Polynomial functions

dr Szymon Murawski

November 26, 2019

1 Polynomial functions

1.1 Introduction

A polynomial function $f(x)$ of a degree n is given by the following equations:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0 x^0, \tag{1}$$

where $a_n \ldots a_1$ are coefficients of the polynomials, and n is an integer number. Some examples of polynomials:

$$f(x) = 3x^3 + 8x^2 - 7x + 8$$
$$f(x) = -2x^2 + \frac{3}{4}x - 1$$
$$f(x) = x^7 + 9$$

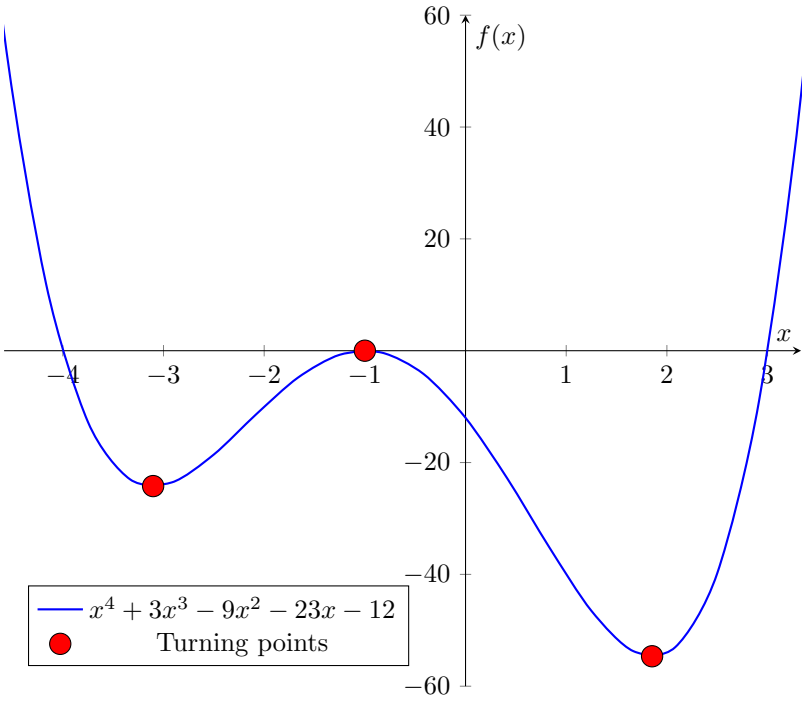
As you might see in the last example, a polynomial function of a degree n might have some coefficients $a_{i<n}$ equal to zero. A degree of a polynomial informs then, what is the highest power of x in the function with coefficient different than zero

While coefficients a are real numbers, power of x must be a positive integer or zero, the following functions are not polynomials:

$$f(x) = 5x^2 + x + x^{-1}$$
$$f(x) = 67x^{3/2} + 8$$
$$f(x) = 4x^3 + \sqrt{x} + 1$$

1.2 Turning points

Polynomial function of a degree n can have up to $n - 1$ turning points, that is points, at which function changes from increasing to decreasing in value (or vice versa). Formally, the gradient of the function changes from positive to negative (or vice versa).



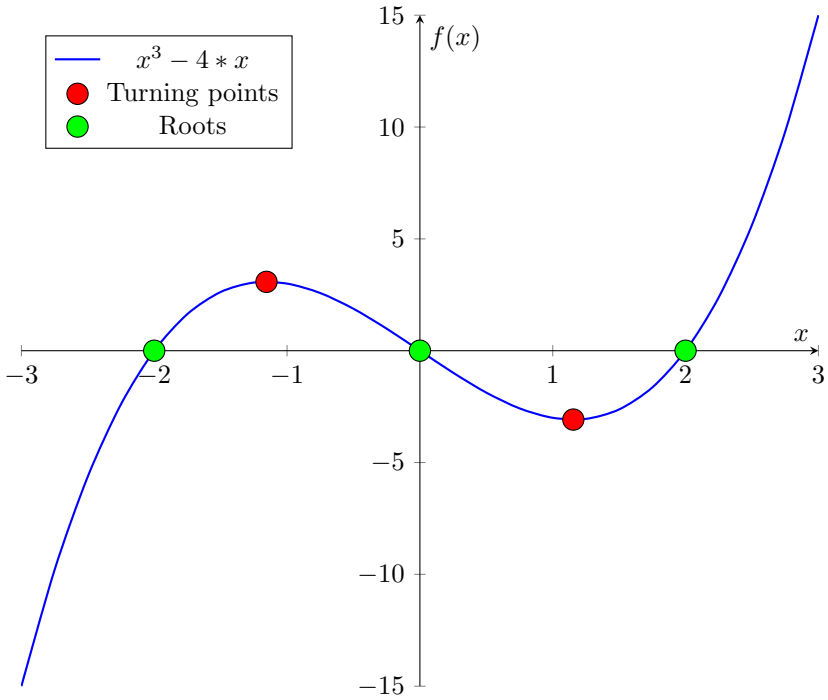
1.3 Roots

Roots are values x, for which $f(x) = 0$. If we know, that function $f(x)$ has roots $r_1, r_2, ...r_n$, then the function can be written as:

$$f(x) = (x - r_1)^{m_1}(x - r_2)^{m_2} \dots (x - r_n)^{m_n}, \tag{2}$$

where $m_1, m_2, \dots, m_n$ denotes multiplicities of a given root. For example:

$$f(x) = x^3 - 4x = x(x - 2)(x + 2) \tag{3}$$



If we know roots of a function, we can sketch the general shape of a function. Rules are:

- Since for very big numbers only the leading term is important, calculate the sign of $f(-\infty)$ and $f(\infty)$. This will give you the idea where the functions starts and ends
- If multiplicity of a root is odd, then the functions crosses this root
- If the multiplicity of a root is even, then function bounces off the root

For example:

$$f(x) = x^7 + 10x^6 + 27x^5 - 57x^3 - 30x^2 + 29x + 20 = (x+1)^3(x-1)^2(x+5)(x+4) \tag{4}$$

Coefficient of x^7 is positive, so function starts at $-\infty$. Function then crosses x axis at $x = -5$, then at $x = -4$, next at $x = -1$ and finally for $x = 1$ function bounces off so for $x = +\infty$ we get $f(+\infty) = +\infty$

2 Interpolation

Interpolation is a process, in which we estimate intermediate values between precise data points. In this process we find a function (or functions) that pass through all the given data points, and then we can evaluate the function values in the unknown region.

There are two methods of obtaining interpolation function:

- Polynomial interpolation, where a polynomial function of order $n-1$ is chosen (n being the number of known points)
- Splice interpolation, where for every consecutive pair of points we choose different function. Depending on the order of the functions we choose we might have linear interpolation, quadratic, cubic, etc.

In this lecture we will look at polynomial interpolation

2.1 Polynomial interpolation - Vandermonde matrix

Suppose we have $n+1$ points $(x_1, y_1), (x_2, y_2), \ldots, (x_{n+1}, y_{n+1})$. According to polynomial interpolation there must exist a polynomial $f(x)$ of order n, that passes through all these points:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0 x^0,$$

If we now insert the values of the points, we get the following system of linear equations:

$$\begin{aligned}
a_n x_1^n + a_{n-1} x_1^{n-1} + \cdots + a_1 x_1^1 + a_0 &= y_1 \\
a_n x_2^n + a_{n-1} x_2^{n-1} + \cdots + a_1 x_2^1 + a_0 &= y_2 \\
\cdots \quad & \cdots \quad \cdots \quad \cdots \\
a_n x_{n+1}^n + a_{n-1} x_{n+1}^{n-1} + \cdots + a_1 x_{n+1}^1 + a_0 &= y_{n+1}
\end{aligned}$$

In the system above $x_1, \ldots, x_n$ are known, as well as $y_1, \ldots, y_n$ and the only unknowns are coefficients $a_0, \ldots, a_n$. This means, that in fact this is a system of linear equations, that can be reduced to augmented matrix notation and solved using one of the methods for solving systems of linear equations!

Example: Suppose we have the following points: $(2, 5), (3, 6), (7, 4)$. Linear equation written in augmented matrix notation would lead us to:

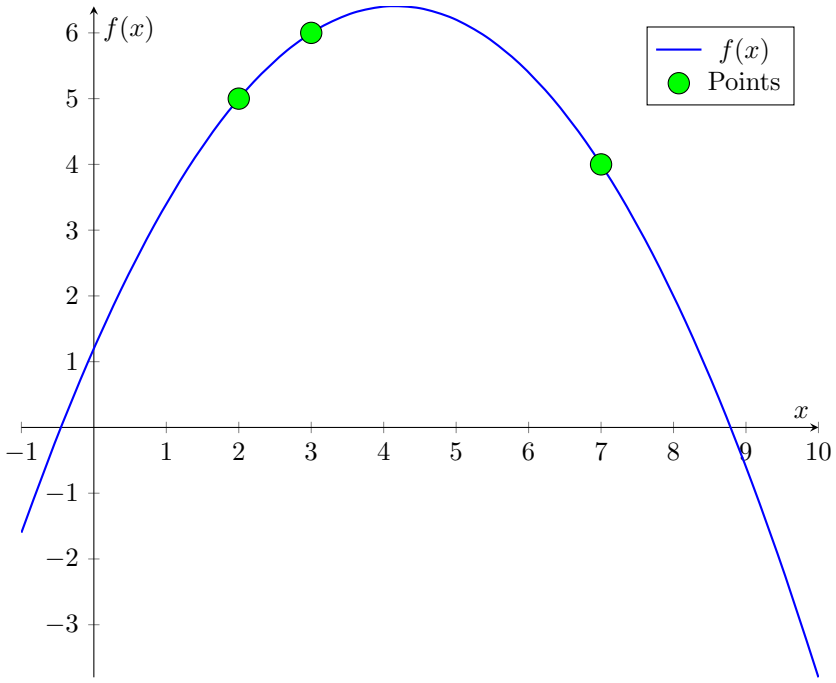
$$\left[\begin{array}{ccc|c} 4 & 2 & 1 & 5 \\ 9 & 3 & 1 & 6 \\ 49 & 7 & 1 & 4 \end{array}\right]$$

From this we get the result

$$\begin{aligned}
a_2 &= -0.3 \\
a_1 &= 2.5 \\
a_0 &= 1.2
\end{aligned}$$

Our resulting polynomial is then given by an expression

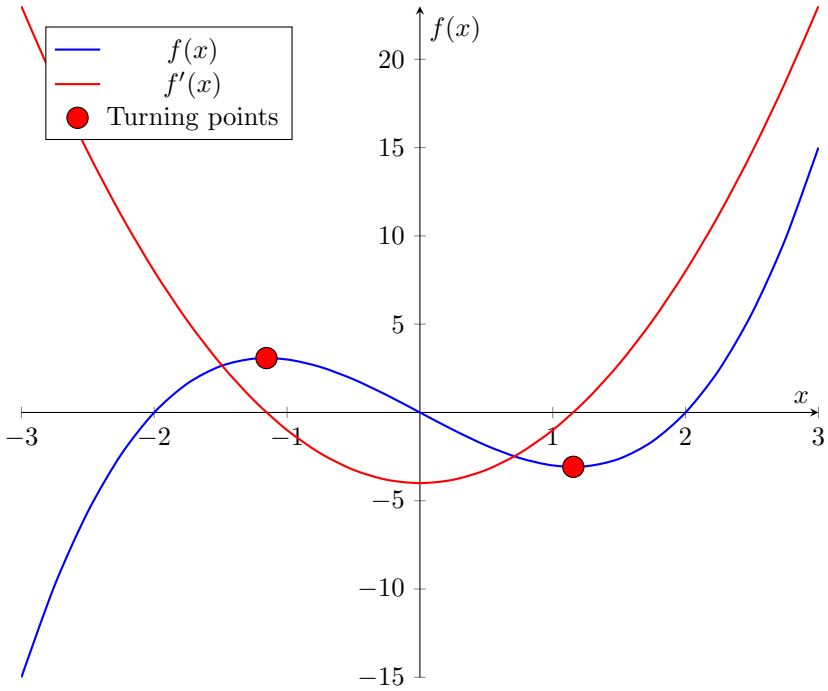
$$f(x) = -0.3x^2 + 2.5x + 1.2$$



3 Derivative of polynomial

A derivative $f'(x)$ of a function $f(x)$ is a function, that tells how change in argument of a function will affect the result. There are three cases:

- $f'(x) < 0$ - function $f(x)$ will be descending
- $f'(x) > 0$ - function $f(x)$ will be ascending
- $f'(x) = 0$ - function $f(x)$ is in a turning point



We can compute derivative of any function by using the following formula:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} \tag{5}$$

Suppose now that we have simple polynomial function $f(x) = ax^n$. The derivative can be then calculated as:

$$\begin{aligned}
f'(x) &= \lim_{\Delta x \to 0} \frac{a(x + \Delta x)^n - ax^n}{\Delta x} \\
&= \lim_{\Delta x \to 0} \frac{a(x^n + nx^{n-1}\Delta x + (n(n-1)/2)x^{n-2}\Delta x^2 + \cdots + \Delta x^n - x^n)}{\Delta x} \\
&= \lim_{\Delta x \to 0} \frac{a(nx^{n-1}\Delta x + (n(n-1)/2)x^{n-2}\Delta x^2 + \cdots + \Delta x^n)}{\Delta x} \\
&= \lim_{\Delta x \to 0} a(nx^{n-1} + (n(n-1)/2)x^{n-2}\Delta x + \cdots + \Delta x^{n-1}) \\
&= anx^{n-1}
\end{aligned}$$

Since polynomial function $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0 x^0$ can be separated into individual functions such as $f_n(x) = a_n x^n, f_{n-1}(x) = a_{n-1}x^{n-1}, \ldots, f_0(x) = 0$, calculating derivative of function $f(x)$ is very easy, as we can express the derivative as a sum of partial derivatives:

$$f'(x) = f_1'(x) + f_2'(x) + \cdots + f_n'(x)$$

Simple pseudocode for this algorithm will look like this

```
1  function = double[n] \\array that holds coefficients of a
     function
2  derivative = double[n-1] \\array that holds coefficients of a
     derivative of function
3  for (i=0..n-2)
4    derivative[i] = (i+1)*function[i+1]
```

4 Finding roots

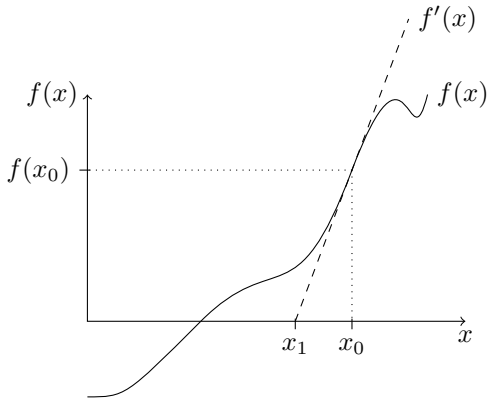
Finding roots of polynomials, and in fact any other non-linear functions, is very hard task. Numerous algorithms were proposed that attempt to solve this problem, but all of them only give approximate locations of roots. What's more, if an algorithm says it did not find a root for a specific function it doesn't mean there is no root, only that this specific method couldn't find one.

Methods for finding roots can be generalized into two categories:

- **Bracketing methods**, where we start with two initial x-values, enclosing the root between them. Then we reduce the distance between the x-values, and provide an approximate answer after the gap is sufficiently small
- **Iterative methods**, where we start with an initial guess of a root and by iteration correct our answer

Method that we will be using for finding roots of polynomials will be iterative Newton's method.

Suppose we have to find a root of a function $f(x)$ and our initial guess is x_0. Our problem would look like on the picture below:



To find a root using Newton's method, we draw a tangent line to function $f(x)$ at $x = x_0$. This tangent line is given by expression

$$y = f'(x_0)(x_1 - x_0) + f(x_0) \tag{6}$$

We can now find a value x_1, for which $y = 0$ and take x_1 as the new approximation of the root.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{7}$$

If we repeat the process multiple times, we can get good approximation of the root. In general then, the iterative formula is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{8}$$

The question now becomes when we stop iterations. One way is to use the formula below for calculating relative approximation error:

$$\epsilon = \frac{x_{n+1} - x_n}{x_{n+1}} * 100\% \tag{9}$$

We now have three stop criteria:

- $\epsilon < 0.0001$, and $f(x_{n+1}) = 0$ - algorithm converged and produced a root
- $\epsilon < 0.0001$, and $f(x_{n+1})! = 0$ - algorithm converged but did not find a root
- $n > 100$ and $\epsilon > 0.001$ - algorithm did not converge after this many iterations and will probably never converge, root cannot be found