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:

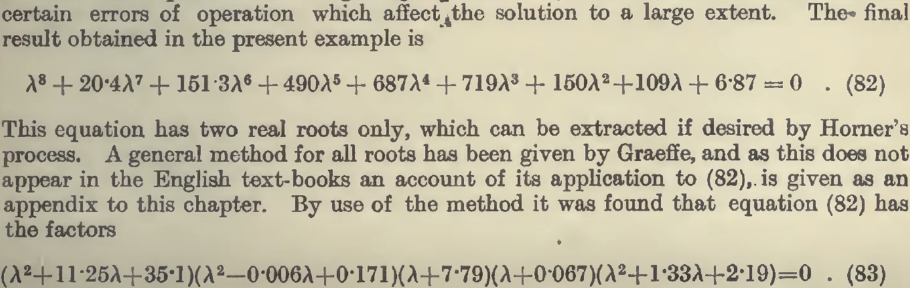
and

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, status)*

# 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*

# Other parameters

*Status* – variable that indicate the status of calculation:

0, if no errors, outputs printed on CLI.

-1, if max\_iter was reached, not all outputs printed on CLI

# 7. Data Type

Integer*: max\_iter*, *status*

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

*character: choice*

long doule*: precision*

# Nietypowe data type

# Function code