'89 韓國自動制御學術會議論文集 1989. 10. 27~28

WHILE 언어를 사용한 32 비트(MC 68020) CPU 제어기에 대한 직접구동방식 로보트의 제어소프트웨어 개발

이주장 신은주 곽윤근

한국과학기술대학 로보틱스연구실

Control Software Development for Direct Drive Arm Robot Using 32 bit(MC 68020) CPU with WHILE Language

Ju-Jang Lee Eun-Ju Shin Yoon-Keun Kwak

Korea Institute of Technology, Robotics Lab.

Abstract

This paper represents the control software development for Direct Drive Arm Robot with the WHILE language composed the 68000 assembly language and high level language modula-2. Direct Drive Arm Robot needs the control program which is fast step and exactly position moving because Direct Drive Arm Robt depends on accuracy.

We found that the self-tuning algorithm for this robot control is very good for the real time control and the floating point operation using the 32 bit CPU(MC 68020) controller

1. 서 론

자기동조제어는 과거 15년동안 활발한 연구분야중의 하나였다. 이 기간동안 안정성(Stability) 과 강건함(Robustness) 같은 이론적인 문제들에서부터 전체적인 동작과 기준위치다음에 뒤따라 오는 점들에 대한 것등 적용의 문제에 이르는 곳까지 많은 문제들이 여러가지 Method 에 의해서 해결되었다.

과거 자기동조제어에 대한 연구는 크게 이론적인 분야, 수식적인 분야, 적용의 분야의 이 세가지 분야에서 어려움을 가졌었다. 그러나 이러한 초기적인 문제는 예보의 구간이 긴 제이기의 이용이나, 가변성의 망각인자(Forgetting Factor) 가 함께 개발되어짐에 따라 극복되어졌다. 이처럼 많은 연구로 이룩된 진보에도 불구하고 자기동조제어기는 널리 보급되어 사용되지 않았고, 제어의 발전면에서 고전적인 삼항조절기와 교체할 어떤 움직임도 보이지 않았다.[1]

이것을 설명하기 위한 이유중의 한가지가 자기동조 제어기는 매개변수를 고정시킨(Fixed- parameter) 고전적인 알고리즘들보다 더 복잡하다는 것이다. 자기동조제어기에서의 매개변수 측정기는 제어가 수행되고 있는 동안 계속 동작되어져야 하고 또한 매개변수의 갯수도 매우 많다. 그러나 이 계산상의 문제는 매우 복잡한 제어시스템의 극히 작은 일부분이기 때문에 이유로서 충분하지가 않다. 중요한 것은 제어계산의 속도를 향상시키기 위한 기술의 발전이 뒤따르지 않았다는 것이다.[2]

이것을 해결하기 위한 것이 자기동조제어의 소프트웨어화이다. 이를 위해 기계적인 알고리즘이 컴퓨터 프로그램으로 변화되어져야 하고, 시스템의 나머지 부분들도 상호 연결되어야 한다.

본 연구는 부동 소숫점 연산 및 실시간 제어를 실현하기 위하여 고속 다기능을 가진 32 비트 마이크로프로세서(MC 68020)를 사용하여 직접구동방식 로보트의 제어기를 설계하였고, 자기동조제어알고리즘을 이용하여 Modula-2 와 68000 어셈블리로 구성된 WHILE 언어를 사용 제어프로그램을 개발하는데 목적을 두고 있다.

2. 직접구동방식 로보트의 제어기 설계

직접구동방식 로보트의 제어기는 세개의 부분으로 구성되어 있다. 통신과 제어계산기(Communications and Control Computer), 축제어계산기(Axis Control Computer)와 이중포트램(Dual-Port RAM) 이다. 이들을 기초로 한 Main 제어보드의 구성도는 그림 1. 에 보여진다.

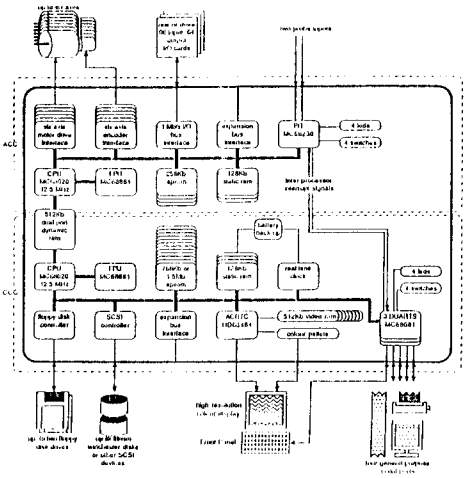


그림 1. Main 제어기 구성도

2.1 통신과 제어계산기(CCC) 의 구성

이 부분에서 사용된 사용된 각 재원은 다음과 같다.

. 프로세서: 12.5MHz 에서 동작하는 모토롤라의 MC 68020 32 비트 마이크로프로세서이고 필요에 따라 부동 소숫점 처리 IC 인 MC 68881 co-processor 를 부착할 수 있다.

. 메모리: 4개의 32 핀 스태틱 램이 있다. 이들 각각은 32K byte 의 메모리를 가지고 있어 총 128K byte 의 메모리 용량을 가지고 있다. 사용시 1M 비트의 칩을 사용한다면 512K byte 까지 메모리를 확장할 수 있다. 이 램들은 외부 전원이 공급되지 않거나 또는 순간 정전시에도 램의 내용이 지워지지 않도록 백업 배터리가 부착되어 있다.

12개의 32핀 EPROM 은 프로그램 저장에 사용되며 용량은 768K byte 이며 최대 1.5M 까지 확장할 수 있다.

256/1024 비트 Serial EEPROM 은 매개변수 저장에 사용된다.

. 통신링크: CCC 위에 있는 MC 68881 DUART 를 이용하기 위해 6개의 비동기 채널을 가지고 있다. 주변장치, DNC 링크등 연결에 사용되는 RS 232/423 인터페이스와 함께 공급되어진 공용의 디버깅 채널이다. 9600 보트레이트의 키보도와 함께 통신에 사용되는 RS 422 채널과는 전기적으로 분리되어져 있다.

. 플로피 디스크제어기: Western Digital 1772 FDC 칩을 사용하여 많은 양의 데이타 저장을 위해 두개의 표준 31/2 인치 플로피 디스크가 하드웨어와 인터페이스하도록 되어 있다.

. SCSI 제어기: LS 380 소형컴퓨터 인터페이스(Small Computer System Interface) 결합부분이다. 이것은 MCB 에 연결된 초고속 대용량의 주변장치를 사용 가능하게 한다.

2.2 축제어 계산기(ACC) 의 구성

. 프로세서: 12.5 MHz 에서 동작하는 모토롤라의 MC 68020 32비트 마이크로프로세서를 사용하고 필요에 따라 MC 68881 을 부착할 수 있다.

. 메모리: 4개의 28핀 스태틱 램이 있고 이들 램 각각의 용량은 32K byte 로 ACC 은 총 128K byte 의 메모리 용량을 가진다.

. I/O 버스: ACC 위에 있는 I/O 버스 인터페이스이다. 이 버스는 RS 422 의 드라이브와 리시브들에 연결되어진다. I/O 버스 인터페이스는 MCB(Main Control Board)로 부터 전기적으로 분리되어 있고 자신의 Power 공급기를 가지고 있다.

. 축제어: 6개의 축 채널이 있다. 이 각각은 구형의 엔코더 인터페이스를 가지고 있다. 디코더는 이 구형신호를 Forward 와 Reverse 펄스속으로 보낸다. 카운터는 이 펄스들을 센다. 또한 엔코더 인덱스 펄스들을 모니터링하기 위한 장치가 있고 구형 입력속의 연속된 Error 들을 찾아낼 수도 있다. 각 채널은 모터 구동에 적합한 ±10V 을 출력하기 위해 12 bit 의 전압 DAC 을 가지고 있다.

. Probe 인터페이스: Probe(Single-ended TTL 또는 RS 422 의 다른 인터페이스)들의 연결을 위한 두개의 입력이다. 이것은 위치 감지에 대해 매우 빠른 응답을 줄 수 있는 소프트웨어의 인터럽트 구조속에 통합되어질 수 있다.

2.3 Dual-Port RAM(DPR) 의 구성

MCB(Main Control Board) 상의 DPR 부분은 통신과 제어계산기(CCC) 와 축제어계산기(ACC) 부분들사이에 공유되어지는 공간으로 32 비트의 512K byte 의 용량을 가지고 있다. 이것은 Dual-port 적용에 사용될 예정으로 SCC74F765 DRAM 제어기에 의해 관리되어진다.

3. 자기동조방식의 제어알고리즘

자기동조제어방식은 간접제어방식과 직접제어방식으로 나눌 수 있다. 간접제어방식은 먼저 시스템의 매개변수를 추정하여 이를 참값이라 가정하고 이를 이용해 제어기 매개변수를 구하는 방식이고 직접제어방식은 시스템의 입력과 출력만을 측정하여 제어기 매개변수를 직접 추정하는 방식으로 계산량이 적은 장점이 있다. 직접제어방식의 구조는 그림 2. 와 같다.

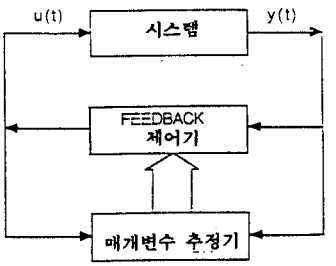


그림 2. 직접제어방식의 자기동조구조

많은 Self-turning 방법이 그렇지만 특히 직접제어방법은 예보적인 디자인을 기초로 한다. 예보제어이론은 시스템 지연상수가 K 일때, 현재 제어입력 u(t) 에 의해 영향을 받는 첫번째 출력 y(t + k) 의 방정식이다.[3].

$$y(t+k) = \frac{B(Z^{-1})}{A(Z^{-1})}u(t) + \frac{C(Z^{-1})}{A(Z^{-1})}e(t+k) \quad \text{----(1)}$$

여기서 예보의 범위는 시스템 지연 K 이고, 이 제어의 효과는 예보 주기 K 와 Disturbance 의 특성에 정확하게 따르는 예보의 정확도에 따른다. 이때 입출력의 결과로 만들어진 $A(Z^{-1})$, $B(Z^{-1})$, $C(Z^{-1})$ 의 모르는 계수 값을 찾아내기 위해 매개변수 추정기를 사용하고 이 매개변수 추정 알고리즘과 연관된 제어법칙으로 시스템을 제어한다.

추정의 갱신 방법은 아래와 같다.[4]

$$\begin{bmatrix}\text{new}\\ \text{estimate}\end{bmatrix} = \begin{bmatrix}\text{old}\\ \text{estimate}\end{bmatrix} + \text{gain} \times (\text{prediction error of the old mode})$$

또한 제어법칙은 변화의 최소화(Minimum Variance) 을 사용하였다. 이것은 이름이 의미하는 것처럼 출력 error 의 평균값을 최소로 하는데 기초를 둔다.

식 (1)에서 C/A 항을 표현하면[3]

$$\frac{C(Z^{-1})}{A(Z^{-1})} = E(Z^{-1}) + Z^{-k}\frac{F(Z^{-1})}{A(Z^{-1})}$$ 이고,

이식을 다시 쓰면

$$Ay(t+k) = Bu(t) + Ce(t+k) \quad \text{----------} \quad (2)$$

$$C = E \cdot A + Z^{-k} \cdot F \quad \text{----------} \quad (3)$$

이다.

각 항에 $E(Z^{-1})$ 을 곱하면

$$E \cdot A \cdot y(t+k) = E \cdot B \cdot u(t) + E \cdot C \cdot e(t+k) \quad \text{---} \quad (4)$$

$$C \cdot y(t+k) - F \cdot y(t) = G \cdot u(t) + E \cdot C \cdot e(t+k) \quad \text{---(5)}$$

이다. 여기서 $G(Z^{-1}) = E(Z^{-1}) \cdot B(Z^{-1})$ 이다.

따라서

$$y(t+k) = \frac{Fy(t) + Gu(t)}{C} + E \cdot e(t+k) \quad \text{-----(6)}$$

이고 최적예보를 $\hat{y}(t+k \mid t)$, error를 $\tilde{y}(t+k \mid t)$로 정의하고 식(6)을 다시 쓰면

$$C \cdot \hat{y}(t+k \mid t) = F \cdot y(t) + G \cdot u(t) \quad \text{----------} \quad (7)$$

$$\tilde{y}(t+k \mid t) = E \cdot e(t+k) \text{ 이고} \quad \text{----------} \quad (8)$$

$$y(t+k) = \hat{y}(t+k \mid t) + \tilde{y}(t+k \mid t) \quad \text{----------} \quad (9)$$

이다. 예를 들어 한 cost 함수 $I = E\{y^2(t+k)\}$ 을 정의하면

$$I = E\{Y^2(t+k)\} = E\{[\hat{y}(t+k \mid t) + \tilde{y}(t+k \mid t)]^2\}$$
$$= [\hat{y}(t+k \mid t)]^2 + E\{[\tilde{y}(t+k \mid t)]^2\} \quad \text{--------(10)}$$

이다. 이 cost 함수가 feedback 제어법칙에 의해 최소화 되어지면

$$Fy(t) + Gu(t) = 0 \quad \text{----------} \quad (11)$$

즉 $u(t) = -(G/F)\,y(t)$ ---------- (12)

이다. 따라서 이것은 K-단계앞에 예보된 error 를 감소시키도록 동작한다.[3]

4. 제어프로그램의 구성[1]

자기동조제어에서 실시간 제어를 위해 사용되는 많은 모델들이 있지만 가장 간단하고 현재의 컴퓨터 제어에서 가장 많이 사용되는 것은 Cyclic Processing Model 이다. 이 방법으로 사이클의 각 요소를 올바르게 동작하기 위해서는 반복주기의 시간이 더 작아져야 한다. 만약 한 요소가 제한시간을 넘어서까지 동작하면 다른 모든 요소들은 그들의 실행속에서 지연된다.

일반적으로 매개변수추정은 제어신호를 계산하는 것보다 더 많은 계산방법이 필요하다. 그러나 이것은 단지 제어기가 이 변수들을 받아들일 때 추정된 새로운 값이 공급되어지므로 매개변수 추정이 실시간에서 동작할 필요는 없다. 자기동조제어기의 다중작업 구조가 그림 3.과 같다.

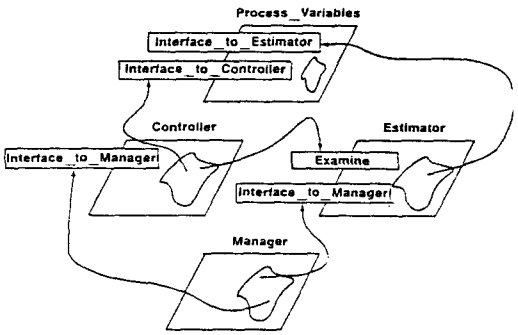


그림 3. 자기동조제어기의 다중작업구조

자기동조제어기를 Process 측정, Set Point 그리고 수행을 위한 생성된 제어신호를 받아들이는 블랙박스로 나타낸 것이 그림 4 이다.

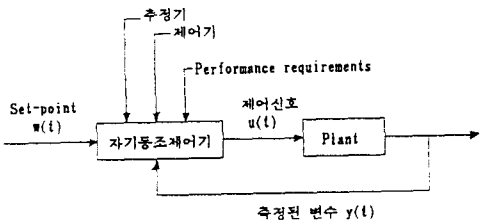


그림 4. 자기동조제어의 블록선도

이 자기동조제어기는 Recursive Estimation Update, a set of Performance Requirements 그리고 Control Calculation 으로 구성되어 있다.

o Self-tuning Controller

. Performance - requirements

Prediction - horizons

Reference - Model - speed

Control - cost

. Estimator

Covariance - Matrix

Forgetting - factor

Parameter - estimates

. Controller

Control - bid

Control - signal

Control - state

. Process - variables

filtered - varialbes

. Performance - requirements 은 예보범위, 기준모델속도, 제어 cost 의 값들을 정한다.

. Estimator 은 초기상태, 곱셈인자에 의해 커질 수 있는 상관계수행렬을 준비하고, 망각인자(Forgetting Factor)를 수정할 수 있다. 추정기로 부터의 매개변수 추정은 제어기를 위해 제거되어지거나 또는 받아들여질 수 있다.

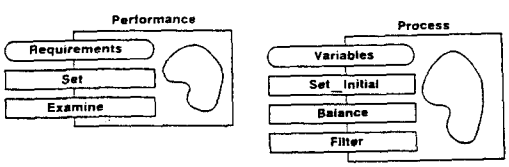
. 제어기는 Process 명령의 안이나 밖에 있다. 그러나 항상 상태(state)와 Process Variables 을 제외하고 Process 명령 자체를 기초로 제어명령을 계산해야만 한다.

제어신호는 Process 을 실행하기 전에 가속기에 의해 생략되어질 수도 있다. Desaturation 은 제어계산후에 수행되어진다.

. Process Variables 은 시작점(Startup) 과 균형을 이루어야만 하고 이는 추정기와 제어기를 위해 준비되어진 data 를 여과시킨다.

. 제어와 추정 알고리즘은 각각 독립적으로 수행해야만 한다.

이 각각에 대한 실행방법이 "Booch"의 그래프를 이용하여 그림 5에 나타나 있다.



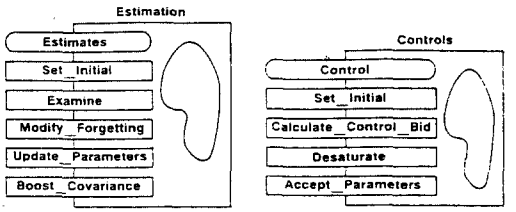


그림 5. 자기동조제어의 실행도

본 프로그램 구성의 일부분이 부록에 수록되어 있다.

5. 결 론

본 연구에서는 자기동조제어기의 디자인을 소프트웨어화 하는 한가지 방법을 구성하여 보았다. 본 프로그램은 Modula-2로 개발되고 있으며 제어기의 하드웨어 동작을 위해 저급언어인 68000 어셈블러로 구성된 프로그램의 개발도 필요하다. 앞으로도 자기동조제어기의 소프트웨어화에 있어서는 실시간 제어, 다중처리 제어기, 시스템들의 상호 관계 등 많은 분야에서 활발한 연구가 필요하다.

Reference

(1) D.P Atherton an Dr K. Warwick, "Implementation of Self-tuning Controllers", IEE Control Engineering Series 35, 1988

(2) C.J. Harris and S.A. Billings, "Slef-tuning and Adaptive Control: Theory and Applications"

(3) Mihailo Ristic, "Dynamic Analysis and Control of Anthropomorphic Robots", Imperial College, pp 181 - 209, 1986

(4) University of Oxford Department of Engineering Science, "Self-tuning and Robust Control Design", University of Oxford, 1988

부록

```
MODULE Main_STC;
================

FROM PERFORMANCE_REQUIREMENTS   IMPORT SETPRQ;
FROM ESTIMATOR                  IMPORT SETEST;
FROM CONTROLLER                 IMPORT SETCON;
FROM PROCESS_VARIABLES          IMPORT SETVAR;


CONST

      UMAX          =     100.0;
      UMIN          =    -100.0;
      H             =       0.1;

BEGIN

      SETPRQ(1,10,1,2.0,0.1,PRG);
      SETSET(2,2,1,1000.0,0.9,EST);
       ....

END Main_STC.


MODULE PERFORMANCE_REQUIREMENTS;
================================

VAR

 N1     : INTEGER;
  (* Prediction horizon start for error signal *)
 N2     : INTEGER;
  (* Prediction horizon end for error signal *)
 NU     : INTEGER;
  (* Horizon on Control signal *)
 PTC    : REAL    ;
  (* Reference model time constant in Samples *)
 RLAM   : REAL    ;
  (* Cost on Control signal increments *)


PROCEDURE SETPRQ(N1,N2,NU,PTC,RLAM,PRQ);
----------------------------------------

VAR

 PRQ : ARRAY [1..6] OF REAL;
  (* Contains static information about
     performance requirements of controller *)

BEGIN

      PRQ[1] := FLOAT(N1);
      PRQ[2] := FLOAT(N2);
      PRQ[3] := FLOAT(NU);
      PRQ[4] := 1.0;
      PRQ[5] := -EXP(1.0/PTC);
      PRQ[6] := PLAM;

END SETPRQ;


 BEGIN

   INITIAL
     (* Initialize performance requirements
        in term of the prediction horizons,
        the reference model speed, and the
        control cost *)
   SETPRQ(N1,N2,NU,PTC,RLAM,PRQ);
   ....

 END PERFORMANCE_REQUIREMENTS.


 MODULE ESTIMATOR;
 =================

 VAR

  NA    :  INTEGER ;
  (* Number of A coefficients to estimate *)
  NB    :  INTEGER ;
  (* Number of B coefficients to estimate *)
  K     :  INTEGER ;
  (* Lower bound on delay of process *)
  PCOV  :  REAL    ;
  (* Initial covariance value *)
  FORG  :  REAL    ;
  (* Forgetting factor between 0.0 and 1.0*)

 PROCEDURE SETEST(NA,NB,K,DCON,FORG,EST);
 ----------------------------------------

  VAR

   EST   :  ARRAY [1..100] OF REAL;
   (* Contains information about the structure
    of estimator and workspace for calculation
     of parameter estimates. *)

  BEGIN

     ....

  END SETEST;
```