MANE 3351

Lecture 9

Classroom Management

Agenda

- Part 2: Bisection Search Error Analysis
- False Position Method
- Review Test 1
- Lab 4
- LANL visit - I will be out of my office for most of today
  - One day extension on Lab 3

Resources

Handouts

- Lecture 9 Slides
- Lecture 9 Marked Slides

Assignments

- Homework 1 (assigned 9/22/2025, due 9/29/2025 (before 11:59 PM)
- Homework 2 (assigned 9/24/2025, due 10/1/2025 (before 9:30 AM - no late submissions)
- Lab 3 (assigned 9/24/2025, due 10/1/2025 (before 9:30 AM))
- Read textbook pages 41-46

Schedule

Lecture/Lab	Date	Topic
7	9/24	Taylor Series, Homework 2 (due 10/1 - no late work), Lab 3 (due 10/1)
8	9/29	Roots of Equations, bisection method (not on Test 1)
9	10/1	Bisection Method Error Analysis, False Position (not on Test 1)
10	10/6	Test 1 (lectures 1-7)

→ 8:15 in regular classroom

Bisection Method: Error Analysis

Bisection Method Theorem

Cheney and Kincaid (2004)[^1] provide a definition of the bisection method

> If the bisection algorithm is applied to a continuous function on an interval $[a, b]$, where $f(a)f(b) < 0$, then, after n steps, an approximate root will have been computed with error at most $(b - a)/2^{n+1}$.

This definition provides the following useful results:

- $|e_n| \leq \frac{1}{2^{n+1}}(b - a)$
- $n > \frac{\log(b-a) - \log 2\varepsilon}{\log 2}$

Percent Relative Error

Chapra and Canale (2015) [^2] provide a formula for the approximate percent relative error

- $\varepsilon_a = \left|\frac{x_r^{new} - x_r^{old}}{x_r^{new}}\right| \times 100\%$ where x_r^{new} is the root for the present iteration and x_r^{old} is the root from the previous iteration

Bracketing Method

[l, u]

False Position

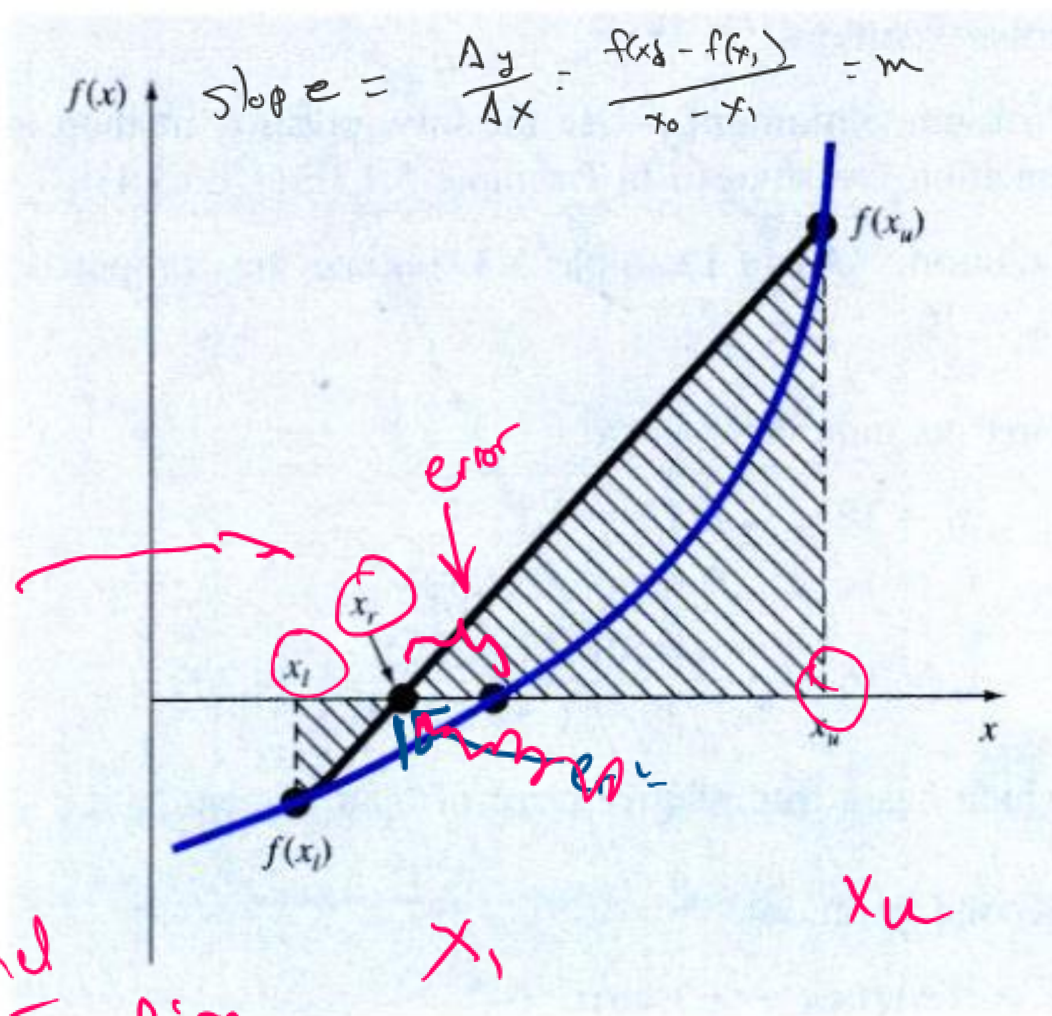
- Attempts to converge faster than bisection method by using functional information
- The new root is found by

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

$= x_u - \frac{f(x_u)}{m}$

– Figure 5.12 (Chapra and Canale(2015)[^2] illustrates the false position method False Position Method

FIGURE 5.12
A graphical depiction of the method of false position. Similar triangles used to derive the formula for the method are shaded.

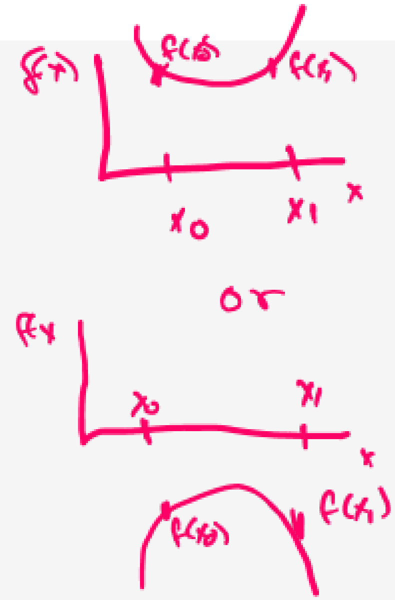


False Position Code

The pseudocode, shown below was taken from [^3]

Complete Pseudocode for Regula Falsi or False Position Method

```
1. Start

2. Define function f(x)

3. Input
       a. Lower and Upper guesses x0 and x1
       b. tolerable error e

4. If f(x0)*f(x1) > 0
       print "Incorrect initial guesses"
       goto 3
   End If

5. Do

       x2 = x0 - ((x0-x1) * f(x0))/(f(x0) - f(x1))

       If f(x0)*f(x2) < 0
               x1 = x2
       Else
               x0 = x2
       End If

   While abs(f(x2) > e          → Repeat/until

6. Print root as x2

7. Stop
```

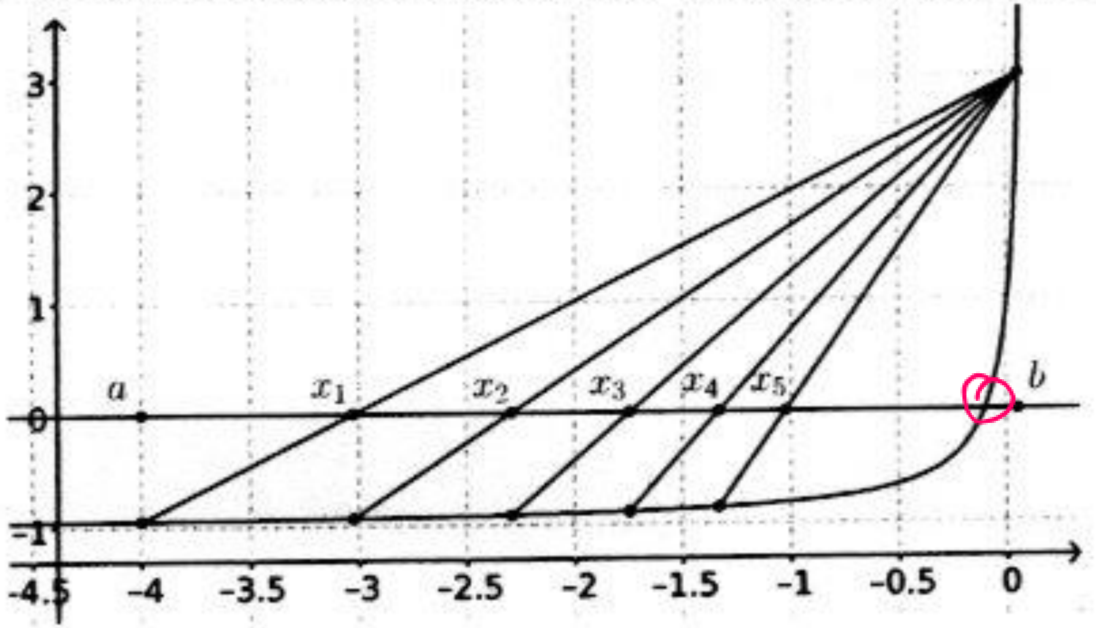


False Position Pseudocode

False Position Error Analysis

- – Much harder to analyze than bisection method
- – Convergence to root depends on shape of $f(x)$
- – At times, false position may converge slowly!, Example from textbook[^4], shown below

Figure 2.7.1: A troublesome function for the bracketed secant method.



False Position troublesome function

Loops in Python

- Python only supports two types of loops: for and while
- Pseudocode used do/while loop

Python Code for False Position

```python
# False Position
from scipy import stats
import math
def f(x):
    return stats.norm.cdf(x)-.25
# input a & b
while True:
    a=float(input("enter the lower bound: "))
    b=float(input("enter the upper bound: "))
    if f(a)*f(b)<0.0:
        break
    print("incorrect bounds, please re-enter")
print("the lower bound is {}".format(a))
print("the upper bound is {}".format(b))
counter=0
# process loop
while True:
    m= a - ((a-b)*f(a))/(f(a)-f(b))
    if f(a)*f(b)<0.0:
        b=m
    else:
        a=m
    counter=counter+1
    if math.fabs(f(m)) <0.0005:
        break
print("the root is {}".format(m))
print("the value at the root is {}".format(f(m)))
print("the number of steps were {}".format(counter))
```

entering a & b & checking that root is bracketed

start →

f(m)

end →

not in the while loop

Test One

- You are allowed one 4 inch by 6 inch notecard containing handwritten notes
  outside my office (shelf)
  front & back
- Calculator is needed; no programmable calculators that can calculate derivatives
- Covers material from Lectures 1 - 7
- Review Homework 1 and 2 and solutions
- Review Test 1
- Resistor information will be provided

2. (24 points) **Error Analysis**

Question 2 analyzes the effectiveness of a third-order Taylor series polynomial used to approximate $f(x) = 2\sin(3x)$. The Taylor series approximation is $T_3(x) = 6x - 9x^3$. **Make sure that your calculator is set to radians and not degrees when working this problem.**

(a) (8 points) What is the value of the absolute error for $f(1)$?

(b) (8 points) What is the value of the relative error for $f(1)$?

What is needed to calculate any error?

$f(x)$ 1. exact value

T_3 2. Approximate value

fall '24

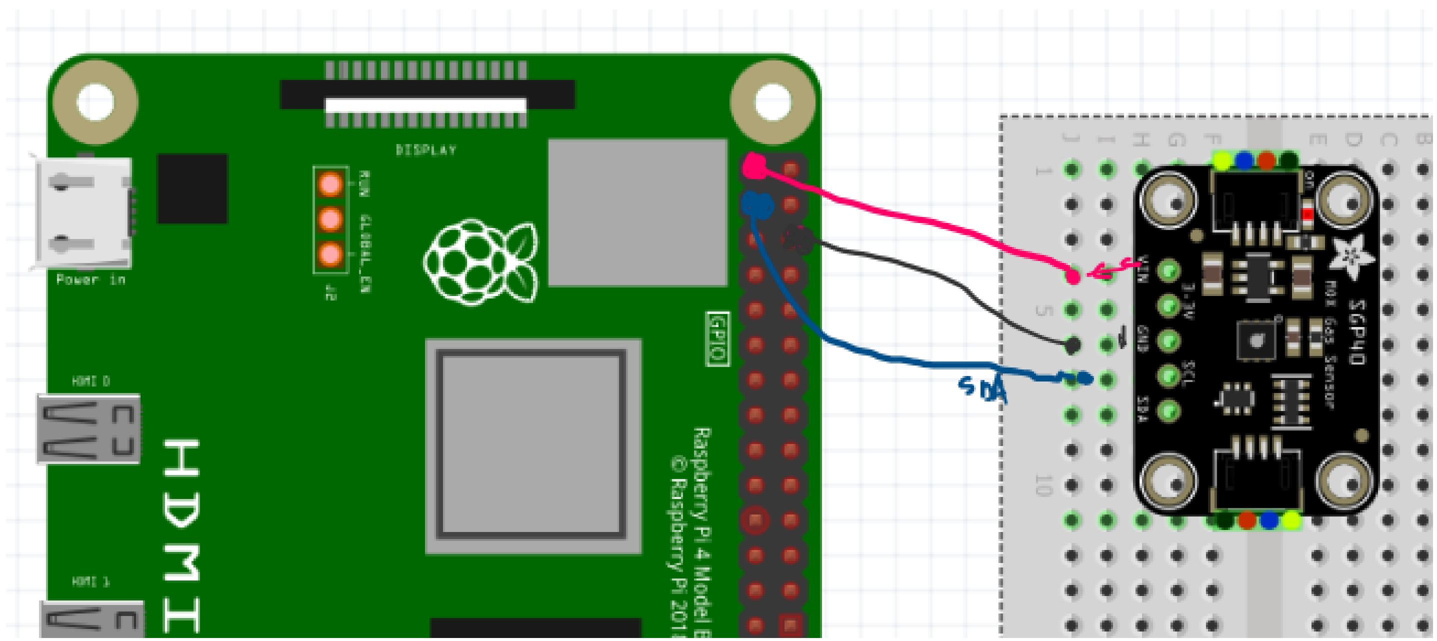
$f(x) = 4\cos(.5x)$ (cos → sin)

$f(x) \to$ 5th order polynomial

derivative $f(x) = 4e^{5x}$

Raspberry Pi	SGP40
3.3V	VIN
GND	GND
GPIO 2	SDA
GPIO 3	SCL

The details of the SGP40 may be hard to read. The pin names are
SCL, and SDA from top to bottom.



References

[1]: Cheney, W. and Kincaid, D. (2004), *Numerical Mathematics and Computing, 5th edition*

[2]: Chapra, S. and Canale, R. (2015), *Numerical Methods for Engineering, 7th edition*

[4]: Brin, L., (2020), *Tea Time Numerical Analysis: Experiences in Mathematics, 3rd edition*