The Cost Function Minimization for Predictive Control by Newton-Raphson Method

B. Durmuş, H. Temurtaş, N. Yumuşak, F. Temurtaş, R. Kazan

***Abstract*-The Newton-Raphson method is one of the most widely used methods for minimization. It can be easily generalized for solving non-linear differential equation systems. In this study, Generalized Predictive Controller (GPC) was applied to a 6R robot manipulator based on joint control. Newton-Raphson (N-R) method was used to minimize the cost function existing in the GPC that represents errors between reference trajectory and actual trajectory in the control of robot. The Newton-Raphson method requires less iteration numbers for convergence and reduces the calculation. This study presents a detailed derivation of the Generalized Predictive Control algorithm with Newton-Raphson minimization method. The results of angular path and position errors belonging to joints were examined and compared with Recursive Least Square (RLS) implemented Generalized Predictive Control. The simulation results showed that Newton-Raphson method improved control performance of the GPC.**

***Keywords*- Predictive control, generalized predictive control, cost function minimization.**

I. INTRODUCTION

Predictive control algorithm was developed from the non-parameter model predictive control algorithm including the Model Algorithm Control (MAC), the Dynamic Matrix Control (DMC) and so on, to the Model Predictive Control (MPC) algorithm on example of which is the Generalized Predictive Control (GPC) algorithm [1-4]. Because the GPC algorithm parameter model is shorter than the non-parameter model, it reduced the GPC algorithm calculation time, and enhanced performance of the control system.

The GPC is used in the control of non-minimum phase plants, open-loop unstable plants and plants with variable or unknown dead time. It is also robust with respect to modeling errors, over and under parameterization, and sensor noise [5-6].

The computational performance of a GPC implementation is largely based on the minimization algorithm chosen for its CFM block. There are several minimization algorithms that have been implemented in GPC such as Non-gradient [7], Simplex, and Successive Quadratic Programming [8,9]. The selection of a minimization method can be based on several criteria such as; number of iterations to a solution, computational costs and accuracy of the solution. In general these approaches are iteration intensive thus making real-time control difficult. Very few papers address real-time implementation or the they used plants with large time constant [8,9]. To improve the usability, a faster optimization algorithm is needed. The Newton-Raphson method is one of the most widely used methods for minimization. It is a quadratic algorithm converging better than others. It requires less iteration numbers for convergence and reduces the calculation.

In this study, Generalized Predictive Control (GPC) was applied to 6R robot manipulator for joint control. Newton-Raphson (N-R) method was used to minimize the cost function existing in GPC that represents errors between reference trajectory and actual trajectory in the control of robot. The results of angular path and angular velocity belonging to joints were examined and compared with results obtained from the Recursive Least Square implemented Generalized Predictive Control. Also, processing times of both algorithms were shown.

II. GENERALIZED PREDICTIVE CONTROL

The Generalized Predictive Control (GPC) introduced by Clarke is a generalization of the model-based control and suitable for controlling of the processes with variable dead time and a plant which is simultaneously non-minimum-phase and open loop unstable. Many research showed the effectiveness of this control algorithm [3].

The GPC system for the robotic manipulator is given in Figure 1. It consists of three components, the robotic manipulator or its simulator, controller and parameter estimator. Where, torque, u, is the control input to the manipulator system, the trajectory, y, is the output, and y_r is the reference output.

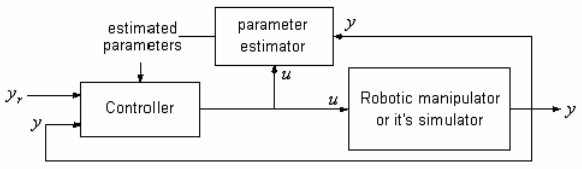


Fig. 1 Block diagram of the GPC system for robotic manipulator

The heart of a model-based predictive controller is the plant model. The Controlled Autoregressive Integrated

Manuscript received November 3, 2007. This work was supported by scientific research fund (University of Sakarya, grant no. 2006.50.02.050).

B.Durmuş is with *Sakarya University, Department of Electric - Electronic Engineering, 54187 Adapazari, TURKEY* (corresponding author phone: +90-533-5494170; fax: +90-264-2955601; e-mail: bdurmus@sakarya.edu.tr).

H. Temurtaş *Dumlupınar University, Department of Electric - Electronic Engineering, 41470 Kutahya, TURKEY* (e-mail: temurtas@dumlupinar.edu.tr).

N. Yumuşak *Sakarya University, Department of Computer Engineering, 54187 Adapazari, TURKEY* (e-mail: nyumusak@sakarya.edu.tr).

F. Temurtaş *Sakarya University, Department of Computer Engineering, 54187 Adapazari, TURKEY* (e-mail: temurtas@sakarya.edu.tr).

R. Kazan *Sakarya University, Department of Mechanical Engineering, 54187 Adapazari, TURKEY* (e-mail: kazan@sakarya.edu.tr).

Moving Average (CARIMA) model is commonly used in the GPC, as it is applicable to many single-input single-output plants:

$$A(q^{-1})\, y(t) \;=\; B(q^{-1})\, u(t-1) \;+ \xi(t) / \Delta \qquad (1)$$

Where, u(t), y(t) are the plant input and output. A and B are polynomials in the backward shift operator q^{-1}:

$$A(q^{-1}) \;=\; \mathrm{I}_m + \sum_{j=1}^{n_a} a_j\, q^{-j} = 1 + a_1 q^{-1} + \ldots + a_{n_a}\, q^{-n_a},$$
$$B(q^{-1}) \;=\; \sum_{j=0}^{n_b} b_j\, q^{-j} = \mathrm{b}_0 + \mathrm{b}_1 q^{-1} + \ldots + b_{n_b}\, q^{-n_b} \qquad (2)$$

Where, $\xi(t)$, is an uncorrelated random sequence, and the use of the operator $\Delta = 1 - q^{-1}$ ensures an integral control law.

Equivalently to the procedure of Clarke [1] an optimal j-step forward predictor is given by [2]:

$$\hat{y}(t+j) \;=\; F_j(q^{-1})\, y(t) + E_j(q^{-1})\, \Delta u(t-1) + G_j(q^{-1}) \Delta u(t+j-1) \qquad (3)$$
free response forced response

The first term of equation (3) is called 'free response', as it represents the plant predicted output $\hat{y}(t+j)$, when there is no future control action. The second term is called 'forced response', as it represents the output prediction due to the hypothetical future control actions $u\,(t+j-1),\ j \geq 1$.

To select a good control sequence, we would wish a minimal tracking error $e(t+j) \;=\; \hat{y}(t+j) - y_r\,(t+j)$ over a certain output horizon. Furthermore, to prevent any 'explosion' of the control action, a second term is generally added, so that the final performance index has a form similar to the following:

$$J(N_1, N_2, N_u) \;=\; \sum_{j=N_1}^{N_2} \left(\hat{y}(t+j) - y_r(t+j) \right)^2 + \sum_{j=1}^{N_u} \lambda(j) \left(\Delta u(t+j-1) \right)^2 \qquad (4)$$

subject to: $\Delta u(t+j) = 0$ for $j \geq N_u$

Where, N_1 and N_2 are the minimum and the maximum costing horizon, N_u is the control horizon and λ is a weighting factor for the control increment sequence to be calculated. Furthermore, the constraints that the control increments are forced to zero $j \geq N_u$ provide a better convergence of the output to the set point.

If we define the two following vectors formed with polynomial solutions of equation 3:

$f \;=\; \left[f(t+N_1) \ldots\ldots f(t+N_2) \right]^T$, the vector of the free response

Additionally, if we denote:

$$\hat{y} \;= \left[\hat{y}(t+N_1) \ldots\ldots \hat{y}(t+N_2) \right]^T \qquad (5)$$

$$\tilde{u} \;= \left[\Delta u(t), \Delta u(t+1) \ldots\ldots \Delta u(t+N_u - 1) \right]^T \qquad (6)$$

And the matrix formed with the coefficients g_i^j of the G_j polynomials, which in fact correspond to the step response values $g_i = g_i^j$:

$$G = \begin{pmatrix} g_{N_1} & \cdots & g_1 & 0 & \cdots & 0 \\ \vdots & & \vdots & & & \vdots \\ \vdots & & \vdots & & & 0 \\ \vdots & & \vdots & & & g_1 \\ \vdots & & \vdots & & & \vdots \\ \vdots & & \vdots & & & \vdots \\ g_{N_2} & \cdots & g_{N_2+N_1-1} & \cdots & & g_{N_2+N_u-1} \end{pmatrix} \qquad (7)$$

The output prediction has the following form:

$$\hat{y} \;=\; G\, \tilde{u} + f \qquad (8)$$

Now, equation (4) can be rewritten in a matrix form:

$$J = \left[G\, \tilde{u} + f - y_r \right]^T \left[G\, \tilde{u} + f - y_r \right] + \lambda\, \tilde{u}^T \tilde{u} \qquad (9)$$

And the optimal control law comes from $\dfrac{\partial J}{\partial \tilde{u}} \;=\; 0$.

$$\tilde{u} \;=\; \left(G^T G + \Lambda \right)^{-1} G^T \left(y_r - f \right) \qquad (10)$$

In this study, firstly, we used Recursive Least Square (RLS) for minimization of cost function in the GPC. In this method $A(q^{-1})$ and $B(q^{-1})$ parameters which are composing G and f are recalculated each of control steps.

We define following formed with the CARIMA model solutions of equation 2 for used Recursive Least Square [10].

$$y(t) = -a_1 y(t-1) - \cdots - a_{n_a} y(t-n_a) + b_0 u(t-1) + \cdots + b_{n_b} u(t-n_b-1) + \xi(t)/\Delta \qquad (11)$$

This equation is simplified;

$$y(t) \;=\; \hat{\Theta}^T(t)\, \Phi(t) + e(t) \qquad (12)$$

If $N \;=\; (n_a + n_b + 1) \cdot m$;

$\hat{\Theta}^T(t)$ is $m \times N$, $\Phi(t)$ and $e(t)$ are $N \times 1$ polynomial matrix. We denote follows:

$$\hat{\Theta}^T(t) \;=\; \left(-a_1, \cdots, -a_{n_a}, b_0, \cdots, b_{n_b} \right) \qquad (13)$$

$$\Phi(t) \;=\; \left(y(t-1)^T, \cdots, y(t-n_a)^T, u(t-1)^T, \cdots, u(t-n_b-1)^T \right)^T \qquad (14)$$

$$e(t) \;=\; \xi(t) / \Delta \qquad (15)$$

$\hat{\Theta}^T(t)$ parameter which is existing $A(q^{-1})$ and $B(q^{-1})$ parameters is updated for each of a control steps as follows:

1. The equation of gain:

$$K(t) \;=\; \frac{P(t-1)\, \Phi(t)}{\mu + \Phi^T(t)\, P(t-1)\, \Phi(t)} \qquad (16)$$

The gain is calculated used *K(t)*. μ is the target factor ($\mu = 0.95$). $P(t)$ parameter is $N \times N$ matrix. $P(0) \;=\; \mathrm{I}_N / \delta$ for first step control. I_N is unit matrix $N \times N$, δ as constant which is value 10^{-6}. The calculation of $P(t)$ for others steps control uses follows equation.

$$P(t) \;=\; \left(\mathrm{I}_N - K(t)\, \Phi^T(t) \right) \frac{P(t-1)}{\mu} \qquad (17)$$

2. The equation of error:

$$e(t) = y(t) - \Theta^T(t-1)\,\Phi(t) \quad (18)$$

Error is calculated by used this equation.

3. $\hat{\Theta}^T(t)$ is calculated the follows equation:

$$\hat{\Theta}^T(t) = \hat{\Theta}^T(t-1) + e(t)\,K(t)^T \quad (19)$$

$A(q^{-1})$ and $B(q^{-1})$ parameters are recalculated from $\hat{\Theta}^T(t)$ in the equation 19. Consequently, the parameters of GPC are updated by this process.

III. COST FUNCTION MINIMIZATION by NEWTON-RAPHSON METHOD

The objective of the CFM algorithm is to minimize J in equation 4 with respect to $[u(n+1), u(n+2), \ldots, u(n+N_u)]^T$, denoted U. This is accomplished by setting the J in equation (4) to zero and solving for U. With Newton-Raphson used as the CFM algorithm, J is minimized iteratively to determine the best U. An iterative process yields intermediate values for J denoted $J(k)$. For each iteration of $J(k)$ an intermediate control input vector is also generated and is denoted as

$$U(k) = \begin{pmatrix} u(n+1) \\ u(n+2) \\ \vdots \\ u(n+N_u) \end{pmatrix}, \quad k = 1, \cdots, \#iterations \quad (20)$$

The Newton-Raphson update rule for $U(k+1)$ is:

$$U(k+1) = U(k) - \left(\frac{\partial^2 J}{\partial U^2}(k)\right)^{-1}\left(\frac{\partial J}{\partial U}(k)\right) \quad (21)$$

where the Jacobian denoted as:

$$\frac{\partial J}{\partial U}(k) = \begin{pmatrix} \frac{\partial J}{\partial u(n+1)} \\ \vdots \\ \frac{\partial J}{\partial u(n+N_u)} \end{pmatrix} \quad (22)$$

the Hessian as

$$\frac{\partial^2 J}{\partial U^2}(k) = \begin{pmatrix} \frac{\partial^2 J}{\partial u(n+1)^2} & \cdots & \frac{\partial^2 J}{\partial u(n+1)\partial u(n+N_u)} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u(n+N_u)\partial u(n+1)} & \cdots & \frac{\partial^2 J}{\partial u(n+N_u)^2} \end{pmatrix} \quad (23)$$

Solving equation (23) directly requires the inverse of the Hessian matrix. These processes could be computationally expensive. One technique to avoid the use of a matrix inverse is to use LU decomposition [11] to solve fort he control input vector $U(k+1)$. This is accomplished by rewriting Equation (23) in the form of a system of linear equations, Ax=b. This result in

$$\frac{\partial^2 J}{\partial U^2}(k)\left(U(k+1) - U(k)\right) = -\frac{\partial J}{\partial U}(k)$$

$$\frac{\partial^2 J}{\partial U^2}(k) = \mathrm{A}$$

$$-\frac{\partial J}{\partial U}(k) = \mathrm{b}, \text{ and}$$

$$U(k+1) - U(k) = \mathrm{x}$$

In this form Equation (23) can be solved with two routines supplied in [11] the LU (Lower/Upper Triangular) decomposition routine (ludcmp), and the system of linear equations solver (lubksb).

After x is calculated, $u(k+1)$ is solved by evaluating $u(k+1) = x + u(k)$. This procedure is repeated until the percent change in each element of $u(k+1)$ is less than some ε ($\varepsilon = 10^{-7}$). When solving for x, calculation of each element of the Jacobian and Hessian is needed to each of Newton-Raphson iteration. The h^{th} element of the Jacobian is

$$\begin{aligned} \frac{\partial J}{\partial u(n+h)} &= -2\sum_{j=N_1}^{N_2}\left(y_r(n+j) - yn(n+j)\right)\frac{\partial yn(n+j)}{\partial u(n+h)} \\ &+ 2\sum_{j=1}^{N_u}\lambda(j)\left(u(n+j) - u(n+j-1)\right)\left(\frac{\partial u(n+j)}{\partial u(n+h)} - \frac{\partial u(n+j-1)}{\partial u(n+h)}\right) \\ &+ \sum_{j=1}^{N_u}\frac{\partial u(n+j)}{\partial u(n+h)}\left(\frac{-s}{\left(u(n+j) - u_{\min} + \varepsilon\right)^2} + \frac{s}{\left(u_{\max} - u(n+j) + \varepsilon\right)^2}\right) \end{aligned} \quad (24)$$

$h = 1, \ldots, N_u$.

The m^{th}, h^{th} element of the Hessian is:

$$\begin{aligned} &\frac{\partial^2 J}{\partial u(n+m)\,\partial u(n+h)} = \\ &2\sum_{j=N_1}^{N_2}\left(\frac{\partial yn(n+j)}{\partial u(n+m)}\frac{\partial yn(n+j)}{\partial u(n+h)} - \frac{\partial^2 yn(n+j)}{\partial u(n+m)\partial u(n+h)}\left(y_r(n+j) - yn(n+j)\right)\right) \\ &+ 2\sum_{j=1}^{N_u}\lambda(j)\left(\frac{\partial u(n+j)}{\partial u(n+m)} - \frac{\partial u(n+j-1)}{\partial u(n+m)}\right)\left(\frac{\partial u(n+j)}{\partial u(n+h)} - \frac{\partial u(n+j-1)}{\partial u(n+h)}\right) \\ &+ 2\sum_{j=1}^{N_u}\frac{\partial u(n+j)}{\partial u(n+m)}\frac{\partial u(n+j)}{\partial u(n+h)}\left(\frac{s}{\left(u(n+j) - u_{\min} + \varepsilon\right)^3} + \frac{s}{\left(u_{\max} - u(n+j) + \varepsilon\right)^3}\right) \end{aligned} \quad (25)$$

$h = 1, \ldots N_u$ and $m = 1, \ldots N_u$.

The $\frac{\partial^2 \Delta u(n+j)}{\partial u(n+m)\partial u(n+h)}$ always evaluates to zero.

The last component needed to evaluate $u(k+1)$ is the calculation of the output of the plant, $yn(n+j)$, and its derivates.

IV. DYNAMIC MODEL OF ROBOT MANIPULATOR

A priori information needed for manipulator control analysis and manipulator design is a set of closed form differential equations describing the dynamic behavior of the manipulators. Various approaches are available to formulate the robot arm dynamics, such as Lagrange-Euler, Newton-Euler and Recursive Lagrange [12,13].

The configuration of the six joint robotic manipulator model and its Denavit-Hartenberg parameters can be seen in Figure 1 and Table I respectively [14]. In this study, Lagrange-Euler is used for dynamics modeling of the six joints robotic manipulator. Lagrange-Euler equation of the motion is,

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_i}\right) - \frac{\partial L}{\partial \theta_i} = \tau_i \quad (26)$$

where τ_i is generalized torque applied to the system from joint *i*, *L* is Lagrangian function ($L = K - P$, K : total kinetic energy of the manipulator, P : total potential energy of the manipulator), θ_i is the angular position of the joint *i*, and $\dot{\theta}_i$ is the first order derivative of the θ_i.

Equations which were used for the calculation of the total kinetic energy of the manipulator are given in (27), (28), and (29).

$$K = \sum_{i=1}^{3} K_i \tag{27}$$

$$K_i = \int dK_i \tag{28}$$

$$dK_i = \frac{1}{2}\left(\dot{x}_i^2 + \dot{y}_i^2 + \dot{z}_i^2\right)dm \tag{29}$$

Equations which were used for the calculation of the total potential energy of the manipulator are given in (30), (31), (32), and (33).

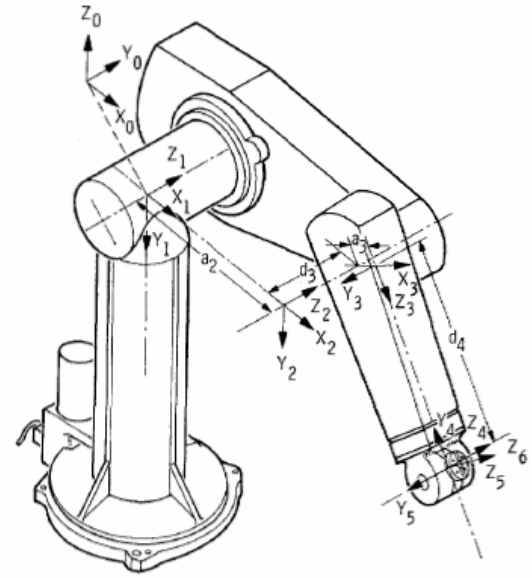


Fig. 2 The Model of 6R Robot Manipulator

Table I. Denavit-Hartenberg Parameters of PUMA 560 Robot Arm

i	α_i (degree)	θ_i	a_i (meter)	d_i (meter)
1	-90	θ_1	0	0
2	0	θ_2	0.4318	0.1491
3	90	θ_3	-0.0203	0
4	-90	θ_4	0	0.4331
5	90	θ_5	0	0
6	0	θ_6	0	0.056

$$P = \sum_{i=1}^{3} P_i \tag{30}$$

$$P_i = -m_i g r_0^i \tag{31}$$

$$r_0^i = A_0^i r_i \tag{32}$$

$$r_i = (x_i, y_i, z_i, 1)^T \tag{33}$$

Where m_i is the mass of the limb *i*, g is the gravity vector, A_0^i is the transition matrix. $|g| = 9.8062\ m/s^2$.

V. SIMULATION RESULTS

In this paper, it was designed 6R (six-DOF) robotic manipulator control using Newton-Raphson (N-R) implemented Generalized Predictive Controller (GPC) algorithm based on joint control. It was compared with RLS (Recursive Least Square) implemented GPC according to the simulation results.

Total simulation time is 10 second and total step number is 10000. In additionally, robot manipulator carries 5 kg load at the end-effecter.

Table II. Some results of simulation robot manipulator by N-R and RLS implemented GPC

Joints	Algorithms	Initial Angle (rad)	Desired Angle (rad)	Actual Angle (rad)	Angular Path Error (rad)
Joint 1	N-R	0.0000	0.1745	0.1744	0.0001
	RLS	0.0000	0.1745	0.1746	0.0001
Joint 2	N-R	-0.0872	0.0000	0.0004	0.0004
	RLS	-0.0872	0.0000	0.0011	0.0011
Joint 3	N-R	0.3490	0.5235	0.5237	0.0002
	RLS	0.3490	0.5235	0.5228	0.0007
Joint 4	N-R	-0.3490	-0.1745	-0.1745	0.0000
	RLS	-0.3490	-0.1745	-0.1745	0.0000
Joint 5	N-R	-0.0872	0.3490	0.3491	0.0001
	RLS	-0.0872	0.3490	0.3480	-0.0020
Joint 6	N-R	0.0000	0.6981	0.6981	0.0000
	RLS	0.0000	0.6981	0.6980	0.0010

For example, the results of angular path were given in Figure 3. Angular velocities and angular position errors were given belonging to robot arm joint 1, 2 and 5 in Figure 4, 5 and 6 respectively. Additionally, initial- desired-actual angles, angular path errors were given in Table II and also squares of angular velocity error values were given in Table III.

As seen in Table II, angle path errors are smaller in the N-R than those in the RLS. On the other hand, the squares of angular velocity errors are proportional to the jerk. The less angular velocity errors at the joints lead to less jerks. With N-R controller happened fewer jolts at the joints.

As seen in Fig. 3, N-R controller tracked to desired trajectory smoother and closer than RLS controller. In tracking performance, both algorithms were found to be satisfactory, according to simulation results of angular velocity given in Figs 4-6. Also, the differences of angular path errors are seen in same Figs.

Table III. Some results of angular velocity errors (rad/sec^2)

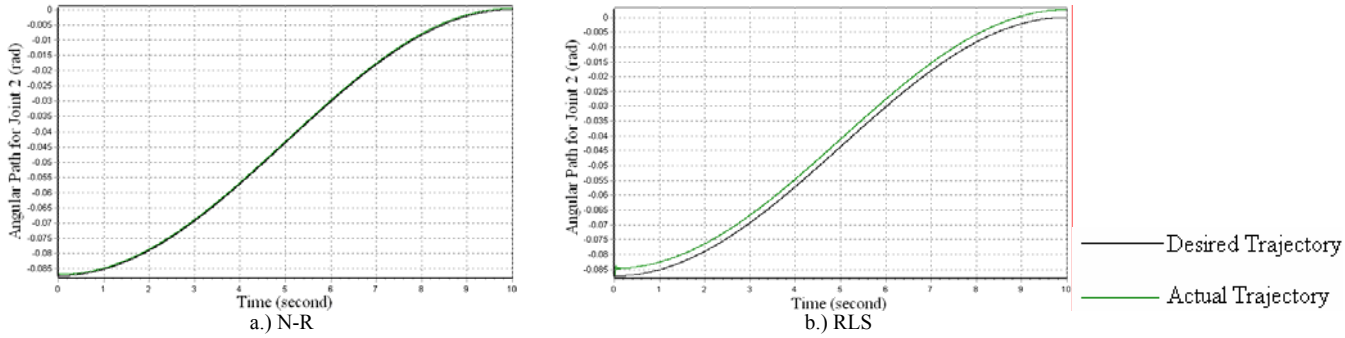
	Joint1	Joint2	Joint3	Joint4	Joint5	Joint6
N-R	0.0000	0.0193	0.0026	0.0000	0.0003	0.000
RLS	0.0016	0.1940	0.0108	0.0004	0.0071	0.001

Table IV. Position errors of robot arm and processing time (CPU's time) of controllers

	Position errors (mm)	Processing Time (ms)
N-R	0.4772	2625
RLS	0.7681	3469

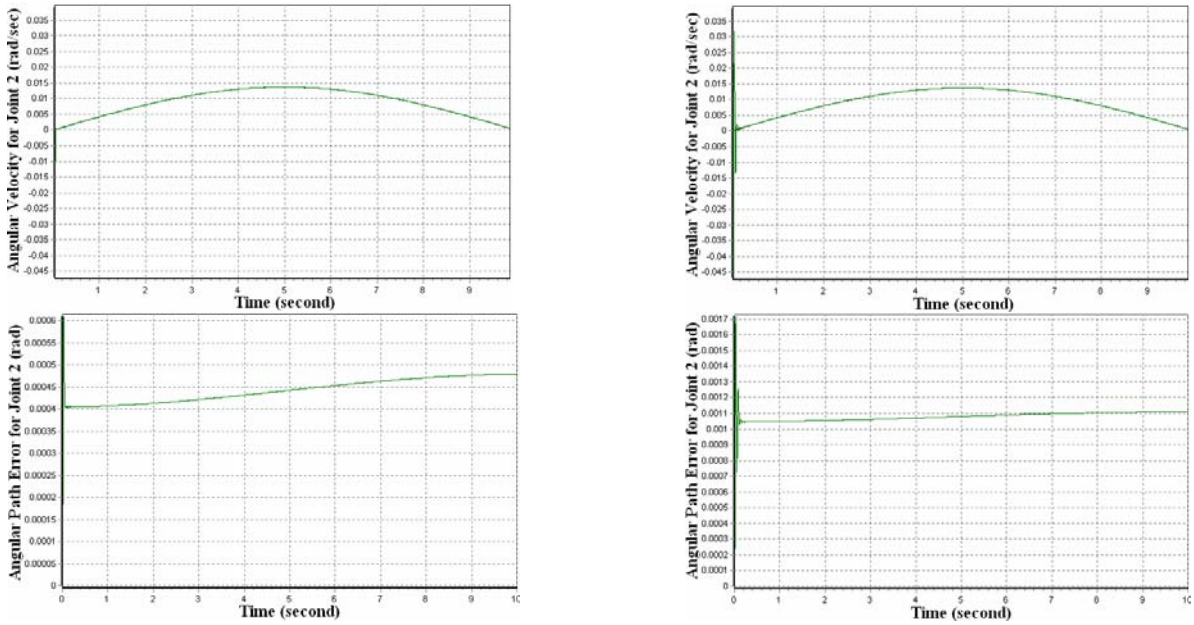
The position errors of robot arm and processing times belonging to control algorithms were given in Table IV. The

position errors of the N-R controller are lesser than those by RLS. For processing time, simulation of the RLS had taken 3469 ms, whereas the N-R controller has taken 2625 ms, for the given trajectory Figs 4-6. Newton-Raphson method reduced the time of cost function minimization and also reduced the processing time.



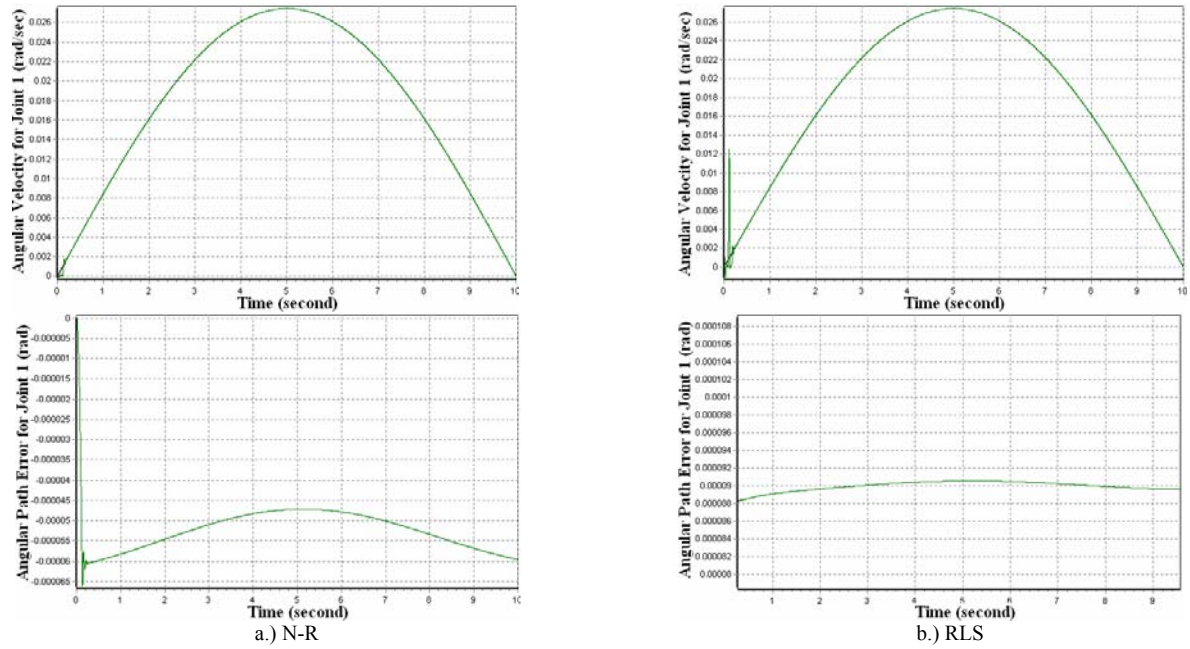
a.) N-R b.) RLS

Fig. 3 Angular Path of Joint 2 by N-R and RLS implemented GPC



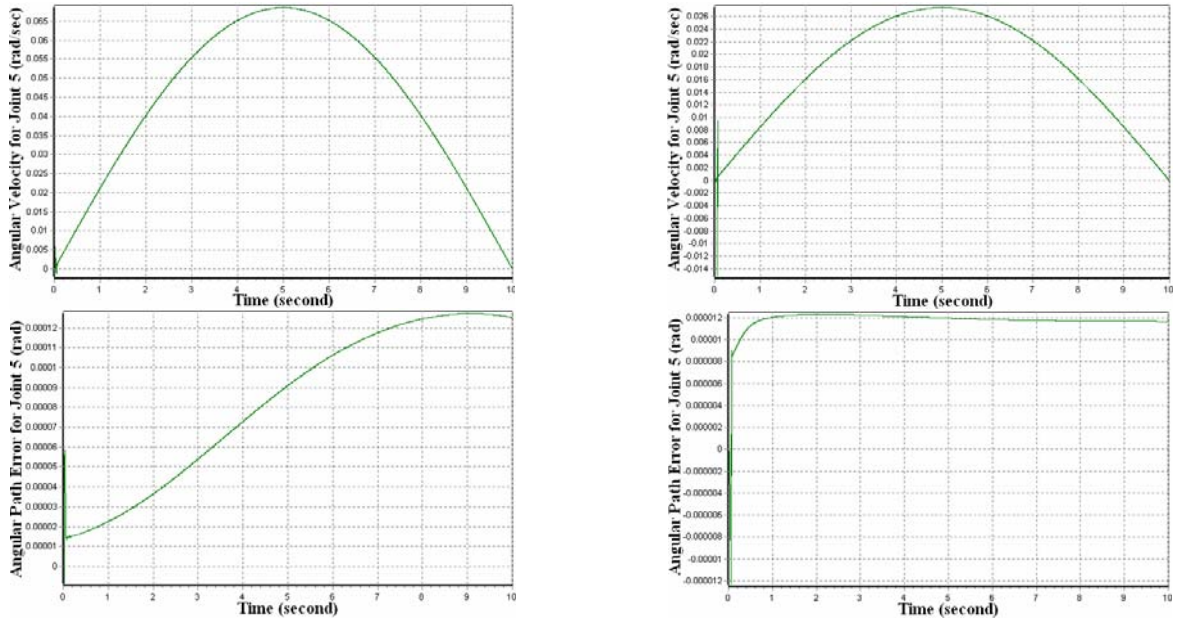
a.) N-R b.) RLS

Fig. 4 Angular Velocity and Angular Position Errors for Joint 2 by N-R and RLS implemented GPC



a.) N-R b.) RLS

Fig. 5 Angular Velocity and Angular Position Errors for Joint 1 by N-R and RLS implemented GPC



a.) N-R b.) RLS

Fig. 6 Angular Velocity and Angular Position Errors for Joint 5 by N-R and RLS implemented GPC

In this paper, computationally efficient of cost function minimization in the GPC algorithms was examined. There is over computation in the minimization of the cost function. Newton-Raphson update method requires less iteration numbers for convergence and reduces the calculation

In this application, Recursive Least Square (RLS) implemented GPC between Newton-Raphson implemented GPC controllers were applied to the 6R robot arm manipulator based on joint control. The results of angular path and angular velocity belonging to joints, position error of end-effecter and also processing time of controllers were compared. According to the simulation results, Newton-Raphson implemented GPC reduced position errors and also processing time for robot manipulator control. This means that the Newton-Raphson improved control performance of the GPC.

REFERENCES

[1] J Clarke D W. Application of generalized predictive control to industrial processes. *IEEE control Syst.* Maga, 1988, 8(2), pp.49-55.

[2] J Richalet, Model predictive heuristic control: Applications to industrial processes, *Automatica*, 1978, 14(5), pp. 413-428.

[3] Rouhani R., Mehra R K. Model algorithmic control (MAC), Basic theoretical properties, Automatica,1982, 18(4), pp. 401-414.

[4] C R Culter, B L Ramaker, Dynamic matrix control: A computer control algorithm, *Proceedings of Joint Automatica Control Conference*, San Francisco, 1980.

[5] Köker R., "Design and performance of an intelligent predictive controller for a six-degree-of-freedom robot using the Elman network", *Information Sciences*, *Volume 176, Issue 12*, pp. 1781-1799, 22 June 2006.

[6] D. Solaway, P. Haley, Neural generalized predictive contol: a Newton-Raphson implementation, Proceeding of the 11_{th} IEEE International Symposium on Intelligent Control, pp. 277-282, 1996.

[7] G. A. Montague, M.J. Willis, M.T. Tham and A.J. Morris, "Artificial Neural Network Based Control", International Conference on Control 1991, Vol. 1, pp.266-271.

[8] Jeong Jun Song and Sunwon Park. "Neural Model-Predictive Control for Nonlinear Chemical Processes", *Journal of Chemical Engineering of Japan*, 1993, V26, N4, pp. 347-354.

[9] D.C. Psichogios and L.H. Ungar, "Nonlinear Internal Model and Model Predictive Control using Neural Networks", *5_{th} IEEE International Symposium on Intelligent Control 1990*, pp. 1082-1087.

[10] F. Temurtas, H. Temurtas, N. Yumusak, C. Oz, Effects of the Trajectory Planning on the Model Based Predictive Robotic Manipulator Control, *Lect. Notes Comput. Sci*., 2869 (2003), pp. 545-552.

[11] W.H. Pres, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling,"Numerical Recipes in C:The Art of Scientific Computing", Cambridge University Pres 1988.

[12] W. M. Silver, On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators, The International Journal of Robotics Research, Vol. 1(2) (1982) 60-70.

[13] C. S. G. Lee, Robot Arm Kinematics- Dynamics, and Control, Computer, Vol. 15 (12) (1982) 62-80.

[14] Tarn T.J., Bejczy A.K, Yun X., "Coordinated Control of Two Robot Arms", CH2282-2/86/0000/1193/01.00 copy IEEE, 1986.