論文

실시간 시뮬레이션용 데이터 검색 알고리즘의 검색속도에 대한 연구

윤석준*, 강현주**

A Study on Searching Speeds of Real-time Searching Algorithms

Sugjoon Yoon* and Hyunjoo Kang**

ABSTRACT

Various parameter values are provided in the form of tables, where data keys are ordered and unevenly spaced in general, for real-time simulation of dynamic systems. Unit intervals including parameter values are searched rather than stored data keys. Since real-time constraint enforces use of a fixed step size in integration of system differential equations because of the inherent nature of input from and output to real hardware, the worst case of iterated probes in searching algorithms is the core measure for comparison. In this study, conventional Bisection, Interpolation, and Fast searches are analyzed and compared in the worst cases, *Big O*'s, as well as the newly developed searching algorithms: Modified fast search and Modified regular falsi search. The numerical tests show that Modified regular falsi search is faster than the other interpolation searches in either expected time or worst case. In the case of non-linear data distribution, Bisection search turns out to be superior to Interpolation search and its derivatives. But, the logic reverses for linear or near linear data distribution because interpolation searches locate key values at once.

초 록

동적 시스템의 실시간 시뮬레이션에서 사용되는 매개변수 값들에 대한 대부분의 데이터 테이블들은 순차가 정렬된 데이터 키(key) 값들이 불연속적이고 비 등 간격으로 배치되며, 검색 대상이 되는 매개변수 값들은 일반적으로 데이터 테이블 내에 바로 존재하지 않는다. 따라서, 매개변수 값을 포함하는 단위 구간에 대한 검색을 수행해야 한다. 비교연구의 기준은 하드웨어로의 입출력이 동반되는 동적 시스템의 실시간 시뮬레이션에서는 고정 적분 프레임을 주로 사용하기 때문에 검색 횟수의 최대값 *Big O*가 된다. 본 연구에서는 이분 검색법, 보간 검색법, fast search 등은 물론 새로이 제시하는 보간 검색의 변형인 modified fast search와 modified regular falsi 기법들의 검색속도 기준인 *Big O*를 분석하였다. 수치실험 결과 Modified regular falsi 기법은 기존의 보간 검색 기법들에 비하여 평균 검색횟수에서나 최대 검색횟수에서 우수한 결과를 나타내었으며, 컴퓨터 상에서 소요되는 실제 검색시간 면에서는 비 선형적인 데이터 분포형태의 경우 이분검색법이, 이분 검색법이 선형적인 분포의 경우 보간 검색법들이 보다 우수하였다.

I. 서 론

항공기, 자동차, 선박, 전동차 등 차량의 시스템 또는 운동의 시뮬레이션에서는 다양한 종류의 불연속적인 매개변수 값들이 데이터 테이블의 형태로 주어지게 되는데, 주로 미분방정식으로 정

† 1999년 11월 4일 접수

* 정회원, 세종대학교 항공우주공학과
** 세종대학교 대학원 응용통계학과

의되는 수학적 모델들을 시뮬레이션하기 위하여 매 적분 스텝마다 데이터 테이블을 이용하여 필요한 매개변수의 함수 값들을 산출하기 위한 소위 함수발생(function generation) 작업들이 수행된다. 이 함수발생은 검색과 보간 과정으로 분리될 수 있는데, 적분 스텝의 크기가 고정되는 일반적인 실시간 시뮬레이션의 제약 하에서 기존의 검색 기법들을 논리적으로, 수치 실험적으로 비교하여 그 특성들을 파악함으로써 주어진 테이블의 데이터 분포 형태에 따른 가장 효과적인 검색 방식을 찾아내는 것이 본 연구의 주목적이다.

동적 시스템의 실시간 시뮬레이션에서 사용되는 매개변수 값들에 대한 대부분의 데이터 테이블들은 순차가 정렬된 데이터 키(key) 값들이 불연속적이고 비 등 간격으로 배치되며, 검색 대상이 되는 매개변수 값들은 일반적으로 데이터 테이블 내에 바로 존재하지 않는다. 따라서, 불연속적으로 저장된 키 값들이 아니라 매개변수 값을 포함하는 단위 구간에 대한 검색을 수행해야 한다. 데이터 베이스 분야에서는 hashing function을 이용한 검색 방법이나, block search, Fibonacci search 등의 매우 효과적인 검색 방법들이 데이터 테이블의 형태에 따라 채택되고 있다. 하지만, 이들은 저장된 키 값들을 검색하는 기법들로 구간을 대상으로 검색하기에는 적합하지 않다. 이러한 이유로 실시간 시뮬레이션 특히 HILS(Hardware-in-the-Loop Simulation) 분야에서는 이분 검색(bisection search)[1], 보간 검색(interpolation search)[1], fast search[2] 등의 방법들이 주로 사용되어왔다.

데이터를 검색하는 방법은 근본적으로 수치해석 분야에서 대수방정식의 근을 찾는 원리와 같다. 수치해석 분야에서 사용되는 기존의 근을 구하는 방식들로는 modified regular falsi method[3], secant method[3], Newton' s method[3] 등이 있다. 이들 중 secant method와 Newtons method는 검색 시작시의 초기 값에 따라 검색에 실패할 가능성이 존재하므로 본 비교연구에서는 배제하였다.

본 논문에서는 이분 검색, 보간 검색과 그 변형인 fast search 이외에도 연속함수에 적용하는 modified regular falsi의 기법을 불연속 함수 형태인 데이터 테이블에 적합하게 새로이 변형하여 제시하고 있다. 이러한 검색 기법들에 대한 비교연구의 기준은 탐색(probe) 횟수의 기대치(expected value)와 최대값(worst case)이나 두 가지 중에서도 하드웨어로의 입출력이 동반되는 동적 시스템의 실시간 시뮬레이션에서는 고정 적분 프레임을 주로 사용하기 때문에 검색 횟수의 최대값이 주 관심의 대상이 된다. 실시간 시뮬레이션에서는 제한된 시간 내에 검색결과를 내야 하기 때문에 검색 알고리즘을 선택하는 기준으로 알고리즘의 평균속도나 하한속도가 아닌 *Big O* [6]로 표현되는 상한속도 즉, 최대 검색횟수를 사용하였다. 본 연구에서는 이분 검색법, 보간 검색법, fast search 등은 물론 새로이 제시하는 보간 검색의 변형인 modified fast search와 modified regular falsi 기법들의 검색속도 기준인 *Big O*를 n개의 data를 가진 1차원의 데이터 테이블을 대상으로 분석하였다.

수치시험으로 각각 1000개의 키가 등간격으로 배치된 3종류의 데이터 테이블에 대해 이분 검색법, 보간 검색법, fast search, modified regular-falsi 등의 검색법들을 시험하였다. 탐색 횟수의 결과를 보면 이분 검색법의 검색 속도는 데이터의 형태에 상관없이 데이터 키의 개수에만 좌우된다는 것을 알 수 있다. 보간 검색 방법들은 데이터의 형태에 따라 많은 영향을 받게 되는데, 데이터가 직선형으로 분포되어 있을 경우 데이터의 개수에 관계없이 빠른 속도로 검색을 하나, 굴곡이 심한 데이터의 경우 검색 속도가 급격히 감소한다. 수치실험 결과 Modified regular falsi 기법은 기존의 보간검색 기법들에 비하여 평균 검색횟수에서나 최대 검색횟수에서 우수한 결과를 나타내었으며, 컴퓨터 상에서 소요되는 실제 검색시간 면에서는 비 선형적인 데이터 분포형태의 경우, 이분 검색법이 가장 우수하였다. 하지만, 데이터의 분포가 선형화 되어 있거나 검색창 기법[7]을 도입할 경우와 같이 선형화 될 수 있을 때는 보간 검색법들은 단 번에 key값을 결정지을 수 있다.

II. 검색속도의 기준

일반적으로 동적 시스템의 시뮬레이션의 경우 데이터 테이블 내에 주어진 키가 아닌 파라미터 값을 포함하는 단위 구간을 검색하게 된다. 따라서 일반적인 검색법을 사용하면 2개의 키를 찾아야 하는데, 본 연구에서는 이러한 조건에 맞추어 단위 구간, 즉 2개의 키를 하나의 단위로 가정하여 각 검색법의 최대속도를 이론적으로 산출하였다. 알고리즘의 속도를 나타내는 방법은 입력의 크기에 따른 수행시간 $T(n)$, $T(n)$의 상한을 나타내는 O-표기법 ($Big\ O$)[8], 하한값을 나

타내는 Ω-표기법(오메가)[8] 등이 있다. 특히, 실시간 시뮬레이션에서는 제한된 시간 내에 검색 결과를 내야 하기 때문에 알고리즘을 선택하는 기준은 알고리즘의 평균속도나 하한속도가 아닌 *Big O*로 표현되는 상한속도가 된다. *Big O*의 계산값은 알고리즘을 수행하는데 필요한 연산식의 최대 개수가 된다.

검색 속도의 비교에서 검색 횟수가 중요할 경우도 있지만, 실제 컴퓨터 상에서 검색에 소요되는 연산 시간이 더욱 중요한 기준이 될 수 있다. 검색 횟수가 작아도 그 알고리즘이 복잡하면, 검색에 소요되는 실제의 시간은 단순한 알고리즘에 검색 횟수가 많은 검색법보다도 클 수가 있다. 즉, 검색 횟수의 최대치가 실시간 시뮬레이션에 적용할 검색 기법 결정에 결정적인 기준이 아닐 수도 있다는 것이다. 또 한가지 주의할 사항은 데이터 테이블의 형태에 따라 각 검색법에서 제시하는 검색 횟수의 최대값이 발생할 확률이 매우 낮을 수가 있다는 사실이다. 이분 검색의 경우는 데이터 테이블의 형태에 비교적 좌우되지 않으나, 검색 횟수의 기대치가 낮은 보간 검색과 그 파생형들은 데이터 테이블이 직선형일 경우 최대값은 나타나지 않는다. 즉, 최적의 검색기법을 선택하기 이전에 시뮬레이션에 사용될 매개변수 데이터 테이블의 형태를 분석할 필요가 있다는 것이다.

III. 기존 검색 알고리즘의 검색 속도

다음은 n개의 데이터를 가진 1차원의 데이터 테이블을 대상으로, 기존의 검색 알고리즘들인 이분 검색법, 보간 검색법, fast search 등의 알고리즘 소개와 검색 횟수의 *Big O*에 대한 분석이다.

3.1 이분 검색법(bisection search)

이분 검색은 원하는 파라미터 값을 포함하는 단위 구간을 찾을 때까지 데이터 테이블의 크기를 반복적으로 이등분하는 방식이다. 다음은 n개의 데이터에 대한 알고리즘으로, i번째 데이터를 $D(i)$로 표시하며, N을 포함하는 구간을 찾을 때까지 반복하게 된다. 이분 검색법의 알고리즘은 다음과 같다.

이분 검색은 데이터의 형태와 무관하게 데이터 개수에만 영향을 받는다. 앞에서 언급한 바와 같이 구간을 단위로 검색한다면, 테이블이 n개의 데이터를 포함하는 경우에 $n-1$개의 구간

Algorithm

$l=1,\ r=n$
For $j=0,1,2,\cdots$ *until satisfied, do:*
 Set $m=(l+r)/2$
 If $D(m+1) < N$, *set* $l=m+1$
 Else if $D(m) > N$, *set* $r=m$
 Otherwise 'N' exist in the inteval $[l, r]$

이 발생하고, 각 반복탐색(iteration)마다 구간의 수가 절반으로 준다. 그림 1에서 단위 구간을 검색하기 위한 반복 탐색의 횟수를 살펴보면 1번에 찾는 경우는 1개 구간이고, 2번에 찾는 경우는 2개구간이 되는데, 이를 일반화시키면 k번에 찾는 경우는 2^{k-1}개의 구간이 된다. 그림 1에 표시된 수는 해당 구간을 검색하는데 소요되는 반복탐색의 횟수를 의미한다.

그림 1. 이분검색의 iteration 수

각각의 반복탐색 내에서는 1번의 계산만이 실행되므로, 반복탐색의 수를 *O*-표기법 (*Big O*)의 기준으로 사용할 수 있다. 이를 이용해 검색 횟수의 최대값을 다음과 같이 산출한다.

k가 최대 반복탐색 횟수라면, 1번부터 k번까지 찾게되는 데이터의 수가 $n-1$이 되어야 하기 때문에 다음 식을 얻는다.

$$\sum_{i=1}^{k} 2^{i-1} = n-1 \tag{1}$$

식(1)를 k에 관해 정리하면,

$$\sum_{i=1}^{k} 2^{i-1} = \frac{1(1-2^k)}{1-2} = n-1 \ \rightarrow\ 2^k = n \tag{2}$$

$$\therefore\ k = \log_2 n \tag{3}$$

그러나 반복탐색의 수는 정수가 되어야 하므로 정확하게는 $\log_2 n$의 올림값이 최대 검색 횟수가 되며, *O*-표기법(*Big O*)으로는 $O(\log_2 n)$으로 표현한다.

3.2 보간 검색법(Interpolation method)

보간 검색은 대수방정식의 근을 구하는 수치해석 기법인 Regular falsi의 변형으로 주어진 데이터 테이블 내의 매개변수 값들이 선형적으로 분포됨을 가정하여 양끝을 잇는 직선 상에서 매개변수 값을 포함하는 단위 구간을 검색하는 방법이다. 데이터 테이블의 형태에 따라 검색속도 차이가 심하게 나타나며, 최악의 경우는 순차 검색을 하게 된다.

보간검색은 데이터의 형태에 많은 영향을 받는다. 데이터의 두 점을 잇는 직선을 이용하여 검색하므로 데이터의 형태가 직선형일 때에는 검색속도가 매우 빠르지만, 최악의 경우 순차검색을 하게된다. 보간검색도 이분검색과 마찬가지로 각각의 반복탐색마다 1번의 계산을 하게되므로 반복탐색의 횟수를 계산하면 된다. 따라서 데이터 형태가 최악의 경우에 검색하는데는 데이터 개수만큼의 반복탐색이 필요하다. 즉, 최대 검색 횟수는 *O*-표기법(*Big O*)으로 $O(n)$이 된다. 그러나 후에 언급하는 수치시험의 결과를 보면 데이터의 구간에 따른 변화가 급격한 경우에는 평균 속도보다 빠르게 찾고 직선형의 경우에는 단 한 번에 찾는 것을 알 수 있다.

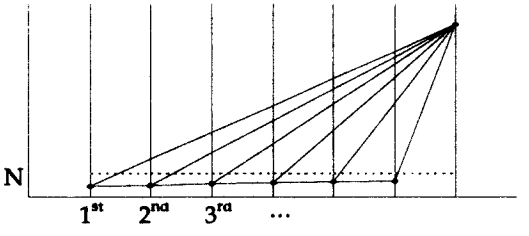

그림 2. 보간검색에서 최대 검색 횟수의 예

3.3 Fast Search

Fast Search는 보간 검색법의 변형으로, 구간의 크기를 정하는 임의의 수를 이용하여 각 반복탐색에서 구간을 줄여 구하는 데이터를 포함하는 단위 구간에 대한 검색속도를 증가시키는 방법이다. 최악의 경우 보간 검색법은 순차 검색을 하기도 하지만, Fast Search는 이러한 경우를 회피할 수 있으므로 최악의 경우에도 검색 속도가 보간 검색법보다 빠르다. Fast Search의 알고리즘은 다음과 같다.

Algorithm

l=1, *r*=*n*, *g*=$\sqrt{n}$
For j=0,1,2,... *until satisfied, do*:
Set $m=(N-D(l))\times(r-l)/(D(r)-D(l))+$
If $r-g<m<l+g$, *set* $m=m$
Else if $m<r-g$, *set* $m=r-g$
If $D(m+1)<N$, *set* $l=m+1$
Else if $D(m)>N$, *set* $r=m$
Set $g=\sqrt{r-l}$
Otherwise N exist in the interval [*l*, *r*]

Fast search는 매 탐색마다 축소하는 단위구간의 크기를 바꾸어 줌으로써 검색 구간의 크기를 평균 n에서 $\sqrt{n}$으로 줄여가면서 검색을 한다. 구간의 크기는 $n^{\frac{1}{2^0}} \rightarrow n^{\frac{1}{2^1}} \rightarrow n^{\frac{1}{2^2}} \rightarrow n^{\frac{1}{2^3}}$와 같은 형태로 줄어들게 되므로 평균 검색속도는 다음과 같이 계산된다.

$$n^{\frac{1}{2^k}}=2 \rightarrow \frac{1}{2^k}\log_2 n=1 \rightarrow 2^k=\log_2 n \quad (4)$$

$$\therefore k=\log_2\log_2 n \quad (5)$$

또한 최대 검색 횟수의 경우는 구간의 크기가 n에서 $\sqrt[k]{n}$으로 감소하기 때문에 다음의 과정을 통해 $O(\log_2 n)^2$의 검색 속도가 계산된다.

$$n^{k^{-\frac{1}{2}}}=2 \rightarrow k^{-\frac{1}{2}}\cdot\log_2 n=1 \quad (6)$$

$$k^{-\frac{1}{2}}=\frac{1}{\log_2 n} \rightarrow k^{\frac{1}{2}}=\log_2 n \quad (7)$$

$$\therefore k=(\log_2 n)^2 \quad (8)$$

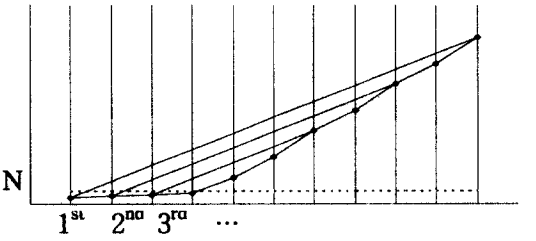

그림 3. Fast search에서 최대 검색 횟수의 예

매 탐색마다 축소되는 단위 구간의 크기를 다음의 알고리즘에서와 같이 고정시키는 경우, Fast search의 검색속도는 논리가 단순해짐에 따라 매 반복탐색에 소요되는 실제 연산시간이 줄어든다. 즉, 실제적인 검색속도의 향상을 기대할 수 있다. 수정된 Fast Search의 알고리즘은 다음과 같다.

Algorithm

$l=1,\ r=n$
For $j=0,1,2,\cdots$ *until satisfied, do*:
Set $m=\dfrac{(N-D(l))*(r-l)}{(D(r)-D(l))}+1$
If $D(m+1)<N$, *set* $l=m+1$
  If also $|m+1, r-1|<|m, l+1|$,
    set $r=r+2$
  Else set $r=r+1$
Else if $D(m)>N$, *set* $r=m$
  If also $|m+1, r-1|<|m, l+1|$,
    set $l=l+2$
  Else set $l=l+1$
Otherwise 'N' exist in the inteval $[l, r]$

이처럼 기존의 Fast search를 수정한 검색 방법(modified fast search)의 경우 최대 검색 횟수는 다음과 같이 산출할 수 있다. 그림 3은 최대 검색 횟수가 발생하는 예이다. 그림 3에서와 같이 Fast search는 최악의 경우에도 1번의 반복탐색으로 3개의 구간을 줄일 수 있다. 따라서 전체 데이터의 개수가 n이라면 다음과 같이 반복탐색의 수를 구할 수 있다.

$$n-3\times k=0 \quad \rightarrow \quad k=n/3 \qquad (9)$$

본 논문의 수치 시험에서 사용한 Fast search의 경우는 축소 단위 구간을 고정시킨 방식이다.

IV. Modified Regular Falsi를 변형한 Fast Search의 검색 속도

Modified regular falsi 방식은 수치해석에서 대수방정식의 근을 찾는 방법으로 이용되어 왔다. 그 원리는 보간 검색법의 변형으로 구간의 양 끝 점 중 하나와 검색하고자 하는 값의 차이를 반으로 줄여서 두 점 사이의 기울기를 축소시킴으로서 키를 포함하는 구간의 크기를 좁혀 검색 속도를 증가시키는 방법이다.

Algorithm

$l=1,\ r=n,\ L=D(1),\ R=D(n)$
For $j=0,1,2,\cdots$ *until satisfied, do*:
Set $m=\dfrac{(N-D(L))*(r-l)}{(D(r)-D(l))}+1$
If $D(m+1)<N$, *set* $l=m+1$ and
  $L=D(m+1),\ R=(D(r)+D(r-1))/2$
Else if $D(m)>N$, *set* $r=m$ and
  $R=D(m),\ L=(D(l)+D(l+1))/2$
Otherwise 'N' exist in the inteval $[l, r]$

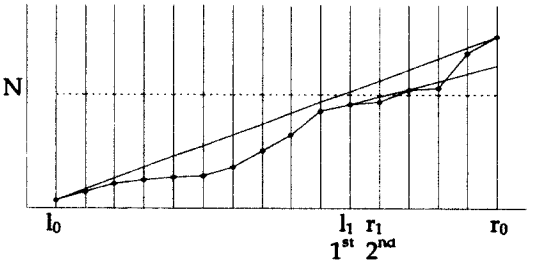

그림 4. Modified regular falsi

Modified regular falsi를 이용한 검색도 보간 검색법을 보완한 방법의 일종이다. 따라서 두 점을 잇는 직선을 이용하게 되는데 두 점 중 한 점의 값을 줄여줌으로써 순차검색이 되는 것을 방지, 검색의 속도를 증가시키는 방법이다. 아래의 3가지 수치시험에서는 가장 우수한 검색속도를 보이지만, 이론적인 최대 검색횟수의 속도는 보간검색과 같다. 데이터의 형태가 연속이 아니기 때문에 고정된 점의 값을 감소시키는 효과가 전혀 나타나지 않기 때문이다. 따라서 최대 검색횟수의 경우 검색하는데 필요한 반복탐색의 횟수는 보간검색과 같이 $O(n)$으로 표현된다.

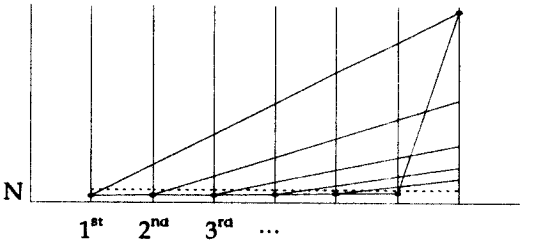

그림 5. Modified regular falsi에서 worst case의 예

V. 검색 창 검색법(dynamic-window search)

검색하고자 하는 매개변수 값이 시스템의 상태 변수, 적분 기법, 적분 프레임 크기 등과 연관되어 임의의 원칙 즉, dynamics를 갖고 매 프레임마다 변한다면, 이 정보를 활용하여 주어진 데이터 테이블 중 일부의 영역으로 검색 대상을 축소할 수 있다. 검색 창[7] 개념을 도입함으로써 이분 검색법이나 보간 검색법 모두에서 데이터 테이블의 검색 대상 영역을 축소하는 효과로 검색 속도가 월등하게 향상되고, 데이터의 형태가 직선형으로 변환되는 효과를 갖는다.

VI. 수치 시험

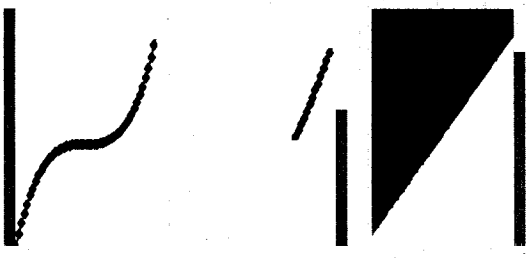

그림 6. 수치시험 데이터 테이블의 유형

그림 6과 같이 각각 1000개의 키가 등간격으로 배치된 3종류의 데이터 테이블에 대해 이분 검색법, 보간 검색법, modified fast search, modified regular-falsi 등의 검색법들을 시험하였다. 등 간격 데이터 테이블을 사용한 이유는 테이블의 작성이 용이하며, 등 간격 데이터 테이블을 대상으로 한 검색기법들의 비교평가 결과가 비 등 간격 데이터 테이블에 대하여도 동일하게 유효하기 때문이다.

표1의 case 1, 2, 3열의 수치들은 단위구간을 찾는데 소요되는 총 탐색(probe) 횟수를 나타내고, Total열의 수치들은 case 1, 2, 3의 산술적 평균과 각 case에서의 최대값을 의미한다. Time열의 수치들은 컴퓨터(Pentium II PC, 266MHz, 64Mb RAM) 상에서 1 probe를 실행하는데 소요되는 시간(초)을 나타낸다. 탐색 횟수와 소요 시간은 작을수록 검색속도가 빠름을 의미한다. 표1에 나타난 탐색 횟수의 결과를 보면 이분 검색법의 검색 속도는 데이터의 형태에 무관하게 데이터 키의 개수에만 좌우된다는 것을 알 수 있다. 그러나 보간 검색 방법들은 데이터의 형태에 따라 많은 영향을 받게 된다. 직선형의 경우 데이터의 개수에 관계없이 빠른 속도로 검색을 하나, 굴곡이 심한 데이터의 경우 검색 속도가 급격히 감소한다. 탐색에 걸리는 시간을 비교해 보면 이분 검색법이 가장 빠르다는 사실을 알 수 있다. 다른 방법들은 이분 검색법에 비해 계산식이 복잡하고, 비교 연산자의 수도 많기 때문에 약 3배의 시간이 소요된다.

VII. 결 론

기존 검색 알고리즘들의 이론적인 검색속도 비교연구를 수행하였다. 본 연구에서 새로이 개발된 보간 검색의 변형인 modified fast search와 modified regular falsi 기법들도 fast search와 같은 order의 최대 검색속도를 가진다. 수치실험 결과 Modified regular falsi 기법은 기존의 보간 검색 기법들에 비하여 평균 검색횟수에서나 최대 검색횟수에서 우수한 결과를 나타내었으며, 컴퓨터 상에서 소요되는 실제 검색시간 면에서는 비선형적인 데이터 분포형태의 경우, 이분 검색법이 가장 우수하였다. 하지만, 데이터의 분포가

표 1. 기존 검색 기법들의 수치시험 비교

	이분 검색		보간 검색		Modified Fast search		Modified Regular search	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max
Case 1 (탐색수)	9.0	10	1	1	1	1	1	1
Case 2 (탐색수)	9.0	10	16.8	65	16.3	64	10.1	48
Case 3 (탐색수)	9.0	10	21.8	98	21.5	97	9.9	37
Total (탐색수)	9.0	10	13.2	98	12.9	97	7.0	48

선형화 되어 있거나 검색창 기법을 도입할 경우와 같이 선형화 될 수 있을 때는 보간 검색법들이 표1에서와 같이 단 번에 key값을 결정지을 수 있다. 따라서, 효과적인 시뮬레이션을 위하여, 주어진 매개변수 테이블의 데이터 분포형태를 사전에 면밀히 검토하여 적절한 검색기법을 적용하는 것이 필요하다.

후 기

본 연구는 과학기술부 선도기술개발사업인 감성공학기술개발의 위탁과제 일부로 수행되었으며, 한국표준과학연구소의 지원에 감사를 드립니다.

참고문헌

1) Warren Burton F. and Lewis Gilbert N., "A Robust Variation of Interpolation Search", Information Processing Letter, 10(4,5), 5 July 1980, pp.198-201.

2) Lewis Gilbert N., Boynton Nancy J. and Burton F. Warren, "Expected Complexity of Fast Search with Uniformly Distributed Data, Information Processing Letter, 10(1), 27 October 1981, pp.4-7.

3) Gonnet G., George J. and Rogers L., "An Algorithmic and Complexity Analysis of Interpolation Search", Acta Informatica 13, Springer-Veriag 1980, pp.39-52.

4) Leon S. Levy, "An Improved List-Searching Algorithm, Information processing letters, 15(1), 19 August 1982, pp.43-45.

5) Yehoshua Perl, Reingold Edward M., "Understanding The Complexity of Interpolation Search", Information processing letters, 6(6), December 1977, pp.219-222.

6) Sedgewick, Robert, ALGORITHM IN C, Addison-Wesley Pub.Co., 1990.

7) Yoon S., "A Study of Searching Algorithms for Real-time Simulation of a Dynamic System", Proceedings of AIAA Flight Simulation Conference, Aug. 1996.