Multiple Dynamic Programming 에 관한 연구

논 문
21～1～2

A Study of Multiple Dynamic Programming

박 영 문*
(Young Moon Park)

Abstract

Dynamic Programming is regarded as a very powerful tool for solving nonlinear optimization problems subject to a number of constraints of state and control variables, but has definite disadvantages that it requires much more computing time and consumes much more memory spaces than other technigues.

In order to eliminate the above-mentioned demerits, this paper suggests a news technique called Multiple Dynamic Programming. The underlying principles are based on the concept of multiple passes that, instead of forming fin lattices in time-state plane as adopted in the conventional Dynamic Programming, the Multiple Dynamic Programming constitutes, at the first pass, coarse lattices in the feasible domain of time-state plane and determines the optimal state trajectory by the usual method of Dynamic Programming, and at the second pass again constitutes finer lattices in the narrower domain surrounded by both the upper and lower edges next to the lattice edges through which the first pass optimal trajectory passes and determines the more accurate optimal trajectory of state, and then at the third pass repeats the same processes, and so on.

The suggested technique insures remarkable curtailment in amounts of computer memory spaces and conputing time, and its applicability has been demonstrated by a case study on the hydro-thermal power coordination in Korean power system.

1. 서 론

制御工學 및 系統工學 분야에서 널리 적용되고 있는 最適化 技法(optimization technique)으로서는 Linear Programming, Quadratic Programming, Geometric Programming, Gradient Method, Conjugate Gradient Method, Maximum Principle, Dynamic Programming 등이 있으나, 系統이 非綜型이고, 制御變數(state variable) 또는 狀態變數(state variable)에 多數의 制限條件(constraint)이 부가되면 체계적인 探索에 기초를 둔 Dynamic Programming에 의하는 것이 유리한 경우가 많다. 그러나, Dynamic Programming은 그 계산결과치의 粗大性, 소요기억용량 및 계산시간의 증가등의 단점을 지니고 있다.

*정회원 : 서울대학교 공과대학 조교수(공학박사)

이와 같은 Dynamic Programming의 단점을 보완하기 위하여 본 논문에서는 Multiple Dynamic Programming 技法을 제안하여, 그 이론적 근거와 장점을 기술함과 동시에 그 효용성을 입증하려 한다.

끝으로 이 새로운 技法을 토대로 한 전자계산기 program을 개발하여 우리나라 전력계통의 水火力協助의 最適化 問題에 실제로 적용한 결과 만족한 결과를 얻었음을 부언한다.

2. 재래의 Dynamic Programming

X를 任意 時刻 t에 있어서의 상태를 표시하는 k次元의 狀態벡터(state vector), U를 狀態를 변화시키는 k'次元의 制御벡터(control vector)라 하면,

$$\frac{d}{dt}\left\{X(t)\right\}=f\left\{X(t),\ U(t),\ t\right\} \tag{1}$$

의 連續動的系統(continous dynamic system)의 汎函數(functional)

$$C\left\{X(o)\right\}=\int_{o}^{t_f} r\{X(t),\ U(t),t\}dt \tag{2}$$

여기서 t_f=終端時間

를 최소로 하는 制御問題는 다음과 같은 離散系統(discrete system)의 制御問題로 近似化할 수 있다. 즉,

$$X_i=X(i-1\cdot\triangle t) \tag{3}$$

$$U_i=U(i\cdot\theta_i\triangle t) \tag{4}$$

여기서, $i=1, 2, \cdots\cdots\cdots, n$

$\triangle t$=時列間隔, $0\leq\theta_i<1$

의 離散量을 가정하고, 制御벡터 U_i는 狀態벡터 X_i를

$$X_{i+1}=T(X_i,\ U_i,\ i) \tag{5}$$

의 變換關係에 의하여 X_{i+1}로 變換한다고 하면, 일반적으로 n個단의 결정과정에 있어서 목적함수(object function)

$$C_n=\sum_{i=1}^{n} r(X_i, U_i, i) \tag{6}$$

를 최소로 하는 n個단의 $U_i[i=1, 2, \cdots\cdots, n]$의 결정 즉

$$g_n(X_1)=\min_{U_i}\left\{\sum_{i=1}^{n} r(X_i,\ U_i,\ i)\right\} \tag{7}$$

를 결정하는 문제로 귀착된다.

그런데 最適性의 원리(principle of optimality)에 의하면, 제 i단 이후의 最適方策(optimal policy)은 그이전의 상태나 制御와는 무관하고 다만 제 i단의 상태 즉 상태벡터 X_i에 의하여 유일하게 결정된다, 따라서 식(7)은

$$g_n(X_1)=\min_{U_1}\left\{r(X_1, U_1, 1)+g_{n-1}(X_2)\right\} \tag{8}$$

또는, 일반적으로

$$g_{n+1-i}(X_i)=\min_{U_i}\left\{r(X_i, U_i, i)+g_{n-i}(X_{i+1})\right\} \tag{9}$$

와 같이 된다. 실제문제에 있어서는 X_i와 U_i는 既知集合 X_{is}와 U_{is}에 대하여

$$X_i \in X_{is} \tag{10}$$

$$U_i \in U_{is} \tag{11}$$

의 制限條件(constraints)을 충족시켜야 하는 경우가 많다. 이 경우, 最適狀態 벡터의 軌跡(trajectory)를 探索(search)하는 통상적인 Dynamic Programming 技法으로서는 X_{is}의 最大値벡터 $X_{is}|_{max}$ 및 最少値벡터$X_{is}|_{min}$에 대하여, 제 i단에서 m個의 等間隔分割로 얻는 $(m+1)$個의 狀態벡터 X_i^j, 즉

$$X_i^j=X_{is}\Big|_{min}+\left\{X_{is}\Big|_{max}-X_{is}\Big|_{min}\right\}\frac{i-1}{m} \tag{12}$$

여기서, $i=1, 2, \cdots\cdots\cdots, n$

$j=1, 2, \cdots\cdots\cdots, m, m+1$

를 택하여 식 (11)의 制限條件下에 식 (9)의 반복식을 만족하는 X_i의 最適狀態벡터의 軌跡 X_i^{oi} 및 最適制御벡터의 軌跡 U_i^{oi}을 표준절차에 따라 결정한다.

3. Multiple Dynamic Programming

여기서 새로이 제안하는 Multiple Dynamic Programming 技法은 時列을 Dynamic Programming에서 처럼 等間隔으로 분활하는 것이 아니라,

$$X_i=X(i-1\cdot\triangle t_i) \tag{13}$$

$$U_i=U(i\cdot\theta_i\cdot\triangle t_i) \tag{14}$$

여기서, $i=1,\ 2,\ \cdots\cdots\cdots,\ n$

$\triangle t_i$=제 i단의 時列間隔

$0\leq\theta_i<1$

처럼 임의의 편리한 간격으로 n個로 분활한 후, 각단의 상태벡터를 미리 m個의 等間隔으로 분활하는 대신, 우선 제1과정에서 $m_1(<m)$個의 粗大한 等間隔으로 분활하여 재래의 Dynamic Programming 技法으로 그 最適狀態벡터의 軌跡을 구하고, 다시 제 2, 3, ……, l과정에 m_2, m_3……, m_e個의 섬세한 간격으로 분활하여 점차로 정밀계산을 행하는 技法으로서, 이를 詳述하면 다음과 같다.

〔제 1 과정〕

제 $i[i=1,\ 2,\ \cdots\cdots,\ n]$단의 狀態벡터 X_i의 可能領域(feasible range)은 식(5), (10) 및 (11)을 동시에 만족시켜야 한다는 조건으로부터

$$X_i \in T(X_{(i-1)s},\ U_{(i-1)s},\ i-1)\cap X_{is} \tag{15}$$

에 의하여 주어지므로, $T(X_{(i-1)s}, U_{(i-1)s}, i-1)\cap X_{is}$의 最大値벡터를 $X_{i1}|_{max}$, 最少値벡터를 $X_{i1}|_{min}$이라 하면, 제 i단의 狀態벡터를

$$X_i^j=X_{i1}\Big|_{min}+\left\{X_{i1}\Big|_{max}-X_{i1}\Big|_{min}\right\}\frac{j-1}{m_1} \tag{16}$$

여기서, $i=1,\ 2,\ \cdots\cdots,\ n$

$j=1,\ 2,\ \cdots\cdots,\ m_1,\ m_1+1$

와 같이, m_1個의 等間隔으로 분활하여, 이로서 얻어지는 (m_1+1)個의 狀態벡터 X_i^j를 택하여 식 (a)의 反復式을 만족하는 最適狀態벡터의 軌跡 X_i^{oi} 및 最適制御벡터의 軌跡 U_i^{oi}를 결정한다.

〔제 2 과정〕

위의 X_i^{oi}은 分割間隔 $\{X_{i1}|_{max}-X_{i1}|_{min}\}/m_1$이 너무 粗大하므로 精度가 낮다. 그런데 最適狀態벡터의 眞經路 $X_i^{o\prime i}$는 閉區間(closed interval) 벡터集合 $[X_i^{Oi}-\{X_{i1}|_{max}-X_{i1}|_{min}\}/m_1, X_i^{Oi}+\{X_{i1}|_{min}/m_1]$인 S_i에 대하여,

$$X_i^{O\prime i} \in T(X_{(i-1)s}, U_{(i-1)s}, i-1)\cap X_{is}\cap S^1{}_i \tag{17}$$

인 경우가 많으므로, $T(X_{(i-1)s},\ U_{(i-1)s}-1)\cap X_{is}\cap S_i^1$의 最大値 벡터를 $X_{i2}|_{max}$, 最少値벡터를 $X_{i2}|_{min}$

이라 하면, 제 i 단에서 m_2個의 等間隔分割로 얻는(m_2+1)個의 狀態벡터 X_i^j 즉,

$$X_i^j = X_{i2}\big|_{min} + \left\{X_{i2}\big|_{max} - X_{i2}\big|_{min}\right\} \tag{18}$$

를 택하여 식(9)의 반복식을 만족하는 最適狀態벡터의 軌跡 X_i^{O2j} 및 最適制御벡터의 軌跡 U_i^{O2i}를 결정한다.

〔제 l 과정〕

일반적으로 第 l 過程에서는 위와 마찬가지 理論으로, 閉區間 벡터集合 〔$X_i^{Ol-1i} - \{X_{i(l-1)}\}_{max} - X_{i(l-1)}\}_{min}\}/m_{l-1}$, $X_i^{Ol-1i} + \{X_{i(l-1)}\}_{max} - X_{i(l-1)}\}_{min}/m_{l-1}$〕 $\cap S_i^l$에 대하여 $T(_{L-1)s}$, $U_{(i-1)s}$, $i-1$) $\cap X_{is}$ $\cap S_i^l$의 最大値벡터를 $X_{il}\big|_{max}$, 最少値벡터을 $X_{il}\big|_{min}$이라 하면 第 i 단에서 m_e個의 等間隔分割로 얻는 (m_l+1)個의 狀態벡터 X_i^j; 즉

$$X_i^j = X_{il}\big|_{min} + \left\{X_{il}\big|_{max} - X_{il}\big|_{min}\right\}\frac{j-1}{m_l} \tag{19}$$

를 택하여 식 (9)의 반복식을 만족하는 最適狀態벡터의 軌跡 $X_i^{Oil}\left(\fallingdotseq X_i^{Oi}\right)$ 및 最適制御벡터의 軌跡 $U^{Oil}\left(\fallingdotseq U_i^{Oi}\right)$을 결정한다.

이상에서와 같이, 第 1, 2, ……, l 과정에서까지 m_1, m_2, ……, m_l의 간격으로 세분하여 가면, 결국 狀態벡터 X_i를 $m_1 m_2 \cdots\cdots m_l/2^{l-1}$個의 等間隔으로 분할한 셈이 되며, 이렇게 분할하여 계산한 결과는 그 精度에 있어서 재래의 Dynamic Programming에서 식(12)에 $m=m_1m_2 \cdots\cdots m_l/2^{l-1}$을 대입하여 계산한 것보다 높거나 같음이 명백하다.

4. Multiple Dynamic Proyramming의 能率

재래의 Dynamic Programming과 Multiple Dynamic Programming의 能率을 비교하기 위하여, 소요계산시간이 대략 狀態벡터의 분할에 의한 選擇數에 비례한다고 가정하면, 제 l 과정까지의 계산을 행한 Multiple Dynamic Programming의 소요계산시간의 재래의 Dynamic Programming의 소요시간에 대한 비, 즉 能率 q는

$$q = \frac{\text{M. D. P. time}}{\text{D. P. time}} \fallingdotseq \frac{m_1+m_2+\cdots+m_l+l}{m+1} \tag{20}$$

과 같이 표시된다. 한편 양자의 분할간격수가 같아야 하므로 즉,

$$m = \frac{m_1\ m_2 \cdots\cdots m_l}{2^{l-1}} \tag{21}$$

의 等式이 성립하여야 하므로 식 (21)을 식 (20)에 대입하면

$$q \fallingdotseq \frac{2^{l-1}\{m_1+m_2+\cdots\cdots+m_l+l\}}{m_1m_2\cdots\cdots m_l+2^{l-1}} \tag{22}$$

와 같이 된다. 그런데 q를 최소로 하는 m_1, m_2, …, m_l을 결정하기 위하여, $\partial y/\partial m_i=0$ 〔$i=1, 2, \cdots\cdots, l$〕의 조건을 구하면

$$m_s = m_1 = m_2 = \cdots\cdots = m_l \tag{23}$$

이 얻어지고, 따라서 식 (23)의 관계를 식 (21)및 (22)에 대입하면,

$$m = \frac{m_s^l}{2^{l-1}} \tag{24}$$

$$q \fallingdotseq \frac{2^{l-1}l(m_s+1)}{m_s^l+2^{l-1}} \tag{25}$$

이 된다. 따라서, Multiple Dynamic Programming의 適用時에는 되도록이면 각 과정의 분활수를 모두 같은 値 m_s로 유지하는 것이 유리하다. 실예를 들자면 Multiple Dynamic Programming을 적용하여 제1과정에서 10분활〔$m_1=10$〕, 제2과정에서 또 10분활 〔$m_2=10$〕한다면, $l=2$이므로, 식 (24)에 의하여, Dynamic Programming에서 50분활 〔$m=10^2/2^{2-1}$〕한 것보다 精度가 높거나 같은 반면 계산속도(소요시간비의 逆數)는 식(25)에 의하여 43% 〔$q=22/51$〕로 단축된다. 이를 다시 제3과정에서 10분활 〔$m_3=10$〕한다면, Dynamic Prorgamming에서 250분활한 것 이상으로 精度가 높으나, 계산속도는 불과 13%〔33/251〕로 단축된다. 따라서 분활수가 높을수록 Multiple Dynamic Programming의 위력이 발휘된다. 뿐만아니라, 어느 한 과정에서는 狀態變數의 선택수가 Dynamic Programming에 의한 것보다 월등히 적으므로 소요기억용량이 현저하게 감소됨이 분명하다.

5. Multiple Dynamic Programming의 적용

남한전역의 연간 발전비용을 최소로 하는 전력계통의 경제적운용을 위하여 한강수계 각 발전소 저수지의 방수량을 最適制御하는 水火力協助問題에 이 Multiple Dynamic Programming에 의한 전자계산프로그램을 개발 적용한 결과 그 효용성이 실증되었다. 한강수계에는 상류에서 부터 화천, 춘천, 의암, 청평순으로 발전소 저수지가 위치하고 있는바, 수도서울을 포함한 流域에 溢流 또는 洪水가 발생해서는 아니된다는 제한조건하에 각 발전소 저수지의 수위를 最適狀態로 변동시킴으로써 연간 화력발전소의 발전원가를 최소로 하는 월별저수지 방수량을 결정하는데 성공하였다.

여기서 狀態벡터는 각 저수지의 수위를, 制御벡터는 각 저수지 월별방수량을, 그리고 목적함수는 연간 발전비용을 뜻한다. 그러므로 Dynamic Programming이 적용될 수 있는 문제는 어떤것이든 Multiple Dynamic Programing도 적용될 수 있으며, 특히 소형전자계산기를 사용할 경우에 그 위력이 발휘된다.

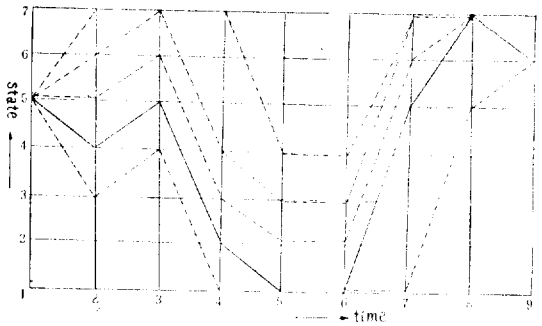

그림. 1. 재래의 D. P.
Fig. 1. Conventional scheme of D. P.

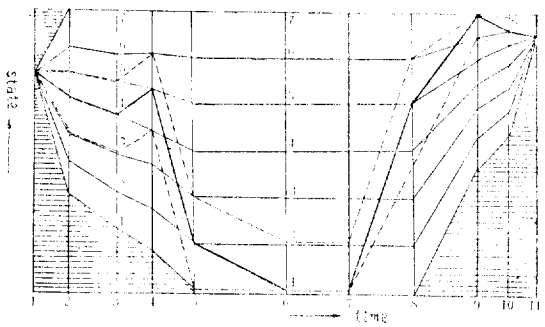

그림. 2. Multiple D. P. 의 제 1 과정
Fig. 2. The first stage of multiple D. P.

6. 결　론

새로이 제안하는 Multiple Dynamic Programming 技法은 재래의 Dynamic Programming 技法의 단점을 보완 한 것으로 다음과 같은 특장이 있다.

(1) 狀態變數의 분활간격수가 양자 동일한 경우, 계산시간이 현저히 단축된다.

(2) 狀態變數의 분활간격수가 양자 동일한 경우, 전자계산기 기억용량이 현저히 절약된다.

(3) 양자의 계산시간이 동일한 경우, 計算精度가 향상되므로 Dynamic Programming 의 경우에 사용되는 精度向上을 위한 補間計算(interpolation)이 불필요하다.

(4) 狀態變數의 분활간격수가 클수록 더욱 유리하다. 그 반면 분활간격수가 너무 적으면(12분활 이하) 효용성이 없다.

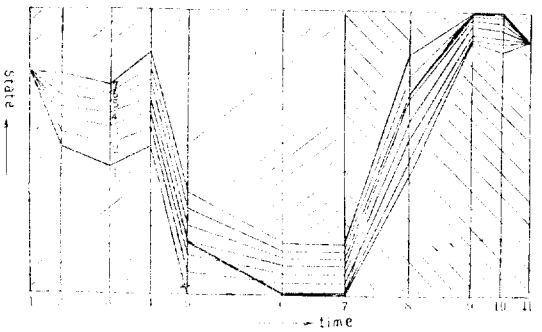

그림. 3. Multiple D. P. 의 제 2 과정
Fig. 3. The second stage of multiple D. P.

참 고 문 헌

1. J. T. Tou; Modern Control Theory' McGraw-Hill, 1964. Chapter 7.
2. G. Hadley; 'Nonlinear and Dynamic Programming' Addison-Wesley, 1964, pp. 350-481.
3. A Discrete Optimal Control Problem IEEE Trans on Automatic Control, Vol. AC-11, No. 2, Aplri, 1966.