전향 차분을 이용한 효율적 곡선 생성 알고리즘

이 상 락† 심 재 홍††

요 약

곡선의 신속한 생성은 컴퓨터 그래픽의 고속 처리를 가능하게하는 중요한 기법 중의 하나이다.이것은 현재 하드웨어나 소프트웨어를 통하여 상당한 효과를 거두고 있으나 급속한 그래픽의 발전은 보다 빠른 곡선 생성 방법을 요구한다. 본 논문은 전향 차분을 이용하므로써 곡선 상의 점의 좌표를 곱셈을 배제하고 덧셈만으로 계산하여 보다 빨리 곡선을 그릴 수 있는 한 알고리즘(CDAUD라 부름)을 제시한다. CDAUD는 곡선이 부드럽고 완전하게 연결된 상태로 그려지도록 하는 방안을 포함하고 있다.CDAUD의 시간 복잡도(time complexity)는 그리는 곡선에 따라 기존의 방법보다 우수한 경우도 있음을 실험을 통하여 확인할 수 있었다.

A Efficient Curve Drawing Algorithm Using Forward Differences

Sang Rak Lee† and Jae Hong Shim††

ABSTRACT

Fast curve generation is one of important techniques which facilitate fast process of computer graphics applications. It is possible to resolve the task through hardware or software. But rapid development of computer graphics area need methods for more fast generation of curves. This paper propose a algorithm (called CDAUD),which computes the coordinates of points on curve only with add operations using forward differences and draw the curve fast. It also contains the method for generation of smooth and fully connected curves. Time complexity of CDAUD shows that it is superior than the existing method for constrained case, and it's superiority was validated through experimental implementation.

1. 서 론

컴퓨터 그래픽에서 곡선의 생성(curve generation, curve drawing,curve rendering)은 디스플레이 화면(이하 화면)에 곡선을 그리는 것을 의미하며, 곡선의 처리가 포함되지 않은 작업은 거의 없기 때문에 컴퓨터 그래픽의 중요한 기본 요소(primitives) 중의 하나이다. 특히 CAD 나 기하 모델링(geometric modelling)에서는 대부분의 작업이 이것을 통하여 이루어진다[4,5,6,7].

본 연구는 곡선의 신속한 생성 방법에 촛점을 두고 있다. 곡선을 신속히 그리려면 곡선을 정의하는 다항식의 값을 신속히 계산하여야 한다. 대화식 그래픽 처리에서 다항식은 매개변수 형태(parameteric form)로 표현되는데 한 매개변수에 대한 다항식의 값이 곡선 상의 점을 결정하기 때문이다. 일반적으로 다항식의 값은 Horner의 규칙을 이용하면 다소 빨리 구할수 있다[11]. Kurokawa[3]는 식의 계산에 쓰이는 수를 대수(logarithmic number)로 나타내어 두 수의 곱셈을 각각의 지수의 덧셈으로 처리하고 또 두 수의 덧셈을 미리 작성한 조견표(look-up table)를 이용하므로써 곡선을 그리는 픽셀의 주소를 간편하고 빠르게 구하는 방법을 제시하였다. Aken[1]은 중간점(mid point) 방법으로 원이나 타원을 빨리 그리는 방법을 제시하였다. 또 Suenaga[2], Danielsson[8], Jordan 등[9]은 한 픽셀 단위 씩 이동하면서 인접한 8방향 또는 4방향의 픽셀 중

† 정 회 원 : 인천대학교 전자계산학과 조교수
†† 정 회 원 : 광운대학교 전자계산학교 교수
논문접수 : 1994년 6월 27일, 심사완료 : 1994년 7월 27일

에서 그리고자 하는 곡선에 가장 가까운 하나의 픽셀을 결정하여 곡선을 그려 나가는 방법을 제시하였는데 이들은 곡선의 신속한 생성보다는 정확한 생성에 더 많은 관심을 두고 있다.

본 연구에서는 그리고자 하는 곡선의 다항식이 주어졌을때 차분 이론에 근거하여 이것을 제일 처음 구한 다항식의 값과 다항식의 계차로써 나타내고, 나아가서 이것을 컴퓨터에서 계산이 용이한 형태인 점화식으로 변환시켰다. 점화식은 곱셈이 배제되고 덧셈만으로 표현되므로써 신속하게 계산된다. 본 연구는 또 구해지는 점의 갯수가 불필요하게 많거나 반대로 부족하게 되는 경우가 발생하지 않도록 적절한 갯수의 점을 구하는 방법을 고안하였다.

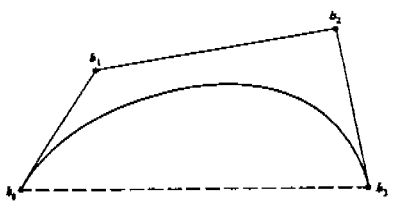
본 연구에서 제시하는 곡선 생성 알고리즘을 CDAUD(Curve Drawing Algorithm Using forward Differences)라고 부르기로 한다.CDAUD 의 적용대상은 3차 베지어 곡선으로 한정하였다. 4차 이상의 다항식의 경우에는 알고리즘 수행의 중간에 나오는 많은 임시 변수의 값을 계산하는데 오버헤드(overhead)가 발생하기 때문에 효용성이 적어 적용 대상에서 제외하였다.

2. 베지어 곡선

Bezier 곡선의 수학적 정의는 식(1)과 같으며 (그림 1)처럼 제어다각형(control polygon 또는 defining polygon 이라고도 함)으로 곡선의 형태를 예상할 수 있다[7].

$$p(t)=\sum_{i=0}^{n} b_i J_{ni}(t) \qquad 0\le t\le 1 \quad \cdots\cdots (1)$$

단, p(t):곡선상의 점 t:매개변수
b_i:제어점(i=0, n−1) n:제어점의 수−1



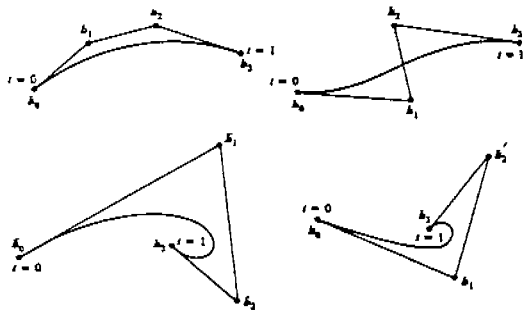
(그림 1) 베지어 곡선과 제어다각형
(Fig. 1) A Bezier curve and it's defining polygon.

위에서 p(t)와 b_i는 벡터로서 식(2)의 의미를 가진다.

$$p(t)=[x(t),\ y(t)]$$
$$b_i=[x_i,\ y_i] \quad \cdots\cdots (2)$$

식(1)에서 $J_{ni}(t)$는 Bezier 다항식 또는 Bernstein 다항식으로 불리우며 식(3)과 같이 정의된다.

$$J_{ni}(t)=\binom{n}{i} t^i(1-t)^{n-i} \quad \binom{n}{i}=\frac{n!}{i(n-i)!} \cdots (3)$$



(그림 2)여러가지 3차 베지어 곡선
(Fig. 2) Cubic Bezier curves

(그림 2)는 몇가지 형태의 제어 다각형에 대한 3차의 베지어 곡선이다. 모든 곡선은 각각의 제어다각형의 convex hull 내에서만 그려지며 이러한 성질을 폐포성(property of convex hull)이라고 한다[7].

3. 기본이론

3.1 전향 차분의 이용

식(1)과 같이 정의된 Bezier 곡선의 식을 다항식으로 표현하면 식(4)와 같이 나타낼 수 있다.

$$p(t)=b_0+(-3b_0+3b_1)t+(3b_0-6b_1+3b_2)t^2 +(-b_0+3b_1-3b_2+b_3)t^3 \quad \cdots\cdots (4)$$

식(4)를 편의상 좀 더 간략한 형태로 표현하면 식(5)와 같다.

$$p(t)=a_0+a_1t_1+a_2t_2+a_3t_3 \quad \cdots\cdots (5)$$

단, $a_0=b_0$
$a_1=(-3b_0+3b_1)$
$a_2=(3b_0-6b_1+3b_2)$
$a_3=(-b_0+3b_1-3b_2+b_3)$

매개변수 t를 등구간으로 나누어
$t_1=\delta t,\ t_2=2\delta t,\ t_3=3\delta t\cdots\cdots t_i=i\delta t\ (i=1,\cdots,m)$이라 하자. 그리고 각 매개변수 t_i에서 다항식의 값이 식(6)과 같다고 하자.

$$\begin{aligned}
p_1&=p(t_1)=a_0+a_1t_1+a_2t_1^2+a_3t_1^3\\
p_2&=p(t_2)=a_0+a_1t_2+a_2t_2^2+a_3t_2^3\\
p_3&=p(t_3)=a_0+a_1t_3+a_2t_3+a_3t_3 \qquad \cdots\cdots(6)\\
&\cdots\quad\cdots\\
p_i&=p(t_i)=a_0+a_2t_i+a_3t_i^2+a_3t_i^3
\end{aligned}$$

그러면 p_1에 대한 k계 (k=1,...4)의 전향 차분(이하 차분) $\triangle^k p_1$은 식(7)과 같이 구해진다.

$$\begin{aligned}
\triangle p_1&=p_2-p_1=a_1\delta t+3a_2\delta t^2+7a_3\delta t^3\\
\triangle^2 p_1&=p_3-2p_2+p_1=2a_2\delta t^2+12a_3\delta t^3\\
\triangle^3 p_1&=p_4-3p_3+3p_2-p_1=7a_3\delta t^3 \qquad \cdots\cdots(7)\\
\triangle^4 p_1&=p_5-4p_4-6p_3-4p_2+p_1=0
\end{aligned}$$

식(7)에서 알수 있듯이 p_1에 대한 제1 제2 제3계차 $\triangle^1 p_1$, $\triangle^2 p_1$, $\triangle^3 p_1$은 $a_0\ a_1\ a_2\ a_3$ 및 δt가 상수이므로 역시 상수이다. 그리고 제4계차 이상은 모두 0이 된다.

차분 이론에 의하면 매개변수 $t=t_k$에 대한 다항식의 값 p_k는 초기값과 계차의 항으로써 식(8)과 같이 나타낼 수 있다[10,12].

$$\begin{aligned}
p_{k+1}&=p_1+\binom{k}{1}\triangle^1 p_1+\binom{k}{2}\triangle^2 p_1+\binom{k}{3}\triangle^3 p_1\\
&\cdots\cdots+\binom{k}{k}\triangle^k p_1 \qquad \cdots\cdots(8)
\end{aligned}$$

지금의 경우 $\triangle^4 p_1$이상의 항은 모두 0이므로 p_{k+1}은 식(9)와 같이 쓸 수 있다.

$$\begin{aligned}
p_{k+1}&=p_1k\triangle p+\frac{k(k-1)}{2}\triangle^2 p_1\\
&+\frac{k(k-1)(k-2)}{6}\triangle^3 p_1 \qquad \cdots\cdots(9)
\end{aligned}$$

여기에 새로운 임시 상수 $T_1\ T_2\ T_3$를 도입하여 식(10)과 같이 정의한다.

$$\begin{aligned}
T_1&=p_1\\
T_2&=\frac{1}{2}\triangle^2 p_1 \qquad \cdots\cdots(10)\\
T_3&=\frac{1}{6}\triangle^3 p_1
\end{aligned}$$

식(10)에서 정의된 각 임시 상수 $T_1\ T_2\ T_3$는 앞의 식(7)을 이용하여 구할 수 있다.

최종적으로 정리하면 식(9)는 식(11)과 같이 간단히 표현된다.

$$\begin{aligned}
p_{k+1}&=p_1+kT_1+k(k-1)T_2\\
&+k(k-1)(k-2)T_2 \qquad \cdots\cdots(11)
\end{aligned}$$

식(11)은 임의의 t_i에서의 함수값 $P(t_i)$는 초기값 p_1과 상수값으로 구할 수 있음을 의미한다.

3.2 점화식 변환

식(11)을 점화식으로 변환하기 위하여 다음의 과정을 거친다.

식(11)에서 k+1을 k로 놓으면 식(12)를 얻을 수 있다.

$$\begin{aligned}
p_k&=p_1+(k-1)T_1+(k-1)(k-2)T_2\\
&+(k-1)(k-2)(k-3)T_33 \qquad \cdots\cdots(12)
\end{aligned}$$

식(11)에서 식(12)를 빼면 식(13)과 같이 된다.

$$\begin{aligned}
p_{k+1}-p_k&=T_1+(k-1)(k-k+2)T_2\\
&+(k-1)(k-2)(k-k+3)T_3\\
\therefore\ p_{K+1}&=p_k+T_1+2(k-1)T_2\\
&+3(k-1)(k-2)T_3 \qquad \cdots\cdots(13)
\end{aligned}$$

또 식(13)으로 부터 g_{k+1}을 식(14)와 같이 두면 식(16)을 얻을 수 있다.

$$\begin{aligned}
g_{k+1}&=T_1+2(k-1)T_2\\
&+3(k-1)(k-2)T_3 \qquad \cdots\cdots(14)\\
g_k&=T_1+2(k-2)T_2\\
&+3(k-2)(k-3)T_3 \qquad \cdots\cdots(15)\\
g_{K+1}-g_k&=(2k-2-2k+4)T_2\\
&+3(k-2)(k-1-k+3)T_3\\
&=2T_2-2+3\cdot2(k-2)T_3\\
g_{k+1}&=g_k+2T_2+3\cdot2(k-2)T_3 \qquad \cdots\cdots(16)
\end{aligned}$$

그리고 h_{k+1}을 식(17)과 같이 두면 식(19)을 얻을 수 있다.

$$\begin{aligned}
h_{k+1}&=2T_2+3\cdot2(k-2)T_3 \qquad \cdots\cdots(17)\\
h_k&=2T_2+3\cdot2(k-3)T_3 \qquad \cdots\cdots(18)\\
h_{k+1}-h_k&=3\cdot2(k-2-k+3)T_3=6T_3\\
\therefore\ h_{K+1}&=h_k+6T_3 \qquad \cdots\cdots(19)
\end{aligned}$$

지금까지의 유도결과를 정리하면 곡선상의 점

의 좌표를 구하기 위한 반복식은 다음의 식(20)과 같이 표현된다.

$$h_{k+1}=h_k+6T_3$$
$$g_{K+1}=g_k+h_{K+1} \quad \cdots\cdots(20)$$
$$P_{K+1}=P_k+g_{K+1}$$

3.3 초기값의 설정

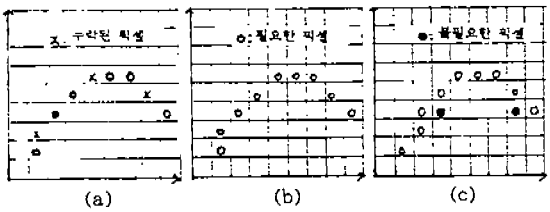
각 변수 h, g의 초기치 g_1, h_1은 각각 식(15)와 식(18)에서 k에 1을 대입하면 다음과 같이 구해진다.

$$g_k=T_1+2(k-2)T_2+3(k-2)(k-3)T_3$$
$$\therefore\ g_1=T_1+2(-1)T_2+3(-1)(-2)T_3$$
$$=T_1-2T_2+6T_3 \quad \cdots\cdots(21)$$
$$h_k=2T_2+3\cdot2(k-3)T_3$$
$$\therefore\ h_1=2T_2+6\cdot(-2)T_3$$
$$=2T_2-12T_3 \quad \cdots\cdots(22)$$

그리고 p_1은 $t=t_1$일 때의 값이므로 식(6)에서 곧바로 구할 수 있다.

3.4 점화식의 반복 횟수

점화식이 한번 계산되면 하나의 점의 좌표가 구해진다. 따라서 점화식을 반복 계산하는 횟수는 적정하게 정해져야 한다. 횟수가 많으면 구해지는 점들은 서로 중복되어 불필요한 것이 있게 되고 횟수가 적으면 점의 수가 부족하여 완전하게 연결되지 못한 곡선이 생성된다.(그림3)은 이러한 현상을 설명하고 있다.



(a)점이 부족한 곡선 (b)점이 알맞은 곡선
(c) 점이 너무 많은 곡선

(그림 3) 곡선의 연결 상태
(Fig. 3) connectivity of curves

그림에서 보면 곡선을 나타내는 픽셀은 모두 서로 8방향 연결(8-way connected)이므로 완전하게 연결된 곡선으로 보인다. 이와같이 곡선이 완전하게 연결된 것으로 그려지도록 하기위하여는 충분한 갯수의 점 즉, 충분한 갯수의 픽셀이 선정되어야 한다.

적절한 반복횟수를 정하기 위해 사용할 수 있는 근거 기준으로서는 먼저 그리고자 하는 곡선의 길이를 생각해 볼 수 있다.그러나 이것은 감당하기 어려운 오버헤드를 초래한다. 다음으로 이용할 수 있는 것은 제어다각형을 구성하는 세 변의 길이이다. 프로그램을 작성하여 실행해 본 결과 상당히 양호한 결과를 얻을 수 있었으나 이 방법 역시 제곱근을 계산해야 하는 부담이 따른다. 따라서 최종적으로 제어다각형의 세변 중에서 수직 또는 수평 방향의 성분이 가장 긴 변을 택하여 그 변의 수직이나 수평 성분의 3배되는 값으로 횟수를 결정하는 기준으로 사용하였다.즉 반복 횟수는 식(23)과 같이 구한다.

$$N=3*\max_{i=1,2,3}\{(x_i-x_{i-1}),\ (y_i-y_{i-1})\} \quad \cdots\cdots(23)$$

실제로 식(23)의 기준을 사용하여 곡선을 그려본 결과 완전하게 연결된 곡선을 얻을 수 있었으며 중복되거나 서로 연결되지 않는 픽셀은 생성되지 않는다는 것을 확인할 수 있었다.

4. 알고리즘의 제시

지금까지의 이론을 바탕으로 하여 3차 매개변수의 베지어 곡선을 차분을 이용한 덧셈만의 방법으로 그리기 위한 알고리즘인 CDAUD를 제시하면 다음과 같다. 실제로 곡선을 그리기 위한 프로그램을 작성할 때는 알고리즘에 나오는 각 계수나 상수는 좌표 x, y에 대하여 각각 계산되어야 한다.

CDAUD 알고리즘

/* 입력 데이타 $b_0(x_0, y_0)$, $b_1(x_1, y_1)$, $b_2(x_2, y_2)$, $b_3(x_3, y_3)$ */

1. 데이타 입력.
2. 곡선식 계수의 계산.

$a_0=b_0$

$a_1=3(b_1-b_0)$

$a_2=3(b_2-2b_1+b_0)$

$a_3=b_3-3b_2+3b_1-b_0$

3. 반복 횟수 및 파라메타 구간값의 계산

```
n=-999
for(i=1; i<=3; ++i)
  {m=max(abs(x_i-x_{i-1}), abs(y_i-y_{i-1}))
   n=max(n, m)}
dt=1/n
```

4. 중간 상수값 계산

$T_1=a_1dt+3a_2dt^2+7a_3dt^3$

$T_2=a_2dt^2+6a_3dt^3$

$T_3=a_3dt^3$

$T_4=6T_3$

5. 각 변수의 초기치 계산

$h=2T_2-12T_3$

$g=T_1-2T_2+6T_3$

$p=a_0+a_1dt+a_2dt^2+a_3dt^3$

6. 곡선상의 점의 좌표 계산

```
for(i=1;i<N; i=i+1)
    h=h+T4
    g=g+h
    p=p+g
    putpixel P(px, py)
end for
```

7. 끝

5. 알고리즘의 분석

5.1 시간 복잡도(time complexity) 분석

CDAUD의 각 스텝별 명령의 종류와 수행횟수를 분석하여 표로 나타내면 〈표 1〉과 같다.

〈표 1〉을 보면 CDAUD는 N개의 점을 구할 경우 3N+21번의 덧셈과 40번의 곱셈 계산이 필요함을 알 수 있다. 만약 Horner의 규칙을 이용하여 N개의 점을 계산한다면 3N번의 덧셈과 3N번의 곱셈이 필요하다[11]. 따라서 CDAUD에서 21번의 덧셈 계산을 고려하지 않는다면 3N의 덧셈에 필요한 시간은 같고 40번의 곱셈과 3N번의 곱셈에 대한 횟수를 비교하면 된다. 따라서 N>14인 경우 CDAUD는 다소 효율적인 곡선 생성 알고리즘이라고 할 수 있다.

〈표 1〉 명령의 종류와 횟수

〈Table 1〉 number of operations in CDAUD

계산단계	덧셈/뺄셈	곱셈/나눗셈
다항식 계수	6	5
반복 회수	6	7
중간 상수	3	18
초 기 값	6	10
루 프	3N	0
계	3N+21	40

5.2. Implementation

실제로 CDAUD 가 Horner의 규칙을 이용한 다항식의 연산에 의한 곡선 그리기와 시간상으로 어느 정도 차이가 나는지를 알아 보기위하여 실험을 해 보았다. 되도록 정확한 시간의 측정을 위하여 불필요한 출력문 같은 것은 제외하였다. 첫번째 비교에서 반복 횟수가 작은 경우 두 알고리즘의 수행 시간은 별 차이가 없어 결과의 신뢰도를 보장하기 어렵다고 보았다. 그래서 반복 횟수가 증가하여도 차이가 없는지를 알아보기위하여 반복 횟수를 증가시키면서 두 알고리즘의 수행 시간을 비교한 결과 〈표 2〉와 같이 횟수가 증가하면 수행 시간도 차이가 커짐을 알 수 있었다. 반복회수가 커지면 이것은 실수로 처리하여야 한다. 두 방법에서 소요되는 시간차이를 알아보기 위한 실험 결과를 정리하면 〈표 2〉와 같다.

〈표 2〉 수행 시간의 비교

〈Table 2〉 Comparision of running time

반복 횟수	CDAUD	Horner
N=1,000	5	6
5,000	9	11
10,000	13	17
50,000	47	68
100,000	89	131

단위:C언어에서 제공되는 CPU time

6. 결 론

컴퓨터 그래픽에서 곡선을 그리는 작업은 많은 비중을 차지한다. 특히 CAD/CAM, 에니메이션 등에서 곡선의 역할은 막중하다. 지금까지 곡선에 대한 연구는 주로 주어진 데이타에 대하여 곡선을 정확하게 그리는 방법에 대한 연구가 대부분이었으며 곡선을 빨리 그리는 방법에 대한 연구는 찾아보기 어렵다.

본 연구는 곡선을 표현하는 식에서 하나의 매개변수에 해당하는 곡선의 식(이하 곡선식)의 값을 구하고 이에 따른 매개변수 값에 해당하는 곡선식의 값을 구할 때 이미 구한 계산정보를 전혀 이용하고 있지 않는 사실에 착안하였다. 즉 한번 곡선식의 값을 구하고, 그 값을 이용하여 계속적인 곡선식의 값을 구하므로써 효율적으로 곡선을 그리게 된다. 따라서 차분이론에서 데이타가 균등하게 분포되어 있는 경우 뒤의 데이타에 대한 함수값은 제일 처음의 함수값과 계차의 항으로 나타낼 수 있음을 이용하였다. 그리고 유도한 식을 컴퓨터에서 처리하기 용이한 점화식으로 변환한 결과 곡선식의 값은 곱셈이 필요한 다항식 표현과는 달리 오직 덧셈 만으로 구할 수 있었다.

CDAUD의 시간 복잡도를 분석해 본 결과 비록 제한적이기는 하지만 비교 대상으로 선정한 Horner 규칙을 이용하여 곡선을 그리는 방법보다 비록 제한적이기는 하지만 우수하다는 것을 알 수 있다. Horner 규칙을 이용한 방법을 비교 대상으로 선정한 것은 이 방법이 보다 다항식의 값을 빨리 계산할 수 있기 때문이다. 또 실제로 프로그램을 작성하여 실행해 본 결과 CDAUD는 비교 대상보다 곡선을 빨리 그릴 수 있음을 확인할 수 있었다.

그러나 본 연구의 방법도 단점이 있다. 즉 곡선의 식이 3차로 제한된다는 점이며 다항식이 4차 이상인 경우는 프로그램이 길어지고 또 알고리즘의 수행과정에서 많은 임시 상수의 계산이 필요하게 되므로 효율이 떨어져 적용하기 어렵다는 단점이 있다. 특히 짧은 구간의 곡선을 그리는 경우에는 오버헤드에 소요되는 시간이 너무 많아서 효율이 떨어진다. 때문에 앞으로 중간의 임시 상수를 효율적으로 구하는 연구가 필요하다.

참 고 문 헌

[1] Jerry van Aken and Mark Novak, "Curve Drawing Algorithms for Raster Displays," ACM Trans., Vol. 4, No. 2, pp. 147-169, 1985.

[2] Suenaga Y., Kamae T., Kobayashi T., "A High-speed Algorithm for the Generation of Straight Lines and Circular Arcs," IEEE Trans.Computers, C-28, 10, pp. 728-736, 1979.

[3] Kurokawa T., Mizukoshi T., "A Fast and Simple Method for Curve Drawing -A New Approach Using Logarithmic Number Systems," J. INF. Process(JAPAN), Vol. 14, No. 2, pp. 144-52, 1991.

[4] Gerald Farin, "Curves and Surfaces for Computer aided Geometric Design:A Practical Guide," 2nd Edition, Academic Press, San Diego, 1990.

[5] Cornel K. Pokorney, Curtis F.Gerald, "Computer Graphics:The Principles behind the Art and Science", Franklin Beedle & Associates, Califonia, pp. 593-621, 1989.

[6] Michael E. Mortenson, "Geometric Modelling", Jone Wiley & Sons, New York, pp. 19-27, 1985.

[7] Rogers D. E. and J. Alan Adams, "Mathematical Elements for Computer Graphics", 2nd Edt., McGRAW Hill, 1990.

[8] P. E. Danielsson, "Incremental Curve Generation," IEEE Trans. Comput. Vol. C-19, pp. 783-793, 1990.

[9] B. W. Jordan Jr. et al., "An Improved Al-

gorithm for the Generation of Nonparametric Curves," IEEE Trans. Comput., Vol. C-22, No 12, pp. 1052-1060, 1973.

[10] Richard L. Burden, J. Douglas Faires, Albert C.Reynolds, "Numerical Analysis", 2dn Edt., Prindle, Weber & Schmidt, pp. 73-119, 1981.

[11] Ellis Horowitz and Sartaj Sahni, "Fundamentals of Computer Algorithms", Computer Science Press, pp. 424-431, 1978.

[12] 양해술, 백청호, 이창석, "수치해석", 상조사, pp. 214-217, 1991.

이 상 락

1971년 서울대학교 사범대학 물리과 졸업
1983년 광운대학교 대학원 전자계산학과 졸업(이학석사)
1988년 광운대학교 대학원 박사과정 수료
1988~1994년 인천대학교 전자계산학과 조교수
관심분야 : 알고리즘, 그래픽스, 자연어 처리

심 재 홍

1967년 서울대학교 수학과 졸업
1980년 고려대학교 대학원(이학석사)
1981~1988년 경희대학교 대학원(이학박사)
1984~1986년 정보과학회 부회장 역임
1992년 광운대학교 총장
현재 광운대학교 전자계산학교 교수로 재직중
관심분야 : 그래프알고리즘, 그래픽스, 수치해석