

Logarithmen und Wurzeln

Viktor Engelmann

30. Oktober 2003

Computer-Prozessoren können im Wesentlichen nur addieren, subtrahieren, multiplizieren, dividieren und modulo-Rechnung. Komplexere mathematische Operationen müssen auf diese Grundoperationen zurückgeführt werden, damit man sie im Computer berechnen kann. Die Exponentialfunktion mit der Eulerschen Zahl e als Basis (diese Funktion nennt man auch exp) und der Logarithmus Naturalis (Logarithmus zur Basis e) sind mit Taylor-Reihen auf Grundoperationen zurückzuführen:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \quad \ln(x) = 2 \cdot \sum_{n=0}^{\infty} \frac{(x-1)^{2n+1}}{(x+1)^{2n+1} \cdot (2n+1)}$$

Mit diesen beiden Funktionen wollen wir nun Wurzeln und Logarithmen auf Grundoperationen zurückführen:

$x = \log_a(y)$ ist Lösung von $a^x = y$ (aufgrund der Injektivität und Surjektivität von e^x ist die Lösung eindeutig). Ferner besagen die Logarithmengesetze, dass $\log_r(p^q) = q \cdot \log_r(p)$. Da e^x und $\ln(x)$ ausserdem invers zueinander sind, gilt

$$\begin{aligned} y &= a^x \\ &= e^{\ln(a^x)} \\ &= e^{x \cdot \ln(a)} \\ \Leftrightarrow \ln(y) &= x \cdot \ln(a) \\ \Leftrightarrow \frac{\ln(y)}{\ln(a)} &= x \end{aligned}$$

$x = \sqrt[a]{y}$ ist Lösung von $x^a = y$ (um Injektivität zu erreichen, schränkt man den Wertebereich für x auf \mathbb{R}^+ ein).

$$\begin{aligned} y &= x^a \\ &= e^{\ln(x^a)} \\ &= e^{a \cdot \ln(x)} \\ \Leftrightarrow \ln(y) &= a \cdot \ln(x) \\ \Leftrightarrow \frac{\ln(y)}{a} &= \ln(x) \\ \Leftrightarrow e^{\frac{\ln(y)}{a}} &= x \end{aligned}$$

Da Logarithmen von negativen Zahlen nicht definiert sind und Division durch 0 nicht definiert ist, müssen wir für die *log* Funktion voraussetzen, dass *a* und *y* größer als 0 sind und für die Wurzelfunktion muss *y* größer als 0 und *a* \neq 0 sein.

Mit Funktionen für *ln* und *exp*

```
float exp(float x) {
    float powerxn = 1;
    float nfak = 1;
    float sum = 0;
    for(int n = 1; n <= 20; n++) {
        sum += powerxn / nfak;
        powerxn *= x;
        nfak *= n;
    }
    return sum;
}
```

```
float ln(float x) {
    if(x <= 0)
        throw "ln(x): x muss positiv sein!";
    float sum = 0;
    float numerator = x-1;
    float denominator = x+1;
    for(int n = 1; n <= 40; n+=2) {
        sum += numerator / (n*denominator);
        numerator *= (x-1)*(x-1);
        denominator *= (x+1)*(x+1);
    }
    return 2*sum;
}
```

kann man nun definieren

```
float log(float a, float y) {
    if( y<=0 || a<=0 )
        throw "log(a,y): a und y muessen positiv sein!";
    return ln(y) / ln(a);
}
```

```
float root(float a, float y) {
    if( y<=0 || a==0 )
        throw "root(a,y): y muss positiv sein und a darf nicht 0 sein!";
    return exp( ln(y) / a );
}
```