MTH 655, Winter 2013, LAB1

The goal of this assignment is to become familiar with the programming environment, to explore the concept of rate of convergence, memory limitations, and of computational complexity. Also, to get started on iterative methods.

TURN IN 1,2,5 and use 3-4 for warm-up.

Background: Consider a sequence of iterates $\{x_n\}$ converging to some number x. That is, the sequence of errors in the n-th iteration $\varepsilon_n := |x - x_n|$ converges to 0.

We say that this iteration has q-convergence rate α if $\varepsilon_{n+1} \leq C\varepsilon_n^\alpha$. C is called a factor.[1] **Note:** with $\alpha = 1$ (and $C < 1$), the convergence is *linear*, $\alpha = 2$, the convergence is *quadratic*. See [Kelley, Chapter 4.1] for additional definitions and conditions.

A good estimate for α can be provided by calculating

$$\alpha \approx \frac{\log(\varepsilon_{n+1}) - \log(\varepsilon_n)}{\log(\varepsilon_n) - \log(\varepsilon_{n-1})}$$

for a few subsequent iterations. When x_n is close to x, the calculated ratio should be close to α.

---

1. Define

- $a_k = 1 + \frac{1}{2^k}$.
- $b_k = 3 + \frac{1}{k!}$.
- $c_1 = 2$; for $k > 1 : c_k = \frac{c_{k-1}^2 + 1}{2c_{k-1}}$
- $d_1 = 2$; for $k > 1 : d_k = d_{k-1} - \frac{d_{k-1}^2 - 1}{4}$

Explore: for each sequence, find its limit and convergence rate, if any.

MATH Hint: Note $a_{k+1} - 1 = \frac{1}{2}(a_k - 1)$. Interpret c), d) as Newton iterations.

Choose some good n here: watch for underflow. (For some problems the convergence will be neither linear nor quadratic, see the text by [Kelley] to identify what kind it is).

---

2. **Explore:** The following formulas come from calculus

$$\frac{\pi}{4} = \sum_{j=1}^{\infty} (-1)^{j+1} \frac{1}{2j-1} = 1 - \frac{1}{3} + \frac{1}{5} - \ldots, \tag{1}$$

$$\frac{\pi^2}{6} = \sum_{j=1}^{\infty} \frac{1}{j^2}. \tag{2}$$

Below use one of these (or both) to approximate π.

Of course, on the computer we can only sum a finite number of terms, say N. The algorithm can be implemented in two ways: a) by setting up an array of values $a(j)$ and then summing those (use MATLAB's `sum` or your own function: how good is it ?), or b) without the array, by summing the values in the loop directly. Also, you have to consider multiplication, e.g., by 4 (where should it go in the code ?).

i) Implement both variants of the algorithm and test its rate of convergence by considering $N = 10, 100, 1000, \ldots$ (what is the largest N you can use for a) and b) ?

ii) Discuss the complexity of the algorithm and time (`tic toc`) the execution of the loop in which you add numbers (not the one in which you set up the array). To get reasonable timings, perhaps that loop should be repeated, for example M=100 times. **Explain**: how many floating point operations per loop have to be performed in a) and b) ? In variant a), you additionally have to account for the time to

[1] (Note: in some other sources α is called the order and C the rate. Sorry, I do not explain it, I just report it.)

access the memory. How do times for a) and b) compare ? What is the work/memory ratio for this algorithm ? (we define the work/memory ratio as the ratio between the number of operations that have to be performed divided by the number of memory locations referenced).
iii) Estimate the performance of the computer you in terms of FLOPS (floating point operations per second), that is, the total number of floating point operations that need to be performed, divided by the time in seconds. Try it on this computer and on some other computers you have access to.

---

3. Fixed point iteration: solve

$$x = \frac{1}{2}\cos(x) \tag{3}$$

by simultaneous iteration. First iterate until you find the solution (around 0.45), next test the q-convergence rate.

For fixed point method a good *estimate* for ε_n follows from Aitken's formula $\varepsilon_n \approx \frac{\lambda_n}{1-\lambda_n}|x_n - x_{n-1}|$, where λ can be approximated with $\frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}$. Test it.

---

4. Newton's algorithm: use an M-file as provided by a link from lab page, or some other code you write (then provide code).

Solve (3) with Newton's algorithm. Test the q-convergence rate. **Explore:** various tolerances and initial guesses. Determine experimentally the interval around the true solution such that if initial guess is included in it, the convergence will be quadratic. **Explain** using information in class (forthcoming) and theory from [Kelley, Chapter 5].

For Newton's method a good *estimate* for ε_{n+1} is $\varepsilon_{n+1} \approx |x_{n+1} - x_n|$. Test it.

---

5. Repeat 3-4 for $x = cos(x)$, and $x^2 - 3x + 2 = 0$. **Explain and explore**.