Proceedings of the 6th WSEAS Int. Conf. on NEURAL NETWORKS, Lisbon, Portugal, June 16-18, 2005 (pp278-283)

# Non-Derivative Optimization using Neural Network Model Based Predictive Control

**P. Aadaleesan*, K. Ramkumar[1], S.Nithya[2], S.M.GiriRajkumar[3]**

*School of Electrical and Electronics Engineering,*
*SASTRA Deemed University, Thanjavur-613 402,*
*Tamil Nadu, India.*

**Abstract**

` Model Predictive Control (MPC) is an online open-loop optimal control and is an advanced control strategy widely used in many process industries now a days. This paper focuses on the use of non-derivative optimization in MPC for a LTI system. The algorithm for the development of such a non-derivative optimization algorithm is also given for bound constraints along with the proof for asymptotic stability. The results of this paper are illustrated with a simple example.

***Keywords:*** Model Predictive Control (MPC), non-derivative optimization, asymptotic stability.

## 1. Introduction

Model Predictive Control (MPC) or Receding horizon control is widely used nowadays for control of many complex processes in many industries since 1970's [2, 5, 6]. MPC being a form of control in which the current control action is obtained by solving *online*, at each sampling instant, a finite horizon open-loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant [3, 4, 9]. Sequential Quadratic Programming (SQP) is the most widely used optimization technique in MPC, as mostly the performance index being quadratic subject to equality and/or inequality constraints. There are well known methods for quadratic programming (QP): BFGS, Newton-Rapson method, etc. These are all derivative based methods. An excellent theory of these methods is given in [2, 5, 6]. Of course, non-derivative methods of optimization are also discussed in [2, 6].

The paper is arranged as follows: in section 2, the non-derivative algorithm is discussed. The above algorithm is used for MPC in section 3. The result obtained by non-derivative method is compared and discussed in section 4.

In this paper any parameter mentioned after; inside the parenthesis is to be taken as for the given value of the parameter.

## *2.* The MPC algorithm

*Prediction and Optimization:*

Model Predictive Control (MPC) is an open-loop optimal control problem, minimizing a cost function, *J*:

$$J_{min\ \Delta u} = \sum_{k=1}^{N-1} (\mathrm{yr(k)}\ \mathrm{y(k)})^{T} Q\ (y_r(\mathrm{k}) - \hat{y}(\mathrm{k}))$$

$+\Delta u^T(\text{k-1})R\Delta u(\text{k-1})$ (2.1)

subject to

$$\Delta y_{min} \le \Delta y \le \Delta y_{max} \quad (2.2)$$
$$\Delta u_{min} \le \Delta u \le \Delta u_{max} \quad (2.3)$$

where,

$y_r$ - set point
$\hat{y}$ - predicted output from the model
$\Delta u$- rate of change of control move
$Q$, $R$- output and control input weighing parameters, respectively.

From the above cost function (2.1), it is evident that the controller, MPC, needs to minimize the error,$e$;

$$e = y_r - \hat{y} \quad (2.4)$$

with minimal rate of change of control effort, $\Delta u$, which is piecewise constant.

The prediction of the plant's behavior in the future over a finite time horizon (prediction horizon) is achieved by making use of the plant's model. The model could be anyone as transfer function, state space model, Matrix Fraction Description (MFD) etc., which are all mathematical models that depicts the dynamics of the plant under consideration. The model could also be obtained as a neural network, which plays a significant role in modeling nonlinear systems [1,8,11], which is a kind of empirical modeling.

The system to be considered here is a linear stable system, so the prediction and control horizon are so selected, $N$ and $M(=N-1)$, respectively, and the principle of linearity, i.e., super position theorem, also holds good.

The objective or cost function could be rewritten, from (2.4), as,

$$J_{min\ \Delta u} = \sum e^T Q\, e + \Delta u^T R \Delta u \quad (2.5)$$

where $Q$ and $R$ are positive symmetric weighing parameters.

$$i.e.,\ J_{min\ \Delta u} = \sum e^2 Q + \Delta u^2 R \quad (2.6)$$

subject to constraints,

$$-e \le e \le e \quad (2.7)$$
$$\Delta u_{min} \le \Delta u \le \Delta u_{max} \quad (2.8)$$

For convenience, some error tolerance is allowable in the system's performance. A strong assumption is made such that the optimum value lie in the interior in the region $\boldsymbol{R^y \times R^u}$ , as the objective function and the constraints are closed and convex [2]. So the weighing parameter, $Q$, is so selected:

$$Q = \left( \frac{error}{error\_tolerance} \right)^2 \quad (2.9)$$

This makes this weighing matrix always be selected dynamically, according to the error, the difference between the set point, $y_r$, and the predicted outputs, $\hat{y}$, instead of having a fixed value like $Q$ = 0.1,1,etc., as did conventionally.

The inputs, $\Delta u$, for every iteration until the prediction horizon (N) is found out by line search within the feasible (or admissible) region in other methods like BFGS method or other gradient based methods of optimization. In this method some finite number of discrete units are made within the bound constraints $\Delta u_{min}$ and $\Delta u_{max}$. This is made to reduce the computational burden, but it is to be noted that smaller the difference between two discrete units, more precise the convergence towards the set point.

All the possible finite inputs $\{\boldsymbol{\Delta u}\}$ made between the bound

constraints, are supplied to the model sequentially and their corresponding output values, $\{\hat{y}\}$, are stored. These inputs $\{\Delta u\}$ and their corresponding outputs, $\{\hat{y}\}$, are substituted in the cost function (2.6). Before substituting $\{\hat{y}\}$ in the cost function, the output values that violate the given constraints are discarded. Only those outputs within the feasible region are substituted in (2.6).

The corresponding set of cost values, $\{J\}$, for the given allowable outputs and its inputs are found out. The minimal cost value of the set obtained is identified. The control inputs $\Delta u$, corresponding to the minimal cost value i.e., $\Delta J_{min}$, is taken as the optimal control move ($\Delta u^*$) for that instant.

The optimal value of the control move, ($\Delta u^*$), is again applied to the model to get the optimal output, $y^*$, for the given optimal control move, and the above process repeats until the end of the prediction horizon.

## 3. Stability

Stability could also be ensured in an asymptotic manner, with the proof of monotonicity property of $\{J^*(\cdot)\}$ [9], if the performance gives a result so as the cost function value always decreases i.e., $J_1^*(\cdot) \leq J_0^*(\cdot)$. The above proof of stability by the monotonicity of $\{J^*(\cdot)\}$ could always be ensured as the optimal point lies in the interior of $\boldsymbol{R}^y \times \boldsymbol{R}^u$, which implies that the system is asymptotically stable.

Once the system state reaches the *terminal constraint region* i.e., the error tolerance ($=X_f$, subset of $\boldsymbol{R}^n$ ) value selected, which is in the neighbourhood of the origin of $\boldsymbol{R}^n$, it could be taken to the origin by a local controller $u=\kappa_f(\cdot)$. This form of model predictive control called dual mode control, was proposed in [Michalska & Mayne 1993], interested reader may refer [9, 10].

## 4. Experimental result

*4.1. System identification with neural network:*

In model predictive control, model plays a vital role of predicting the future response of the system, with the current measurement from the actual plant. Here it is achieved using a neural network, as did conventionally. No special emphasis is given here to say about the system identification using neural network, as an excellent literature is available on this [1, 7, 8, 12].

The network is trained with a set of input-output data, for the network to capture the dynamics of the system. As here only a linear system is taken, the network could capture the linearity very easily.

A pseudo random binary sequence (PRBS) is used to train the network, with the output being feedback after delayed by one time unit. See Fig.3.1.

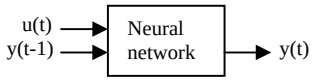


**Fig.3.1**. *Recurrent type neural network*

*4.2. Model Predictive Control design:*

The above said non-derivative algorithm for MPC is illustrated here with a simple example, by taking a stable linear time invariant (LTI), SISO system:

$$\frac{Y(s)}{U(s)} = \frac{5.9}{50s+1} \tag{3.1}$$

which must satisfy the performance index,

$$J_{min\ \Delta u} = \sum (y_r - \hat{y})^T Q\,(y_r - \hat{y}) + \Delta u^T R \Delta u \tag{3.2}$$

subject to constraints,

$$0 \le \hat{y} \le 0.5 \qquad (3.3)$$

$$-0.5 \le \Delta u \le 0.5 \qquad (3.4)$$

where the set point, $y_r$, is to be tracked by the closed loop system. The equation (3.2), thereafter takes the form (2.6), as the weighing matrices ($Q$ and R) are positive symmetric and the error, $e=y_r-\hat{y}$. The weighing matrix $Q$ is so selected for an error tolerance = 0.001:

$$Q = \left( \frac{e}{0.001} \right)^2 \qquad (3.5)$$

where $R$ = **1** (identity matrix) i.e., there is no dynamically changing control move suppression made. The performance of this dynamically changing weighing matrix (only $Q$ here) is shown in Fig.3.2.

The Fig.3.2 shows that the control move is somewhat a deadbeat like performance. It could also be noted that there is some error in the response, which is because of the error tolerance that is been accepted in the beginning.

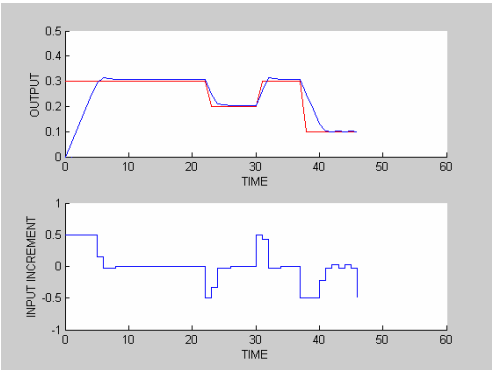


**Fig.3.2**. *Response of the NN-MPC with no control move suppression, R =1*

This performance is not a satisfactory as with a deadbeat like control. So the dynamically changing weighing matrix is also used for the control move suppression:

$$R = (\Delta u_{max}/\delta)^2 \qquad (3.6)$$

where $\delta$ is the discretization step size between $\Delta u_{max}$ and $\Delta u_{min}$, which could be chosen as small as possible. As already mentioned, smaller the difference between two discrete units, more precise the convergence towards the set point. But this so a trade off is to be made between the
will obviously increase the computation, final error and the computational burden by the proper selection of the value of $\delta$.

The result of the performance of the system by selecting the control move suppression (or weighing matrix), $R$, as above, for $\delta$ = 0.025, is shown in Fig.3.3 It could be noted that the performance is with a smooth control action, not as dead-beat type of control as in Fig.3.2. But this is achieved at the cost of a longer settling time. A trade off could be made between the faster settling time and the smoother control move, by proper selection of the value of $\delta$.

## 5. Comparative result

The result obtained by the new algorithm could be well compared with the conventional derivative type of optimization, like Sequential Quadratic Programming (SQP), to show the comparative performance between the two.

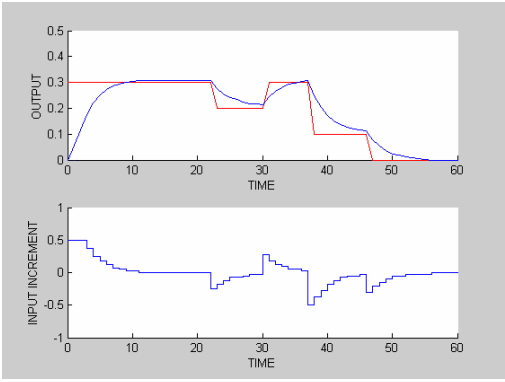


**Fig.3.3**. *Response of NN-MPC with dynamic move suppression.*

The above novel method of optimization by dynamically changing the weighing parameters is compared with a standard SQP optimization method, namely BFGS method [2, 6]. The BFGS method is basically a quasi-Newton gradient-based method in which the Hessian matrix of the objective function is also included. The Hessian matrix to be positive definite is the sufficient condition for optimality (minimization). But with other methods that incorporate Hessian matrix, needs calculation of the Hessian and an inverse of the Hessian at each iteration. This may be impossible always, where the quasi-Newton method proves its success by avoiding direct calculation of Hessian and its inverse. So BFGS is one of the famous and widely used quasi-Newton method. In this method an easy way to update the Hessian matrix is also provided with the use of the objective function and its Jacobian itself.

This newly developed optimization algorithm is compared with the result obtained using the BFGS method to show the comparative performance of the two in Fig.4.

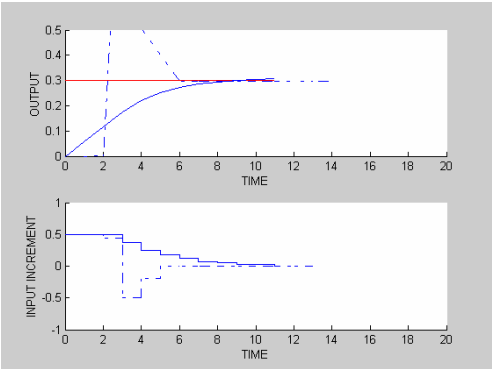


*Fig.4. Comparison of the new algorithm (solid line) and BFGS method (dashed)*

## 6. Discussion

It could be found that the new algorithm provides a better move suppression than that of the BFGS method. But the BFGS method converges faster than that of the new method, but some thing like a deadbeat controller. So it could be said that a trade off is to be made between the faster convergence and the lesser control move at each time instant.

## 7. Conclusions

The non-derivative method of optimization provides a better performance than the conventional optimization method(s). This method greatly reduces the computational complexity of differentiating the objective function subject to constraints. This approach is developed only for a linear stable system, taking the advantage of the properties of linearity. This approach could be easily extended to a MIMO (Multiple input multiple output) linear system.

## References

[1] Andreas Draeger, Sebastin Engell & Horst Ranke (1995). “Model predictive Control using neural networks”. *IEEE Control Systems Magazine.* pp. 61-66.

[2] Bazaraa, M. S., Sherali, H. D., & Shetty, C. M.(2004). Nonlinear programming-Theory and algorithms. 2e,John-Wiley & Sons, Inc.,(Asia).

[3] Camacho, E.F., & Bordons, C.(2004). Model Predictive Control, Springer-Verlag, 2e, London.

[4] Chen, H., & Allgöwer, F.(1998). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica,* 34(10), 1205-1218.

[5] Edgar, Himmanbleu.(2001).Optimization of chemical processes. McGrqw Hill,2e, Sing-apore

[6] Fletcher,R. (1999). Practical methods of optimization.John-Wiley,2e, London.

[7] Haykin, S.(1999). Neural Networks-A comprehensive foundation. 2e, Prentice-Hall Inc., Upper Saddle River, New Jersey.

[8] Ishida, M., & Zhan, J.(1997). The multi-step control of nonlinear SISO processes with a neural model predictive control method.

*Comp. Chem.. Engg*, 21(2),pp. 201-210.

[9] Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. M.(2000). Constrained model predictive control: Stability and optimality. *Automatica,* 36,789-814.

[10] Michalska, H., & Mayne, D. Q.(1993). Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. on Automatic Control*, 38, 1623-1632.

[11] Morari, M., & Lee, J. H.(1999). Model predictive control: past, present and future. *Comp. and Chem.*23, 667-682.

[12] Narendra, K.S., & Parthasarathy, K.(1990). Identification and control of dynamical systems using neural networks. *IEEE Trans on Neural Netrorks.*1(1), pp. 4-27.