AMSC460: Homework 1

James Zhang[*]

September 10, 2024

3. (a) How many distinct positive numbers can be represented in a floating point system using base $\beta = 10$, precision $t = 2$ and exponent range $L = -9$, $U = 10$?
(Assume normalized fractions and don't worry about underflow.)

Solution. In general, a floating point number can be expressed in the representation

$$fl(x) = \pm \left(\frac{\tilde{d}_0}{\beta^0} + \frac{\tilde{d}_1}{\beta^1} + \cdots + \frac{\tilde{d}_{t-1}}{\beta^{t-1}} \right) \times \beta^e$$

The problem statement specifies that we are looking for positive integers, $t = 2$, $\beta = 10$, and e is bounded by -9 and 10. Applying this information, we now have the more specific representation

$$fl(x) = + \left(\tilde{d}_0 + \frac{\tilde{d}_1}{10} \right) \times 10^e$$

Since we assume normalized fractions, $\tilde{d}_0 \neq 0$, so it can attain the digits $1 - 9$, or 9 possibilties. $\tilde{d}_1$ can be any digit, so 10 possibilities. Finally, e can be any number from -9 to 10, so 20 possibilities. Multiplying these together yields

$$9 \times 10 \times 20 = 1800 \text{ distinct positive integers}$$

□

[*]Email: jzhang72@terpmail.umd.edu

13. Consider the linear system

$$\begin{pmatrix} a & b \\ b & a \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

with $a, b > 0$; $a \neq b$.

(a) If $a \approx b$, what is the numerical difficulty in solving this linear system?

Solution. To solve this system, if the square matrix is invertble, we would invert the matrix and solve for the vector $\begin{pmatrix} x & y \end{pmatrix}^T$. By the Invertible Matrix Theorem, one of the conditioning for checking if a matrix is invertible is if its determinant is nonzero. Note that the determinant of the square matrix is

$$\det \begin{pmatrix} a & b \\ b & a \end{pmatrix} = a^2 - b^2 = (a-b)(a+b)$$

If $a \approx b$, and specifically if we take the limit $a - b \to 0$,

$$\lim_{a-b\to 0} \det \begin{pmatrix} a & b \\ b & a \end{pmatrix} = 0$$

because the $a - b$ approaches 0. Therefore, the matrix is almost singular and so the system is very ill-condtioned, meaning the system output is sensititve to small changes in coefficients and that small errors in arithmetic will get quickly propagated throughout the calculations. □

1. Apply the bisection routine `bisect` to find the root of the function

$$f(x) = \sqrt{x} - 1.1$$

starting from the interval $[0,2]$ (that is, $a = 0$ and $b = 2$), with `atol` = 1.e-8.

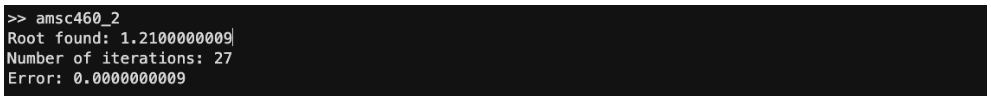
(a) How many iterations are required? Does the iteration count match the expectations, based on our convergence analysis?

(b) What is the resulting absolute error? Could this absolute error be predicted by our convergence analysis?

Solution.

(a) Let us apply the Bisection Method

```
function [root, iter] = bisection_sqrt()
  % Define the function
  f = @(x) sqrt(x) - 1.1;

  % Set the tolerance and initial interval [a, b]
  atol = 1e-8;
  a = 0;
  b = 2; % Initial guess for the root search range

  % Check if the interval is valid
  if f(a) * f(b) > 0
      error('f(a) and f(b) must have opposite signs');
  end

  iter = 0; % Counter for number of iterations

  % Bisection method loop
  while (b - a) / 2 > atol
      iter = iter + 1;
      c = (a + b) / 2; % Midpoint of interval
      if f(c) == 0
          break; % We've found the exact root
      elseif f(a) * f(c) < 0
          b = c; % Root lies in the left subinterval
      else
          a = c; % Root lies in the right subinterval
      end
  end

root = (a + b) / 2; % Approximate root

% Display the result
fprintf('Root found: %.10f\n', root);
fprintf('Number of iterations: %d\n', iter);
fprintf('Error: %.10f\n', root - 1.21)
end
```



```
>> amsc460_2
Root found: 1.2100000009
Number of iterations: 27
Error: 0.0000000009
```

We can compute the expected number of iterations as

$$\text{Expected Iterations} = \text{ceil}\left(\frac{\log(b-a) - \log(2 * \text{atol})}{\log 2}\right) = \text{ceil}(26.57) = 27$$

and this matches with our true number of iterations, 27.

(b) The resulting absolute error is $0.0000000009 = 9 \times 10^{-10} < \text{atol} = 1 \times 10^{-8}$. Using a convergence analysis, we know that the error is always less than half the length of the current interval. In words, after the first interval, we know that our error will be less than $\frac{1}{2} = 2^{-1}$. After the second iteration, the error will be less than $\frac{1}{4} = 2^{-2}$. To generalize this, after the i-th iteration, our error will be less than 2^{-i}. Since we computed that the bisection method will have an expected number of iterations equal to 28, we can say that our error is upper bounded by

$$\text{Predicted error} = 2^{-28} = \frac{1}{2^{28}} \approx 0.00000000372529 \approx 3.72529 \times 10^{-9}$$

and we can correctly observe that

$$\begin{aligned} \text{True error} &\leq \text{Predicted Error} \leq \text{atol} \\ 9 \times 10^{-10} &\leq 3.72529 \times 10^{-9} \leq 1 \times 10^{-8} \end{aligned}$$

$\square$

2. Consider the polynomial function[8]

$$\begin{aligned} f(x) &= (x-2)^9 \\ &= x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 \\ &\quad + 2304x - 512. \end{aligned}$$

   (a) Write a MATLAB script which evaluates this function at 161 equidistant points in the interval $[1.92, 2.08]$ using two methods:

      i. Apply nested evaluation (cf. Example 1.4) for evaluating the polynomial in the expanded form $x^9 - 18x^8 + \cdots$.

      ii. Calculate $(x-2)^9$ directly.

      Plot the results in two separate figures.

   (b) Explain the difference between the two graphs.

   (c) Suppose you were to apply the bisection routine from Section 3.2 to find a root of this function, starting from the interval $[1.92, 2.08]$ and using the nested evaluation method, to an absolute tolerance 10^{-6}. *Without computing anything*, select the correct outcome:

      i. The routine will terminate with a root p satisfying $|p-2| \le 10^{-6}$.

      ii. The routine will terminate with a root p *not* satisfying $|p-2| \le 10^{-6}$.

      iii. The routine will not find a root.

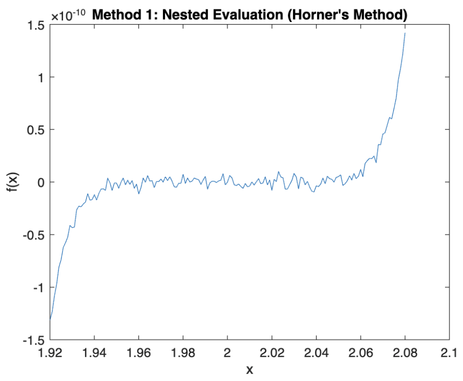
      Justify your choice in one short sentence.

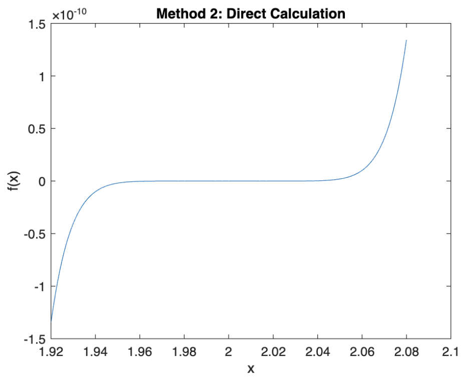
Solution.

(a) Below is my Matlab code

```
% Define the polynomial coefficients
c = [1, -18, 144, -672, 2016, -4032, 5376, -4608, 2304, -512];
% Nested evaluation function (Horner's method)
function y = nested_eval(x, c)
    n = length(c) - 1;
    y = c(1);
    for j = 2:n+1
        y = y.*x + c(j);
    end
end
% Direct calculation function
f2 = @(x) (x - 2).^9;
% Generate 161 equidistant points in [1.92, 2.08]
x = linspace(1.92, 2.08, 161);
% Evaluate the function using both methods
y1 = nested_eval(x, c);
y2 = f2(x);
% Plot the results
figure(1)
plot(x, y1)
title('Method 1: Nested Evaluation (Horner''s Method)')
xlabel('x')
ylabel('f(x)')
figure(2)
plot(x, y2)
title('Method 2: Direct Calculation')
xlabel('x')
```

```
ylabel('f(x)')
```





(b) The graph of Method 2 is far more smooth and symmetric about $x = 2$ then Method 1, which shows many more oscillations due to the numerical instability of the method. Method 1 involves multiple arithmetic operations with large and small numbers. Each operation introduces small rounding errors due to the finite precision of floating-point arithmetic in computers. Furthermore, in the expanded form, errors from calculations propagate and amplify through subsequent operations. In direct calculation, the output is more numerically stable because it avoids the intermediate large numbers and repeated operations that cause issues in Method 1.

(c) In the case where we use the textbook code for Bisection Method, I would select choice *ii* because in that code, there is no check for if we have found the exact root ($f(p) == 0$), so it will continuously make the interval smaller, and since Nested Evaluation is numerically unstable here, it is likely that Bisection Method converges to a root that does not satisfy the provided atol $= 1 \times 10^{-6}$. However, in a more efficient implementation of Bisection Method, I propose the following alternative answer of Choice *i*.

Choice *i*: despite the Nested Evaluation method being more numerically unstable than Direct Evaluation, let us actually think about if we were to proceed using the Bisection Method. In the first iteration $a = 1.92, b = 2.08 \implies p = 2$ (which is the exact root of $f(x) = (x-2)^9$) and since 2 is an integer and the terms in the polynomial will not result in integer overflow, there will be no floating point or rounding errors, and the Bisection Method will find the exact root $p = 2$ in the first iteration, satisfying the tolerance condition perfectly.

$\square$