Descartes’ Rule of Signs

Bernardo Dias, Asier Olano, Filippo Riva

Road map

DEFINITION AND RULE EXPLANATION

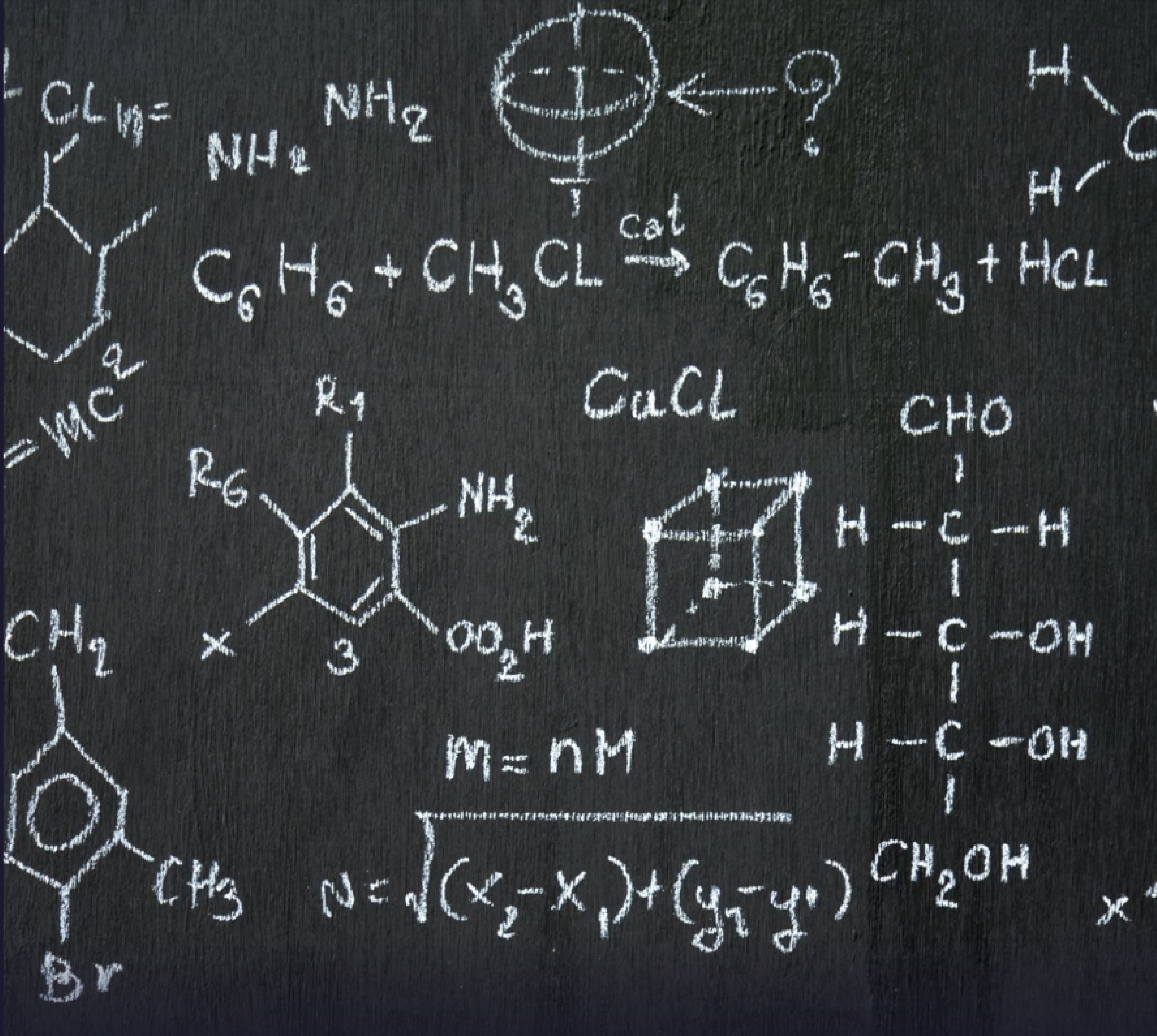
DEVELOPMENT OF THE ALGORITHM

GRAPHING

REAL WORLD APPLICATION

Definition

Descartes' Rule of Signs is a mathematical theorem used to determine the possible number of positive and negative real roots of a polynomial equation



Real Positive and Negative roots

Root Definition: A root of a function $f(x)$ is any value of x where $f(x)=0$

Positive Root:

•Occurs when $x>0$

•The function intersects the x-axis at a positive x-coordinate.

Negative Root:

•Occurs when $x<0$.

•The function intersects the x-axis at a negative x-coordinate.

How do we find real positive zeros?

The number of positive real zeros is equal to the number of sign changes in your function f(x), or, the number of sign changes in your function f(x) minus an even integer greater than 0

$$f(x) = +10x^6 + 7x^5 + 7x^3 - 8x^2 + 4x - 3$$

0 0 1 1 1

Change of signs = 3

Other possible results = 3 – 2 = 1

Real positive zeros = 3 , 1

How do we find real negative zeros?

The number of negative real zeros is equal to the number of sign changes in your negative function f(-x), or, the number of sign changes in your function minus an even integer greater than 0

$$f(x) = +10x^6 + 7x^5 + 7x^3 - 8x^2 + 4x - 3$$

$$f(-x) = +10x^6 - 7x^5 - 7x^3 - 8x^2 - 4x - 3$$

1 0 0 0 0

Change of signs = 1

Other possible results = 1 – 2 = - 1 → -1 < 0 not a possible result

Real negative zeros = 1

Road map

DEFINITION AND RULE EXPLANATION

DEVELOPMENT OF THE ALGORITHM

GRAPHING

REAL WORLD APPLICATION

STEP 1
How do we input a function in a list that we want to work with?

$$f(x) = +10\,x^6 + 7x^5 + 7x^3 - 8x^2 + 4x - 3$$

f = [-3, 4, -8, 7, 0, 7, 10]

STEP 2

Develop a function able to count how many times the signs are changing in a list

```python
def noc(f):
    j = 0
    c = 0
    f = f[:]  # Create a copy of the list to avoid modifying the original

    # Remove zeros from the list
    while j < len(f): # while loop to go through the list and deleting all zeros
        if f[j] == 0:
            del f[j]
        else:
            j = j + 1

    # Iterate through the list and check if the sign changes
    for i in range(1, len(f)):
        if f[i] * f[i-1] < 0:  # If the multiplication result is negative, the signs have changed!
            c = c + 1 #Increase the counter every time the sign changes

    return c
```

- Create a copy of the initial list
- Two counters
- Multiplication formula to decide if the sign is changing

$$f(x) = +10\,x^6 + 7x^5 + 7x^3 - 8x^2 + 4x - 3$$

\+ \+ - - -

STEP 3
Develop a second function that is able to give us the real positive and negative roots of any given function

Real positive roots:

```
def drs(f):
    c = noc(f)  # Get the count of sign changes for positive roots
    pr = [] #open teh list to store the positive roots

    # Subtract even numbers from c to get possible positive roots
    for i in range(0, c + 1, 2): # for loop going from 0 to c every 2(smallest even number)
        result = c - i
        if result >= 0:
            pr = pr + [result] #paste the results in the list
```

- Use the counter from the past function
- Create a loop that subtracts 2 to the counter to get the results
- Open a pr[] list where you store the results

Real negative roots:

```
# Now we simulate f(-x) by flipping the signs based on the index
negf = []
for i in range(len(f)):
    if i % 2 == 0:  # If the index is even, keep the value as it is
        negf = negf + [f[i]]
    else:  # If the index is odd, negate the value
        negf = negf + [-f[i]]

# Count the number of sign changes in the negated list (f(-x))
cneg = noc(negf)

nr = []
# Subtract even number     m cneg to get possible negative roots
for i in range(0, cneg      2):
    result = cneg - i
    if result >= 0:
        nr = nr + [res

return pr, nr  # Retur      h positive and negative roots
```

- Create a new list negf[] to store the f(-x)
- Function to obtain f(-x)
- Create a new counter
- Create a new list to store real negative roots

noc(f) → f = [-3, 4, -8, 7, 7, 10]

$+ 7x^4 + 7x^3 - 8x^2 + 4x - 3$

$f(-x) = -1 \ldots 7x^4 - 7x^3 - 8x^2 - 4x - 3$ → [2,0]

noc(negf) → f = [-3, 4, -8, 7, 0, 7, 10]

$f(x) = 10x^6 + 7x^5 + 7x^3 - 8x^2 + 4x - 3$

$f(-x) = 10x^6 - 7x^5 - 7x^3 - 8x^2 - 4x - 3$ → [1]

Road map

DEFINITION AND RULE EXPLANATION

DEVELOPMENT OF THE ALGORITHM

GRAPHING

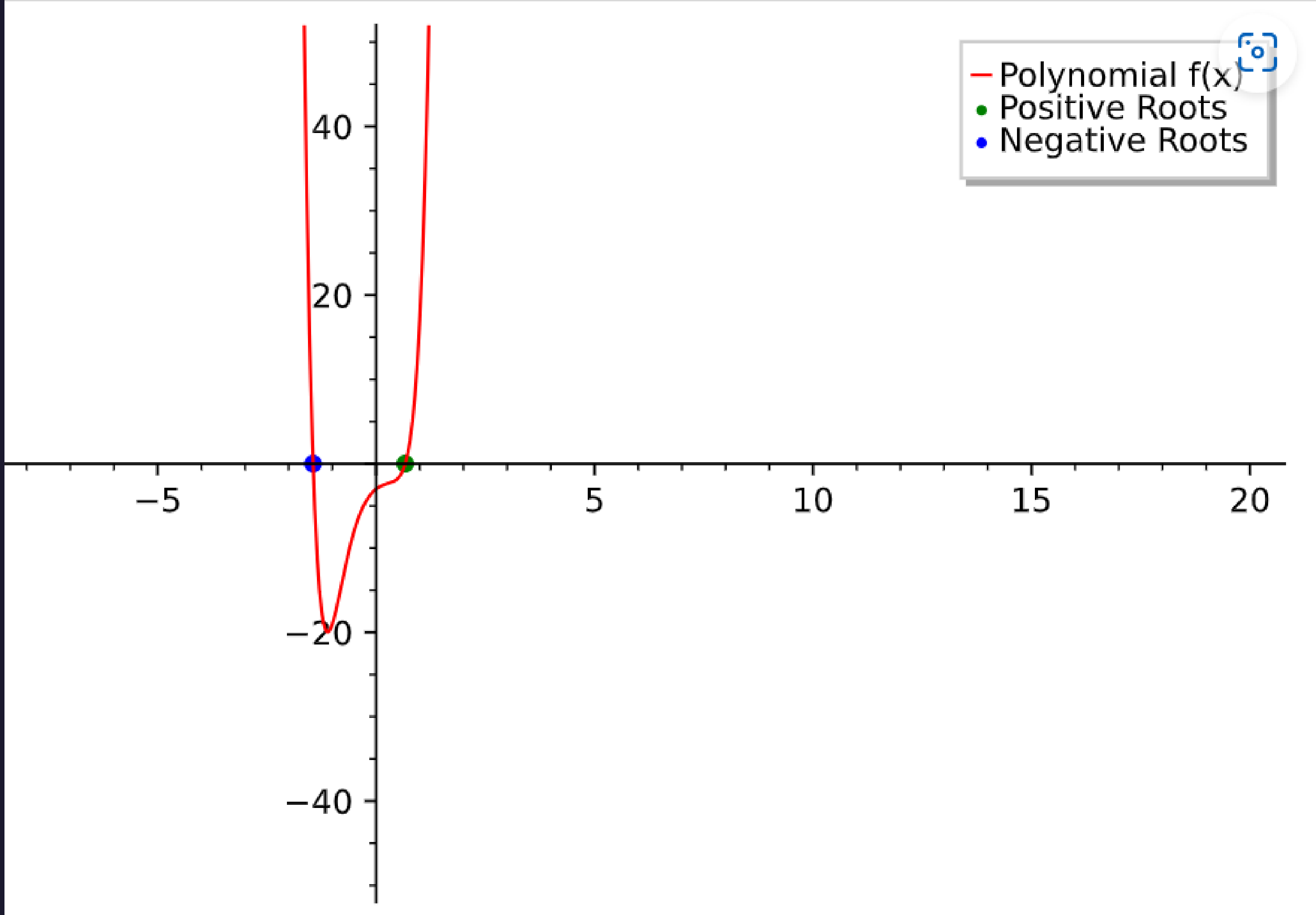
REAL WORLD APPLICATION

Graphing

```
x = var('x')
f = 10*x^6 + 7*x^5 + 7*x^3 - 8*x^2 + 4*x - 3

# Calculate the real roots of the polynomial
roots = f.roots(ring=RR, multiplicities=False)  # Find real roots

# Separate positive and negative roots
positiveroots = [r for r in roots if r > 0]  # Filter positive roots
negativeroots = [r for r in roots if r < 0]  # Filter negative roots

# Generate the polynomial plot over the interval [-20, 20]
plotpolynomial = plot(f, (x, -20, 20), ymin=-50, ymax=50, color="red", legend_label="Polynomial f(x)")

# Add points for positive and negative roots
plotpositivepoints = point([(r, f.subs(x=r)) for r in positive_roots], color="green", size=30, legend_label="Positive Roots")
plotnegativepoints = point([(r, f.subs(x=r)) for r in negative_roots], color="blue", size=30, legend_label="Negative Roots")

# Show the final plot with the points
show(plotpolynomial + plotpositivepoints + plotnegativepoints)
```

- *f.roots* function used to find and highlights the roots

Graphing



Polynomial f(x)
Positive Roots
Negative Roots
40
20
−20
−40
−5
5
10
15
20

Road map

DEFINITION AND RULE EXPLANATION

DEVELOPMENT OF THE ALGORITHM

GRAPHING

REAL WORLD APPLICATION

Business Profit Maximation

A company models its profit P(x), where x represents the number of units produced and sold, as a polynomial. P(x) represents the total profit when producing and selling x units.

When P(x) = 0, the profit is zero, therefore, the total revenue equals total costs, which is a break-even point. This helps the company calculate the minimum sales volume required to cover all expenses and start generating profit.

$$P(x) = 2x^4 - 5x^3 + 3x^2 - x + 7$$

The profit function P(x) has at most 4 positive roots (4, 2 or 0) (break-even production levels)