nedeljainformatike v2.0

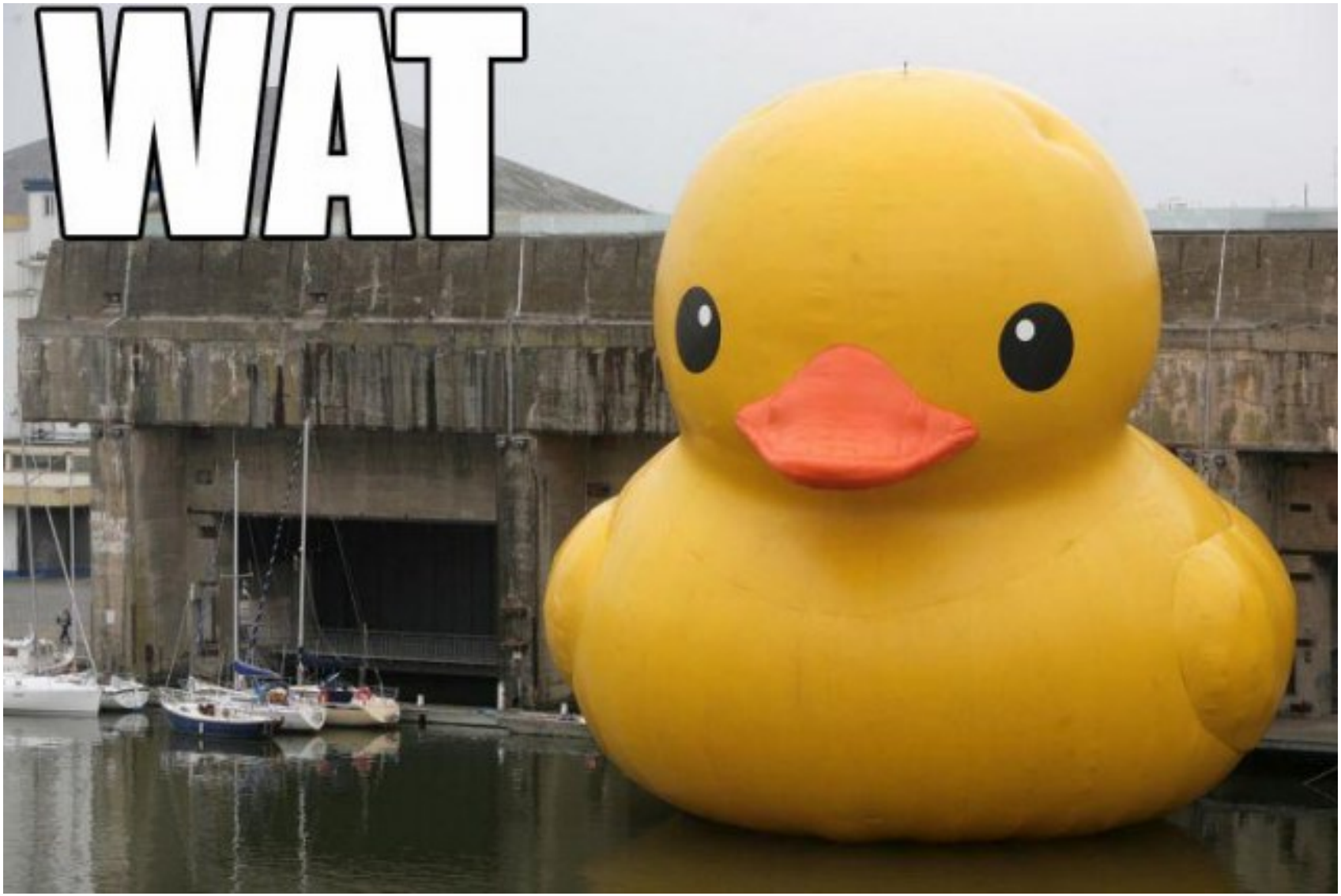
MATLAB kroz primere

Uvod + Vidim plave krugove

Lazar Mitrović

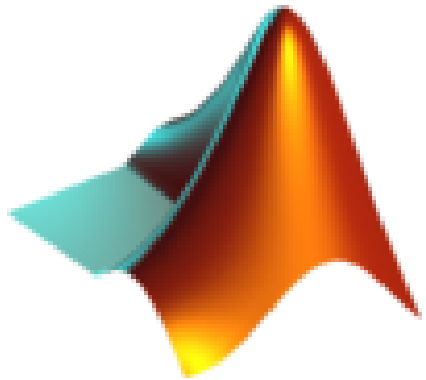
Matematička gimnazija
Nedelja informatike v2.0

15. decembar 2015.


WAT

WAT?

- *MATLAB* (**mat**rix **lab**oratory - ne Mathematics laboratory) je programski paket namenjen numeričkim izračunavanjima (ili mu je bar to nekada bila jedina namena)
- Vuče korene iz 1970-tih, a firma koja ga razvija (MathWorks) to radi još od 1984.
- Verzija u vreme pisanja ove prezentacije je R2015b



WAT?

- Razvijen kako bi studenti imali pristup bibliotekama za numerička izračunavanja i linearnu algebru bez potrebe da uče Fortran
- Brzo našao primenu u nauci i tehnici i počeo ekspanziju oblasti od interesa
- Trenutno paket zauzima ∼8GB, radi na svim većim operativnim sistemima (JVM) i sadrži više od 40 toolbox-eva koji proširuju funkcionalnosti osnovnog programa:

  `Aerospace, Financial, Image Processing, Optimization, Signal Processing, Statistics and Machine Learning, Curve Fitting, Bioinformatics, Filter Design HDL Coder, Simulink, Parallel Computing, Neural Network, Image Acquisition..`

Alternative

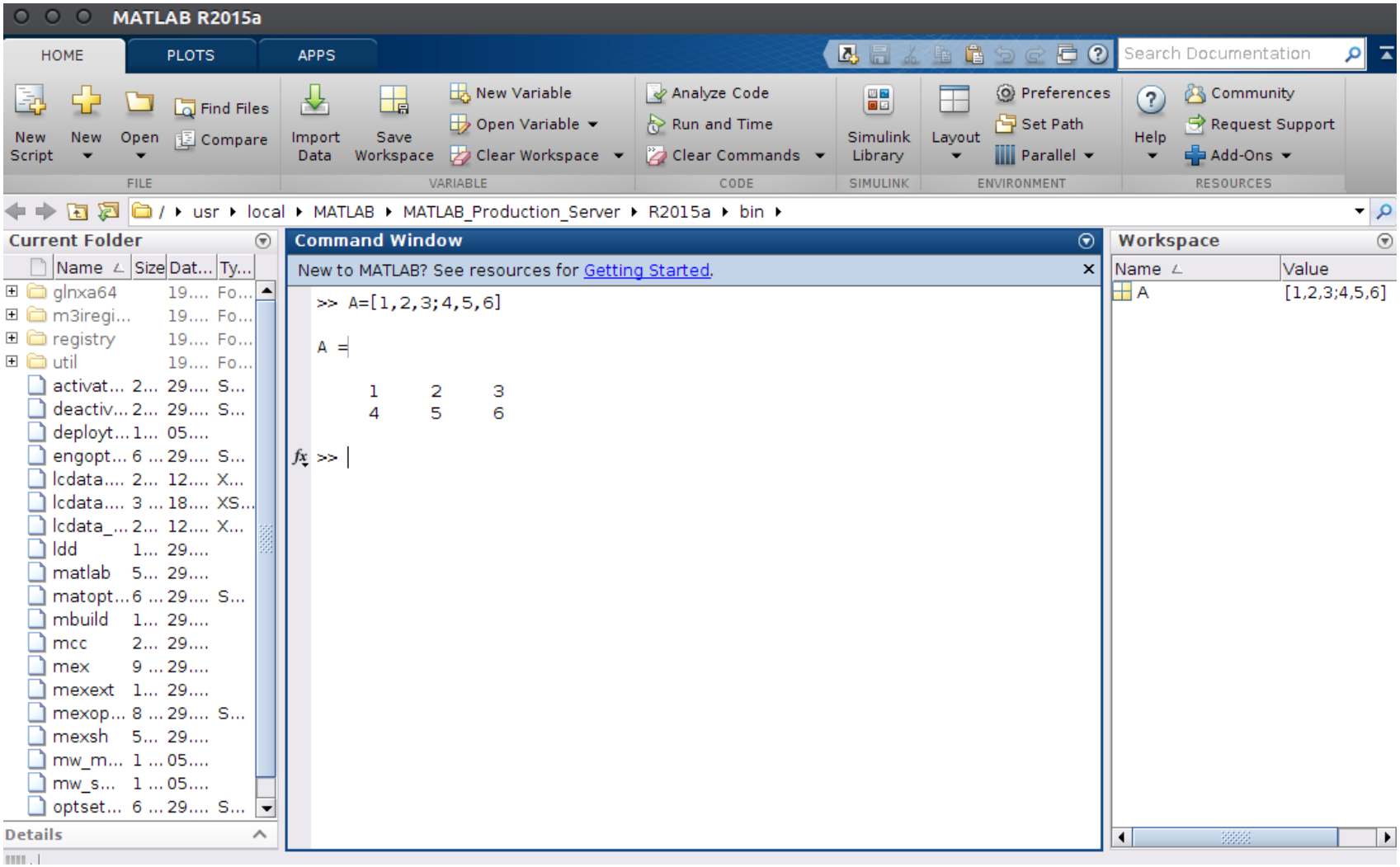
- $135 za kućnu - $2650 za standardnu licencu (osim za Kembridžovce)
- Open source alternative: *GNU Octave* i *SciLab* (GPL i GPL compatible)

- Nisu 100% kompatibilne sa Matlab-om ali možete da kontribjutujete
  http://www.gnu.org/software/octave/get-involved.html
  https://www.scilab.org/development/contribute

Okruženje

- Sva podešavanja su dostupna iz konzole (GUI je dodatak*)
- Postoji path varijabla (kao i u Win32 i *nix sistemima)
- Workspace sadrži vrednosti svih varijabli i može se izlistati (i grafički i konzolno preko `whos`), a i sačuvati u *.mat fajl
- Ispisivanje vrednosti varijabli može se uraditi dvoklikom na varijablu u workspace-u ili kucanjem imena varijable u konzoli
- Postoji moćan editor i još moćniji debugger koji se pozivaju preko `edit imefajla.m`
- Toolbox-evima se pristupa preko stavke `APPS` na ribonu

** Offtopic: Postoji čak i GUI Builder za vaše projekte*


MATLAB R2015a
HOME
PLOTS
APPS
Search Documentation
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
Layout
Preferences
Set Path
Parallel
Help
Community
Request Support
Add-Ons
FILE
VARIABLE
CODE
SIMULINK
ENVIRONMENT
RESOURCES
/ ▸ usr ▸ local ▸ MATLAB ▸ MATLAB_Production_Server ▸ R2015a ▸ bin ▸
Current Folder
Name
Size
Dat...
Ty...
glnxa64
m3iregi...
registry
util
activat...
deactiv...
deploylt...
engopt...
lcdata....
lcdata....
lcdata_...
ldd
matlab
matopt...
mbuild
mcc
mex
mexext
mexop...
mexsh
mw_m...
mw_s...
optset...
Details
Command Window
New to MATLAB? See resources for Getting Started.
>> A=[1,2,3;4,5,6]
A =
1 2 3
4 5 6
>>
Workspace
Name
Value
A
[1,2,3;4,5,6]

Jezik

- Skript jezik (`Python`, `JavaScript`, `Bash`, `Batch`, `GML`..)
- C-olik po sintaksi ali *weakly typed* - implicitno konvertuje tipove
- Proceduralan i iterativan - ali su petlje vrlo često loša praksa
- Interpreterski $\rightarrow$ interpreterska konzola
- Ekstenzija `*.m`
- Interpreter paralelizuje većinu matričnih operacija $\rightarrow$ što više izbegavati petlje

Jezik

- ; na kraju reda nije obavezna, ali zaustavlja output komande u konzolu
- % označava komentar (samo jednolinijski)
- Konzola se briše pomoću `clc`, workspace se briše preko `clear all`, a otvoreni prozori gase pomoću `close all`
- NE MEŠATI `clc` i `clear` (obrisaće vam sve varijable iz projekta)
- (Odličnu) referencu za funkcije koje vas zanimaju možete naći preko `help imefunkcije`. Samo `help` lista sve funkcije po kategorijama.
- Kôd većine default funkcija može da se ispiše (`type imefunkcije`) i forkuje (`edit imefunkcije`)

Jezik

- Sve (skoro) je matrica (olakšava linearnu algebru)

```
» A = [1,2,3; 4,5,6]
A =
     1 2 3
     4 5 6
```

- Indeksiranje počinje od 1

```
» A(2,2)
ans = 5
```

- Case-Sensitive (varijabla a nije isto što i varijabla A)
- Operacije sa matricama trivijalne (sabiranje + oduzimanje - množenje * transponovanje .' množenje skalarom .* deljenje skalarom ./ stepenovanje ^ stepenovanje element-wise .^)

Jezik

- Jednodimenziona matrica je vektor

```
» A = [1,2,3]
A =
     1  2  3
```

- Nadovezivanjem vektora možemo dobiti matricu

```
» B = [A;A.*2]
B=
     1  2  3
     2  4  6
```

- Predefinisane matrice su `zeros([brojredova,brojkolona])` i `ones([brojredova,brojkolona])`

Jezik

- Vektori se mogu zadavati i preko colon notacije (start:korak:stop), jednolinijski

```
» A = [0:0.1:1]
A =
 Columns 1 through 7
  0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000
 Columns 8 through 11
  0.7000 0.8000 0.9000 1.0000
```

- Većina funkcija kao argument prihvata vektor

```
» B = sin(A)
B =
 Columns 1 through 7
  0 0.0998 0.1987 0.2955 0.3894 0.4794 0.5646
 Columns 8 through 11
  0.6442 0.7174 0.7833 0.8415
```

Jezik

- Izdvajanje podmatrice može se vrlo jednostavno vršiti Colon notacijom:

```
» C=B(1:5)
C =
    0 0.0998 0.1987 0.2955 0.3894
```

- Moguće je i klasično iterativno rešenje (ali je sporije - 69us u odnosu na 7us)

```
» D=[]; for (i=1:5) D=[D,B(i)]; end; D
D =
    0 0.0998 0.1987 0.2955 0.3894
```

Jezik

- Kompleksni brojevi se mogu predstaviti kao `a+bi` ili `a+bj`
- 2*i nije isto što i 2i (u prvom slučaju i je varijabla dok je u drugom imaginarna jedinica)‼1! Isto važi za j
- Sve operacije koje su definisane na prethodnim slajdovima su validne i za kompleksne brojeve

Jezik

- If-else grananja:

```
if r == c
    A(r,c) = 2;
elseif abs(r-c) == 1
    A(r,c) = -1;
else
    A(r,c) = 0;
end
```

- Funkcije:

```
function izlaz = brojjedan(argument)
p=1;
return % nije obavezno, vraća kontrolu roditelju
end;
```

Čudovišne matrice

- Zadatak: *Napisati program koji vraća vrednost date matrice nakon transformacije koja je transponuje a zatim svaki novodobijeni element zameni sa vrednošću sinusa kvadrata tog elementa.*

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 7 & 5 & 3 & 1 & -1 & -3 \\ 4 & 8 & 16 & 32 & 64 & 128 & 256 \end{pmatrix}$$

Čudovišne matrice

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 9 & 7 & 5 & 3 & 1 & -1 & -3 \\ 4 & 8 & 16 & 32 & 64 & 128 & 256 \end{pmatrix}$$

```
A=[1:7; 9:-2:-3; 2.^(2:8)]
A.'
[A.'].^2
sin([A.'].^2)
sin([[[1:7; 9:-2:-3; 2.^(2:8)].'].^2])
```

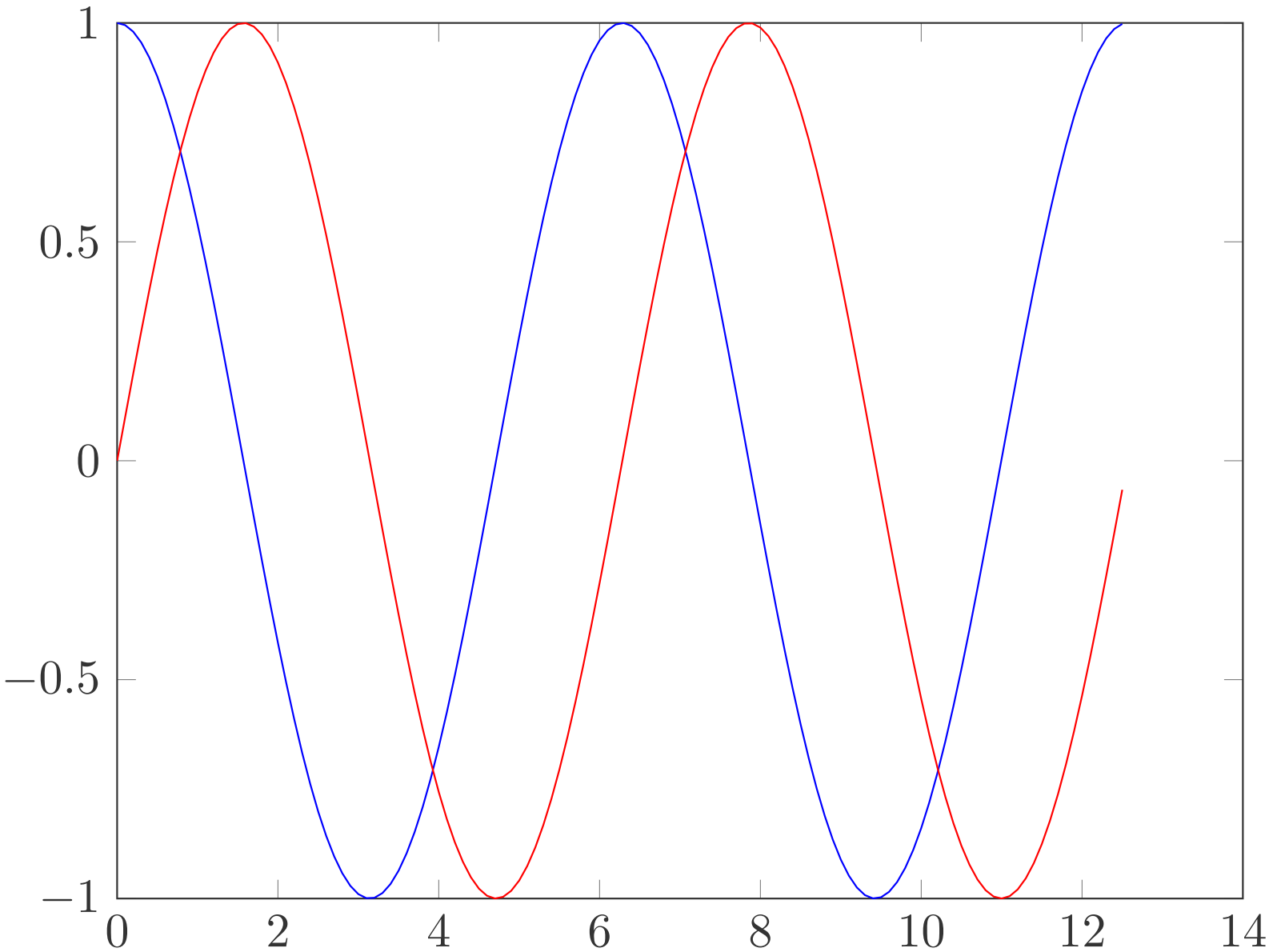
Uradi mi domaći iz analize (nemoj)

- Zadatak: *Nacrtati grafik sinusne (crvenom) i kosinusne funkcije (plavom) od 0 do 4π*

Uradi mi domaći iz analize (nemoj)

- Zadatak: *Nacrtati grafik sinusne (crvenom) i kosinusne funkcije (plavom) od 0 do 4π*

```
t=[0:0.1:4*pi];
plava=cos(t);
crvena=sin(t);
plot(t,plava,'b')
hold on
plot (t,crvena,'r')
```


1
0.5
0
−0.5
−1
0
2
4
6
8
10
12
14

Predriblaj Eratostena

- Zadatak: *Napisati funkciju koja proverava da li je broj prost*

Predriblaj Eratostena

- Zadatak: *Napisati funkciju koja proverava da li je broj prost*

```
function p = prost(n)
t=floor(sqrt(n));
for m = 2:t
    if mod(n,m) == 0
        p=0;
        return
    end
end;
p=1;
end
```

- Funkcija isprime() je u opštem slučaju do 7x brža (Matlab čuva u memoriji mapu sa često korišćenim prostim brojevima, a one koje ne zna računa paralelizovanjem)

Vidim plave krugove

- Zadatak: *Upasti na Nedelju informatike™*

Hvala na pažnji

Pitanja?

nedeljainformatike v2.0

MATLAB kroz primere

IQ sampling i FM demodulacija (kako napraviti radio prijemnik)

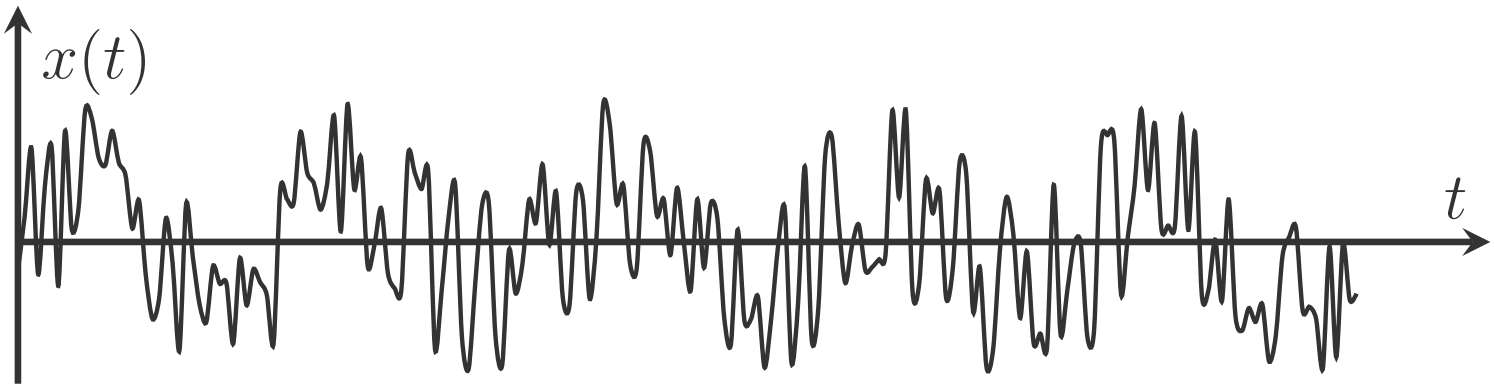
Petar Veličković

Matematička gimnazija
Nedelja informatike v2.0

15. decembar 2015.

Naš današnji zadatak

- Želja nam je da dizajniramo jednostavan softverski **radio prijemnik**; program koji učitava snimljene podatke sa radio spektra, i, znajući određene dodatne informacije (npr. sa koje frekvencije odašilje izabrana radio stanica), rekonstruiše originalne informacije koje je ta stanica poslala.



- MATLAB je, kao što ćemo uskoro videti, izuzetno podesan za ovakav zadatak.

Matematički uvod

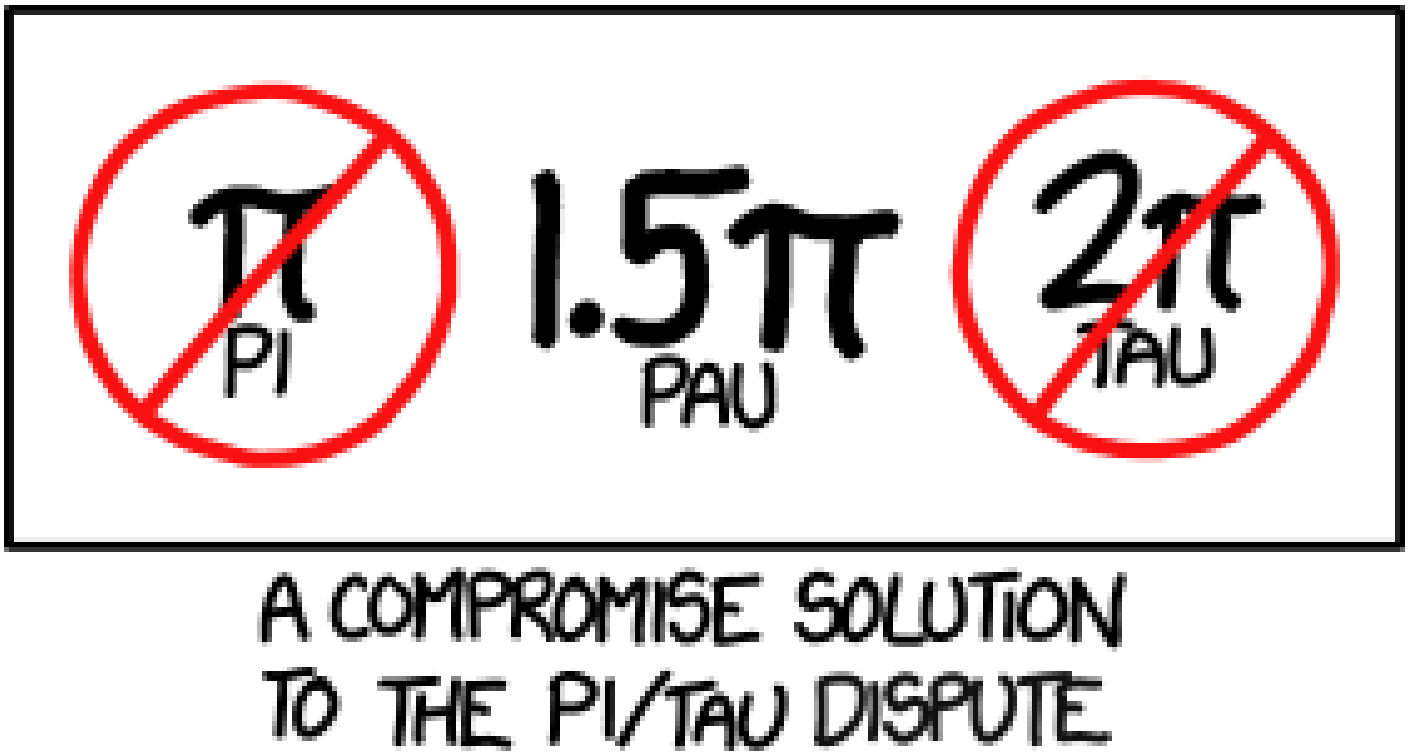
- Da bi mogli da celokupno prođemo kroz ovu demonstraciju, neophodno je uvesti nekolicinu matematičkih pojmova koje možda niste videli ili koristili ranije.

- Do kraja školovanja u MG imaćete priliku da se detaljno upoznate sa svim ovim konstrukcijama, tako da nećemo prelaziti potpune definicije—trudiću se da vam sada dam što je minimalniji mogući uvod bez gubljenja jasnoće.

Stepeni i radijani

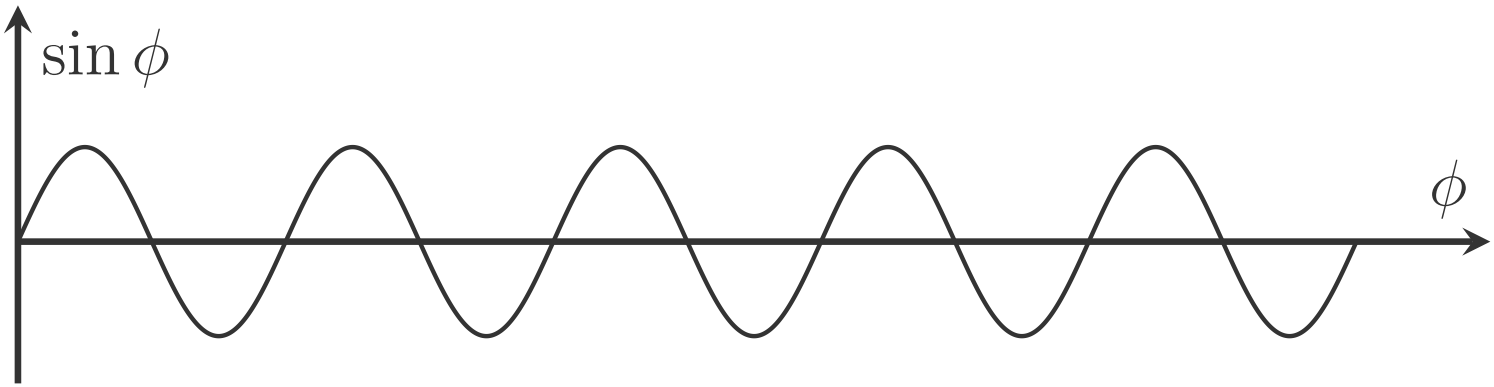
- Do sada se određen deo vas verovatno susretao sa uglovima iskazanim isključivo u vidu jedinice *stepena* (pun krug je $360°$). Karakteristični uglovi: $0°, 30°, 45°, 60°, 90°, \ldots$
- Međutim, daleko korišćenija jedinica za uglove u matematici su *radijani*—dužine kružnog luka nad datim uglom, ukoliko je poluprečnik kruga 1 (pun krug je 2π). Karakteristični uglovi: $0, \frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{2}, \ldots$
- (Za tauiste, karakteristični uglovi su: $0, \frac{\tau}{12}, \frac{\tau}{8}, \frac{\tau}{6}, \frac{\tau}{4}, \ldots$)

Kratko skretanje: π vs. τ (`http://xkcd.com/1292/`)



Sinusna funkcija

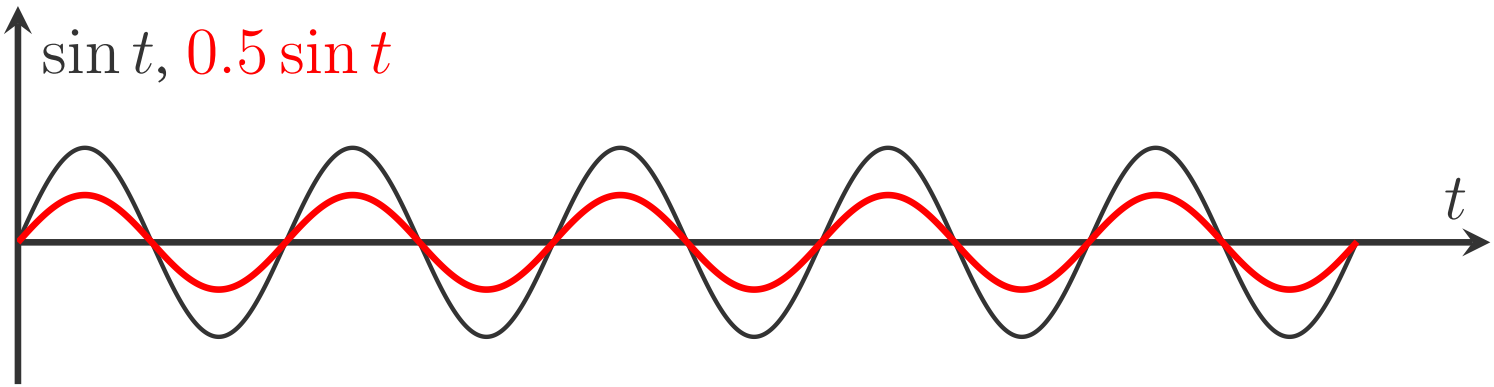
- **Sinusna funkcija**, $\sin\phi$, predstavlja vrednost ordinate tačke na krugu poluprečnika 1 sa centrom u koordinatnom početku, kada je ugao između x-ose i linije koja spaja centar sa tačkom jednak ϕ.



- Ova funkcija je uvek u intervalu $[-1, 1]$, periodična sa periodom 2π, i važi $\sin 0 = 0$.

Generalizovane sinusne funkcije

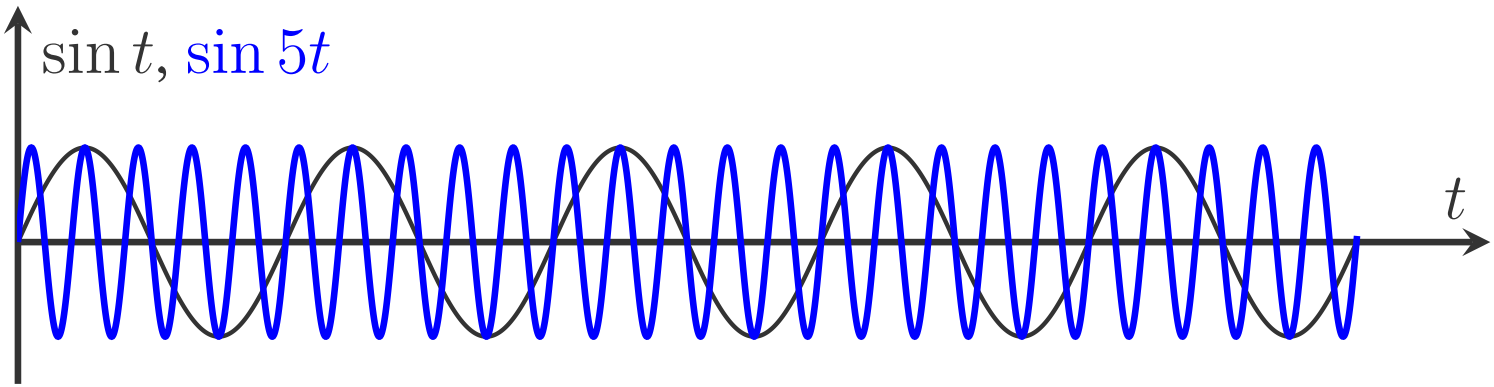
- Ovaj osnovni oblik sinusne funkcije možemo menjati na nekoliko načina; generalni oblik je $A\sin(\omega t + \varphi)$.



- Parametar A (**amplituda**) kontroliše interval vrednosti funkcije (postaje $[-A, A]$).

Generalizovane sinusne funkcije

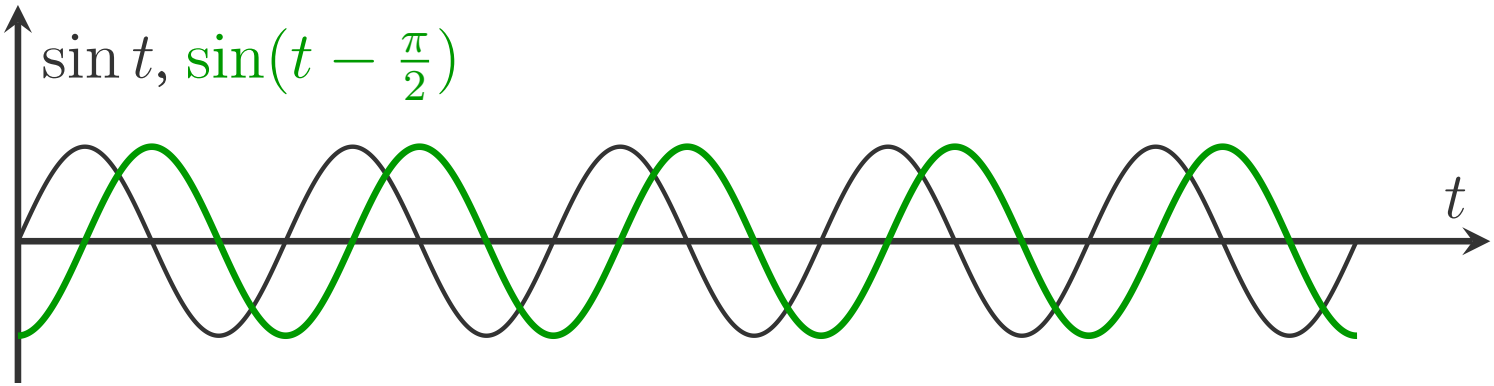
- Ovaj osnovni oblik sinusne funkcije možemo menjati na nekoliko načina; generalni oblik je $A\sin(\omega t + \varphi)$.



- Parametar ω (**ugaona frekvencija**) kontroliše period funkcije (postaje $\frac{2\pi}{\omega}$).
- Često se koristi smena $\omega = 2\pi f$; tada period možemo izraziti kao $\frac{1}{f}$, i f se naziva *frekvencijom*.

Generalizovane sinusne funkcije

- Ovaj osnovni oblik sinusne funkcije možemo menjati na nekoliko načina; generalni oblik je $A\sin(\omega t + \varphi)$.



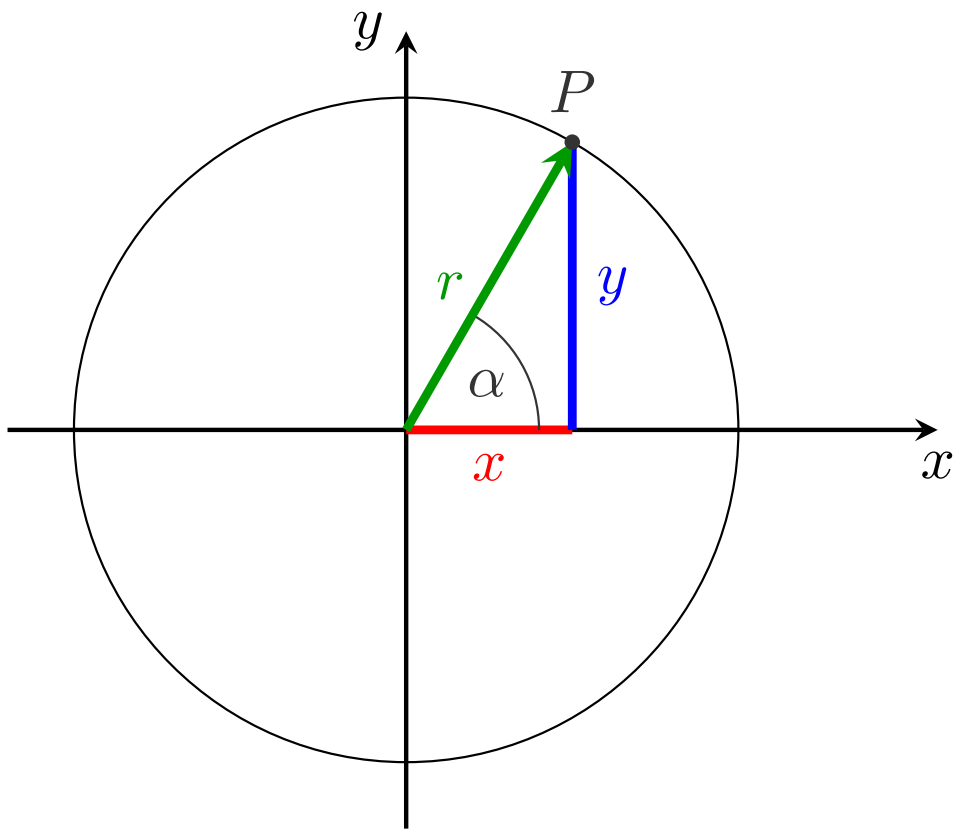
- Parametar φ (**faza**) kontroliše početnu vrednost funkcije (u tački $t = 0$).
- Specijalno, funkcija koja kasni $\frac{\pi}{2}$ $(90°)$ za sinusnom funkcijom se naziva *kosinusom*: $A\cos(\omega t) = A\sin(\omega t + \frac{\pi}{2})$.

Koordinatni sistemi (u jednom slajdu!)

- U **pravouglom** (Dekartovom) koordinatnom sistemu tačku P predstavljamo pomoću x i y.
- U **polarnom** koordinatnom sistemu tačku P predstavljamo pomoću r i α.
- Prelaz iz jednog sistema u drugi:

$$x = r \cos \alpha$$
$$y = r \sin \alpha$$

$$r = \sqrt{x^2 + y^2}$$
$$\tan \alpha = \frac{y}{x}$$



Izvodi i integrali

- U sklopu ove demonstracije, u par navrata biće spominjani *izvodi* i *integrali*. Here Be Dragons?
- Na sreću, jedino što treba da zapamtite su sledeće tri stvari:
  - Izvod, $f'(x)$, odgovara "brzini promene" funkcije $f(x)$ u tački x. Možemo ga približno izračunati kao

$$f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

    za neko malo Δx;
  - Određeni integral, $\int_a^b f(x)\, dx$, odgovara površini ispod grafika funkcije $f(x)$ na intervalu $x \in (a, b)$;
  - Izvod i integral su *inverzni operatori*—vrlo grubo govoreći, jedan "poništava" drugog i obratno.

Radio odašiljači

- Na radio stanici, mikrofon pretvara zvuk (audio talase) u električni signal.

- Ove signale ne možemo direktno slati; antene na prijemnicima bi morale da budu **ogromne** da uhvate ove (relativno niske) frekvencije. Takođe, moramo paziti na to da nema interferencije između različitih radio stanica koje odašilju u isto vreme.

Modulacija

- Problem se rešava tako što, umesto originalnog talasa, šaljemo talas na nekoj mnogo višoj frekvenciji (do ~ 100 MHz za FM). Svaka radio stanica dobija neku frekvenciju na kojoj sme da odašilje (kao i određen prostor oko te frekvencije).

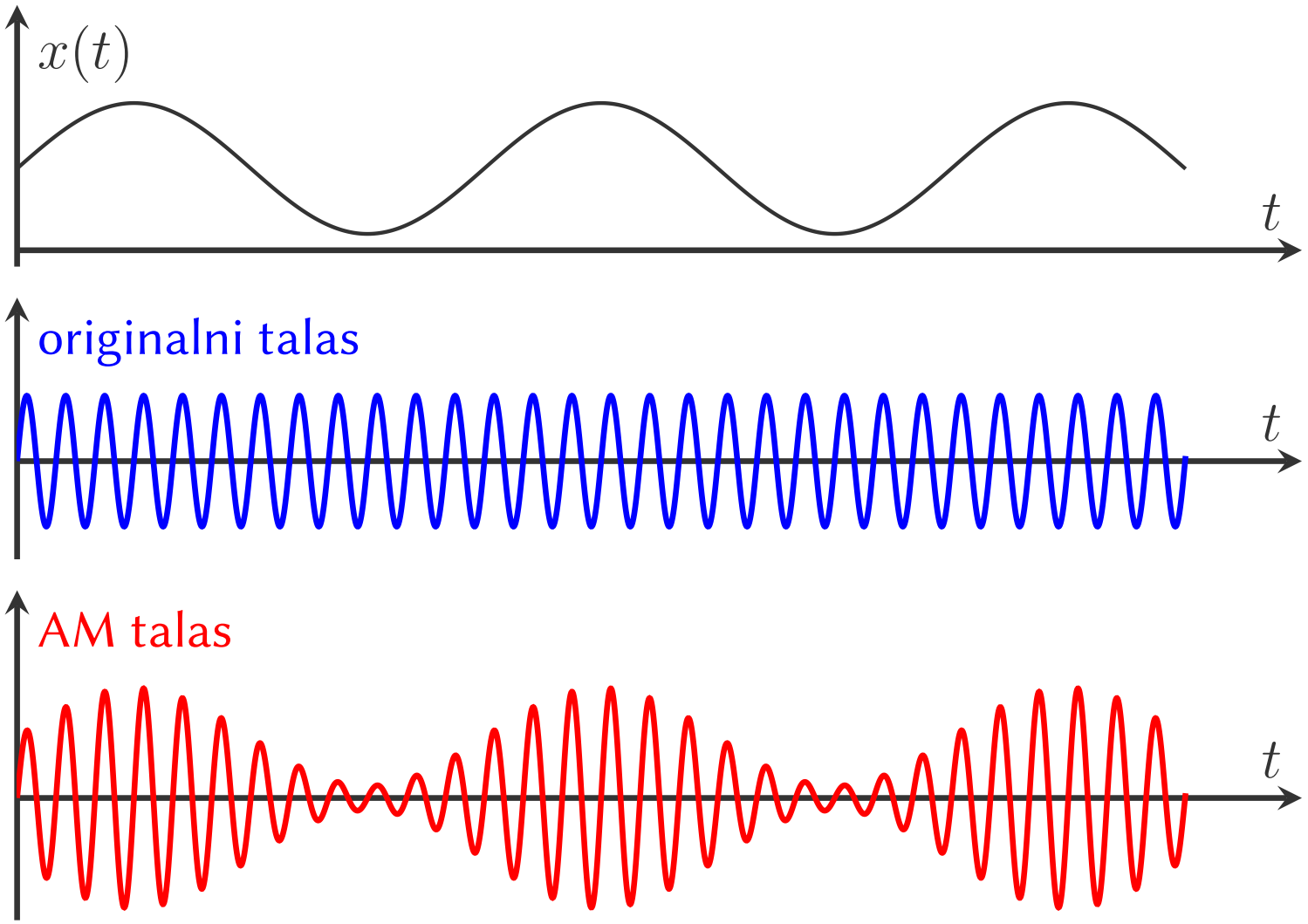
- Ovaj talas ćemo izmeniti (**modulirati**) na neki način koji sadrži informacije o originalnom zvučnom talasu.
- Dva najčešća načina modulacije su **amplitudna modulacija** (AM) i **frekvencijska modulacija** (FM).

Amplitudna modulacija (AM)

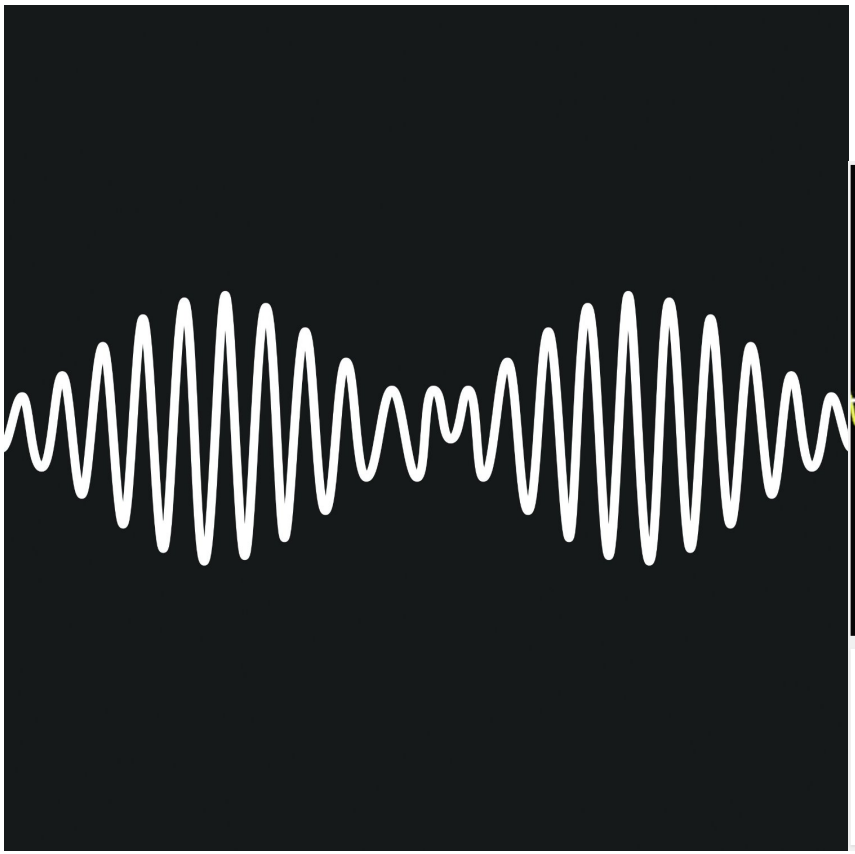
- Pretpostavimo da je radio talas originalno oblika $\sin(2\pi f t)$, gde je f frekvencija na kojoj stanica odašilje.

- Ukoliko želimo da pošaljemo signal $x(t)$ (za koji ćemo pretpostaviti da stalno važi $x(t) \geq 0$) amplitudnom modulacijom, poslaćemo talas $x(t)\sin(2\pi f t)$ (jačina signala menja amplitudu talasa).

Amplitudna modulacija (AM)



Arktički majmuni (AM)



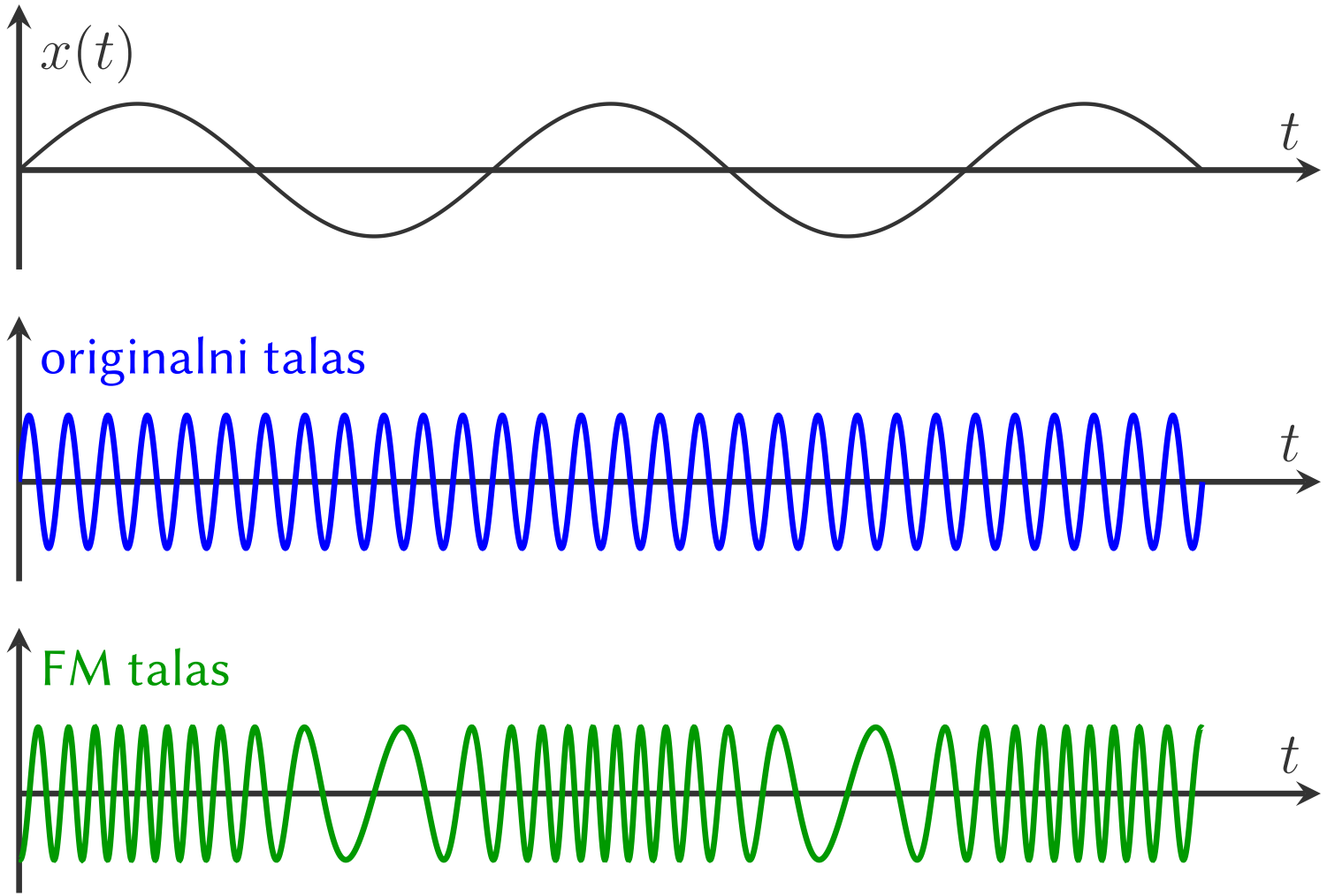
Frekvencijska modulacija (FM)

- AM radio je danas uglavnom zapostavljen u korist FM radia, zato što je ranjiv na spoljašnje zvuke, koji mogu da promene registrovanu amplitudu talasa (npr. grmljavina).
- Kod frekvencijske modulacije, amplitudu signala držimo konstantnom, dok se frekvencija menja u zavisnosti od jačine signala. Za originalni talas oblika $\sin(2\pi f t)$ i dozvoljeno odstupanje f_Δ, odgovarajući FM talas za signal $x(t)$ (ograničen sa $|x(t)| \leq 1$) je

$$\sin\left(2\pi f t + 2\pi f_\Delta \int_0^t x(\tau)\, d\tau\right)$$

- Naravno, ovo zahteva širi pojas frekvencija na kojoj radio stanica sme da odašilje signal u odnosu na AM.

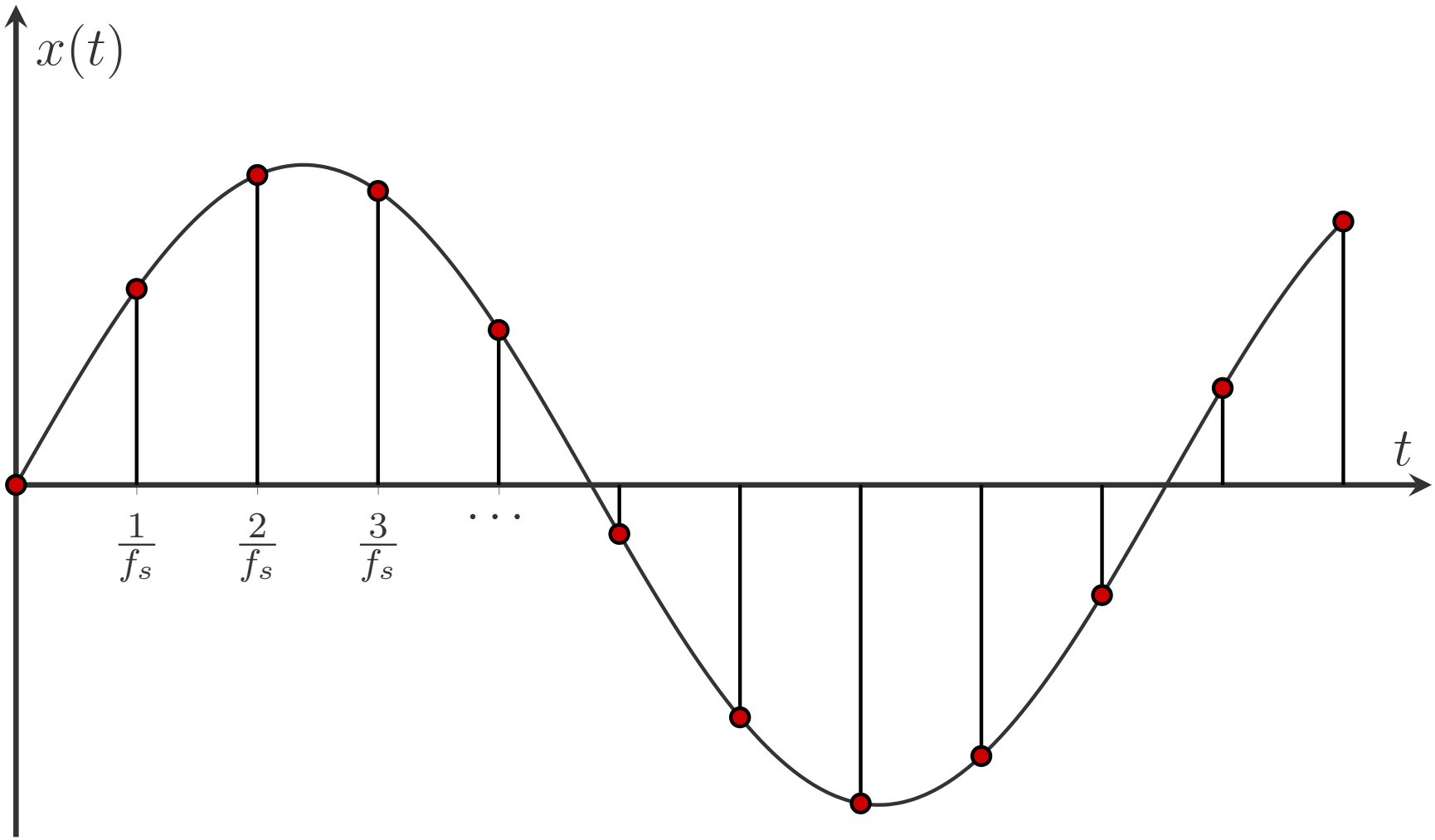
Frekvencijska modulacija (FM)



Sampling

- Sada želimo da vršimo obradu ovih signala u softveru. *Kako predstaviti sve podatke o nekom neprekidnom signalu u memoriji računara?*
- Odgovor: *ne čuvamo ceo signal*, već beležimo vrednosti njegove funkcije u određenim tačkama!
- **Sampling** (grub prevod: *uzorkovanje*) funkcije sa frekvencijom f_s herca podrazumevanje beleženje vrednosti funkcije na svakih $\frac{1}{f_s}$ sekundi.
- *Nikvistova teorema* (*Nyquist's sampling theorem*) kaže da, za dovoljno veliku vrednost f_s, **možemo kompletno rekonstruisati ceo signal** iz ovog niza vrednosti!

Sampling



IQ sampling

- Frekvencija kojom moramo vršiti samplovanje da bismo mogli da rekonstruišemo signal je proporcionalna najvišoj frekvenciji signala; za FM radio signale ovo je bespotrebno veliko (morali bismo samplovati na frekvenciji ~ 200 MHz da bismo izvukli korisne informacije iz nekoliko kHz).

- Jedan od načina da se neophodna frekvencija smanji je da se, umesto direktnog samplovanja signala, vrši **IQ sampling**.

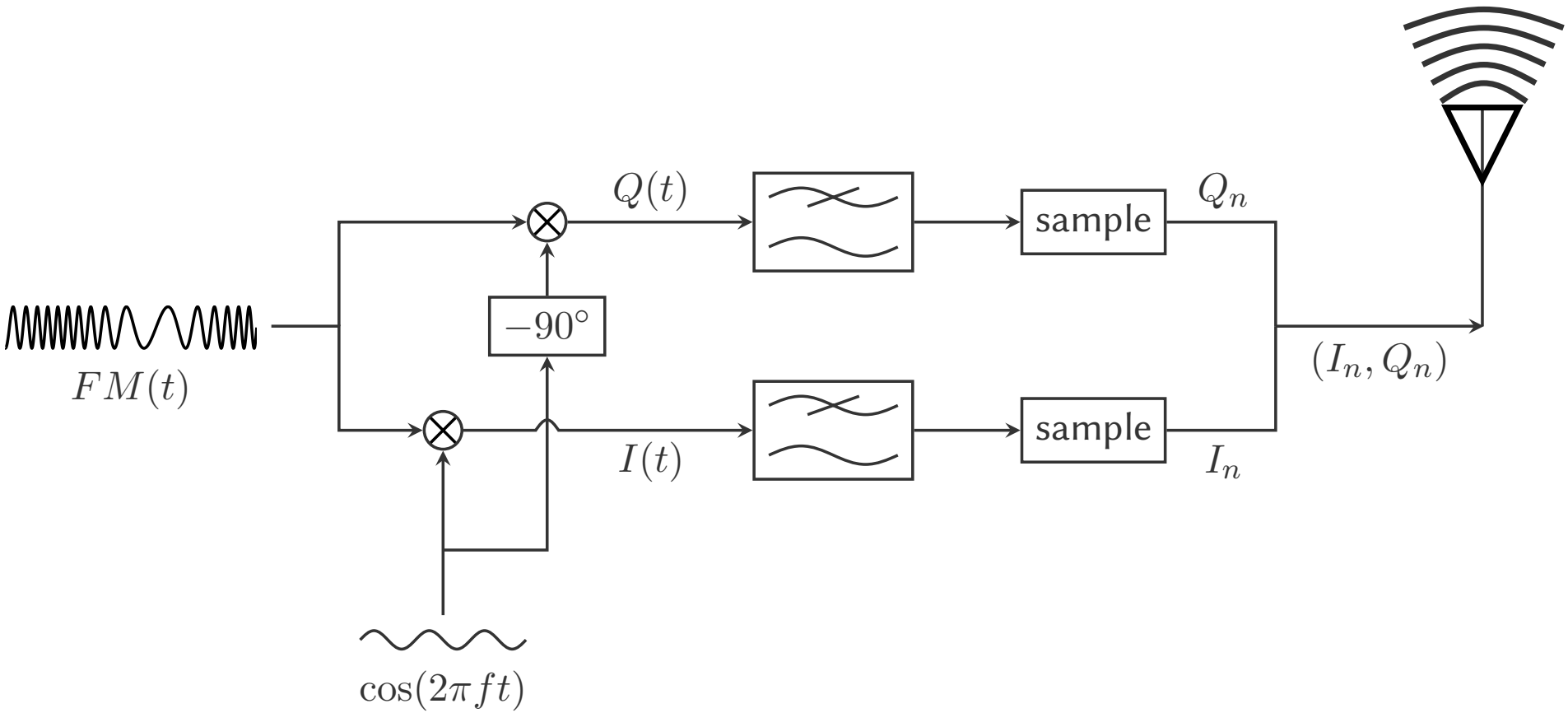
IQ sampling, *cont'd*

- IQ sampling podrazumeva snimanje dva signala koji zavise od signala $FM(t)$ odjednom:
  - $I(t) = FM(t)\cos(2\pi ft)$;
  - $Q(t) = FM(t)\sin(2\pi ft)$,

  gde je f frekvencija sa koje odašilje radio stanica.

- Pre samplovanja, ova dva signala se propuštaju kroz **nisko-propusni filter** (*low-pass filter*), koji eliminiše sve "visoko frekventne" elemente tih signala.

Kolo za IQ sampling



Šta zapravo dobijamo?

- Koristićemo dva trigonometrijska identiteta:

$$\sin(x) \cdot \cos(y) = \frac{1}{2}\sin(x-y) + \frac{1}{2}\sin(x+y)$$
$$\sin(x) \cdot \sin(y) = \frac{1}{2}\cos(x-y) - \frac{1}{2}\cos(x+y)$$

- Podsetnik:

$$FM(t) = \sin\left(2\pi f t + 2\pi f_{\Delta} \int_0^t x(\tau)\, d\tau\right)$$

Šta zapravo dobijamo? (*cont'd*)

- Odavde možemo zaključiti (precrtane izraze eliminiše filter):

$$
\begin{aligned}
I(t) &= FM(t)\cos(2\pi f t) \\
&= \frac{1}{2}\sin\left(2\pi f_\Delta \int_0^t x(\tau)\,d\tau\right) + \cancel{\frac{1}{2}\sin\left(\underline{4\pi f t} + 2\pi f_\Delta \ldots\right)} \\
Q(t) &= FM(t)\sin(2\pi f t) \\
&= \frac{1}{2}\cos\left(2\pi f_\Delta \int_0^t x(\tau)\,d\tau\right) - \cancel{\frac{1}{2}\cos\left(\underline{4\pi f t} + 2\pi f_\Delta \ldots\right)}
\end{aligned}
$$

- Ostali smo sa dva signala koja ne mogu imati veću frekvenciju od $f_\Delta \ll f$, tako da možemo samplovati sporije.

FM demodulacija iz IQ podataka

- Kako iz IQ podataka, samplovanih na frekvenciji f_s, sada izvući originalni signal $x(t)$?
- Primetiti da $I(t)$ i $Q(t)$ određuju tačku čije su koordinate u polarnom sistemu tačno: $r = \frac{1}{2},\ \alpha = 2\pi f_\Delta \int_0^t x(\tau)\, d\tau$.
- Pošto je $x(t)$ u izrazu za polarni ugao α dat unutar *integrala*, možemo ga (do na konstantu) izvući kao *izvod* tog ugla! Tačnije:

$$\alpha_n = \arctan \frac{Q_n}{I_n}$$

$$x_n = \frac{\alpha_{n+1} - \alpha_n}{\Delta t} = (\alpha_{n+1} - \alpha_n) \cdot f_s$$

Demo

- Za današnju demonstraciju, napravićemo jednostavan radio prijemnik u MATLAB-u.
- Kao ulazne podatke, naš prijemnik prima IQ podatke sa radio stanice, samplovane na 240 kHz; podaci su snimljeni tačno na centralnoj frekvenciji radio stanice *BBC Radio Cambridgeshire*, tako da ne moramo raditi dodatne manipulacije podataka.
- Izlaz prijemnika treba da bude `.wav` fajl koji odgovara početnom signalu poslatom sa stanice.

Kompletan MATLAB izvorni kod

```
% PARAMETERS
f_sample = 240000;
f_filter = 16000;
f_output = 48000;

% READ IQ DATA
f = fopen('iq-fm-96M-240k.dat', 'r', 'ieee-le');
c = fread(f, [2, inf], '*float32');
fclose(f);

I = c(1,:);
Q = c(2,:);
```

Kompletan MATLAB izvorni kod, *cont'd*

```
% FM DEMODULATION
angles = atan2(Q, I);
d_alpha = diff(angles);
X = unwrap(d_alpha) * f_sample;

% LOW-PASS FILTER
Wn = f_filter / (0.5 * f_sample);
[b, a] = butter(5, Wn, 'low');
f_X = filter(b, a, X);

% DOWNSAMPLE, NORMALISE
f_X_down = f_X(1:5:end);
f_X_norm = f_X_down / max(f_X_down);
audiowrite('radio_out.wav', f_X_norm, f_output);
```

nedeljainformatike v2.0

MATLAB kroz primere

Probijanje PayPal HIP-a

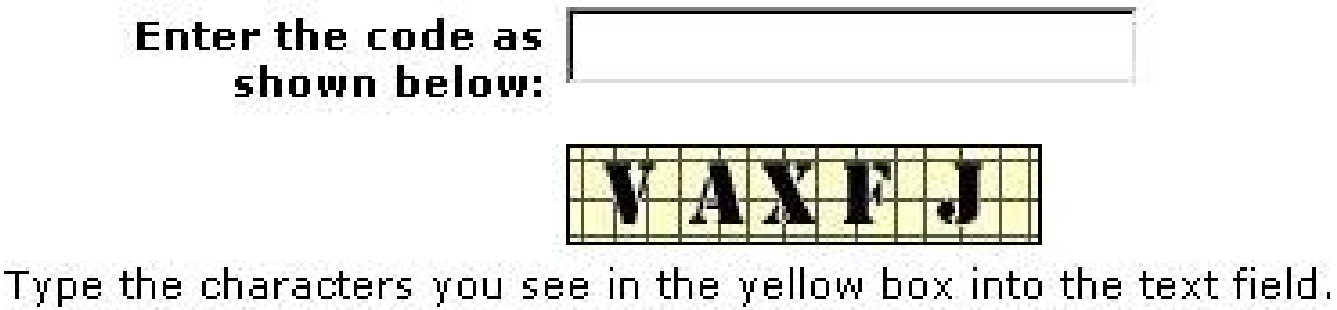
Nikola Nedeljković

Matematička gimnazija
Nedelja informatike v2.0

15. decembar 2015.

Šta je HIP?

- HIP (*Human Interactive Proof*) je metod korišćen za automatsko razlikovanje legit čoveka od maliciozne mašine.
- Naziv HIP se preferira u odnosu na naziv CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*), sa kojim ste verovatno upoznati.



Osobine HIP-a

- HIP treba da bude lako generisan od strane mašine i lako rešiv od strane čoveka.
- Takodje, rešavanje ovog HIP-a bi za svaku drugu mašinu(logično) trebalo da bude nemoguće ili jako teško.
- Dakle, treba pronaći težak *AI* problem koji će čovek da reši bolje od mašine.
- Neki od tih problema su *natural language processing*, *character recognition*, *image understanding*...

Metod rada

- U ovoj MATLAB demonstraciji za probijanje PayPal HIP-a korističemo standardni OCR(*Optical character recognition*) proces.
- OCR predstavlja proces prevodjenja napisanih ili odštampanih slika u format razumljiv mašinama.
- OCR proces se može podeliti u tri glavna dela: obrada(*pre-processing*), segmentacija(*segmentation*) i klasifikacija(*classification*).

Tipovi slike(ukratko)

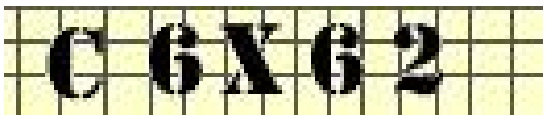
- Kao što znate slike su sastavljene od piksela koji su u računaru predstavljeni bitovima.
- Binarna(*binary image*, crno-bela) slika uzima za vrednost svakog piksela jedan bit(koji može biti *on* ili *off*).
- Kod *Grayscale* slike je svaki piksel predstavljen jednim bajtom(tj. može uzimati 256 vrednosti).
- RGB(*red-green-blue*) slika je kombinacija tri '*grayscale*' slike u ovim bojama.
- Kako redom izgledaju sve, videćemo kroz obradu HIP-a.

Neke tehnike obrade

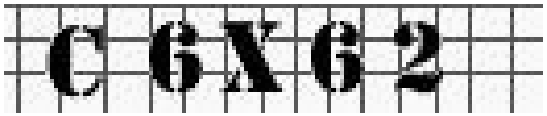
- U slučaju loseg papira, stampača, skenera slika moze sadržati dosta šumova(*noise*).
- Zaostali 'šumovi' na slici mogu da se uklone filtriranjem slike.
- *Thresholding* je proces postavljanja svih vrednosti vecih od odredjene na *on*, a manjih na *off* i cesto se koristi kao metod 'binarizovanja' slike.
- *Thresholding*-om se takodje može ukloniti zvuk, ako neki pikseli imaju vrlo velike ili male vrednosti.

Obrada slike

- Prva slika sa kojom se susrećemo je RGB. Potrebno ju je pretvoriti u *grayscale*. To, kao i mnoštvo drugih stvari kojima ćemo se kasnije koristiti se u MATLAB-u radi vrlo lagano.



- *greyScale = rgb2gray(i);* i dobijamo:



- Zatim 'binarizujemo' sliku i otklanjamo šumove 'thresholdovanjem'.
- *thresholded = greyScale < 30;*

Obrada slike - nastavak

- Nakon toga se izvršava dodatno 'čišćenje' kojim se uklanjaju pikseli iz redova cija je suma piksela jako mala, i 'bounding' kojim se okružuje '*on*' deo slike.

```
rows = size(thresholded, 1);% SIZE prva dimenzija
row = 1;
while (row < rows)
rowSum = sum(thresholded(row,:));
if (rowSum < 5 && rowSum > 0)
thresholded(row,:) = 0;
end
row = row + 1;
end
bb = regionprops(double(thresholded), 'BoundingBox');% BOUNDING
bounded = imcrop(thresholded, bb.BoundingBox);
end
```

- Poslednja slika se predaje segmenteru.

Cilj i tipovi segmentacije

- Proces segmentacije koji ovde radimo predstavlja podelu slike na više slika od kojih svaka sadrži tačno jedan karakter.
- Postoje dva pristupa segmentaciji koja se mogu primeniti.
- Pristup disekcijom(*Dissection Approach*) podrazumeva podelu slike zasnovanu na osobinama karaktera kao što su visina, širina, rastojanje od susednih karaktera.., nakon čega sledi klasifikacija karaktera. Pri ovoj demonstraciji ćemo koristiti ovaj pristup.
- Pristup klasifikacijom - Kroz sliku se iterativno traže komponente koje najvise liče već poznatim 'klasama' alfabeta.

Kako segmentovati?

- Kako biste vi, koristeci bit reprezentaciju slike, podelili sliku na odgovarajuće segmente karaktera?

Kako segmentovati?

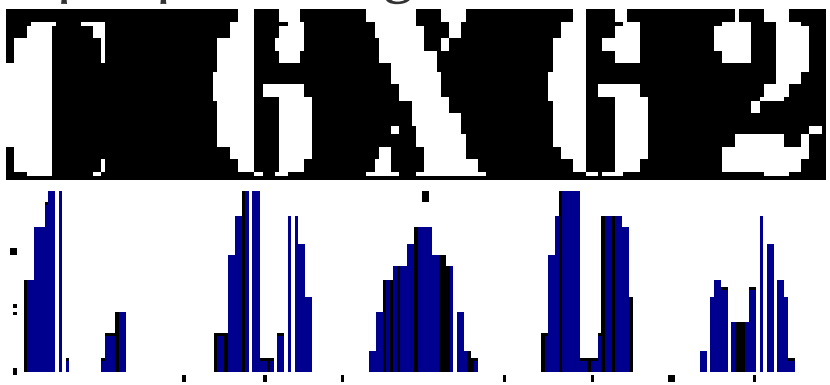
- Kako biste vi, koristeci bit reprezentaciju slike, podelili sliku na odgovarajuće segmente karaktera?
- Neki od mogućih načina segmentacije:
- Projekcijama(horizontalne i vertikalne)
- Vertikalna(horizontalna) projekcija predstavlja jednostavno brojanje *on* piksela u kolonama(redovima). Ako je vrednost vertikalne projekcije jako mala ta kolona je kandidat za podelu slike. Mi ćemo koristiti ovaj metod.
- Kolone koje dele sliku se mogu naći i kao '*peakovi*' izvoda vertikalnih projecija.
- Povezane komponente(grafovi)?
- Mogući problemi?

Bitne osobine PayPal HIP-a za segmentaciju

- Proces segmentacije je olakšan jer svaki HIP ima tačno 5 karaktera.
- Eksperimentalno je utvrdjeno da je svaki karakter širok bar 10 piksela.
- Više karaktera - veće šanse za grešku.

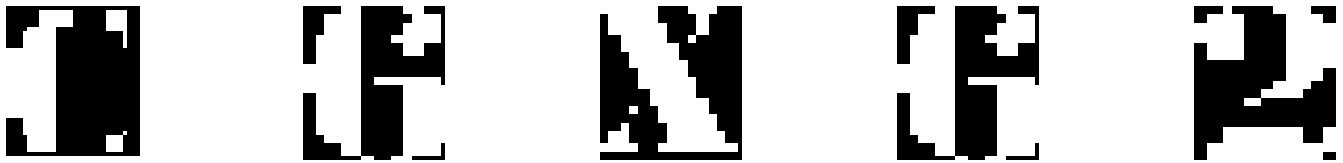
Segmentacija slike - kod

- Input predat segmenteru i vertikalne projekcije, redom:



```
[rows cols] = size(bounded);
col=3;
startCol = 1;
charIndex = 1;
while(col < cols) % do kraja slike
col = col + 10;
while (col+2 <= cols) && (sum(sum(bounded(:,col:col+2))) > 0)% dok nije prazna kolona
col = col + 1;
end
a = imcrop(bounded, [startCol 1 (col - startCol) rows]);% koordinate pravougaonika
% Pad out to 20 rows and 20 cols with 0's
[charRows charCols] = size(a);
a = padarray(a, [(20 - charRows) (20 - charCols)], 'post');
retVal(:,:,charIndex) = a;
charIndex = charIndex + 1;
col = col + 1;
startCol = col;
% Napred dok je belo
while (col <= cols) && (sum(sum(bounded(:,col))) == 0)
col = col + 1;
startCol = startCol + 1;
```

Rezultat segmentacije

- Nakon segmentacije:
- 
- Nakon popunjavanja(zbog klasifikacije):
- 
- Ove slike se predaju klasifikatoru.

Postupak treniranja

- Kako bi mogla da se izvrši klasifikacija prvo moramo napraviti templejt fajlove, pomoću kojih ćemo kasnije prepoznati slike koje je poslao segmenter.
- Templejt fajlovi su napravljeni na osnovu 20 trening HIP-ova iz direktorije 'training'.
- HIP-ovi su *randomly* odabrani i u sebi sadrže sve karaktere(sem I,0,Q,O i 1 koje PayPal ionako ni ne koristi).
- Neka sa s_i^c predstavljamo matricu karaktera c
- Kako se neko slovo može pojavljivati više puta, konačna matrica t^c(templejt) karaktera c se računa po formuli

$$t^c = \left(\sum_{i=1}^{n} s_i^c\right)/n$$

Treniranje - kod

- Sve primere iz 'training' direktorije takodje obradjujemo i segmentiramo.
- *for i=1:numTrainingSamples*
  bounded = preprocess(imread(filename));
  chars = segment(bounded);
- Zatim dodajemo na templejt matricu i tražimo aritmetičku sredinu(*asciiz* je indeks karaktera).
- *templates(:,:,asciiz) = templates(:,:,asciiz) + chars(:,:,i);*
  counts(asciiz) = counts(asciiz) + 1;
  end
  for i=1:36
  templates(:,:,i) = templates(:,:,i) / counts(i);

Klasifikatori

- Prilikom klasifikacije želimo da što bolje povežemo zadate slike iz segmentera sa odgovarajućim templejtom.
- Svaki karakter(matricu) dobijen iz segmentera i templejtove možemo predstaviti *feature* vektorom.
- Tada najsličniji vektori predstavljaju najsličnije karaktere.
- Koeficijent koorelacije(*CORR2*) izmedju dve matrice i i j racunamo kao

$$\frac{\sum_m \sum_n (i_{mn} - i_{sr})(j_{mn} - j_{sr})}{\sqrt{\left(\sum_m \sum_n (i_{mn} - i_{sr})^2\right)\left(\sum_m \sum_n (j_{mn} - j_{sr})^2\right)}}$$

gde su i_{sr} i j_{sr} srednje vrednosti matrica i i j.

Klasifikatori - nastavak

- Što je koeficijent korelacije veći to su dve predate matrice sličnije.
- Prema *Cauchy-Schwartz*-u vidimo da je maksimalna vrednost koeficijenta korelacije 1.
- *Feature* vektor može biti broj *on* piksela.
- Takodje, za *feature* vektor možemo uzeti vertikalne projekcije.
- Ovde je za *feature* vektor uzeta cela matrica dobijena od segmentera, jer su tako dobijeni najbolji rezultati, i na nju je primenjeno računanje koeficijenta koorelacije zadatog po prethodnoj formuli.
- Ukupnu pouzdanost za klasifikaciju dobijamo kao prozivod 5 maksimalnih koeficijenata koorealcije.

Kod - isečak

```
confidence = 1.0;
for i=1:5 % za svaki char
allCorrs = zeros(1, 36);
for j=1:36 % za svaki templejt
temp = templates(:,:,j);
in = chars(:,:,i);
allCorrs(j) = corr2(temp, in); % CORRCOEF za [char,templejt]
end
index = find(allCorrs == max(allCorrs), 1); % najvece poklapanje
confidence = max(allCorrs) * confidence;
end
```

Demo

- Pomoću funkcije *recognizeAll.m* puštamo prepoznavanje svih slika iz direktorijuma '*test*'
- Ispisujemo rezultate...

```
charAcc = charCorrect / (charCorrect + charWrong);
hipAcc = hipCorrect / numTestingSamples;
avgConfidence = sum(confidences) / numTestingSamples;
minConfidence = min(confidences);
fprintf('Character Accuracy: %f\n', charAcc);
fprintf('HIP Accuracy: %f\n', hipAcc);
fprintf('Average confidence: %f\n', avgConfidence);
fprintf('Minimum confidence: %f\n', minConfidence);
```

XKCD

- Svi materijali cele prezentacije **MATLAB kroz primere** mogu se naći na linku `https://github.com/mgcsweek/MATLAB-Examples`.