Programming Project 2

Due on Mar 4th.

1 Goal

A tracking device has been attached to a moving vehicle. The goal of this project is to construct the routes of the vehicle, by interpolating the signals obtained from the tracking device.

2 Detailed description

2.1 Method

The tracking device records information about, among other parameters, their positions at different times as a matrix of the form:

$$\begin{bmatrix} t_0 & x_0 & y_0 \\ t_1 & x_1 & y_1 \\ \vdots & \vdots & \vdots \\ t_n & x_n & y_n \end{bmatrix} \tag{1}$$

The pair (x_i, y_i) are xy coordinates of the vehicle at time t_i, and we want to find a parametric curve $(x(t), y(t))$ describing the path of the vehicle at any time t. To find it, we will use polynomial interpolation to get a description of the evolution of the variables x and y with respect to t, and evaluate it at different times in the interval $[t_0, t_n]$.

2.2 Approach: Cubic Spline Interpolation

2.2.1 Definition of Cubic Spline

Here we want to use so-called **cubic spline interpolation** to interpolate the function $f(t)$ given point values $f(t_i) \equiv f_i$ for $t_0 < t_1 < \ldots < t_n$. We consider a piecewise polynomial $S(t)$ called the cubic spline, written in the form

$$S(t) = \begin{cases} s_0(t) & t \in [t_0, t_1] \\ \vdots & \vdots \\ s_{n-1}(t) & t \in [t_{n-1}, t_n] \end{cases}.$$

By the definition, the cubic spline $S(t)$ satisfies the following conditions

a. The function $S(t)$ restricted on each subinterval $[t_i, t_{i+1}]$, denoted by $s_i(t)$, is a cubic polynomial of the form

$$s_i(t) = f_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3 \quad \text{for } i = 0, 1, \ldots, n - 1. \tag{2}$$

b. $S(t)$, $S'(t)$ and $S''(t)$ are continuous.

c. **either** (i) natural boundary condition:

$$S''(t_0) = S''(t_n) = 0,$$

or (ii) clamped boundary condition:

$$S'(t_0) = f'_0 \quad \text{and} \quad S'(t_n) = f'_n$$

are satisfied. If clamped boundary condition is used, then some numerical differentiation method is needed to compute the derivatives f'_0 and f'_n.

2.2.2 Solving Cubic Spline

We want to solve the coefficients b_i, c_i, d_i for $i = 0, 1, \ldots, n-1$ in cubic spline representation (2). The solution process can be formulated in two steps:

1. **Solving second order derivatives** $S''(t_i)$.

   We denote $m_i = S''(t_i)$ for $i = 0, \ldots n$, and $h_i = t_{i+1} - t_i$ for $i = 0, 1, \ldots, n-1$. The second derivatives $m_0, \ldots, m_n$ at nodes can be obtained by solving the following tridiagonal linear system

$$A\boldsymbol{m} = \boldsymbol{r} \tag{3}$$

   where

$$A = \begin{bmatrix} 2 & \mu_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \lambda_0 & 2 & \mu_1 & 0 & \cdots & \cdots & 0 \\ 0 & \lambda_1 & 2 & \mu_2 & \cdots & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & 0 \\ \vdots & & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & \lambda_{n-2} & 2 & \mu_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & \lambda_{n-1} & 2 \end{bmatrix}, \ \boldsymbol{m} = \begin{bmatrix} m_0 \\ m_1 \\ \vdots \\ m_{n-1} \\ m_n \end{bmatrix}, \text{ and } \boldsymbol{r} = \begin{bmatrix} r_0 \\ \frac{6}{h_1+h_0}\left(\frac{f_2-f_1}{h_1} - \frac{f_1-f_0}{h_0}\right) \\ \vdots \\ \frac{6}{h_{n-1}+h_{n-2}}\left(\frac{f_n-f_{n-1}}{h_{n-1}} - \frac{f_{n-1}-f_{n-2}}{h_{n-2}}\right) \\ r_n \end{bmatrix}.$$

   Here

$$\mu_i = \frac{h_i}{h_{i-1}+h_i}, \quad \lambda_{i-1} = 1 - \mu_i = \frac{h_{i-1}}{h_{i-1}+h_i} \quad \text{for} \quad i = 1, \ldots, n-1.$$

   In addition, if the natural boundary condition is applied, then

$$\mu_0 = \lambda_{n-1} = 0, \quad r_0 = r_n = 0, \quad m_0 = m_n = 0.$$

   If a clamped boundary condition is applied, then

$$\mu_0 = \lambda_{n-1} = 1, \quad r_0 = \frac{6}{h_0^2}(f_1 - f_0) - \frac{6}{h_0} f_0', \quad r_n = \frac{6}{h_{n-1}} f_n' - 6\frac{f_n - f_{n-1}}{h_{n-1}^2}.$$

2. **Solving the coefficients.**

   After applying the interpolation conditions and continuity of the spline at the interior nodes, we have

$$\begin{cases} b_i &= \frac{f_{i+1}-f_i}{h_i} - \frac{h_i(m_{i+1}+2m_i)}{6}, \\ c_i &= \frac{m_i}{2}, \\ d_i &= \frac{m_{i+1}-m_i}{6h_i}. \end{cases} \tag{4}$$

Then we obtain the cubic splines $s_i(t)$ for each subinterval $[t_i, t_{i+1}]$ as defined in (2). They join together to define $S(t)$ on the whole interval $[t_0, t_n]$.

2.3 Application of cubic splines to the tracking problem

We consider cubic spline interpolation of $(x(t), y(t))$ for each time interval $[t_{i-1}, t_i]$. More concretely, we construct cubic splines $S^x(t)$ for $x(t)$ and $S^y(t)$ for $y(t)$ respectively. Then the parametric curve $(S^x(t), S^y(t))$ represents the path of the vehicle.

3 What to submit

You are expected to submit to the CCLE page of the course a `.zip` file containing:

1. MATLAB/Octave files implementing the following algorithms:

   (a) Construction of cubic splines as described in Section 2.2. The implementation should provide options to use natural boundary condition and clamped boundary condition.

   (b) Functions for solving the tracking problem as described in Section 2.1, using both natural boundary condition and clamped boundary condition.

2. A `.pdf` file with a brief report of the project, including the following sections:

   - **User Guide:** A concise description of all the routines and applications: what do they do, and meaning of each input/output variable.
   - **Solutions:** The data file is provided as "`data.mat`", and you can load the data by using the command "`load('data.mat')`". The variable "`ip`" are interpolating data points in the form of (1). Test your code by plotting the path of the vehicle.
   - **Experiments:** Calculate the velocity of the vehicle at each data point using the cubic splines, which are denoted by $\boldsymbol{u_0}, \boldsymbol{u_1}, \ldots, \boldsymbol{u_n}$. As a comparison, calculate the the velocity using three-point mid-point method (end points treated by appropriate methods), which are denoted by $\boldsymbol{v_0}, \boldsymbol{v_1}, \ldots, \boldsymbol{v_n}$. Note that the velocity of a vehicle is a 2D vector. Plot the velocity field $\boldsymbol{u}$ and $\boldsymbol{v}$ using the MATLAB/Octave built-in "`quiver`" function. Which velocity field is more likely to be accurate?