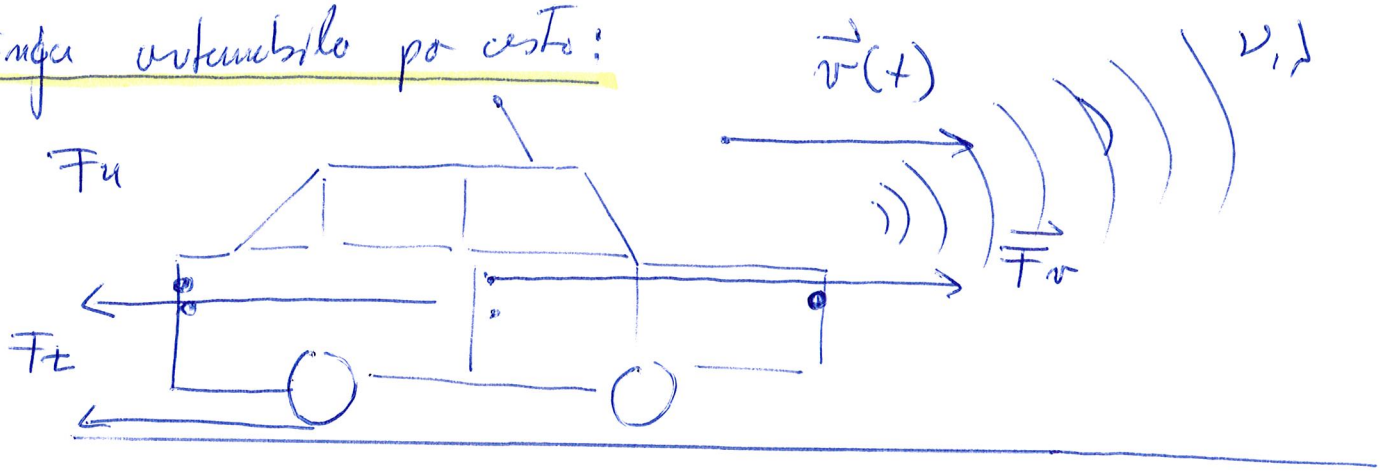
Uvod

14. 2. 2023

MATEMATIČNO FIZIKALNI SEMINAR

Cilj predmeta: Razumevanje in uporaba numeričnih metod v fizikalnih izračunih.

Vožnja avtomobila po cesti:



Šafer potna $v(t)$: Od tod lahko izračunamo prevoženo pot in pospešek z integracijo in odvajanjem hitrosti po času:

$$\vec{a}(t) = \frac{d\vec{v}}{dt} = \dot{\vec{v}}$$ — To navedbo znamo vedno izračunati, ker je pozicijska funkcija odvisnost.

$$\vec{s}(t) = \int_0^t \vec{v}(t')\,dt'$$ — Tak integral pa ne znamo vedno izračunati analitično

Tedaj se poslužimo numerične integracije.

Hitrost izmerjena le v diskretnih točkah: ⇒ Analitičen račun nisa možen, saj ne poznamo prave funkcijske odvisnosti. ⇒ Za izračun pospeška in hitrosti se zatečemo numeričnemu odvajanju in integraciji

Poznamo sile, ki delujejo na avtomobil:

$$x: \quad \sum_i F_i = m \cdot a = F_v - F_u - F_t$$

$$m \cdot a = F_v(t) - c_x \rho_z \cdot \frac{S \cdot v^2}{2} - mg \cdot k_t$$

$$\boxed{m \ddot{x}(t) = F_v(t) - \frac{c_x \rho_z S}{2} \dot{x}(t)^2 - mg\, k_t}$$

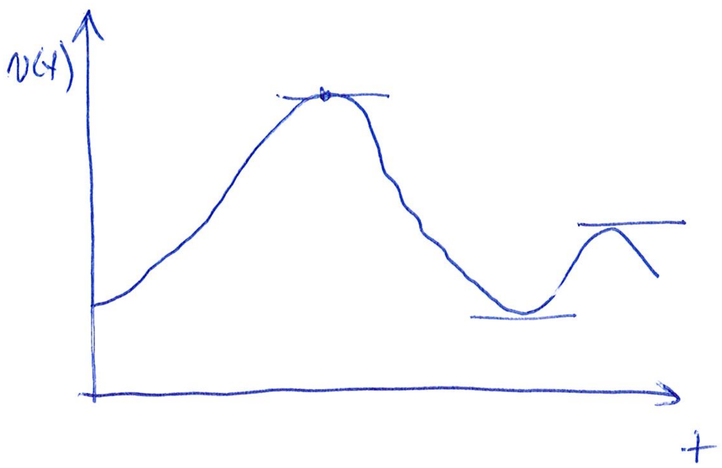
Zapletene diferencialne enačbe ne znamo rešiti analitično ⇒ Numerično reševanje diferencialnih enačb.

↓

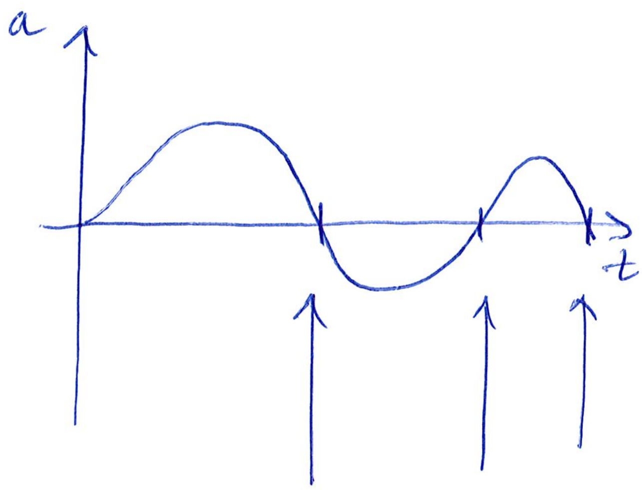
Reševanje parcialnih diferencialnih enačb. (npr. Difuzijska enačba)

Reševanje "običajnih" diferencialnih enačb v več dimenzijah.

Kdaj avto začne zavirati ⇒ Iskanje ničel pospeška



⇒



Iskanje ekstremov funkcij

Numerično iskanje ničel funkcij

Avtomobil med vožnjo oddaja zvok s frekvenco ν_0. Mi zaradi Dopplerjevega efekta slišimo zvok pri drugačnih frekvencah, glede na to ali se nam avto približuje ali oddaljuje.

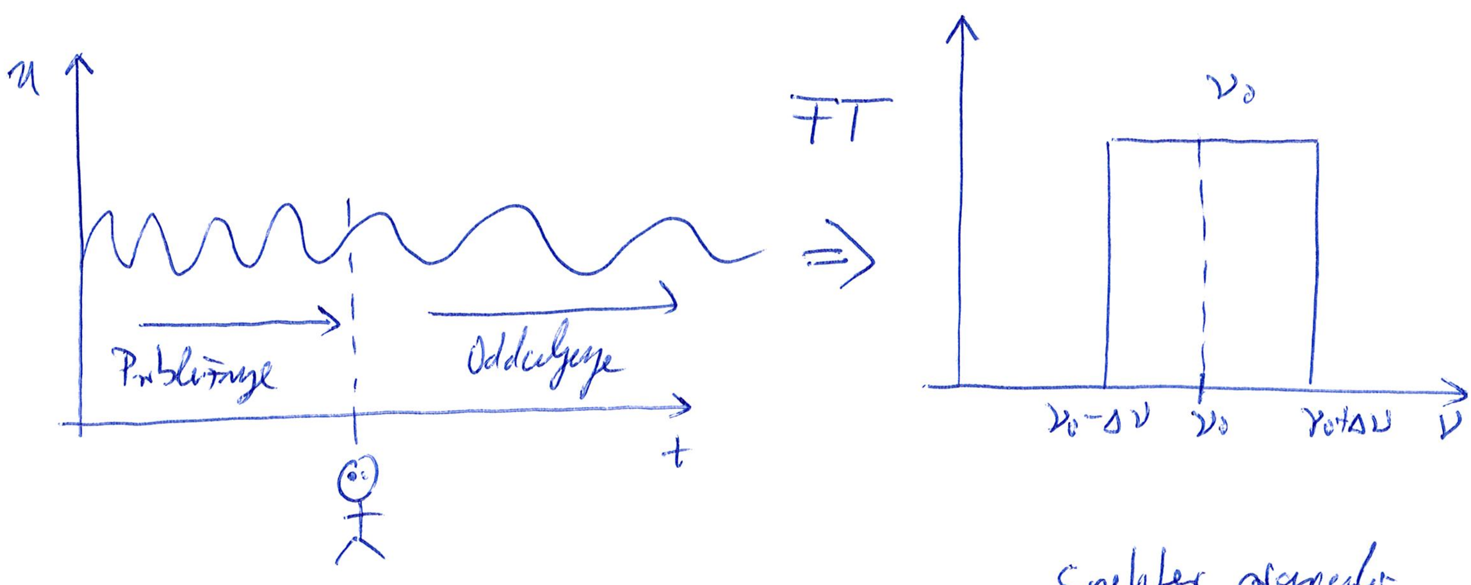
Zvok: $u(t) = u_0 \cdot \sin(2\pi \nu t)$

$$\boxed{\nu = \nu_0 \cdot \left(1 \pm \frac{v}{c}\right)^{-1}}$$

Če se vir premika.

$\pm$ → Približuje ali oddaljuje.

Kako analizirati zvok avtomobila ⇒ Fourierova analiza



Časovna odvisnost signala

Spekter signala

Določimo mu hitrost.

Znova hočemo uporabiti diskretne vzorce (posnetka zvoka!) ⇒ Numerične Fourierove transformacije in hitre Fourierove transformacije

(*) **Dodatni motivacijski zgledi** ⟶

Kdaj uporabljamo numerične metode?

– Ko drugih možnosti ne poznamo:
⟶ kot bomo videli $\int_0^1 e^{x^2}\,dx$

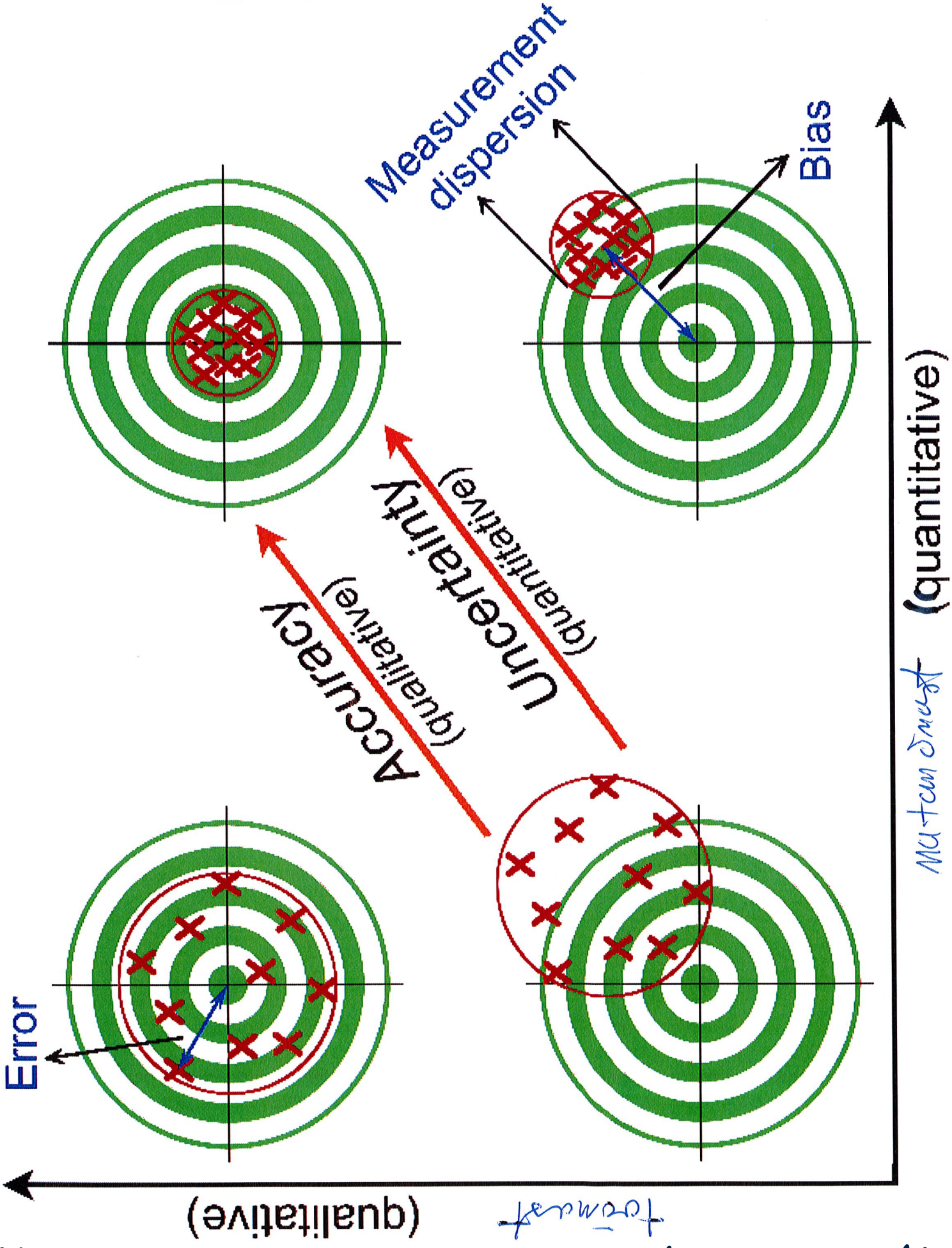
– kadar je to udobnejše oziroma manj zahtevno od iskanja analitičnih rešitev

Od dobrih numeričnih metod pričakujemo:

- EKONOMIČNOST: časovno (število operacij)
  prostorsko (poraba spomina)
- UNIVERZALNOST: da odpove le pri redkih izjemah
- NUMERIČNA STABILNOST: čim manjše numerične napake.

Preveč podrobno se o tem ne bomo ukvarjali. Uporabljali bomo že izdelane algoritme.

Todnest vs. natamdnast



Od rezultatov, ko jih dobimo z numeričnim orodjem (bodisi modelskim napovedi bodisi analitičnih podatkov) pričakujemo, da so karseda točni, natančni pa tudi medsebojno nestrukturni.

ZAPIS ŠTEVIL V BINARNI BAZI

V računalniku so vsa števila predstavljena z zaporedjem bitov 0, 1. Števila predstavimo z več dolgimi 32, 64 bitov (večina se bolj uporablja, lahko pa so tudi dolgi)

"char" = 8 bit
"int" = 32 bit
"long" = 64 bit

Celo število: (Hp "long")

Predznak (sign)

Največje število: $2^{63} - 1$ (zaradi ničle)

Najmanjše število: -2^{63}

Pomni v Matematiki, kaj je to za eno število!

Celo število nima problem, ker pa omejena.

Zapis števil:

$$x = (-1)^s \sum_{i=0}^{63} b_i \cdot 2^i$$

Zgled: Pretvori število 0000000....01010102

6 5 4 3 2 1 0

s

64 bitov

v desetiško sistem:

$$x = (-1)^0 \cdot \left[2^5 + 2^3 + 2\right] = 32 + 8 + 2 = \underline{\underline{42}}$$

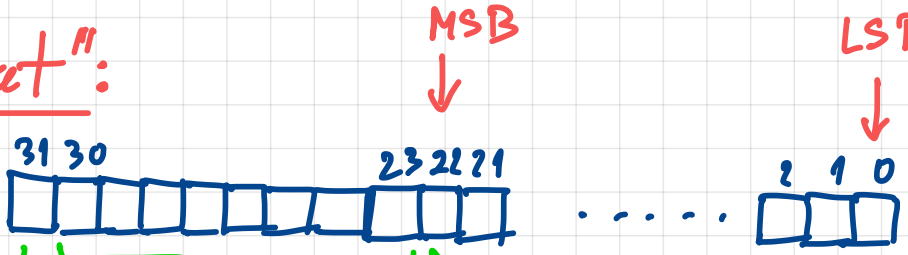
Realna števila

Decimalno vejico imenujemo tudi kopasto število.

Za predstavitev realnih števil uporabljamo aritmetiko v plavajoči decimalni vejici. V računalniku jih predstavimo z natančnostjo 32 oziroma 64 bitov. Pri tem uporabljamo standard IEEE 754.

- Števila v plavajoči vejici v enojni natančnosti (v C/C++ "float")
- Števila v plavajoči vejici v dvojni natančnosti (v C/C++ "double")

"Float":



Števila so predstavljena na naslednji način:

$0 < E < 255$: "normalizirane vrednosti"

$$fp(x) = (-1)^S \cdot 2^{E-127} \cdot \left[1 + \sum_{i=1}^{23} b_{23-i}\, 2^{-i}\right]$$

$$= (-1)^S \cdot 2^{E-127} \cdot \left(1 + \frac{b_{22}}{2^1} + \frac{b_{21}}{2^2} + \dots \frac{b_0}{2^{23}}\right)$$

$E = 0$: "denormalizirane vrednosti"

$$fp(x) = (-1)^s \cdot 2^{-126} \cdot \left[0 + \sum_{i=1}^{23} b_{23-i} \, 2^{-i} \right]$$

$$= (-1)^s \cdot 2^{-126} \cdot \left(\frac{b_{22}}{2^1} + \frac{b_{21}}{2^2} + \dots \frac{b_0}{2^{23}} \right)$$

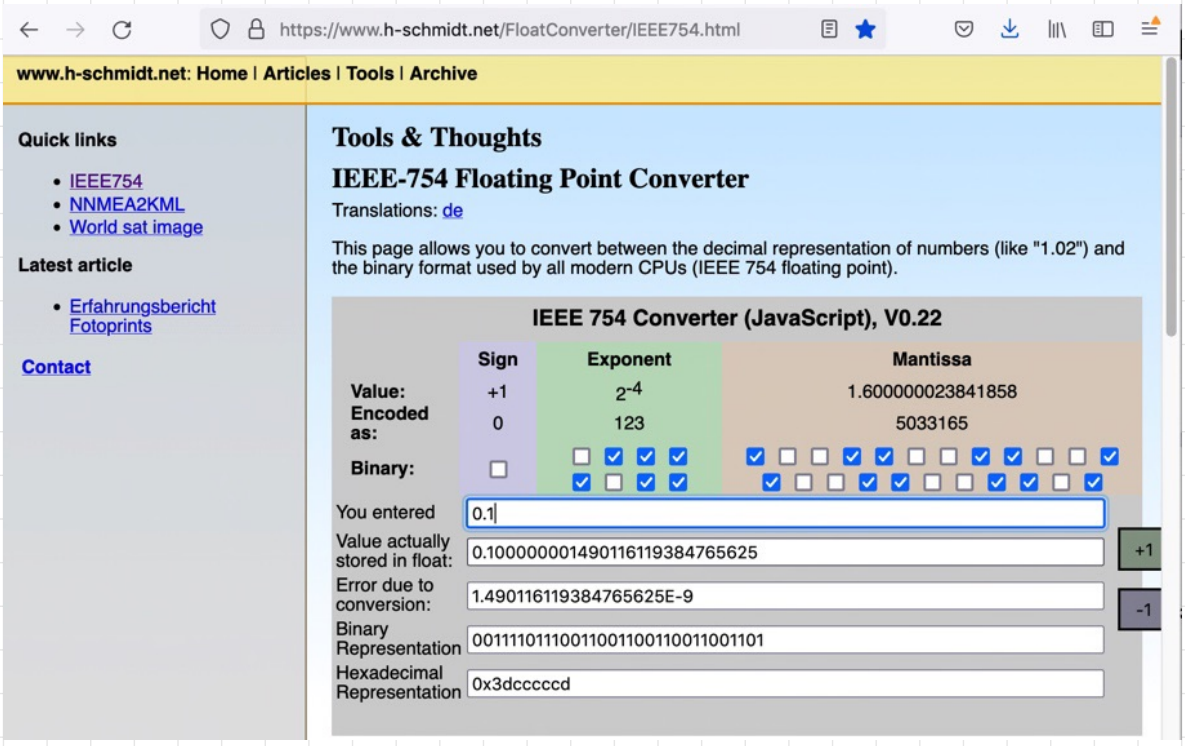
Zgledi: (Marko Horvat)

```
0 10000000 00000000000000000000000 = +1 * 2**(128-127) * 1.0 = 2
0 10000001 10100000000000000000000 = +1 * 2**(129-127) * 1.101 = 6.5
1 10000001 10100000000000000000000 = -1 * 2**(129-127) * 1.101 = -6.5

0 00000001 00000000000000000000000 = +1 * 2**(1-127) * 1.0 = 2**(-126)
0 00000000 10000000000000000000000 = +1 * 2**(-126) * 0.1 = 2**(-127)
0 00000000 00000000000000000000001 = +1 * 2**(-126) *
                                     0.00000000000000000000001 =
                                     2**(-149)  (Smallest positive value)
```

Najmanjše število predstavljivo v enojni natančnosti.

Web calculator



Posebna stanja:

$E = 255$ in $F \neq 0 \Rightarrow fl(x) = NaN$
("Not a number")

$E = 255$ in $F = 0$ in $S = 1 \Rightarrow fl(x) = -\infty$

$E = 255$ in $F = 0$ in $S = 0 \Rightarrow fl(x) = +\infty$

Težava nedenormaliziranih števil je ta, da se bližamo operaciji odštevanja enakih števil.

Natančnost aritmetike interpretiramo kot največje število, za katero velja:

$$\boxed{fl(1+\varepsilon) = 1}$$

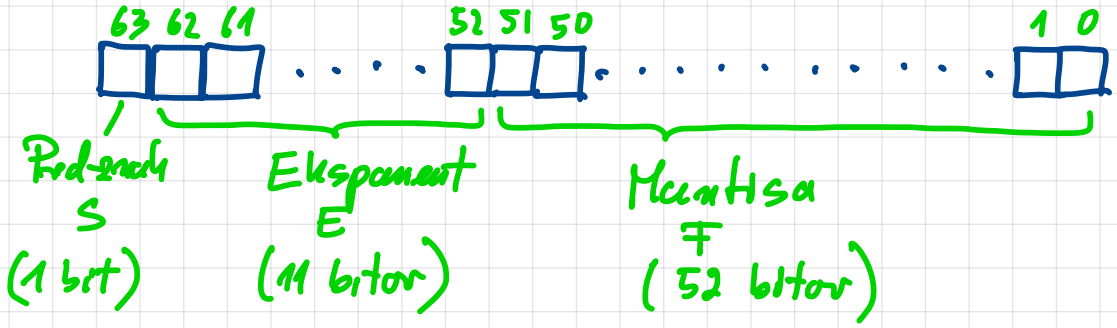
ε: Ta se nikjer ne pojavi, kar je manjše od definirane mantise.

Odvisno je od zaokroževanja. Po IEEE standardu zaokrožujemo k najbližjemu predstavljenemu rezultatu:

$\varepsilon \approx \frac{\varepsilon_M}{2}$ — Strojna natančnost.

$\varepsilon_M = 2^{-23} \approx 1.2 \cdot 10^{-7}$

"Double":



Števila so predstavljena na naslednji način:

$0 < E < 2047$: "normalizirane vrednosti"

$$f_p(x) = (-1)^S \cdot 2^{E-1023} \cdot \left[\underline{1} + \sum_{i=1}^{52} b_{52-i} \, 2^{-i} \right]$$

$$= (-1)^S \cdot 2^{E-1023} \cdot \left(1 + \frac{b_{51}}{2^1} + \frac{b_{50}}{2^2} + \dots \frac{b_0}{2^{52}} \right)$$

$$= (-1)^S \, 2^{E-1023} \, \underline{1.} \underbrace{b_{51} b_{50} b_{49} \dots b_0}_{F}$$

binarna beseda

$E = 0$: "denormalizirane vrednosti"

$$f_p(x) = (-1)^S \cdot 2^{-1022} \cdot \left[\underline{0} + \sum_{i=1}^{52} b_{52-i} \, 2^{-i} \right]$$

$$= (-1)^S \cdot 2^{-1022} \cdot \underline{0.} \underbrace{b_{51} b_{50} b_{49} \dots b_0}_{F}$$

binarna beseda

Posebna stanja:

$E = 2047$ in $F \neq 0 \Rightarrow fl(x) = NaN$
("Not a number")

$E = 2047$ in $F = 0$ in $S = 1 \Rightarrow fl(x) = -\infty$

$E = 2047$ in $F = 0$ in $S = 0 \Rightarrow fl(x) = +\infty$

Težava računanja s plavajočo vejico je ta, da se bližamo operacije obravnavane z napako.

Natančnost aritmetike interpretiramo kot največje število, za katero velja:

$$\boxed{fl(1+\varepsilon) = 1}$$

Ta se mnogo ne razlikuje od delitvene natančnosti.

Odvisno je od zaokroževanja. Po IEEE standardu zaokrožujemo k najbližjemu predstavljenemu rezultatu:

$$\varepsilon \approx \frac{\varepsilon_M}{2}$$

Strojna natančnost.

$$\varepsilon_M = 2^{-52} \approx 2.22 \cdot 10^{-16}$$

Največji eksponent: $E_{max} = 1023$

Najmanjši eksponent: $E_{min} = -1022$

Najmanjše predstavljeno število: $4.94 \cdot 10^{-324}$

Največje normalno število: $1.80 \cdot 10^{308}$

Za več informacij glej: Štrca, RMF, str 2, 595

Takeaway message:

- Z realnimi števili v enojni natančnosti, ie 32 bit ("float") lahko računamo na 7 mest natančno!
- Z realnimi števili v dvojni natančnosti, ie 64 bit ("double") lahko računamo na 16 mest natančno
- Pazi, ko odštevaš, primerjaš števila! Upoštevaj ε natančnost!

Zgled PlavajocaVejica.py

```python
#!/usr/bin/env python3

import numpy as np


x=np.finfo(np.float64);
print("Lastnosti stevil tipa: ", x.dtype)
print("- Najmanjse stevilo je", x.min)
print("- Najvecje stevilo je", x.max,'\n')
print("- Najmanjsi eksponent v dvojiski bazi je ", x.minexp)
print("- Najvecji eksponent v dvojiski bazi je ", x.maxexp,'\n')
z=1
print("- Najblizji vecje stvilo stevila",z," je: ",'%.20f'%np.nextafter(z,z+1))
print("- Najblizji manjse stvilo stevila",z," je: ",'%.20f\n'%np.nextafter(z,z-1))



# The smallest number in double: 1/2^53
dx=1./pow(2,52);
print("Najmanjša predstavljiva razlika dx je ", dx);
print('1+dx  = ', '%.20f\n'%np.double(1+dx));

dx=1./pow(2,53);
print("Ze premajhna razlika dx je ", dx);
print('1+dx  = ', '%.20f\n'%np.double(1+dx));

print('Zadnja decimalka v stevilu 2.0000000000000001 ', '%.20f\n'%np.double(1+dx));


def shownum(num):
    y=np.double(num)
    print('Stevilo', num,'se izpise kot:')
    print('%.20f\n'%y)
    return 0

# sprememba je premajhna, da bi jo lahko shranili v double
shownum('2.0000000000000001')

# majhne vrednosti (na robu) pa se shranijo narobe.
shownum('2.000000000000001')

# Primer nauka, kadar imamo dve skoraj enaki veliki stevili moramo paziti, kako ju
# odstevamo in gledamo razliko.

```

```
In [187]: runfile('/Users/miham/Desktop/Sola/MFSEM-2022/Uvod/PlavajocaVejica/
PlavajocaVejica.py', wdir='/Users/miham/Desktop/Sola/MFSEM-2022/Uvod/PlavajocaVejica')
Lastnosti stevil tipa:  float64
- Najmanjse stevilo je -1.7976931348623157e+308
- Najvecje stevilo je 1.7976931348623157e+308

- Najmanjsi eksponent v dvojiski bazi je  -1022
- Najvecji eksponent v dvojiski bazi je  1024

- Najblizji vecje stvilo stevila 1  je:  1.0000000000000022204
- Najblizji manjse stvilo stevila 1  je:  0.99999999999999988898

Najmanjša predstavljiva razlika dx je  2.220446049250313e-16
1+dx  =  1.0000000000000022204

Ze premajhna razlika dx je  1.1102230246251565e-16
1+dx  =  1.0000000000000000000

Zadnja decimalka v stevilu 2.00000000000000001  1.0000000000000000000

Stevilo 2.00000000000000001 se izpise kot:
2.0000000000000000000

Stevilo 2.000000000000001 se izpise kot:
2.0000000000000088818
```