Calculating complex roots of a polynomial with real coefficients using the Bairstow's method.

# Application

Function *Bar* calculate all complex and real roots of given polynomial. User is provided a choose between 3 different representations: floating point, floating point converted to intervals, intervals.

# Description of the method

Having a polynomial we can divide it by quadratic equation . That way:

If , roots of quadratic equation are roots of polynomial.

Finding right quadratic equation is iterative way. It starts by performing first synthetic division of :

and again, performing a synthetic division on dividing by :

And

what also means that:

Similar for second division:

.

The quadratic evenly divides the polynomial when:

while and can be written as a function of and :

what allows using iterating Newton–Raphson method what can be briefly written as:

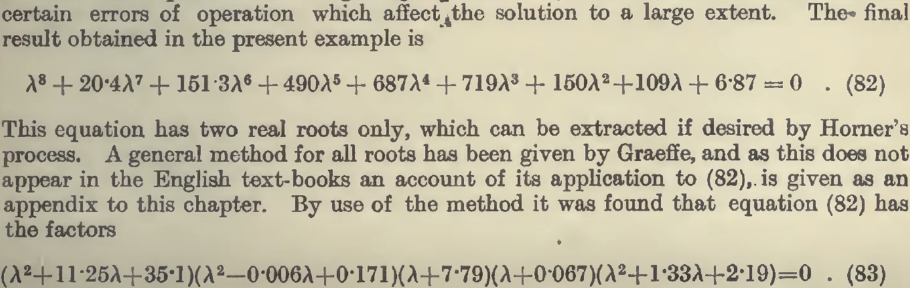
and

That way we find quadratic equation that solves the polynomial, then solve the quadratic equation to find roots :

with possible complex results.

That kind of iterations are performing as long as degree is bigger or equal 3.

This method was invented and firstly used by Leonard Bairstow in *Applied Aerodynamics* where he needs to find a root of polynomial that is 8th degree.



# Function call

*Bar (choice, precision, max\_iter, polynomial, real\_roots, complex\_roots)*

# Inputs

*choice* – mode of number calculation; ‘f’ for floating; ‘s’ for interval from single number; ‘i’ for interval calculation from interval input

*precision –* precision of comparison, -1 for maximum possible precision

*max\_iter -* maximum integration that will be performed, -1 for no limit

*polynomial* - polynomial coefficients starting from to, for ‘i’ mode 2 floating-point number, for ‘s’ and ‘f’ just floating-point number.

# Outputs

*real\_roots – array of real roots of given polynomial; ‘I’ and ‘s’ it is interval; ‘f’ floating-point number*

*complex\_roots – array of complex roots of given polynomial; ‘I’ and ‘s’ it is two intervals; ‘f’ floating-point numbers*

# 7. Data Type

Integer*: max\_iter*, *status*

*vector: polynomial, real\_roots, complex\_roots*

character*: choice*

long doule*: precision*

# Custom data type

*Vector polynomial- dynamic array of coefficients of a polynomial*

*Vector real\_roots- dynamic array of real roots*

*Vector complex\_roots –* dynamic array of dynamic array containing real and imaginary part of complex number

# Function code

All of the codes are publicly available here: <https://github.com/TMPkl/Bairstow-s>

# Example

1. Polynomial:

Input:

-1 <- set the precision

-1 <- set the maximum iteration

0.9 1.0

-2.1 -2

-5 -4.9

6 6

Output:

[3.052133758492346e+00, 4.148089245674311e+00] szerokość: 1.096e+00

[-1.935646440475890e+00, -8.396909532939260e-01] szerokość: 1.096e+00

[9.000000000000000e-01, 1.000000000000000e+00] szerokość: 1.000e-01

1. Polynimial:

Input:

1e-15 <- precision

-1 <- maximum iteration

1

10

34

38

-73

-160

150

Output:

1.000000001654560e+00

-3.000000000000000e+00

9.999999983454404e-01

-5.000000000000000e+00

[-2.000000000000000e+00 + 2.449489742783178e+00i]

[-2.000000000000000e+00 + -2.449489742783178e+00i]

c) Polynomial:

Input:  
1e-50

3

1

5

-10

100

Output:

(none output because iteration limit exceed)