1.2.2.12 IEEE-Gleitkommazahlen

IEEE-Gleitkommazahlen im 64 Bitformat sind von der Form

$$x_d = s \cdot 2^e \cdot \sum_{k=1}^{53} f_k \cdot 2^{-k} \tag{1.1}$$

mit genau 53 signifikanten Dualstellen, wobei $s = \pm 1, -1021 \leq e \leq 1024$ und $f_i \in \{0,1\}$ mit $f_1 \neq 0$ oder

$$x_d = s \cdot 2^{-1021} \cdot \sum_{k=2}^{53} f_k \cdot 2^{-k} \tag{1.2}$$

(inklusive der 0). Zusätzlich sind folgende drei Ausdrücke vordefiniert:

NaN	not a number
$+infty$	$+\infty$
$-infty$	$-\infty$

Man beachte, daß sich die Zahl $\frac{1}{3}$ nicht exakt im IEEE-Format darstellen läßt. Die Darstellung einer Zahl im IEEE-Format erfolgt dabei durch das folgende Bitmuster:

s	e[52:62]	f[51:32]
63	62 … 52	51 … 32

f[31:0]
31 … 0

Werte werden dabei entsprechend der folgenden Tabelle angenommen:

Double-Format Bitmuster	Wert
$0 < e < 2047$	$(-1)^s \cdot 2^{e-1023} \cdot 1.f$ (normal numbers)
$e = 0, f \neq 0$ (mindestens ein Bit in f ist ungleich 0)	$(-1)^s \cdot 2^{-1022} \cdot 0.f$ (subnormal numbers)
$e = 0, f = 0$ (alle Bits in f sind 0)	$(-1)^s \cdot 0.0$ (signed zero)
$s = 0, e = 2047, f = .000 \cdots 0$ (alle Bits in f sind 0)	+INF (positive infinity)
$s = 1, e = 2047, f = .000 \cdots 0$ (alle Bits in f sind 0)	-INF (negative infinity)
$s = u, e = 2047, f \neq 0$ (mindestens ein Bit in f ist ungleich 0)	NaN (not a number)

- **Maximum positive normal number** — die größte im IEEE Double-Format darstellbare endliche Zahl
- **Minimum positive normal number** — die kleinste im IEEE Double-Format darstellbare positive Zahl, bei der sicher ist, daß nur die letzte Binärstelle unsicher ist.
- **Maximum subnormal number** — die größte im IEEE Double-Format darstellbare Zahl, bei der es möglich ist, daß mehr als die letzte Binärstelle unsicher ist.
- **Minimum positive subnormal number** — die kleinste im IEEE Double-Format darstellbare positive Zahl; hier ist es möglich, daß mehr als die letzte Binärstelle unsicher ist.

Es ergeben sich dabei folgende Beispiele:

Üblicher Name	Bitmuster (Hex)	Äquivalenter Wert
+0	00000000 00000000	0.0
−0	80000000 00000000	−0.0
1	3ff00000 00000000	1.0
2	40000000 00000000	2.0
max normal number	7fefffff ffffffff	1.7976931348623157e+308
min positve normal number	00100000 00000000	2.2250738585072014e−308
max subnormal number	000fffff ffffffff	2.2250738585072009e−308
min positve subnormal number	00000000 00000001	4.9406564584124654e−324
$+\infty$	7ff00000 00000000	Infinity
$-\infty$	fff00000 00000000	−Infinity
Not-a-Number	7ff80000 00000000	NaN

Im IEEE 32 Bitformat (single precision) gilt analog:

Gleitkomma Single-Precision Format	
s = sign (1 bit) e = biased exponent (8 bits) f = fraction (23 bits) u = undefined	
Normalized value ($0 < e < 255$)	$(-1)^s \cdot 2^{e-127} \cdot 1.f$
Subnormal value ($e = 0$)	$(-1)^s \cdot 2^{-126} \cdot 0.f$
Zero ($e = 0$)	$(-1)^s \cdot 0$
Signalling NaN	$s = u, e = 255\,(\max)\; f = .0uu \cdots uu$ (mindenstens ein Bit in $f \neq 0$)
Quiet NaN	$s = u, e = 255\,(\max)\; f = .1uu \cdots uu$
$-\infty$	$s = 1, e = 255\,(\max)\; f = .000 \cdots 00$
$+\infty$	$s = 0, e = 255\,(\max)\; f = .000 \cdots 00$

Dazu noch einmal vergleichend das 64 Bitformat

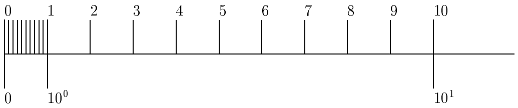
Gleitkomma Double-Precision Format	
s = sign (1 bit) e = biased exponent (11 bits) f = fraction (52 bits) u = undefined	
Normalized value ($0 < e < 2047$)	$(-1)^s \cdot 2^{e-1023} \cdot 1.f$
Subnormal value ($e = 0$)	$(-1)^s \cdot 2^{-1022} \cdot 0.f$
Zero ($e = 0$)	$(-1)^s \cdot 0$
Signalling NaN	$s = u, e = 2047\,(\max)\; f = .0uu \cdots uu$ (mindenstens ein Bit in $f \neq 0$)
Quiet NaN	$s = u, e = 2047\,(\max)\; f = .1uu \cdots uu$
$-\infty$	$s = 1, e = 2047\,(\max)\; f = .000 \cdots 00$
$+\infty$	$s = 0, e = 2047\,(\max)\; f = .000 \cdots 00$

und das 128 Bitformat:

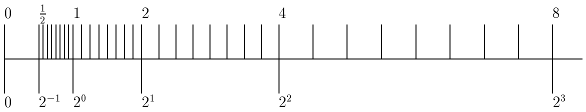
Gleitkomma Quad-Precision Format	
s = sign (1 bit) e = biased exponent (15 bits) f = fraction (112 bits) u = undefined	
Normalized value ($0 < e < 32767$)	$(-1)^s \cdot 2^{e-16383} \cdot 1.f$
Subnormal value ($e = 0$)	$(-1)^s \cdot 2^{-16382} \cdot 0.f$
Zero ($e = 0$)	$(-1)^s \cdot 0$
Signalling NaN	$s = u, e = 32767\,(\max)\; f = .0uu \cdots uu$ (mindenstens ein Bit in $f \neq 0$)
Quiet NaN	$s = u, e = 32767\,(\max)\; f = .1uu \cdots uu$
$-\infty$	$s = 1, e = 32767\,(\max)\; f = .000 \cdots 00$
$+\infty$	$s = 0, e = 32767\,(\max)\; f = .000 \cdots 00$

Zur Veranschaulichung der absoluten Abstände in Gleitkomma-Zahlensystemen:

Decimal Representation



Binary Representation



Die Lücken zwischen darstellbaren Single-Precision Format Gleitkommazahlen:

x	**Nachbarzahl von x in Richtung $+\infty$**	**Lücke**
0.0	1.4012985e-45	1.4012985e-45
1.1754944e-38	1.1754945e-38	1.4012985e-45
1.0	1.0000001	1.1920929e-07
2.0	2.0000002	2.3841858e-07
16.000000	16.000002	1.9073486e-06
128.00000	128.00002	1.5258789e-05
1.0000000e+20	1.0000001e+20	8.7960930e+12
9.9999997e+37	1.0000001e+38	1.0141205e+31

Die üblichen Gleitkomma-Zahlensysteme im Vergleich:

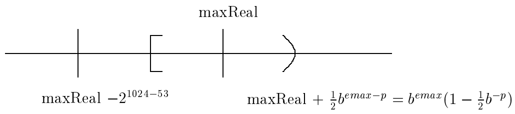
Wertebereich und Genauigkeit				
Format	**Signifikante Binärstellen**	**min positiv normal number**	**max normal number**	**Signifikante Dezimalstellen**
single	24	$1.175\ldots10^{-38}$	$3.402\ldots10^{+38}$	6-9
double	53	$2.225\ldots10^{-308}$	$1.797\ldots10^{+308}$	15-17
double extended (SPARC)	113	$3.362\ldots10^{-4932}$	$1.189\ldots10^{+4932}$	33-36
double extended (x86)	64	$3.362\ldots10^{-4932}$	$1.189\ldots10^{+4932}$	18-21

Format	**Underflow-Schwelle**	
Single	Smallest Normal Number	1.17549435e-38
	Largest Subnormal Number	1.17549421e-38
Double	Smallest Normal Number	2.2250738585072014e-308
	Largest Subnormal Number	2.2250738585072009e-308
Double extended (SPARC)	Smallest Normal Number	3.3621031431120935062626778173217526e-4932
	Largest Subnormal Number	3.3621031431120935062626778173217520e-4932
Double extended (x86)	Smallest Normal Number	3.36210314311209350626e-4932
	Largest Subnormal Number	3.36210314311209350590e-4932

IEEE double precision entspricht dem Zahlenbereich $S(2, 53, -1021, +1024)$. Damit ist die größte darstellbare Zahl

$$maxReal = \sum_{k=1}^{53} 2^{-k} \cdot 2^{1024} = (1 - 2^{-53}) \cdot 2^{1024} = (1 - 2^{-p}) \cdot 2^{emax}$$

und für $[maxReal, +\infty)$ ergibt sich bei der Rundungsart r_{near} und $b = 2$:



Beispiele für gute IEEE-Algorithmen

1. **Invarianz von Eigenschaften in $\mathbb{R}$ auch in `real`:**

```
function arithMittel(x,y : real) : real; {IEEE}
begin
    if sign(x)=sign(y) then
        arithMittel := x + (y-x)/2.0
    else
        arithMittel := (x+y)/2.0
end; {arithMittel}
```

Dieser Algorithmus weist folgende Eigenschaften auf, wobei ohne Einschränkung davon ausgegangen werden kann, daß $x \leq y$ gilt:

(a) Fehler $\leq 1\,ulp$, wobei ulp für *unit in last place* steht

(b) Allgemein gilt $x \leq arithMittel \leq y$; falls zwischen x und y ein weiterer Gitterpunkt liegt, so gilt $x < arithMittel < y$.

(c) Es tritt in keinem Fall Over- oder Underflow auf.

Die angegebene zweite Eigenschaft des Algorithmus ist keineswegs selbstverständlich. Es gibt nämlich $x, y; x \neq y$ mit:

$$\frac{x \oplus y}{2} \notin [\min(x, y), \max(x, y)]$$

Man betrachte etwa im 13-stelligen Dezimalsystem die Zahlen

$$\begin{aligned} x &= 0.9\cdots96 \cdot 10^0 \\ y &= 0.9\cdots98 \cdot 10^0 \end{aligned}$$

Dann ist $x \oplus y = 0.19\cdots9 \cdot 10^1$ und somit

$$\frac{x \oplus y}{2} = 0.9\cdots95 \cdot 10^0.$$

Es wird nun der Zahlenbereich $S(10, 2, -5, +5)$ mit $S' := S \cup \{Underflowbereich\}$ und die Operation

$$\frac{x}{2} + \frac{y}{2}$$

behandelt.

(a) Für

$$x = 0.13 \cdot 10^{-5}, \quad y = 0.50 \cdot 10^{-4}$$

erhält man:

$$\begin{aligned} \frac{x}{2} &= 0.065 \cdot 10^{-5} \overset{in\,S'}{=} \begin{cases} 0.07 \cdot 10^{-5} & \text{bei grad. Underflow} \\ 0.0 & \text{trad.} \end{cases} \\ \frac{y}{2} &= 0.25 \cdot 10^{-4} \\ \frac{x}{2} + \frac{y}{2} &= \begin{cases} 0.257 \cdot 10^{-4} \overset{in\,S'}{=} 0.26 \cdot 10^{-4} & \\ 0.25 \cdot 10^{-4} & \text{trad.} \end{cases} \end{aligned}$$

(b) Für

$$x = 0.13 \cdot 10^{-5}, \quad y = 0.10 \cdot 10^{-5}$$

erhält man:

$$\begin{aligned} \frac{x}{2} &\overset{in\,S'}{=} 0.07 \cdot 10^{-5} \\ \frac{y}{2} &\overset{in\,S'}{=} 0.05 \cdot 10^{-5} \\ \frac{x}{2} + \frac{y}{2} &\overset{in\,S'}{=} 0.12 \cdot 10^{-5} \end{aligned}$$

(c) Sind

$$x = 0.13 \cdot 10^{-5}, \quad y = 0.11 \cdot 10^{-5},$$

so erhält man bei trad. Rechnung $\frac{x}{2} + \frac{y}{2} = 0$ und $x + \frac{y-x}{2} = x$. In S' ist jedoch

$$\begin{aligned} \frac{x}{2} &\overset{in\,S'}{=} 0.07 \cdot 10^{-5} \\ \frac{y}{2} &\overset{in\,S'}{=} 0.06 \cdot 10^{-5} \\ \frac{x}{2} + \frac{y}{2} &\overset{in\,S'}{=} 0.13 \cdot 10^{-5} \end{aligned}$$

während

$$\begin{aligned} y - x &\overset{in\,S'}{=} -0.02 \cdot 10^{-5} \\ (y - x)/2 &\overset{in\,S'}{=} -0.01 \cdot 10^{-5} \\ x + (y - x)/2 &\overset{in\,S'}{=} 0.12 \cdot 10^{-5}. \end{aligned}$$

2. **Monotonie** (Voraussetzung für Nullstellensuchverfahren, z.B. Bijektion,...)
   Der Graph von

$$p(x) = 170.4x^3 - 356.41x^2 + 168.97x + 18.601$$

liegt in $[1.09160791, 1.09160805] \times [-2 \cdot 10^{-13}, 5 \cdot 10^{-13}]$.

In beiden Fällen wurde mit 16-stelliger Arithmetik gerechnet bzw. auf ebenso viele Stellen (automatisch) verifiziert. Die beiden positiven Nullstellen des obigen Polynoms liegen sehr nahe zusammen, wie die Zerlegung in Linearfaktoren

$$p(x) = 170.4 \cdot \left(x - \frac{5+\sqrt{35}}{10}\right)\left(x - \frac{5-\sqrt{35}}{10}\right)\left(x - \frac{18601}{17040}\right)$$

und die unter Verwendung von ADE berechneten Einschließungen

$$\frac{5+\sqrt{35}}{10} = 1.09160797830996_1^2 \quad \text{und} \quad \frac{18601}{17040} = 1.09160798122065_7^8$$

zeigen.

3. **Lösung einer quadratischen Gleichung (Auslöschung):**
Es wird der Rundungsfehlereinfluß bei 4-stelliger Arithmetik, also bei Gleitkommazahlen mit 4 Mantissenstellen betrachtet. Wir wollen das wohlbekannte Problem

$$ax^2 + bx + c = 0.0$$

nach der Lösungsformel $x_{1/2} = (-b \pm \sqrt{b^2 - 4.0 \cdot a \cdot c})/(2.0 \cdot a)$ mit 4-stelliger Arithmetik (d.h. alle Zwischenergebnisse haben nur 4 signifikante Stellen!) lösen für

$$a = +1.0004, \quad b = -200.01, \quad c = +1.0003 \tag{1.3}$$

<u>1. Fehler:</u> Representation von a, b, c nur vierstellig, also

$$\bar{a} = +1.000, \quad \bar{b} = -200.0, \quad \bar{c} = 1.000$$

Die Rechnung ergibt im 1. Schritt den <u>2. Fehler</u>, nämlich einen Rundungsfehler bei der Subtraktion:

$$\bar{b}^2 - 4.0 \cdot \bar{a} \cdot \bar{c} = 40000 - 4.000 = 40000$$

Der 2. Schritt liefert

$$\bar{x}_1 = (200.0 + 200.0)/2.0 = 200.0$$

und im 3. Schritt ergibt sich durch einen <u>Folgefehler</u> des 2. Fehlers

$$\bar{x}_2 = (200.0 - 200.0)/2.0 = 0.0$$

Die exakten Lösungen lauten jedoch auf 4 signifikante Ziffern genau

$$\bar{x}_1 = 200.0$$
$$\bar{x}_2 = 0.005000$$

Die exakten Lösungen des gestellten Problems (1.3) wären auf 4 Ziffern genau

$$x_1 = 200.0$$
$$x_2 = 0.004999$$

gewesen.
Als stabilere Lösungsidee bietet sich für $a \neq 0$ nach der Umformung

$$ax^2 + bx + c = 0 \quad \Leftrightarrow \quad x_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

die vorzeichengünstige Wurzel

$$x_1 := \frac{-b + \operatorname{sign}(-b)\sqrt{b^2 - 4ac}}{2a}$$

an. Der entscheidende Vorteil ist, daß die Summanden im Zähler das gleiche Vorzeichen besitzen. Nach dem Satz von Vieta ist dann die zweite Wurzel gegeben durch:

$$x_2 := \frac{c}{a \cdot x_1},$$

wegen $x_1 x_2 = \frac{c}{a}$ und $x_1 + x_2 = -\frac{b}{a}$.

4. **Unvergleichbare IEEE-Zahlen:**
Man betrachte den folgenden Programmausschnitt (Fraktale/Apfelmännchen);

```
⋮
y1 = 0 ;
y2 = 0 ;
for (i=0;i<ITERATIONS;i++)
{
      x1 = y1 ;
      x2 = y2 ;
      y1 = x1*x1-x2*x2+c1 ;
      y2 = 2.0*x1*x2+c2 ;
      n2 = y1*y1+y2*y2 ; /* Norm ↑ 2 */
      if (n2>4.0) then
         /* skip */
      else plot(y1,y2) ;
}
```

Geplotten wird, wenn $\|y\|_2 \leq 2.0$, *aber:* Falls y_i sehr groß (nahe $+\infty$), so wird wegen $y_1 = +\infty - \infty + c = NaN$ ebenfalls geplottet! Besser wäre also:


```
if (n2>4.0) then
      ...
else if unordered(n2,4.0) then
      ...
else ···
```

Beachte also:

```
if (a>b) then
      ...
else {a ≤ b ∨ unordered(a,b)}
      ...
```

Extremsituationen

Ausnahmebedingungen:

IEEE-Ausnahmebedingungen	Grund des Auftretens	Beispiel	Ergebnisse ohne Exception-Header
Ungültige Operation	Die Operation ist für den auftretenden Operanden nicht definiert. Diese Ausnahmebedingung kann beim x86 auch bei Stack-Fehlern auftreten	$\sqrt{(-x)}$ für $x > 0$ fp_op (signaling_NaN) $0 \cdot \infty, 0/0, \infty/\infty, x$ rem 0 ungeordneter Vergleich unerlaubte Umwandlung	Quiet NaN
Division durch Null	Der Divisor ist gleich Null und der Dividend ist eine endliche Zahl ungleich Null. Allgemeiner: ein exaktes unendliches Ergebnis wird durch eine Operation auf endlichen Operanden geliefert.	$x/0$ für $x \neq 0, \infty$ oder NaN log(0)	∞ mit richtigem Vorzeichen

Tabellenfortsetzung

<table>
<tr><td>Überlauf</td><td>Ein korrekt gerundetes Ergebnis ist größer als die größte darstellbare Zahl, d.h. der Bereich des Exponenten ist erschöpft.</td><td>Double precision:
MAXDOUBLE+1.0e294
exp(709.8)
Single precision:
(float)MAXDOUBLE
MAXFLOAT+1.0e32
expf(88.8)</td><td>Abhängig von der Rundungsart (RM) und dem Vorzeichen des Zwischenergebnisses<table><tr><td>RM</td><td>+</td><td>−</td></tr><tr><td>RN</td><td>$+\infty$</td><td>$-\infty$</td></tr><tr><td>RZ</td><td>$+\max$</td><td>$-\max$</td></tr><tr><td>R−</td><td>$+\max$</td><td>$-\infty$</td></tr><tr><td>R+</td><td>$+\infty$</td><td>$-\max$</td></tr></table></td></tr>
<tr><td>Unterlauf</td><td>Tritt auf, wenn das exakte Ergebnis von Null verschieden ist, liegt stets im zwischen $\pm 2^{e_{min}}$</td><td>Double precision:
nextafter(min_normal,$-\infty$)
nextafter(min_subnormal,$-\infty$)
MINDOUBLE/3.0
exp(-708.5)
Single precision:
(float)MINDOUBLE
nextafterf(MINFLOAT,$-\infty$)
expf(-87.4)</td><td>Subnormal oder Null</td></tr>
<tr><td>Ungenauigkeit</td><td>Das gerundete Ergebnis stimmt nicht mit dem exakten Ergebnis überein.</td><td>2.0/3.0
(float)1.12345678
log(1.1)
MAXDOUBLE+MAXDOUBLE, falls kein Überlauf auftritt</td><td>Ergebnis der Operation (gerundet), Unter- oder Überlauf</td></tr>
<tr><td>Unnormalisierter Operand (nur bei x86)</td><td>Versuch, eine arithmetische Operation mit unnormalen Operanden durchzuführen</td><td>1.0+MINDOUBLE/2.0</td><td>Ergebnis der Operation (gerundet), Unter- oder Überlauf</td></tr>
</table>