0 Instructions

This problem set is **due on Tuesday, Sept. 24 by midnight**. Submit your solutions on Canvas. Include the names of everyone you worked with on this problem set. Include any code you used to solve the problems as part of your submission.

1 Practice

For these problems, we will use the function

$$f(x) = x^{23} + e^{\sin(x)} + \ln(x).$$

Problem 1.1. Use three iterations of bisection to approximate the zero of $f(x)$. Be sure to explain your choice of starting interval.

Problem 1.2. For the same $f(x)$ as above, use three iterations of fixed point iteration with the function $g(x) = x - f(x)$ and starting guess $x_0 = 0.3$. Would you be able to perform a fourth iteration?

Problem 1.3. For the same $f(x)$ as above, use three iteration of Newton's method with starting guess $x_0 = 0.3$.

Problem 1.4. Explain the difference in performance between the second and third approaches.

2 Bisection

Recall the bisection algorithm for finding zeroes of a given function $f(x)$ described in class.

1. Initialize a_0, b_0 such that $f(a_0)f(b_0) < 0$.
2. Set $m_i = \frac{a_{i-1}+b_{i-1}}{2}$ and update $a_i = m_i, b_i = b_{i-1}$ if $f(m_i)f(b_{i-1}) < 0$ or $a_i = a_{i-1}, b_i = m_i$ if $f(a_{i-1})f(m_i) < 0$.
3. Iterate step 2 until reaching a desired stopping criterion. For this problem, we will use the stopping criterion of iterating until the error is acceptably low.

Problem 2.1. Implement the bisection algorithm described above. It should take the following inputs

1. f: function to find zeroes of
2. a: initial left endpoint of interval containing zero of f
3. b: initial right endpoint of interval containing zero of f
4. e: desired error tolerance for stopping criterion

and return the approximate zero that was found using bisection along with a reasonable upper bound on the error of the approximation. Feel free to adapt the code presented in class.

Problem 2.2. Use your code to approximate all zeroes of

$$f(x) = \frac{1}{x} + \ln(x) - 2$$

with an error of at most 2^{-10}. How many steps does your algorithm take to converge? What is the minimum error you can achieve for this approximation?

3 Approximating π

On the last homework, we represented π using an infinite sum

$$S = \sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6} \implies \pi = \sqrt{6S}.$$

This gives a method for approximating π by taking partial sums. However, this approximation is inherently inaccurate and can only deliver an estimate that is accurate to within roughly $\sqrt{\varepsilon}$, where $\varepsilon = 2^{-52}$. Even worse, this required approximately $n = \frac{1}{\sqrt{\varepsilon}} = 2^{26}$ terms of the sum.

Problem 3.1. Use your bisection algorithm from the previous part to approximate π as the zero of some function $f(x)$. How many steps does your bisection take to converge? What is the error of your approximation?

Problem 3.2. Construct a fixed point problem to approximate π as a zero of $f(x) = \sin(x)$. Use Python to run a few iterations manually. (You should be able to get convergence in 3 iterations.) What is the error of your approximation? Explain why fixed point iteration was so effective here.

Approximating π using the partial sums S_n requires an enormous amount of computation. In contrast, the approximation using bisection was vastly more efficient and the fixed point approximation even more so.

Problem 3.3. Which steps in bisection and fixed point iteration are the most computationally expensive in approximating π? If we enlarged floating point numbers (e.g. from 64 bits to 128 bits or more), would these approaches be just as efficient at obtaining good approximations? Explain why or why not.

Analyzing the error in fixed point iteration is somewhat subtle. In fixed point iteration, we iteratively compute $x_{n+1} = g(x_n)$ to try to converge to a fixed point x. This makes sense mathematically, but computationally speaking, the best we could hope for is $x_{n+1} = fl(g(x_n))$, which might accumulate floating point errors over time. To see what happens, we'll study the properties of the floating point fixed point iterations using the sequence $y_{n+1} = fl(g(y_n))$ with $y_0 = fl(x_0)$.

Problem 3.4. Suppose that $g(x)$ is a function satisfying the requirements for fixed point iteration and has a fixed point x. (Differentiable on $[a, b]$, $g(x) \in [a, b]$ for $x \in [a, b]$, and $|g'| < 1$ on $[a, b]$.) If we further assume that $|g'| < \frac{1}{2}$, show that the relative errors

$$r_n = \frac{|y_n - x|}{|x|}$$

satisfy the inequalities

$$r_{n+1} \leq \frac{1+\varepsilon}{2} r_n + \varepsilon.$$

Problem 3.5. Let $\theta = \frac{1+\varepsilon}{2}$. Using the inequalities from problem 2.4, show that

$$r_{n+1} \leq \theta^{n+1} r_0 + \frac{1}{1-\theta}\varepsilon \approx 2\varepsilon$$

for n sufficiently large.

This problem tells us that this method of approximating π can (very quickly) get us an estimate with a relative error of approximately 2ε, a great improvement over $\sqrt{\varepsilon}$. As we've seen before, blindly throwing more computational power at a problem won't always lead to improvements, and we actually need to make fundamental improvements in our techniques instead.