Exercise Medical Robotics CS4270 – Exercise Sheet 7

Institute for Robotics and Cognitive Systems
boettger@*rob.uni-luebeck.de*

Motion Prediction

SOLUTIONS

Please submit your solutions before next Monday at 14:15. The names of all group members must be included in the solution sheets/files. Handwritten solutions can be either submitted in the Institute for Robotics and Cognitive Systems (postbox in front of room 98) or scanned and uploaded in Moodle. MATLAB codes must be properly and briefly commented and uploaded in Moodle.

For all MATLAB solutions, see attached codes in the Moodle page.

1 Motion Prediction I – MULIN (10 Points)

In this exercise we will use the MULIN algorithm to predict the positions of a robotic treatment system. Typically respiration is periodic. Figure 1 shows the ideal situation, with a perfect sine wave (blue curve "Patient"). The time is discretized, i.e. at fixed time intervals (i.e. every 50 milliseconds) we must send a motion command to the robot. This command will tell the robot where to go. However, due to the time lag (computing time, robot motion time, etc.) the robot will reach the desired position after 100 milliseconds.

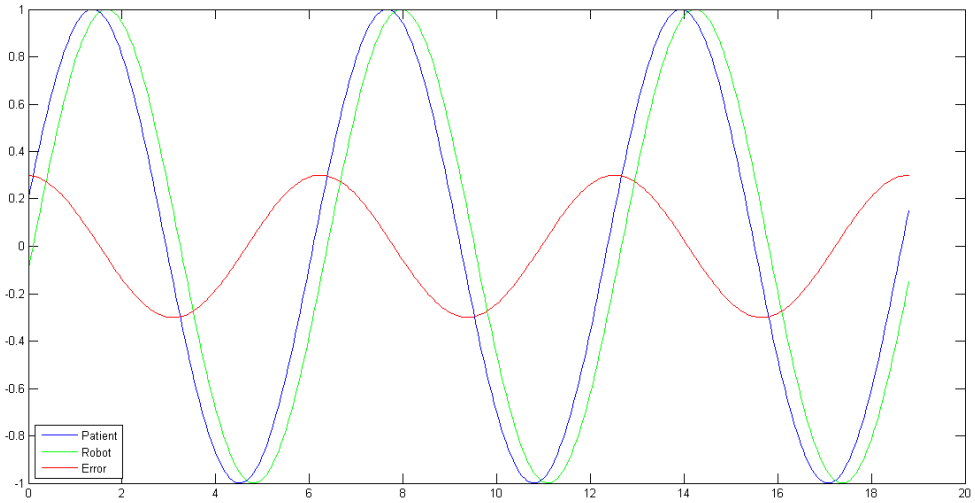

Figure 1: Error of robot positions respect to patient positions (without prediction)

The idea of prediction is the following. We will send a motion command to the robot. This command tells the robot not to move to the current position of the target, but to the position where the target will be in $100\ ms$. Based on the past history of the motion curve, we can determine this future position. Thus it is our goal to extrapolate the blue curve by a fixed amount δ_t.

Figure 2 shows an example of patient positions prediction using the MULIN algorithm of order 0. The green curve is the predicted positions for every patient position in n horizon $\delta_t = 100\ ms$ (i.e. two samples). We can see that the prediction improves the respiratory tracking and decrease the error (red curve) of the robot position.

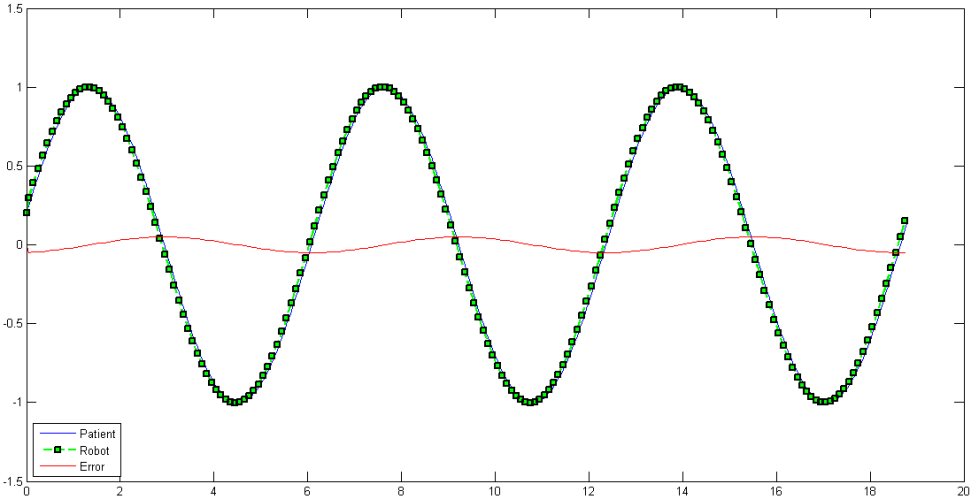

Figure 2: Predicting robot positions respect to patient positions (MULIN algorithm of order 0)

Write a MATLAB function $Ex07.m$ where you implement the following functions:

(a) First, plot the curve of the patient positions. The x-axis represents the time discretized at fixed time intervals (i.e. every 50 milliseconds) $t = [0: 0.05: 2]$. The y-axis represents patient positions $P_t = \sin(t + 0.2014)$. (1P)

(b) Write a function $MULIN0.m$ of order 0, and predict for each patient position the next position where the robot should drive.
If $\hat{p}(k+1) = p(k) + \big(p(k) - r(k)\big)$, note that $r(k) = p(k-1)$, where $\hat{p}(k+1)$ is the predicted patient position (i.e. the desired robot position at that time) at $k+1$, $r(k)$ is the current position of the robot. Additionally, assume the initial robot position is $r(0) = p(0)$. (2P)

(c) Write a function $MULIN1.m$ of order 1, and predict the robot positions. (3P)

(d) Write a function $MULIN2.m$ of order 2, and predict the robot positions. (4P)

(e) Write a function $RMS.m$ calculating the RMS of the prediction error between the patient positions and the robot positions. Which order of the MULIN algorithm has the smallest error? Comment the results. (6P)

2 Least mean squares (LMS) (16 Points)

Apart from the MULIN algorithm from the last exercise, the least mean squares (LMS) techniques can also be used for motion prediction. The LMS algorithm for prediction minimizes a quadratic error function by learning weights throughout the time series. In its simplest form it is given by:

$$e_n = y_n - \hat{y}_n$$

$$\hat{y}_{n+k} = w_n y_n$$

$$w_{n+k} = w_n + e_n y_n$$

where

e_n is the error at current time step n,
$\hat{y}_{n+k}$ is the prediction at time step $n + k$,
y_n is the original signal, and
w_n is the weight factor.

Consider the following signal

$$y_0 = \frac{\sqrt{2}}{2}, y_1 = 1, y_2 = \frac{\sqrt{2}}{2}, y_3 = 0.$$

(a) Use the LMS prediction algorithm to compute $\hat{y}_4$ (pen and paper). (4P)

To calculate $\hat{y}_4$ using LMS, it is important to calculate the weight factor w_4 as well.

n	y_n	$\hat{y}_n$	e_n	w_n
0	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{2}}{2}$	0	1
1	1	$\frac{\sqrt{2}}{2}$	0.29	1
2	$\frac{\sqrt{2}}{2}$	1	-0.29	1.29
3	0	0.91	-0.91	1.08
4	--	**0**	--	**1.08**

(b) The vector-based version of LMS allows for increasing the depth M of the past history which we take into account for prediction. It is described as follows:

$\hat{y}_{n+k} = w_n^T u_n$, $w_{n+1} = w_n + \mu(y_n - \hat{y}_n)u_n$, with $n \geq k$.

$u_n = \left(u_{n,1}, \dots, u_{n,M}\right)^T = (y_{n-M+1}, \dots, y_n)^T$.

Starting with $\hat{y}_1 = \cdots = \hat{y}_k = y_1$, and $w_1 = (0, \dots, 0,1)^T$.

Here u_n is the signal history used to calculate the next prediction, namely $\hat{y}_{n+k}$, and μ is the learning parameter. The weight factor for a prediction horizon k becomes a vector $w_n \in \mathbb{R}^M$.

Implement the LMS algorithm as a MATLAB function which takes a signal y as input, as well as parameters μ and M. You can assume $k = 1$. Run your implementation on real respiration data for $M = 2,4,8$ and different values of μ (hint: take values close to 0.001). Calculate the RMS of the error between the original and the predicted signal. Plot the prediction and original signals. You can download the file $test\_resp.mat$ from the Moodle. (10P)

(c) Comment on what happens as the value of M and μ change. (2P)

M is the parameter used to control the depth of the past history, which the algorithm considers for prediction. Therefore, the higher M is, the lower the RMS we can see. An improvement in prediction is therefore noticed.

μ controls the step length. It is difficult to select its optimal value, because it can be too short for good prediction results, or too large causing an overstep as seen in the given example. This is actually one of the disadvantages of LMS.