# Reprezentácia čísel v MatLabe[1]

LITERATÚRA:

http://www.mathworks.com/moler/intro.pdf (kapitola 1.7)
http://steve.hollasch.net/cgindex/coding/ieeefloat.html
http://standards.ieee.org/
http://www.cs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF

Čísla sú v MatLabe reprezentované v dvojkovej sústave, väčšinou v normatívnom tvare tzv. *floating-point* aritmetiky s dvojnásobnou (*double*) presnosťou. Na záznam čísla je vyhradených 64 bitov.

| | |
|---|---|
| 1 bit | pre znamienko čísla |
| 11 bitov | pre exponent |
| 52 bitov | pre mantisu |

**Znamienko** čísla *S* má hodnotu 0 pre kladné a hodnotu 1 pre záporné čísla. Matematicky to možno vyjadriť ako $(-1)^S$ .

**Exponent** *e* je 11-bitový a MatLab si ho pamätá v podobe $E = e+1023$. Pokiaľ má *E* hodnoty medzi 1 a $2^{11}-2$, zodpovedá to exponentom –1022 až 1023. Krajné hodnoty 0 a $2^{11}-1$ (v dvojkovej sústave 00000000000 a 11111111111) sú rezervované pre zvláštne prípady (čísla v nenormovanom tvare).

**Mantisa** 1.f sa skladá z implicitnej (tj. neukladá sa v pamäti a automaticky sa predpokladá) 1 na začiatku, tj. prvej nenulovej cifry v binárnom zápise čísla, binárnej bodky a tzv. zlomku *f*, kde sú v 52 bitoch uložené ďalšie cifry čísla.

Príklad: Ako si MatLab pamätá číslo 44.5 ?

Preložíme číslo do dvojkovej sústavy: $44.5 = 32+8+4+0.5 = 101100.1_2$
V normovanom tvare je to: $1.011001_2 * 2^5 = 1.011001 \text{ e } 101_2$

MatLab si teda pamätá znamienko 0, exponent $E=5+1023=1028=10000000100_2$ , a zlomok mantisy $f=011001000\ldots_2$

| Znam. | Exp. | Mantisa |
|---|---|---|
| 0 | 10000000100 | 0110010000 0000000000 0000000000 0000000000 0000000000 00 |

Úlohy: Nájdite reprezentáciu v MatLabe pre nasledujúce čísla (ak treba, zaokrúhlite):

5479.375 7.1 pi

[1] Testované na MatLabe 5.3.0.10183 R11 a 6.0.0.88 R12

**Najväčšie číslo v MatLabe – tri cesty k nemu**

**A.** Určíme ho skusmo – testujme rôzne mocniny dvojky. Najvyššia mocnina, ktorú MatLab dokáže registrovať, je $2^{1023}$.

```
>> 2^1023
                ans =       8.988465674311580e+307

>> 2^1024,  2^1023*1.999999999999999

                ans =   Inf
                ans =  1.797693134862315e+308
```

2^1024 síce prekročilo možnosti MatLabu, avšak k tejto hodnote sa dá tesne priblížiť. Najvyššie číslo, ktoré sme získali, je 1.797693134862315e+308.

**B.** Položme otázku priamo MatLabu: » realmax

ans = 1.797693134862316e+308

**C.** Na základe teoretických vedomostí o reprezentácii čísel v MatLabe by najvyššou reprezentovateľnou hodnotou malo byť:

+1.1111111111 1111111111 1111111111 1111111111 1111111111 11 **e** +1111111111

čo je v desiatkovej sústave približne 1.9999999999999998 *2^1023

Skúsme si to overiť – zostavíme uvedené číslo postupne tak, ako je zapísané v dvojkovej sústave:[2]

```
>> t=0; for i=0:52, t=t+2^(-i); end, x=(t)*2^1023

x =    1.797693134862316e+308
```

Číslo je rovnaké sme dostali v **B**. Teraz ho mierne zväčšíme:

```
>> x =    1.797693134862317e+308

x =   Inf
```

To sme prehnali, skúsme pridať jednotku za poslednú pozíciu binárneho zápisu:

```
>> x=(t+2^(-53))*2^1023

x =   Inf
```

Ani toto nerozchodil... jednotka na 53. mieste za bodkou (54. na mantisi) spôsobila zaokrúhľovanie nahor na 2^1024, a to je priveľa. Sme na hranici, nájdené najväčšie číslo nezväčšíme.

[2] Poznámka: Prečo nemôžeme toto číslo počítať ako 2-(2^-53)*2^1023 ?

**Zaokrúhľovanie I.**

Mantisa, ako sme už spomenuli, je zložená z prvej implicitnej jednotky (tzv. leading digit – vedúca cifra) a 52 ďalších cifier za binárnou bodkou. Čísla s väčším počtom cifier, prípadne reálne čísla s nekonečne dlhým binárnym rozvojom sa musia vhodne zaokrúhliť.

a) Cifry na 52. pozícii za bodkou ML registruje. Nasledujúce číslo si zachová presne:

1.0000000000 0000000000 0000000000 0000000000 0000000000 01

» q = 52; r = 2^(-q); a = 1+r; b = a-1; [r,b]

ans = 1.0e-016 *

2.220446049250313 2.220446049250313

b) Cifru 1 na 53 pozícii už ML nevie uchovať presne a preto zaokrúhľuje smerom nadol:

» q=53; s=2^(-q); a=1+s; b=a-1; [r,b]

ans = 1.0e-015 *

0.1110 0

c) V prípade, ak cifre na 53. pozícii predchádza 1, zaokrúhľovať sa bude nahor:

» a=1+r+s; b=a-1; [r+s, b]

ans = 3.330669073875470e-016 4.440892098500626e-016

Zhrnutie:

1. MatLab zaokrúhľuje na 52 cifier za binárnou bodkou. Jednotka na 53. pozícii (a za ňou už nič, tj. nerozhodná situácia) sa zaokrúhľuje nahor, ak na 52. pozícii je 1 a nadol, ak je na 52. pozícii 0.

2. Jednotka na poslednej pozícii, čiže hodnota $2^{-52}$ pri nulovom exponente e, zodpovedá tzv. počítačovému epsilon:

» [eps, 2^-52]

ans = 2.220446049250313e-016 2.220446049250313e-016

**Zvláštne čísla v MatLabe I.**

Ak hodnota čísla presiahne hodnotu *realmax*, nastáva tzv. *overflow*. MatLab vypíše na obrazovku Inf (nekonečno) a túto hodnotu reprezentuje číslami

$$f=00000\ldots,\ E=2^{11}-1=11111111111,$$

tj. v počítačovej množine je to 1.0 e 1024.

S týmto číslom vie normálne pracovať. Platí (pre ľubovoľné k):

Inf+Inf=Inf
1/Inf=0
k*Inf=Inf
k+Inf=Inf atď.

Ak hodnotu čísla nemožno jednoznačne určiť, napr. pre 0/0 alebo Inf-Inf, ML hodnotí výsledok ako NaN (*not a number*, nie je to číslo) a reprezentuje ho číslami

$$f>0,\ E=2^{11}-1.$$

**Najmenšie kladné číslo v MatLabe**

**A.** Najmenšie normované kladné reálne číslo v ML je podľa teórie

$1.0 * 2^{-1022}$ , tj. 1.0 e-1111111110.

Zodpovedá mu reprezentácia f=000000… a E=00000000001.

```
» 2^-1022
                ans =    2.225073858507201e-308
```

**B.** Oficiálna odpoveď MatLabu na otázku o najmenšom reálnom čísle:

```
» realmin
                ans =    2.225073858507201e-308
```

**C.** Experimentálne určenie najmenšieho čísla – počítajme hodnoty záporných mocnín dvojky. Najmenšia mocnina dvojky, ktorú MatLab dokáže registrovať ako odlišnú od nuly, je až 2^(-1074)!

```
>> mp=2^(-1074)
                mp =    4.940656458412465e-324

>> t=2^(-1075), t-0
                t =     0
                ans =     0
```

Polovica *mp* je už pre MatLab zhodná s nulou. Skúmajme ďalej:

```
>> mp/1.999999999999999
                ans =    4.940656458412465e-324
```

Vidíme, že mp/1.999999999999999 je pre MatLab stále to isté ako *mp*. Pozorované zaokrúhľovanie svedčí o tom, že mp je minimálne kladné číslo v MatLabe a nižšie sa nedostaneme. Otázkou však je, prečo je *mp* iné, než by sme na základe teórie očakávali.

## Zvláštne čísla v MatLabe II.

V jednoduchých výpočtových prostrediach sa číslo (zadané alebo získané výpočtom) menšie než *realmin* hodnotí ako *underflow* a zaokrúhľuje sa na *realmin* alebo 0. V niektorých prostrediach, vrátane MatLabu, však je možné zobraziť aj čísla menšie ako *realmin* – hovorí sa im *subnormálne* resp. *denormálne (denormalizované) čísla*. Spôsob ich zobrazenia v rámci počítačovej množiny čísel je neštandardný – exponent sa formálne kladie rovný -1023 (tj. E=0) a vtedy ML číta číslo inak – exponent považuje za rovný -1022 (ako pri E=1) a zároveň „vypína“ pevnú jednotku pred mantisou. Týmto spôsobom sa dá zobraziť aj číslo menšie ako *realmin*, avšak s menšou presnosťou.

V *subnormálnom* režime možno dosiahnuť **najnižšiu** kladnú hodnotu pre f= 0000…01, E=00000000000 a bude to:

0.0000000000 0000000000 0000000000 0000000000 0000000000 01 e-1111111110,

čo je v desiatkovej sústave

2^(-52)*2^(-1022) = 2^(-1074)= 4.940656458412465e-324.

V denormálnom režime sa zobrazuje aj 0, keďže v normovanom tvare s implicitnou 1 na začiatku ju zobraziť nemožno. Číslu 0 zodpovedajú hodnoty

f= 0000…0, E=00000000000, so znamienkom + alebo -. [3]

**Najvyšším** kladným *subnormálnym* číslom je 0.11111…1 e –1022.

» t=0; for i=1:52, t=t+2^(-i); end, x=(t)*2^-1022

x = 2.225073858507201e-308

Rozdiel medzi najmenším normovaným kladným číslom a najvyšším subnormálnym číslom (v zápise v desiatkovej sústave rozdiel nevidíme) je

x- realmin = -4.940656458412465e-324

Táto hodnota je rovnaká ako hodnota najmenšieho subnormálneho čísla a získať ju môžeme aj ako eps*realmin.

[3] 0 sa dá zobraziť dvoma rôznymi spôsobmi, ale pri výpočtoch sú obe hodnoty ekvivalentné.

**Zaokrúhľovanie v MatLabe II.**

Zadajme nasledujúce číslo a pozrime sa, ako ho ML akceptoval:

```
» 77.546
                ans =    7.754600000000001e+001
```

Bez toho, aby sme čokoľvek robili, ML číslo mierne zmenil. Dôvod tejto zmeny sa ukáže, keď po malých krokoch a pozorne urobíme presne to, čo ML.

A. Napíšeme číslo 77.546 v desiatkovej sústave.
B. ML číslo skonvertujeme do dvojkovej sústavy.

77=64+8+4+1+0.546
77=2^6+2^3+2^2+2^0+…=1001101. …

Ostalo 0.546 a na mantisi 46 miest. Ostatné cifry za bodkou sa ručne hľadajú zdĺhavo a s hrozbou omylov, preto necháme počítať ML (po 52. pozíciu):

```
» x=0.546; v=zeros(1,52);
» for i=1:50, a=x-2^(-i); if a>0 v(i)=1; x=a; end, end, v
```

Odpoveďou je vektor, ktorého cifry zaradíme za bodku a dostávame:

$77.546_{10}$ =

1001101. 1000101111 0001101010 0111111011 1110011101 101100100010 $\ldots_2$

Žltou sú zvýraznené cifry, ktoré už ML nevie akceptovať a musí sa s nimi vysporiadať zaokrúhlením. Prvá "žltá jednotka" nie je poslednou a preto sa bude zaokrúhľovať nahor. ML si nakoniec zapamätá toto:

1001101. 1000101111 0001101010 0111111011 1110011101 101101

Vektor *v* obsahujúci cifry za bodkou sa zmení nasledovne:

```
» v(46)=1; v(47:52)=0;
```

C. ML toto číslo opäť skonvertuje do 10-kovej sústavy a vypíše.
Hodnota čísla v 10-kovej sústave je (získame s pomocou ML):

```
» y=0; for i=1:46, y=y+v(i)*2^(-i); end, 77+y

                ans =    7.754600000000001e+001
```

**<u>Úloha 1:</u>** Zopakujte tento postup pre číslo 77.564 .
**<u>Úloha 2:</u>** Počítajte (0.3/0.1-3) najprv na papieri a potom v MatLabe. Vysvetlite, prečo vychádzajú rôzne výsledky.