Numerical Methods for Digital Computers (Fall 2020) Homework: 3

ICSI 401 – Numerical Methods

Instructor: Abram Magner Date: Fall 2020

Please answer the following questions in complete sentences, showing all work (including code and output of programs, when applicable). I prefer that you type your solutions (e.g., using LaTeX, with Overleaf, TeXworks, etc., or Word), but will accept handwritten notes. If the grader cannot read your handwriting, then they cannot award you points.

Certain problems require you to write Matlab code. You should include in your submission a printout of all Matlab code that you write, in addition to the output that you get when you run it.

Due date: Tuesday, 10/27/2020, 11:59 p.m., on Blackboard.

3.1 More root finding and related topics

- This question will test your understanding of the intermediate value theorem, which you will recall is the theorem that motivates bisection search.

  Consider the following function $f(x)$:

  $$f(x) = \begin{cases} x & x < -1/2 \\ x+2 & x \geqslant -1/2 \end{cases} \tag{3.1}$$

  (This notation is standard and means that $f(x) = x$ whenever $x < -1/2$ and $f(x) = x + 2$ whenever $x \geqslant -1/2$.)

  Note that $f(x)$ is defined for every real x, but it has no roots. That is, there is no x_* such that $f(x_*) = 0$. Nonetheless, we can find an interval $[a, b]$ such that $f(a) < 0 < f(b)$: just choose $a = -1, b = 1$. Why can't we use the intermediate value theorem to conclude that f has a zero in the interval $[-1, 1]$?

- This question will test your understanding of the limitations of bisection search and Newton's method. Consider the function $f(x) = \frac{1}{2}|x|$.

  - Can we use bisection search to find one of its roots? Why or why not?
  - Can we use Newton's method to find one of its roots? Why or why not?

3.2 Fixed point iteration

- Complete problem 4.7.12 in the textbook. This will test your knowledge of what a fixed point of a function is, how to find them, and how to determine when iteration will converge to a fixed point.

- **Application question. Here's how this works: You get full points for a reasonable attempt. So you MUST attempt this question. You get extra credit if you're correct.** This question presents a simple model of population dynamics, and analyzes its equilibrium states. Models like this, but more complicated, can be used to predict global population trends, for instance.

  Let N_t denote the number of individuals in a population at time t. We will assume that N_t evolves at each time step according to the following equation:

  $$N_{t+1} = N_t + rN_t(1 - N_t/K), \tag{3.2}$$

  where

  - r is the birth minus death rate (per existing individual) parameter. Let us assume that it is larger than 0.
  - K is a positive constant representing fundamental resource limitations for the population. For example, on Earth, there is only a finite amount of of consumable biomass, and so the number of humans on Earth probably cannot grow to infinity. Note that when $N_t > K$, we have $N_{t+1} - N_t < 0$.

  Supposing we start with some initial population N_0, we can calculate N_t as follows:

  - Define $F(x) = x + rx(1 - x/K)$.
  - Define $x = N_0$.
  - Compute $N_t = F \circ F \circ ... \circ F(x)$, where F is applied t times.

  Now, we want to determine cases where this process converges. Suppose $r > 0$ and $K > 0$ are fixed.

  - **Determine all non-negative values of x for which F is a contraction.**

    Hints: Recall that we say that a function F is a contraction if its Lipschitz constant L is strictly less than 1. In other words:

    $$|F(z) - F(z')| \leqslant L \cdot |z - z'| \tag{3.3}$$

    for all $z, z' \geqslant 0$. Remember that you can get an upper bound on L by upper bounding $|F'(z)|$ for every z.

    You should get an answer of the form "$x > g(K)$" for some explicit function g that you have to determine.
  - Suppose that $x \leqslant K$. **Show that $F(x) \geqslant x$.**
  - Suppose that $x > K$. **Show that $F(x) < x$.**
  - What we've shown, then, is that F is a contraction on the interval $(g(K), \infty)$, and, furthermore, if we fix *any* $L > g(K)$, $U > K > L$, then F maps any value $y \in [L, U]$ to some value $F(y) \in [L, U]$. Thus, by Banach's fixed point theorem, we can conclude that F has a unique fixed point x_* in the interval $[L, U]$, and iterated application of F converges to x_*. Here, x_* is the limiting population size!

    Furthermore, since K is guaranteed to be in $[L, U]$, we see that $x_* = K$. That is, if we start at *any* positive population size $> g(K)$, the population will eventually converge to K (in fact, with more work, one can show that this happens for any positive initial population size). This makes intuitive sense: remember that K represents the resource constraints on the population, so this says that the population will converge to the capacity of its environment.

You don't have to do anything for this bullet point. Just make sure that you understand it!

3.3 Condition numbers and algorithmic stability

Recall that the relative condition number $\kappa(x)$ of a function $f(x)$ is approximately the factor by which a relative error in the input gets magnified in the relative error in the output. I.e., if x is perturbed to $\hat{x}$, then

$$\left|\frac{f(x) - f(\hat{x})}{f(x)}\right| \approx \kappa(x) \left|\frac{x - \hat{x}}{x}\right|. \tag{3.4}$$

We gave a formula for $\kappa(x)$ in class, and it also appears in the textbook.

- Qualitatively speaking, if the relative condition number of a function is large, does this make the function ill-conditioned, or well-conditioned (choose one)?
- Suppose that a problem has a very small condition number for a given input, but the relative error of the output of an algorithm for the problem is large. Is this the fault of the problem or of the algorithm (choose one)?
- Complete problem 6.3.5 on compound interest in the textbook, and make sure that you understand how they derived $\mathcal{I}_n(x)$. Also, note that $\lim_{n\to\infty} \mathcal{I}_n(x) = e^x$.

  For part (d), demonstrate your method in Matlab.

3.4 Some numerical linear algebra

- This problem will teach you how to work with the LU decomposition of a matrix programmatically to solve a linear system.

  The Matlab function `lu(A)` returns `[L, U, P]`, where L is a lower triangular matrix, U is an upper triangular matrix, and P is a permutation matrix, such that

  $$A = P^T LU. \tag{3.5}$$

  Complete the following code to produce a solution to the equation $Ax = b$, *without multiplying the input matrices.*

```
function x = solve_with_LU(L, U, P, b)
%
% Given a lower triangular matrix L, an upper triangular matrix U,
% a permutation matrix P, and a vector b,
% return a solution to the equation P'LUx = b.
%
```

```
    z = P'\b
    y = %FILL THIS IN!
    x = %FILL THIS IN!
end
```

- In Matlab, compute the matrices P, L, and U from the LU decomposition of the matrix

$$A = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix} \tag{3.6}$$

  and use the completed function above to obtain the solution to the system $Ax = b$, where

$$b = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{3.7}$$

Please note that you have to turn in Matlab code for this question, including the completed version of the function above.