TASK 2

Robotzii

Overview

The **AICitizens** team from the **"Alexandru Ioan Cuza" National College in Focșani** won the **FIRST Tech Challenge Robotics World Championship**, held between **April 15-20, 2024**, in **Houston**, **USA**, being one of the **231** participating teams.

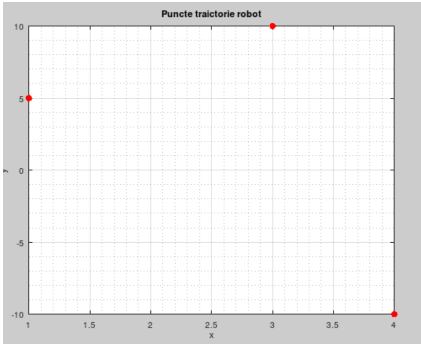
The test of the competition consisted of two stages: the first, **autonomous**, in which the robot executes pre-calculated movements, and the second, **controlled by joystick**, giving commands to the robot motors.

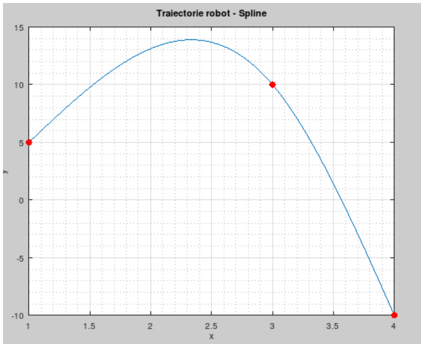
This performance would not have been possible without the ultra-advanced techniques learned by the team using **numerical methods**.

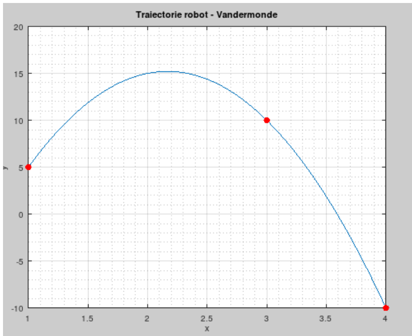
Purpose

In this topic, we will simulate the **autonomous** part of the robot at a minimalist level through the interpolation methods learned in the course. We consider that the robot is on a two-dimensional terrain without bumps, and it needs to discover a trajectory such that it passes through a set of points: (x0,y0),(x1,y1),(x2,y2),...,(xn,yn)$(x_0,y_0),(x_1,y_1),(x_2,y_2),...,(x_n,y_n)$, where xi<xi+1,   i=0:n−1$x_i<x_{i+1}$, $i=0:n-1$. Knowing these points, you will determine this trajectory using two interpolation methods: **Vandermonde** and **Natural Cubic Splines**.

For example, having three points through which we need to pass, the trajectories will be the following:







Cubic Spline Interpolation

We are going to imeplement an interpolation method using C^2 class spline functions. For each interval $[x_i, x_{i+1}]$, we will consider a third-degree polynomial with the following expression:

$$s_i : [x_i,\ x_{i+1}] \to \mathbb{R}, \quad s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad i = 0 : n - 1$$

The spline polynomial will be:

$$P_{\text{spline}}(x) = \begin{cases} s_0(x), & \text{if } x \in [x_0, x_1) \\ s_1(x), & \text{if } x \in [x_1, x_2) \\ \vdots \\ s_{n-1}(x), & \text{if } x \in [x_{n-1}, x_n] \end{cases}$$

System of equations

We notice that **n such polynomials** will be constructed, each with **4 coefficients** a_i, b_i, c_i, d_i in total, **4n unknown variables**. We will thus create a system with **4n equations** to find these coefficients:

- **n+1 equations** like this: the value of the polynomials at the ends of the intervals is equal to the value of the ordinates of the points in the interpolation support;

$$\begin{cases} s_i(x_i) = y_i, & i = 0 : n - 1 \\ s_{n-1}(x_n) = y_n \end{cases}$$

- **3(n-1) equations** like this: at the intersection points, every two consecutive polynomials will have the same value, the same slope (equal values of the derivatives), respectively the same convexity (equal values of the 2nd-order derivatives);

$$\begin{cases} s_i(x_{i+1}) = s_{i+1}(x_{i+1}), & i = 0 : n - 2 \\ s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}), & i = 0 : n - 2 \\ s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}), & i = 0 : n - 2 \end{cases}$$

- **2 equations** like this: natural splines have the property that the 2nd derivatives at the ends of the interpolating support are 0.

$$\begin{cases} s_0''(x_0) = 0 \\ s_{n-1}''(x_n) = 0 \end{cases}$$

Having the 4n equations, and knowing the expression of si*si*, we deduce the 1st and 2nd order derivatives as follows:

$$\begin{aligned} s_i'(x) &= b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2, & i = 0 : n - 1 \\ s_i''(x) &= 2c_i + 6d_i(x - x_i), & i = 0 : n - 1 \end{aligned}$$

Substituting into the above expressions, we deduce the 4n equations needed to form the system. We will build the matrix **A** of the coefficients of the described equations, the vector of the coefficient $coef = [a_0\ b_0\ b_0\ c_0\ a_1\ b_1\ c_1\ d_1 ... a_{n-1}\ b_{n-1}\ c_{n-1}\ d_{n-1}]^t$ of each polynomial, respectively **b** of t he free variables. Thus, it is required to solve for coef*coef* the equation A·coef=b

For example, given $\{(x_0, y_0), (x_1, y_1), (x_2, y_2)\} = \{(1, 5), (3, 10), (4, -10)\}$ we derive the following system, putting each equation in order, on every row of the matrices:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & -1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 12 & 0 & -1 & 0 & 0 \\ 0 & 0 & 2 & 12 & 0 & 0 & -2 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6 \end{pmatrix} \begin{pmatrix} a_0 \\ b_0 \\ c_0 \\ d_0 \\ a_1 \\ b_1 \\ c_1 \\ d_1 \end{pmatrix} = \begin{pmatrix} 5 \\ 10 \\ -10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \Rightarrow coef = \begin{pmatrix} a_0 \\ b_0 \\ c_0 \\ d_0 \\ a_1 \\ b_1 \\ c_1 \\ d_1 \end{pmatrix} = \begin{pmatrix} 5 \\ 10 \\ 0 \\ -1.875 \\ 10 \\ -12.5 \\ -11.25 \\ 3.75 \end{pmatrix}$$

Thus, the s1*s*1 and s2*s*2 polynomials and have the following expressions:

$$s_0(x) = 5 + 10(x - 1) + 0(x - 1)^2 - 1.875(x - 1)^3$$
$$s_1(x) = 10 - 12.5(x - 3) - 11.25(x - 3)^2 + 3.75(x - 3)^3$$

Octave Tasks

You must implement the following functions:

- **function [x, y] = parse_data(filename)**

  This function reads numbers from the file named <filename>. On the first line of the file there is a natural number **n**. On the second line there are **n+1 integers** representing the abscissas of the interpolation support points $x_0\ x_1\ x_2\ ...\ x_n$. On the third line there are **n+1 integers** representing the ordinates $y_0\ y_1\ y_2\ \cdots\ y_n$ of these points. The function returns two collumn vectors **x** and **y**.

- **function coef = spline_c2 (x, y)**

  This function solves the matrix equation system described. The sent parameters are the collumn vectors: $[x_0\ x_1\ x_2\ ...\ x_n]^t$ and $[y_0\ y_1\ y_2\ \cdots\ y_n]^t$ respectively. The result of the function is the collumn vector coef= $[a_0\ b_0\ b_0\ c_0\ a_1\ b_1\ c_1\ d_1 ... a_{n-1}\ b_{n-1}\ c_{n-1}\ d_{n-1}]^t$. Remember that, in **GNU Octave**, indexing of vectors and matrices is done from **1**, and not from **0** (for ease you can shift all indices by one position to the right).

- **function y_interp = P_spline (coef, x, x_interp)**
  This function receives the previously determined matrix of coefficients, the collumn vector **x** with the points $[x_0\ x_1\ x_2\ ...\ x_n]^t$, and the abscissa vector **x_interp**, in which the values of the polynomial *Pspline* are to be found. The function returns a collumn vector **y_interp**, where yinterp(i)=Pspline(x_interp(i)), i=0:length(xinterp)−1

Vandermonde Interpolation

We want to determine the polynomial P_Vandermondee with the following property: PVandermonde(xi)=yi,   i=0:

$$P_{Vandermonde} : [x_0, x_n] \to \mathbb{R}, \quad P_{Vandermonde}(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

System of equations

To determine this polynomial, we write its value at each point and determine the following system:

$$\begin{cases} P_{Vandermonde}(x_0) = y_0 \\ P_{Vandermonde}(x_1) = y_1 \\ \vdots \\ P_{Vandermonde}(x_n) = y_n \end{cases} \Leftrightarrow \begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + ... + a_n x_0^n = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + ... + a_n x_1^n = y_1 \\ \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + ... + a_n x_n^n = y_n \end{cases}$$

Solving the system

Written in matriceal form, the system becomes:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ 1 & x_2 & x_2^2 & ... & x_2^n \\ ... & ... & ... & ... & ... \\ 1 & x_n & x_n^2 & ... & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ ... \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ ... \\ y_n \end{pmatrix}$$

For the example given at splines: {(x0,y0),(x1,y1),(x2,y2)}={(1,5),(3,10),(4,−10)} the above system is solved in the following way:

$$\begin{pmatrix} 1 & 1 & 1^2 \\ 1 & 3 & 3^2 \\ 1 & 4 & 4^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 5 \\ 10 \\ -10 \end{pmatrix} \Rightarrow coef = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} -20 \\ 32.5 \\ -7.5 \end{pmatrix}$$

And the Vandermonde polynomial is:

$$P_{Vandermonde} : [1, 4] \to \mathbb{R}, \quad P_{Vandermonde}(x) = -20 + 32.5x - 7.5x^2$$

Octave Tasks

You must implement the following functions:

- **function coef = vandermonde (x, y)**

This function solves the matrix equation system described. The sent parameters are the collumn vectors: $[x_0\ x_1\ x_2\ ...\ x_n]^t$ and $[y_0\ y_1\ y_2\ ...\ y_n]^t$ respectively. The result of the function is the collumn vector coef= $|a_0\ a_1\ a_2\ ...\ a_n|^t$

- **function y_interp = P_vandermonde (coef, x_interp)**

This function receives the previously determined vector of coefficients, **coef**, and the abscissa vector **x_interp**, in which the values of the polynomial PVandermonde are to be found. The function returns a collumn vector **y_interp** with these values.

Debugging

For debugging, we provide you the following functions:

- **function plot_points(x, y)** – plots the points from the interpolation support;
- **function plot_spline (x, y, nr_points)** – plots splines for **nr_points** equidistant points;
- **function plot_vandermonde (x, y, nr_points)** – plots the Vandermonde polynomial using **nr_points** equidistant points.