FORTRAN언어로 작성된 로보트 연속 경로 제어 알고리즘의 하드웨어적 실현

정광조 · 정재문 · 박종구*

A Hardware Implementation of the Continuous-Path Control Algorithm for Robot

*Kwang Jo, Chung · Jae Moon, Chung · Jong Gu, Park**

1. 서 론

산업용 로보트 뿐 아니라 機構學的 構造를 가지고 있는 대부분의 기계운동은 수학적 모델링이 가능하다. 따라서 이들을 제어하는 것은 그러한 모델의 입출력 사이의 관계를 규명하는 알고리즘이 얼마나 효율적으로 구성되었는가 하는데 그 성패가 달려 있다고 볼 수 있다.

5軸 이상의 多關節形 로보트에서도 그 문제는 더욱 현실화되어 많은 알고리즘이 개발되고 解가 존재하지 않는 경우에 대해서도 近似解法 등의 방법으로 결국 解는 구할 수 있게 되었다 할 수 있다.

그러나 이러한 알고리즘이 凡用 컴퓨터를 바탕으로 만들어 졌고 解를 얻기까지의 Running시간이 상당히 길어 진다면 실시간 제어를 요구하는 로보트의 제어 알고리즘으로는 부적합하다고 볼 수 있다.

본 논문에서는 한국기계연구소에서 개발한 KIMMBOT-Ⅱ의 CP제어용 FORTRAN언어를 제어장치에 實裝하는 과정을 중심으로 고찰하였다.

2. KIMMBOT-Ⅱ의 CP제어 시퀜스

2.1 KIMMBOT Ⅱ의 제어 알고리즘

보통 다관절 로보트에 응용되는 제어 알고리즘에 사용되는 Direct kinematics는 여러방법이 있으나 모두 실용성이 있고 어떠한 형상의 로보트에도 잘 적용된다. KIMMBOT-Ⅱ에서는 그중

* 창원본소 자동제어실 : Member of Automatic Control Lab.

Homogeneous transformation에 의한 방법을 채택하였다.

그러나 그 逆의 과정인 Inverse kinematics는 로보트의 형태에 따라 Closed form의 解를 구하지 못하는 경우 Iteration방법으로 近似値를 구해야 한다.

KIMMBOT-Ⅱ의 형상은 5축의 Offset distance가 있음으로 인하여 Closed form을 찾지 못하고 Newton-Raphson에 의한 Iteration으로 계산하였다.

이러한 解法을 가진 로보트에서는 계산시간이 상당히 소요되기 때문에 高速演算 Processor의 도움을 받아야 하고 실제 Implementation에 상당한 제약이 따른다.

또한 Algorithm의 복잡성으로 인하여 High level언어상의 디버깅과 보조 Software가 필요하게 된다.

2.2 CP제어 시퀜스

KIMMBOT-Ⅱ에서 채택한 CP수행 시퀜스는 그림 1과 같다.

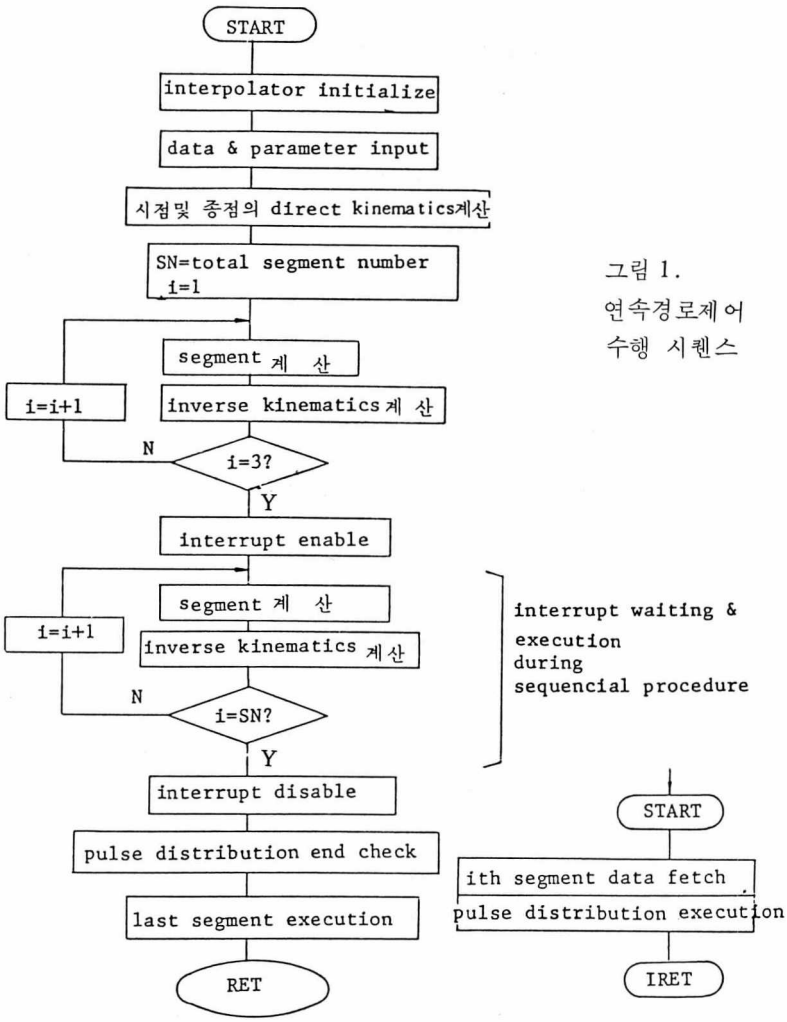


그림 1.
연속경로제어
수행 시퀜스

이중에서 始點과 終點 그리고 각 Segment에서의 Direct kinematics와 그 Inverse는 FORTRAN으로 작성되어 컴퓨터로 충분한 디버깅을 거친 후 주 프로그램과 Link시켜 사용되었다.

그림에서와 같이 실시간 Pulse분배를 행하는 경우에는 계산시간중에도 Interrupt에 의해 이미 계산된 세그먼트를 수행시켜야 하므로 계산속도는 샘플링 시간과 직접적인 관계를 가지게 되므로 FORTRAN으로 작성된 Program을 제어시스템이 얼마나 빨리 계산해 내는가 하는 것이 그 制御裝値의 성능을 대변한다고 하여도 이의가 없을 것이다.

3. Hardware for FORTRAN algorithm process

KIMMBOT-Ⅱ에서 채택한 CP처리시스템은 8086CPU와 8087NDP(Numeric data processor)인데 이러한 구성은 계산기능의 비중이 큰 多關節 로보트의 경우 상당히 강력한 구성이라고 볼 수 있다.

그 장점을 몇가지로 요약해 보면

○ 실장의 편이성

8087은 Coprocessor이기 때문에 86CPU와 거의 모든 Line이 Direct 접속되고 부수적인 Interface가 필요하지 않아 Implementation이 간편하고 더불어 가격, 회로 크기면에서도 유리하다.

○ 정확성

8087의 내부 Register가 80bit로 되어 있어서 일반 컴퓨터의 Resolution보다 유효숫자 3자리 이상과 10배 이상의 Exponential range를 가지고 있다.

○ 계산속도

8087 시스템과 몇 가지의 PC 또는 컴퓨터와의 계산속도를 비교해 보면 표1과 같이 대략 중형컴퓨터의 속도에 육박하고 있음을 볼 수 있다.

표1. 8087 speed benchmarks(seconds)

PROGRAM／COMPUTER	50x50 matrix multiply	5000 square roots
APPLEII+BASIC	1796	130
IBMPC+INTERPRETER	1200	52
IBMPC+COMPILER	140	6
8087 routine	8	0.35
DEC2060+BASIC	5.2	0.4
VAX780+FORTRAN	1.6	0.2
IBM3081+BASIC	0.11	0.22

4. FORTRAN－86 프로그램 development process

4.1 개요

FORTRAN으로 작성된 프로그램은 최종적으로 주 프로그램과 연결시켜 Running되기까지 그림 2와 같은 과정을 거치게 된다. 마지막에 얻어지는 것은 실제 Controller의 ROM에 기억시킬 수 있는 절대 Object code가 된다.

실제로 이 과정은 컴파일러, Locator, Linker 등의 많은 Utility software가 Back-up되어야 하지만 Language가 FORTRAN이 아닌 PASCAL이나 PL/M이라 하여도 컴파일러 이외의 과정은 동일하다.

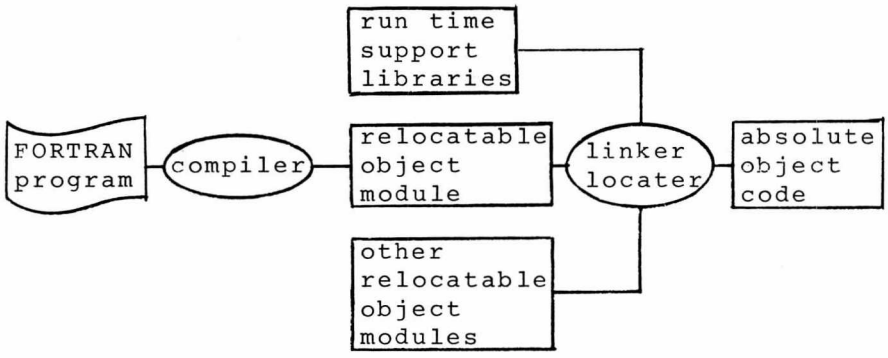


그림 2. FORTRAN-86 Program development process

4.2 FORTRAN compiler

FORTRAN으로 작성된 프로그램은 Compiler를 통해 86 또는 87 CPU의 언어로 번역이 된다. 예를 들면 S2＝SQRT(1－C2*C2)와 같은 FORTRAN은 다음과 같은 8087언어만으로 번역된다.

```
FLD     CS:CONST+2H ; load constant"1"
FLD     ES:C2       ; load C2
FMUL    ST(0)       ;C2*C2
FSUBRP              ;1-C2*C2
FSQRT               ; squre root
FSTP    ES:S2       ; store to S2
```

이때 Compiler는 FORTRAN 1 line마다 8087 내부 Stack을 새로이 사용하기 때문에 가능한 한 중복이 되지 않는 한도에서 Link수를 줄여주는 것이 바람직하고 서브루틴에서의 passing parameter수를 줄이기 위해 Common variable을 많이 사용하는 것이 유리하다.

4.3 Parameter transfer 및 calling procedure

CP algorithm은 주 프로그램에 대한 서브루틴으로 작성하였기 때문에 Call될때 Argument variable들에 대한 Address를 주기 위한 조작이 필요하게 된다.

KIMMBOT-Ⅱ에 사용된 FORTRAN 서브루틴중 Inverse k.를 구하는 부분은 다음과 같은 Argument를 사용한다.

```
SUBROUTIN INVER (DF1-DF6,NV1-NV6)
COMMON/LEE/CDE, SDE, IS, TES
  ⋮
```

여기서 DF1-6는 Inverse된 角軸座標置(radian)이고 NV1-6는 계산된 Feedrate이다.

이때 이 프로그램을 Assembly언어로 된 주 프로그램에서 호출하는 것은 Stack을 통해 각 Argument의 Address를 다음과 같이 넘겨준다.

```
  ⋮
LEA AX,DFI      ; argument address
PUSH DS           passing to stack
PUSH AX
  ⋮
LEA AX, NV6
PUSH DS
PUSH AX
CALL INVER      ;FORTRAN calling
  ⋮
```

4.4 Link & locate process

KIMMBOT-Ⅱ개발과정에서 작성 또는 사용된 모듈 또는 Library는 다음과 같다.

ROBOT.OBJ	; 주 프로그램
RADDEG.OBJ	; 축이동, 모터제어 프로그램
ATINT.OBJ	; CP제어 알고리즘
RINT.OBJ	; CP알고리즘(FORTRAN)
CEL87.LIB	; floating point support library
F86RNØ-4.LIB	; runtime support library
DCON87.LIB	; data conversion library
8087.LIB	; for 8087 used format

이중 RINT.OBJ는 FORTRAN으로 작성하여 컴파일한 것으로 8Kbyte 정도이고 나머지의 Object 모듈은 14Kbyte, 그리고 Runtime library는 실제로 사용되는 것만 Link되어서 Link후의 메모리 비중은 1K 미만이다. 그러므로 총 소모 프로그램은 23Kbyte(데이터 제외)이었다.

5. CP동작실험 및 고찰

5.1 실험

실험은 User가 프로그램을 작성, 입력 및 수행까지의 과정에서 Direct kinematics와 그 Inverse 의 계산 소요시간에 중점을 두었다. 입력한 프로그램 및 위치 데이터는 다음과 같다.

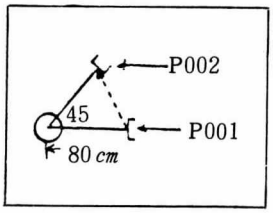
-Program-

```
J000 N000 G00 F05 P000*
     N001 G00 F05 P001*
     N002 G01 F20 P002 S30000*
```

-Position data-

J000 P000	P001	P002
0°	0°	-45°
0°	0°	0°
0°	0°	0°
0°	90°	90°
0°	90°	90°
0°	-90°	-90°

프로그램에서 실제 CP제어(직선)은 N002블록에서 이루어지며 그때 F code는 Feedrate로서 20 ㎝/sec를 입력하였다. CP제어의 位置座標는 P001과 P002로 각각 始點과 終點이 주어진다. 이것은 위치좌표에서 보는 바와 같이 이는 1축만을 45°만큼 이동시킨 점간의 직선보간인데 아암의 길이가 80㎝이므로 직선 이동거리는 대략 60㎝정도(그림 3참조)가 된다.



$D = 2\pi \times 80 / 8$
$= 62.83$ [㎝]

그림 3. CP직선 이동거리

그러므로 샘플링 주기(T)=10msec로 하면 속도 f=20㎝/sec이므로 총 세그먼트 수는

n=1×T/f=60×10/20=30EA로 추정된다.

5.2 결과 및 고찰

위의 入力條件에 따른 동작시험 결과는 다음과 같다.

- segment no ; 30〔EA〕
- direct Ki. 계산시간 ; 19.5〔ms〕
- inverse routine iteration no. ; 3〔times /seg.〕
- inverse Ki. 계산시간(1 segment 당)
  - MAX. : 37.2〔ms〕
  - MIN. : 34.2〔ms〕
  - average : 35.6〔ms〕

실험에서의 시간측정은 사용MDS내부Timer에 의한 측정치이다. 측정기준은 몇 가지로 바꿔 보았으나 계산시간은 별 차이가 없었다.(±1〔ms〕정도). 실험결과에 따르면 이러한 시스템에서의 실시간 펄스분배를 위한 샘플링 주기는 40〔ms〕이상으로 제한된다. 또한 계산과 펄스분배를 분리시켜야 할 경우 샘플링 주기는 4〔ms〕이내로 가능하다. 실제로 실험모델이 30개의 Inverse K.를 계산하는데 소요시간은 측정결과 1158〔ms〕이었다. 이중의 펄스분배 지령시간은 (1158/30)-35.6=3.0〔ms〕로 계산된다. 또한 CP알고리즘의 Inverse k.가 Closed form의 解를 가지는 형상의 로보트인 경우 Inverse 계산시간은 1/4정도 즉, 10〔ms〕 이내로 줄일수 있을 것으로 예상된다.

6. 결 론

본 연구에서는 Offset 축이 있는 NACHI형의 6軸 多關節 로보트의 連續經路제어 알고리즘을 FORTRAN으로 작성하여 8086CPU와 8087 co-processor로 구성된 주 制御裝置에서 실현시켰다.

실험결과에 따르면 이러한 시스템에서는 실시간 펄스분배시 최소 40〔ms〕정도의 샘플링 시간을 얻을 수 있었다. 물론 로보트가 NACHI type이 아니고 PUMA와 같은 Iteration이 필요없는 Type인 경우 샘플링 시간은 10〔ms〕이하로 줄일 수 있다. 또한 40〔ms〕 정도의 샘플링 시간은 아크 용접용 로보트와 같은 경우 상당히 높은 精度를 보장할 수 있을 뿐 아니라 日本産 凡用 로보트의 경우(50-100〔ms〕정도)보다 더 앞선 성능이 보장되리라 본다. 또한 계산기능의 추가가 IC(8087)의 추가만으로 가능하며 전체시스템에 대한 가격비중이 적은 것도 큰 장점이라 볼 수 있다.

《參 考 文 獻》

1. Richard Startz; *"8087 Applications and Programming for the IBM PC and other PCs"*, Prentice-Hall, 1983.
2. Yoram Koren; *"Robotics for Engineers"*, McGraw-Hill, 1985.
3. Rector, Alexy; *"The 8086 book"*, McGraw-Hill, 1980.
4. Richard P. Paul; *"Robot Manipulators"*, MIT press, 1981.
5. *"iAPX 86, 88 User's manual"*, INTEL corp., 1981.