots, a_n$ are positive integers. Many of the most famous irrational numbers can be expressed as infinite continued fraction expansions. Truncating these expansions after a relatively few terms often gives an excellent approximation to the irrational number.

(a) Write a function in MATLAB that computes the continued fraction expansion for a given array (or vector) of numbers a_j, $j = 1, 2, \ldots, n$. Your code should take as input an array containing the continued fraction coefficients a_j and return the value x using formula in (4).

(b) The continued fraction coefficients for $\sqrt{2}$ are $a_1 = 1$ and $a_j = 2$, for $j = 2, 3, \ldots$. Create an array containing these numbers for $n = 20$ and use your function from part (a) to approximate $\sqrt{2}$. Report the approximation you obtain to 16 digits and report the absolute error in the approximation.

(c) Use the *Online Encyclopedia of Integer Sequences* to find the integer sequence A003417 for the continued fraction coefficients that can be used to compute Euler's number e. Use these coefficients with your function from part (a) to approximate e to full machine precision (16 digits). Report your result and report the minimum n you need to use to obtain the approximation to full machine precision.

3. (**Newton's method**, 15pts) The almost universally used algorithm to compute $\sqrt{a}$, where $a > 0$, is the recursion

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right), \tag{5}$$

easily obtained by means of Newton's method for the function $f(x) = x^2 - a$. One potential problem with this method is that it requires a floating point division, which not all computer processors support, or which may too expensive for a particular application. For these reasons, it is advantageous to devise a method for computing the square root that only uses addition, subtraction, multiplication, and division by 2 (which can be easily done by shifting the binary representation one bit to the right). The trick for doing this is to use Newton's method to compute $\frac{1}{\sqrt{a}}$, and then obtain $\sqrt{a}$ by multipling by a. Write down your recursion formula for computing $\frac{1}{\sqrt{a}}$ in a manner similar to (5). This formula should only involve addition/subtraction, multiplication and division by 2.

Try you algorithm on the problem of computing $\sqrt{5}$. As an initial guess use $x_0 = 0.5$. Report the values of x_0, x_1,..., x_5 in a *nice* table and verify that your algorithm is working by comparing these numbers to the true value of $\sqrt{5}$.

To see where this sort of software assisted acceleration is used in gaming, see the course webpage for a link to the article : *Origin of Quake3's Fast InvSqrt()*.

4. (**Using `fzero`**, 15pts)

(a) Consider the Colebrook equation for the friction factor in a fully developed pipe flow

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\frac{\epsilon/D}{3.7} + \frac{2.51}{\mathrm{Re}_D\sqrt{f}}\right)$$

where f is the Darcy friction factor, ϵ/D is the relative roughness of the pipe material, and Re is the Reynolds number based on a pipe diameter D. Use `fzero` to find the friction factor f corresponding to parameter values $\epsilon/D = 0.0001$ and $\mathrm{Re}_D = 3 \times 10^5$. Use a tolerance 10^{-8} with `fzero`. In your homework write-up give the value of f as well as the code you used to solve the problem. Do not include the `fzero` code just how you called it.

Hint : Use the function `optimset` to set up the tolerance `TolX`.

(b) David Peters (SIAM Review, 1997) obtains the following equation for the optimum damping ratio of a spring-mass-damper system designed to minimize the transmitted force when an impact is applied to the mass:

$$\cos\left[4\zeta\sqrt{1-\zeta^2}\right] = -1 + 8\zeta^2 - 8\zeta^4$$

Use `fzero` to find the $\zeta \in [0, 0.5]$ that satisfies this equation with a tolerance 10^{-12}. In your homework write-up give the value of ζ as well as the code you used to solve the problem. Do not include the `fzero` code just how you called it.

5. (**Freezing water mains**, 10pts) NCM 4.16. The function erf in this problem is called the *error function* and is available in MATLAB using `erf`. Also, you can use `fzero` instead of `fzerotx` to solve this problem.

6. (**Matrix computations**, 10pts) Matrices of size m-by-n with random entries can be created in MATLAB with the command `rand(m,n)` (note that if $m = n$ then the command `rand(n)` can be used). Create four random matrices A, B, C, and D, such that A is 2-by-3, B is 3-by-3, C is 3-by-2, and D is 3-by-3. Perform the following operations on these matrices using MATLAB . If the operation cannot be performed, explain (mathematically) why not; do not just give the output from the MATLAB command window.

(a) $2A + C$	(b) $C - 3B$	(c) $3B - 2D$
(d) AD	(e) CA	(f) AC
(g) BD	(h) DB	(i) BC
(j) CB	(k) DAB	(l) $2D^T + B$
(m) D^2	(n) A^2	(o) $C^T D$
(p) BA^T	(q) $-2A^T + 5C$	(r) $B^T + AD$
(s) $\frac{1}{2}(B + B^T)$	(t) $\frac{1}{2}(B - B^T)$	(u) CC^T
(v) $C^T C$	(w) $(CC^T)^{-1}$	(x) $(C^T C)^{-1}$
(y) $B(AD)^T$	(z) ADB^T	

Note that C^T means the transpose of the matrix (see MATLAB function `transpose` or just use the single quote character '). Also, none of the above operations involving multiplication or squaring are to be interpreted as elementwise operations

Do not turn in the individual output of each of the valid operations, simply turn in the MATLAB statements you used for each of the operations.