CONTROL PREDICTIVO CON REDES NEURONALES COMO MODELO, UTILIZANDO EL METODO DE NEWTON-RAPHSON PARA LOGRAR LA MINIMIZACION DE LA FUNCION DE COSTO

Neural model predictive control using newton-raphson method

RESUMEN

En presente documento se implementará un controlador predictivo basado en el modelo utilizando redes neuronales, para esto se describen cada uno de los pasos necesarios, como el calculo de la derivada de la función de costo, el calculo de la primera y segunda derivada de la red neuronal y por ultimo se presenta la implementación del algoritmo de Newton-Raphson para la minimización de la función de costo.

PALABRAS CLAVES: control inteligente, newton-Rapshon, redes neuronales.

ABSTRACT

In this paper a Neural Network Model Predictive control is implemented, for this is describe each one of the steps, as the calculus of the function cost derivate, the calculus of the first and second derivate of the neural network and finally is present the implementation of the Newton-Raphson algorithm for the minimization of the function cost.

KEYWORDS: *intelligent control, neural network, newton-Raphson.*

JOSÉ ALFREDO JARAMILLO VILLEGAS
Docente Auxiliar Ingeniería de Sistemas y Computación
Universidad Tecnológica de Pereira
jj@sirius.utp.edu.co

LINA MARÍA PÉREZ PÉREZ
Ingeniera en Sistemas y Computación.
Grupo de Investigación Sirius
Universidad Tecnológica de Pereira
lina@sirius.utp.edu.co

ANDRÉS GUILLERMO VELÁSQUEZ GÓMEZ
Ingeniero en Sistemas y Computación.
Grupo de Investigación Sirius
Universidad Tecnológica de Pereira
andres@sirius.utp.edu.co

1. INTRODUCCIÓN

El control predictivo basado en el modelo (*Model Predictive Control*), se dio a conocer en primera instancia como *Generalized Predictive Control* en 1987 por D.W Clarke y sus colaboradores [1], [2]. La técnica MPC ha sido analizada e implementada satisfactoriamente en procesos industriales desde 1970 y continúa vigente. Esta técnica fue creada originalmente utilizando modelos lineales de la planta para realizar la predicción, esto por la facilidad en los cálculos. Para modelos no lineales las características de MPC de hacer buenas predicciones puede ser mejorada utilizando redes neuronales para aprender la dinámica de la planta, y se conoce como *Neural Model Predictive* Control (NMPC).

2. ARQUITECTURA DE UN CONTROLADOR PREDICTIVO UTILIZANDO REDES NEURONALES COMO MODELO

La arquitectura de un sistema NMPC se puede ver **Figura 1.** Este está conformado por tres componentes principales, la planta que se va a controlar, el algoritmo de minimización de la función de costo y la red neuronal del modelo [2] [3].

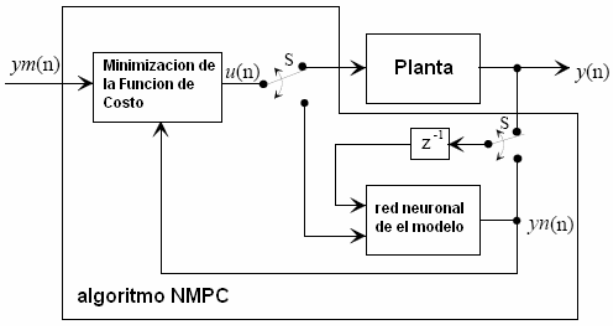

Figura 1: diagrama de bloques del algoritmo NMPC
Fuente: *Neural Generalized Predictive Control, A Newton-Raphson Implementation*

El algoritmo de NMPC inicia con el valor al cual se quiere llevar la planta este valor se denomina *ym(n)* siendo utilizado como entrada para el bloque de minimización de la función de costo, este bloque da como resultado la señal de control *u(n)* que puede tomar

Fecha de Recepción: 26 de enero de 2009
Fecha de Aceptación: 19 de mayo de 2009

dos direcciones como se observa en la **Figura 1** el valor de *u(n)* es la entrada de la red neuronal hasta hallar un valor para *u(n)* que minimice el error, este valor es el que se le presenta posteriormente a la planta. El algoritmo se presenta a continuación [3]:

1) Se genera la trayectoria al algoritmo, esta puede ser constante *ym(n)*.
2) Utilizando la señal de control anterior y el modelo de la planta se calcula el comportamiento futuro de de la planta.
3) Se calcula una nueva señal de control que minimice la función de costo.
4) Se repiten los pasos 2 y 3 hasta que se obtiene el valor mínimo deseado.
5) Se repite todo el proceso.

La eficiencia del algoritmo NMPC depende principalmente de dos factores claves, uno es la selección del método de minimización de la función de costo y el otro es la aproximación de la planta.

Existen varios tipos de algoritmos de minimización para la función de costo como el método simplex, programación cuadrática, Newton-Raphson, y Newton-Gauss. La desventaja que presentan algunos de estos métodos es que necesitan de muchas iteraciones para encontrar un mínimo haciendo difícil la implementación de un controlador en tiempo real, por esto para la selección del método de minimización se debe de tener en cuenta la velocidad de convergencia, Newton-Raphson, Newton-Gauss son métodos de minimización donde la convergencia de estos no está garantizada, pero en el caso de que esta exista la convergencia es cuadrática lo que garantiza alta velocidad en la minimización.

La función de costo es la que se busca minimizar sobre un horizonte de predicción finito y esta definida como:

$$J = \sum_{j=N_1}^{N_2} [ym(n+j) - yn(n+j)]^2 + \sum_{j=1}^{N_u} [\Delta u(n+j)]^2 \quad (1)$$

Donde:

- *N1* es el valor mínimo para el horizonte de predicción
- *N2* es el valor máximo para el horizonte de predicción
- *Nu* es el horizonte de control
- *Ym* es el valor al que se quiere llevar la planta
- *Yn* es el valor predicho por la red neuronal
- $\Delta u(n+j)$ es el cambio en la señal de control *u* y está definido como *u(n+j)-u(n+j-1)*

Se puede observar en la ecuación (1) que no solo se minimiza el error entre el valor predicho por la red neuronal *(Ym)* y el valor al cual se quiere llevar la planta *(Yn)*, sino que también se minimiza el cambio entre las señales de control.

Para una buena aproximación de la planta, de sistemas que presenta un comportamiento lineal existen técnicas para hallar un modelo matemático de esta, pero en la práctica existen sistemas que no presentan un comportamiento lineal, en este caso la mejor opción para aproximar el comportamiento de la planta es utilizar redes neuronales que cuentan con la capacidad de aproximar el comportamiento de sistemas no lineales. A continuación se profundizara en cada uno de los componentes del algoritmo, utilizando como método de minimización el método de Newton-Raphson.

3. ALGORITMO DE MINIMIZACIÓN PARA LA FUNCIÓN DE COSTO

Como se mencionó anteriormente el objetivo principal cuando se utiliza MPC es minimizar la función de costo (1) para esto se calcula el jacobiano ecuación (2) de la función de costo:

$$J = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \quad (2)$$

Para realizar la minimización de la función de costo se utiliza el método de Newton-Raphson para hallar el valor donde el jacobiano es mínimo, a continuación se presentarán los pasos del algoritmo de minimización:

El algoritmo de Neuton-Raphson tiene la siguiente forma:

$$U(k+1) = U(k) - \left(\frac{\partial^2 J}{\partial U^2}(k)\right)^{-1} \frac{\partial J}{\partial U}(k) \quad (3)$$

Donde *U(k+1)* es un vector donde se almacenan los valores de la señal de control **u** y tiene la siguiente forma:

$$\boldsymbol{U(k)} \triangleq \begin{bmatrix} u(n+1) \\ u(n+2) \\ \vdots \\ u(n+N_u) \end{bmatrix}, k = 1, \ldots, \#iteraciones \qquad (4)$$

Y la matriz hessiana:

$$\frac{\partial^2 J}{\partial u^2}(k) \equiv \begin{bmatrix} \frac{\partial^2 J}{\partial u(n+1)^2} & \cdots & \frac{\partial^2 J}{\partial u(n+1)\partial u(n+N_u)} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u(n+N_u)\partial u(n+1)} & \cdots & \frac{\partial^2 J}{\partial u(n+N_u)^2} \end{bmatrix} \qquad (5)$$

Este proceso se repite hasta que el porcentaje de cambio de los valores de **U** sea menor que un error determinado.

Resolviendo la función de costo para el cálculo del jacobiano se obtiene:

$$\frac{\partial J}{\partial u(n+h)} = -2\sum_{j=N_1}^{N_2}[ym(n+j) - yn(n+j)]\frac{\partial yn(n+j)}{\partial u(n+h)} +$$

$$2\sum_{j=N_1}^{N_2}[\Delta u(n+j)]\frac{\partial \Delta u(n+j)}{\partial u(n+h)}, h = 1, \ldots, N_u \qquad (6)$$

Donde $\frac{\partial \Delta u(n+j)}{\partial u(n+h)} = \frac{\partial}{\partial u(n+h)}(u(n+j) - u(n+j-1))$ se puede escribir en términos de la función Delta de Kronecker:

$$\frac{\partial u(n+j)}{\partial u(n+h)} - \frac{\partial u(n-1+j)}{\partial u(n+h)} = \delta(h,j) - \delta(h,j-1) \qquad (7)$$

Donde $\delta(h,j)$ se define como:

$$\delta(h,j) = \begin{cases} 1 \; si \; h = j \\ 0 \; si \; h \neq j \end{cases} \qquad (8)$$

Cada uno de los elementos de la matriz hessiana está dado por la expresión:

$$\frac{\partial^2 J}{\partial u(n+m)\partial u(n+h)} = \sum_{N_1}^{N_2}\left\{\frac{\partial yn(n+j)}{\partial u(n+m)}\frac{\partial yn(n+j)}{\partial u(n+h)} - \frac{\partial^2 yn(n+j)}{\partial u(n+m)\partial u(n+h)}[ym(n+j) - yn(n+j)]\right\} \qquad (9)$$

$$+2\sum_{j=1}^{N_u}\left\{\frac{\partial \Delta u(n+j)}{\partial u(n+m)}\frac{\partial \Delta u(n+j)}{\partial u(n+h)} + \Delta u(u+j)\frac{\partial^2 \Delta u(n+j)}{\partial u(n-1)\partial u(n+h)}\right\}, h = 1, \ldots, N_u, m = 1, \ldots, N_u$$

(10)

Donde el término $\frac{\partial \Delta u(n+j)\partial \Delta u(n+j)}{\partial u(n+m)\,\partial u(n+h)}$ se remplaza por $\frac{\partial \Delta u(n+j)\partial \Delta u(n+j)}{\partial u(n+m)\,\partial u(n+h)} = (\delta(h,j) - \delta(h,j-1))(\delta(m,j) - \delta(m,j-1))$ utilizando la función Delta de Kronecker. El último término que hace falta por calcular es la derivada de la red neuronal.

4. ARQUITECTURA DE LA RED NEURONAL

Para el algoritmo de NMPC el modelo de la planta es una red neuronal. El entrenamiento de la red neuronal se realiza antes de utilizar la red. El diagrama de entrenamiento se muestra en la Figura 2 . Para realizar el entrenamiento de la red neuronal se utilizan las mismas entradas de la planta denominadas *u(n)* y una entrada adicional que es la salida de la planta retrasada un instante de tiempo, esto con el fin de ayudar a la red neuronal a capturar las características de la planta. El error de la red neuronal con respecto a la planta es utilizado para realizar el cálculo de los pesos de la red neuronal utilizando el algoritmo de Levenberg-Marquardt.

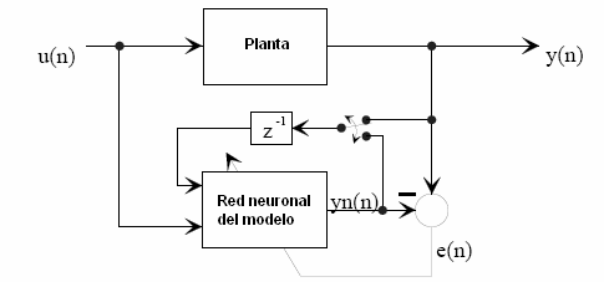

Figura 2. Entrenamiento de la red neurona

A continuación en la Figura 3 se puede observar la arquitectura de la una red con una capa oculta y con las entradas retardadas en el tiempo. Las entradas de la red neuronal consisten de una entrada *u(n)* que es el la señal de control actual y la señal *y(n-1)* que es la salida anterior de la red neuronal, adicional a estas entradas se encuentras los valores anteriores de *u(n)* los cuales están determinados por n_d que es la cantidad de valores anteriores de *u(n)* y los valores anteriores de *y(n-1),* determinados por d_d [2].

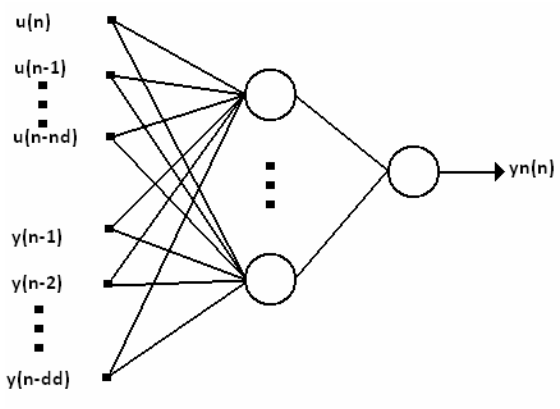

Figura 3. Red multicapa con entradas retardadas en el tiempo
Fuente: *Neural Generalized Predictive Control, A Newton-Raphson Implementation*

Este tipo de arquitectura utiliza como función de activación en la capa oculta la función:

$$y = \frac{2}{1+e^{-2x}} - 1 \quad (11)$$

Y en la capa de salida utiliza una función identidad de la forma:

$$y = x \quad (12)$$

La red neuronal de la **¡Error! No se encuentra el origen de la referencia.** se puede expresar de la siguiente manera:

$$yn(n) = \sum_{j=1}^{hid} w_j f_j\left(net_j(n)\right) + b$$

$$net_j(n) = \sum_{i=0}^{n_d} w_{j,i+1} u(n-i) + \sum_{i=1}^{d_d} w_{j,n_d+i+1} y(n-i) + b_j \quad (13)$$

Donde:

- $yn(n)$: salida de la red neuronal
- $f_j(\cdot)$: salida de la función de activación en la j^{th} neurona de la capa oculta
- $net_j(n)$: Valor de entrada para la j^{th}neurona de la capa oculta
- hid : Cantidad de neuronas en la capa oculta
- n_d : es la cantidad de valores retardados relacionados con la señal de control **u,** sin tener en cuenta el valor actual**.**
- d_d : es la cantidad de valores retardados relacionados con y (.)
- w_j: son los pesos sinápticos de la capa de salida.

5. CALCULO DE LA DERIVADA DE LA RED NEURONAL

Para minimizar la función de costo utilizando el algoritmo de Newton-Raphson es necesario realizar el cálculo de la derivada para la red neuronal, la cual se realizara a continuación.

Para evaluar el jacobiano y la matriz hessiana, el cálculo de la primera derivada y la segunda es necesario, este se realiza con respecto a n(n+h) donde:

$$\frac{\partial yn(n+k)}{\partial u(n+h)} = \sum_{j=1}^{hid} w_j \frac{\partial f_j(net_j(n+k))}{\partial u(n+h)} \quad (14)$$

A la ecuación anterior es necesario aplicarle la regla de la cadena a $\partial f_j(net_j(n+k))/\partial u(n+h)$ para el cálculo de la derivada, lo cual da como resultado:

$$\frac{\partial f_j(net_j(n+k))}{\partial u(n+h)} = \frac{\partial f_j\left(net_j(n+k)\right)}{\partial net_j(n+k)} \frac{\partial net_j(n+k)}{\partial u(n+h)} \quad (15)$$

Donde $\partial f_j(net_j(n+k))/\partial u(n+h)$:

$$\frac{\partial net_j(n+k)}{\partial u(n+h)} = \sum_{i=0}^{n_d} w_{j,i+1} \begin{cases} \delta(k-i,h), k-N_u < i \\ \delta(N_u,h) \quad , k-N_u \geq i \end{cases} + \sum_{i=1}^{\min(k,d_d)} w_{j,i+n_d+1} \frac{\partial yn(n+k-i)}{\partial u(n+h)} \delta_2(k-i-1) \quad (16)$$

Con el cálculo de la primera derivada ya es posible calcular el jacobiano solo falta realizar la segunda derivada para calcular la matriz hessiana. Derivando las ecuaciones se obtiene la segunda derivada de la red neuronal:

$$\frac{\partial^2 yn(n+k)}{\partial u(n+h)\partial u(n+m)} = \sum_{j=1}^{hid} w_j \frac{\partial^2 f_j(net_j(n+k))}{\partial u(n+h)\partial u(n+m)} \quad (17)$$

Donde:

$$\frac{\partial^2 f_j\left(net_j(n+k)\right)}{\partial u(n+h)\partial u(n+m)} = \frac{\partial f_j\left(net_j(n+k)\right)}{\partial net_j(n+k)} \frac{\partial^2 net_j(n+k)}{\partial u(n+h)\partial u(n+m)}$$

$$+ \frac{\partial^2 f_j(net_j(n+k))}{\partial net_j(n+k)^2} \frac{\partial net_j(n+k)}{\partial u(n+h)} \frac{\partial net_j(n+k)}{\partial u(n+m)}$$

$$\frac{\partial^2 net_j(n+k)}{\partial u(n+h)\partial u(n+m)} =$$

$$\sum_{i=1}^{min(k,d_d)} w_{j,i+n_d+1} \frac{\partial^2 yn(n+k-i)}{\partial u(n+h)\partial u(n+m)} \delta_1(k-i-1) \quad (18)$$

6. DESEMPEÑO DEL CONTROLADOR

a continuación se presentan los resultados obtenidos de la implementación de el controlador utilizando como software de simulación Matlab.

Se puede observar en la función de costo, que el segundo factor de la ecuación encargado de asegurar que los cambios en las señales de control sean mínimos, esta multiplicado por una constante denominada α; está constante puede tomar valores entre 0 y 1, y es utilizada para darle relevancia o no al segundo factor de la función de costo, a continuación se puede observar el efecto que tiene α en el desempeño del controlador.

Para las pruebas a continuación se llevara la planta a dos estados, primero la salida deseada de la planta es 0.5 y después de 100 segundos se cambia el valor por 0.9, esto se realizara para $\alpha = [0, 0.5, 1]$

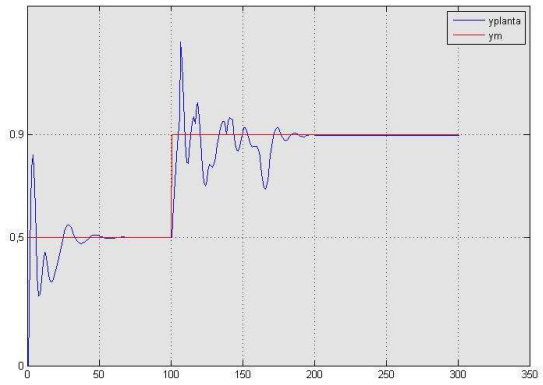

Figura 6: Comportamiento Deseado Vs Respuesta de la planta para $\alpha = 0$

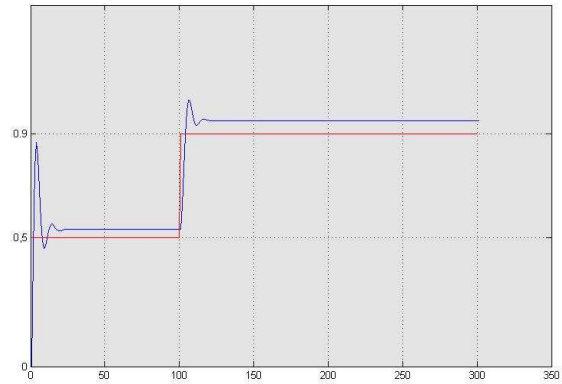

Figura 4: Comportamiento Deseado Vs Respuesta de la planta para $\alpha = 0.5$

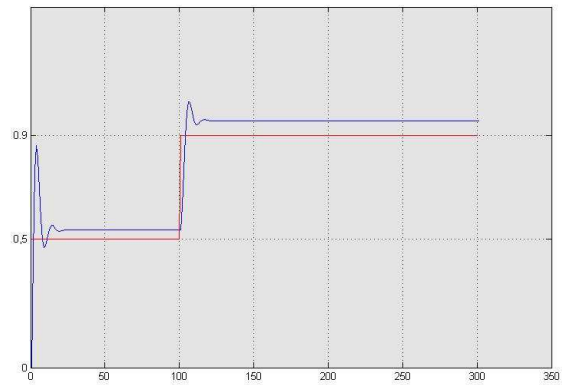

Figura 5: Comportamiento Deseado Vs Respuesta de la planta para $\alpha = 1$

7. CONCLUSIONES Y RECOMENDACIONES

Se puede observar el efecto que tiene α en el comportamiento del controlador asegurando que no se den cambios bruscos en la señal de control esto es una característica muy importante que debe tener el controlador para asegurar la estabilidad del sistema.

Para valores de α cercanos a 1 se incrementa el error de estado estable del controlador de la planta, es por esto que se debe encontrar un equilibrio entre estabilidad y precisión.

El desempeño del controlador en tiempo de convergencia se ve afectado cuando la cantidad de iteraciones se incrementa, por esto se recomienda que para una implementación de dicho controlador se deben utilizar lenguajes de programación diferentes a Matlab que permitan incrementar la velocidad de ejecución de el controlador.

8. BIBLIOGRAFÍA

[1]. **D W, Clarke, Mohtadi, C and P S Tuffs.** *Generalized Predictive Control.* 1984.

[2]. **Soloway, Donald and J. Haley, Pamela.** *Neural Generalized Predictive Control, A Newton-Raphson Implementation.* NASA Langley Research Center. Hampton : s.n., 1997.

[3]. **J. A, Rossiter.** *Model-based Predictive Control: A Practical Approach.* s.l. : CRC Press, 2003.

[4]. **Ollero Baturone, Aníbal.** *Robótica: Manipuladores y robots móviles.* s.l. : Marcombo, 2001.

[5]. **CYTED-Conicit.** *aplicaciones de las redes neuronales en supervision, diagnosis y control de procesos* . s.l. : Equinoccio, 1999.

[6]. **Sarangapani, Jagannathan.** *Neural Network Control of Nonlinear Discrete-Time Systems.* s.l. : CRC/Taylor & Francis, 2006.