Primena računara u fizičkoj hemiji

Nastavnik: ***Miloš Mojović, v.prof.***

Asistent: ***Aleksandra Pavićević***

Čemu ovaj predmet?

Program predmeta:

- računarski sistem
- hardver računara
- razlozi za alternativne OS i njihovu upotrebu
- mrežni parametri i *on line* naučne baze podataka
- rad u programima koji su relevantni za fizičku hemiju (MATLAB, Mathematica, LabVIEW ...)
- upoznavanje sa osnovama računarskih simulacija
- osnove savremenog programiranja
- računari i merni instrumenti, akvizicija podataka

Literatura i ispit:

- **Literatura:**
  1. Predavanja
  2. Internet
  3. Knjige

- **Ocenjivanje**

1.	aktivnost u toku predavanja	**10** (% dolaznosti na predavanja/10)
2.	praktična nastava	**30** (izrada, sređivanje i odbrana vežbi)
3.	pismeni ispit	**60** (zadatak **30*** + teorijska pitanja **2x15**)

* ispitni zadatak moze biti zamenjen seminarskim radom

Računarski sistem:

- **Definicija:**

  Skup mašina (hardvera) i pridruženih metoda (softvera) organizovanih radi vršenja automatske obrade podataka.

- **Struktura:**

  - Centralni procesor (CPU)
  - Unutrašnja memorija
  - Različiti I/O uređaji

Softver računara:

- Softver je ključna komponenta koja omogućava da računar obavlja različite zadatke.
- Postoje dva osnovna tipa softvera:

  - sistemski softver

  - aplikativni softver
- **Sistemski softver** kontroliše i koordiniše rad različitih uređaja koji su sastavni delovi računarskog sistema. Najvažniji deo sistemskog softvera je skup programa koji se zajedno nazivaju **operativni sistem**.
- **Aplikativni softver** dopušta korisnicima da koriste računar za rešavanje najrazličitijih zadataka. Postoji ogromna količina različitog aplikativnog softvera (za pisanje i formatiranje teksta, grafički softver, matematički softver, igrice ...)
- Sistemski i aplikativni softver pišu po zadatoj potrebi kompjuterski programeri koristeći sintaksu koja je karakteristina za određeni programski jezik.
- Kompjuterski jezici: Od 1943. pa do 2009. napravljeno je oko 260 različitih programskih jezika od **ENIAC** (**E**lectronic **N**umerical **I**ntegrator and **C**omputer) do **Cω** (C-Omega) jezika.

Operativni sistemi (OS):

- Predstavljaju najvažniji deo sistemskog softvera.
- Omogućavaju računaru da izvršava osnovne funkcije (rad sa aplikativnim softverom).
- Sadrže interfejs prema korisnicima (operativno okruženje) i čine rad sa računarom manje-više konfornim.
- U globalu (recimo definicija): ***Operativni sistem je sistemski softver koji upravlja aktivnostima računara, kontroliše hardver računara i izvršavanje korisničkih programa.***
- Osnovni ciljevi kvalitetnog OS su da:

  - obezbedi udoban rad korisnika
  - obezbedi efikasnije korišćenje računarskih resursa
  - obezbedi stabilnost i sigurnost u radu sa računarom

- Problem je što efikasnost, udobnost i stabilnost često ne idu zajedno (DOS i Windows).

Funkcije OS:

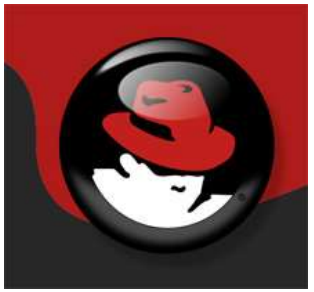
- Postoji bitna korelacija između snage računarskog sistema i mogućnosti operativnog sistema.
- Računari veće snage mogu da podrže (ali i zahtevaju) moćniji operativni sistem.
- U opštem slučaju, funkcije savremenih operativnih sistema mogu se podeliti u četiri kategorije:

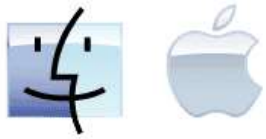
  - zauzimanje i dodeljivanje sistemskih resursa (CPU, memoriji, I/O uređajima dodeljuje sve potrebne resurse za izvršenje nekog posla). To radi skup programa OS koji se zove **supervizor** (**kernel**).

  - raspoređivanje različitih poslova (određivanje na koji način će resursi obavljati neki posao).

  - nadgledanje aktivnosti sistema (npr. obaveštava o greškama).

  - ostvarivanje interfejsa između korisnika i računara (nekada je bio linijski režim rada kao **DOS** ili **UNIX**, ali danas se koristi **GUI** (**G**raphical **U**ser **I**nterface) režim npr.: Windows, Mac OS ili Linux.





Tipovi operativnih sistema:

- Operativne sisteme možemo podeliti prema:

  - tehnologiji rada (UNIX ili Windows ...)

  - vlasništvu i licenci (vlasništvo neke firme ili open source)

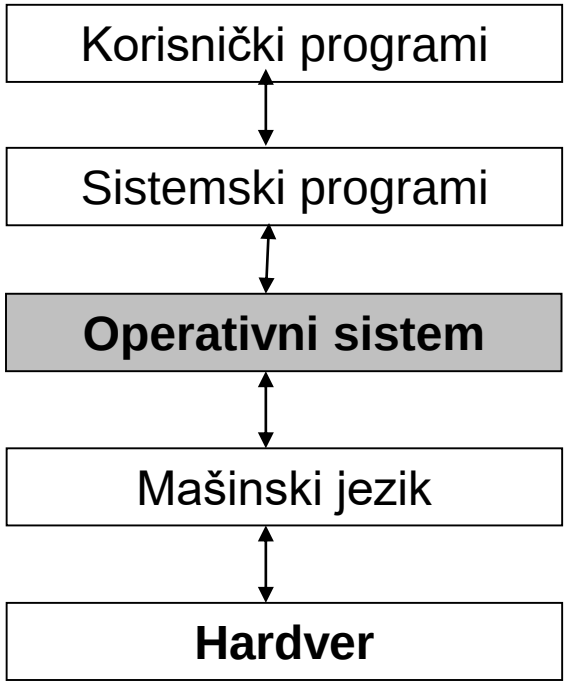
  - radnom okruženju (stariji kao DOS ili OS/2 ili noviji kao Windows, Linux)

  - svrsi (istraživanje, proizvodnja, zabava, razvoj ...)

- Trenutno postoji ogroman broj OS koji se koriste u različitim uređajima (mobilnim telefonima, konzolama za igru, serverima, kućnim računarima, industrijskim uređajima ...) npr. Windows CE, Symbian, Palm OS, Android OS ....
- Na ovome kursu biće reči o **Linux OS** (**SUSE i UBUNTU**).
- Počinjemo od samog početka tj. instalacije Windows i Linux OS na prazan računar.

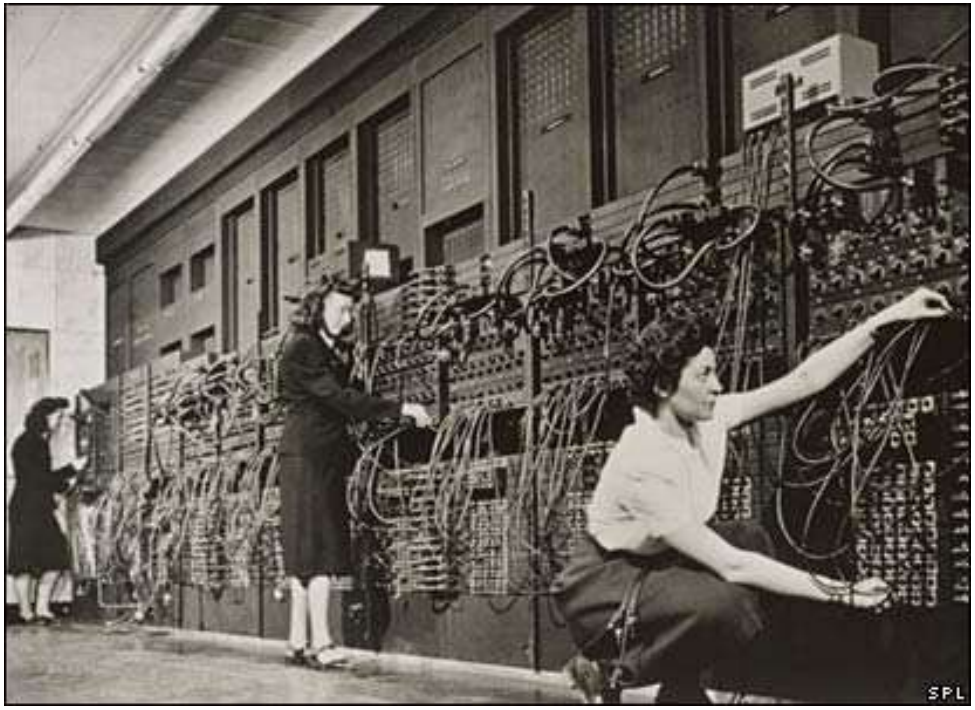
Gde se u hijerarhiji nalaze OS?

- OS koristi predefinisanu bazu podataka da bi omogućio hardver-softver interakciju.
- Zbog toga, sa korisničkim programima komuniciramo preko OS pošto oni sami nemaju direktan prisup hardveru računara.
- OS je sam po sebi program ali njegovi prioriteti nisu isti kako kod korisničkih programa.
- OS korist tzv “kernel” mod za razliku od korisničkih programa koji koriste tzv “user” mod.
- Razlika je u tome što su sve instrukcije hardveru validne u “kernel” modu što nije slučaj za “user” mod.



Istorija operativnih sistema

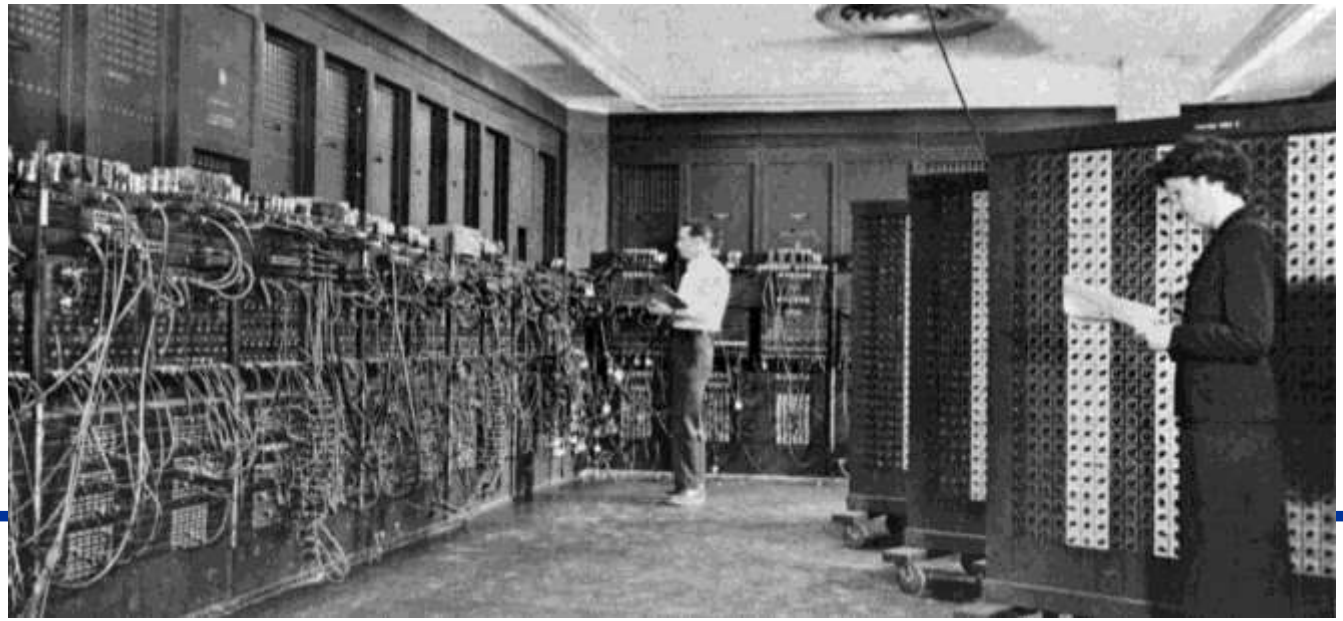
- Sve je počelo 40-tih godina prošlog veka sa razvojem hardvera.



ENIAC 1943

Istorija operativnih sistema

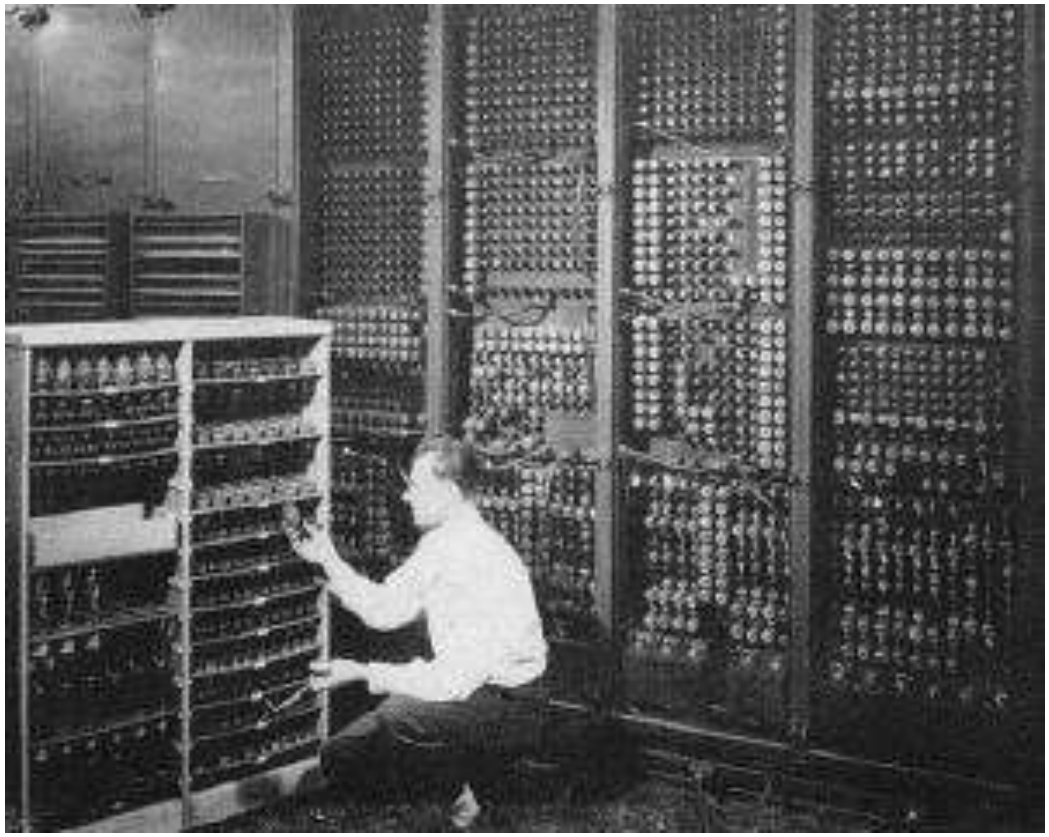
- **ENIAC** (Electronic Numerical Integrator and Computer), razvijen je od strane U.S.A. za potrebe vojske.
  - izgrađen 1940-tih,
  - težio 30 tona,
  - 2.5 m visok, 1 m širok i 30 m dugačak,
  - U njegovom sastavu bilo je 18 000 vakuumskih cevi koje je hladilo 80 rashladnih uređeja.



Istorija operativnih sistema

ENIAC vakuumske cevi

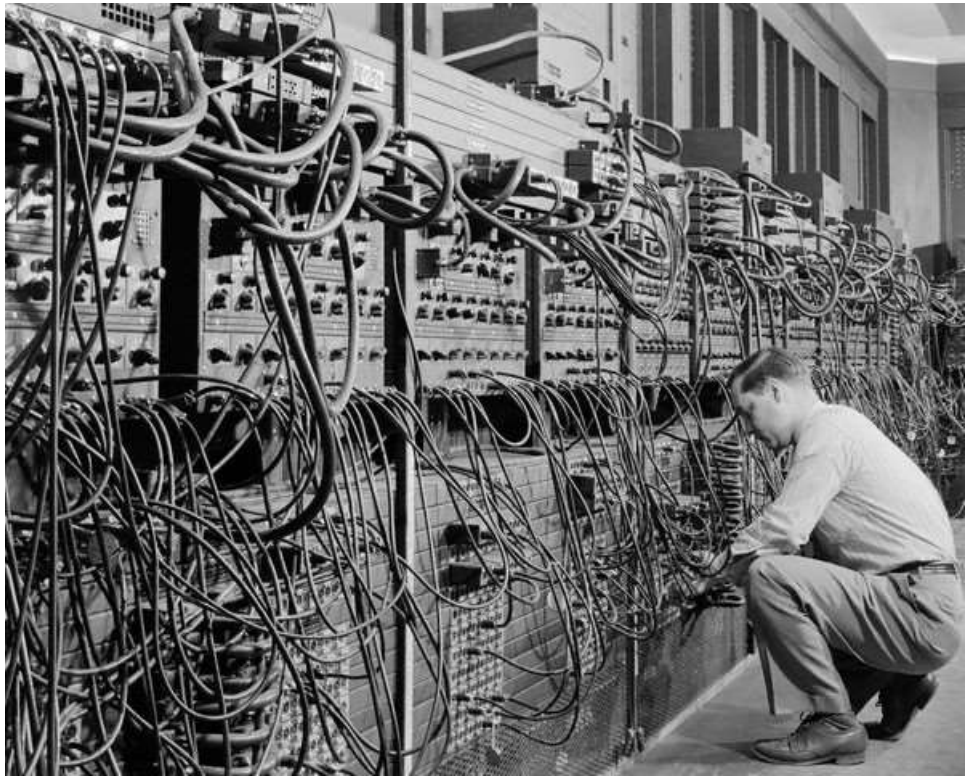
Istorija operativnih sistema



ENIAC – zadnja strana

Istorija operativnih sistema

Programi su se unosili u memoriju ručno pomoću prekidača, bušenih kartica i papirnih traka

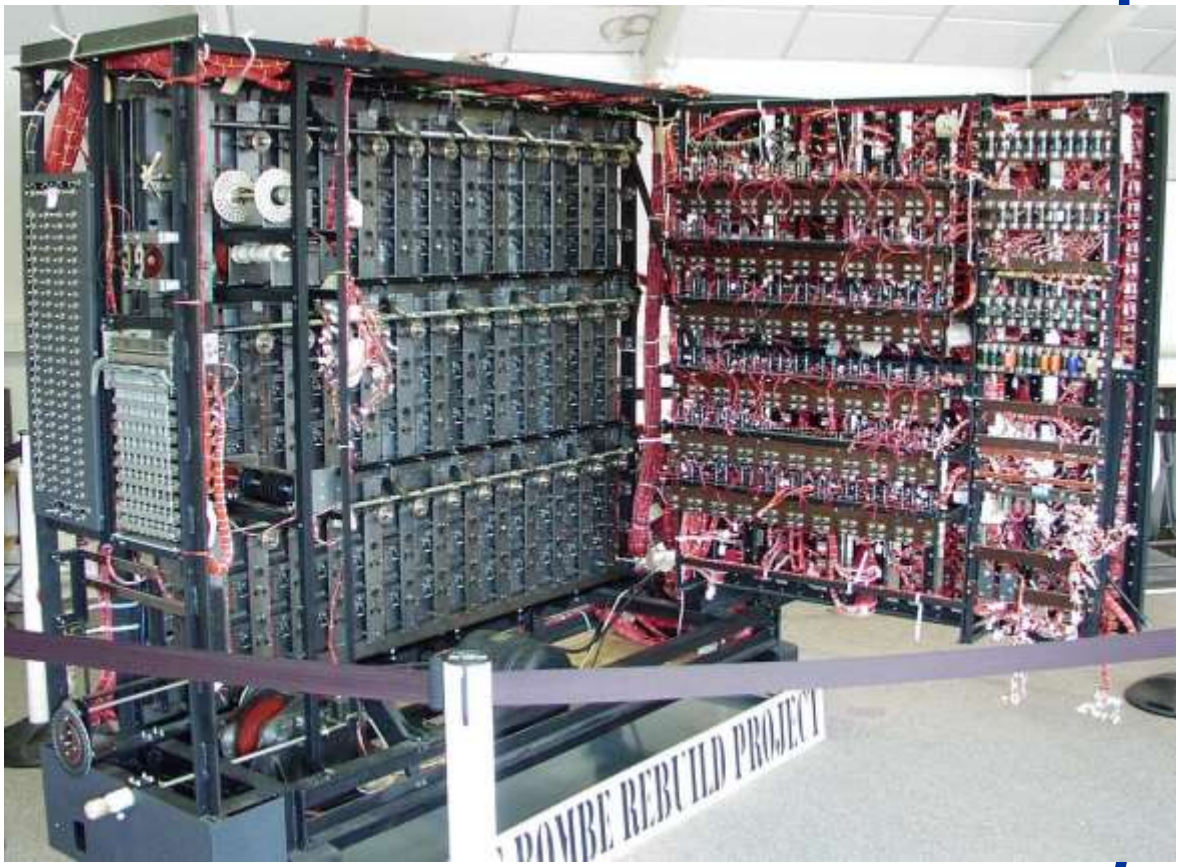


ENIAC : “kodiranje” pomoću kablova

Istorija operativnih sistema

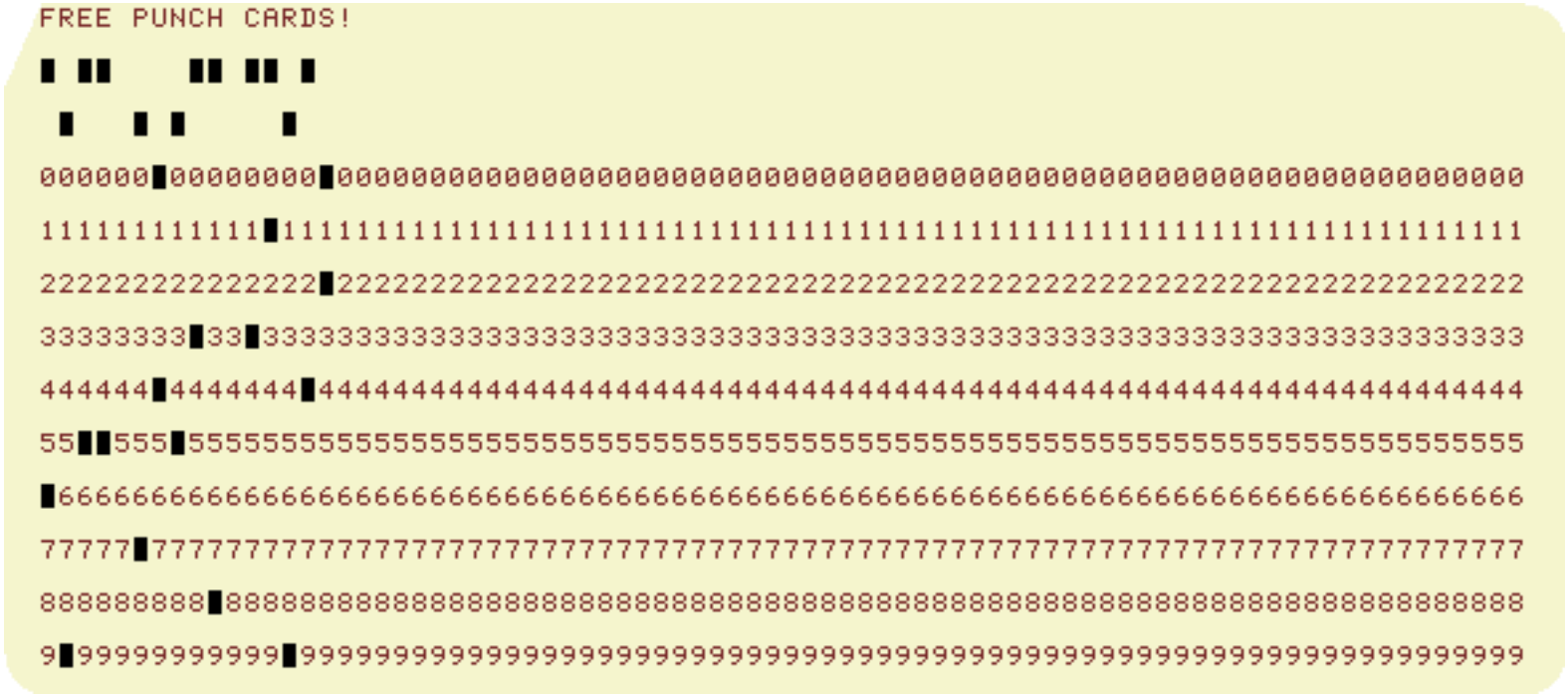
- Turingova mašina korišćena za dekriptovanje “Enigme” (1940).

Alan Turing



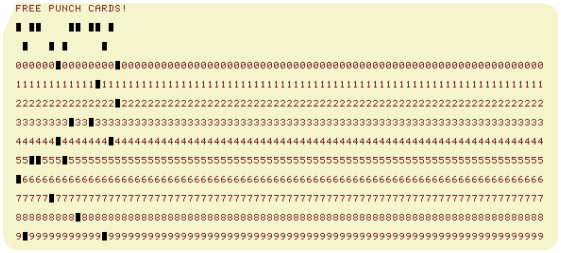
Istorija operativnih sistema

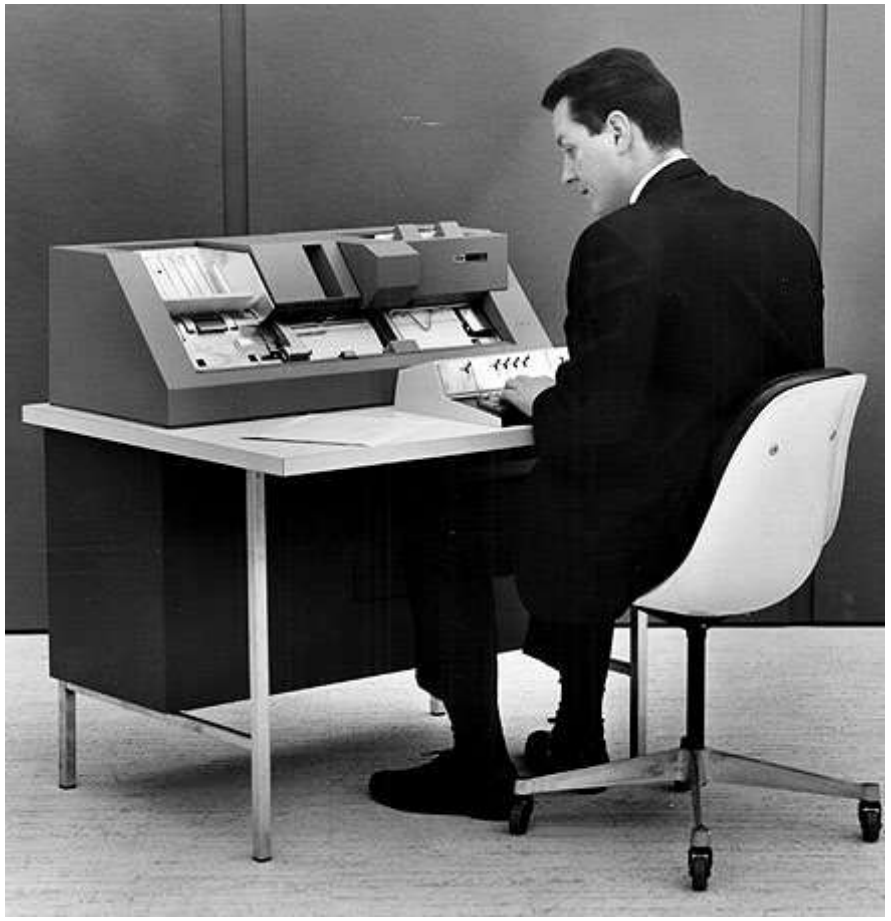
Bušena kartica



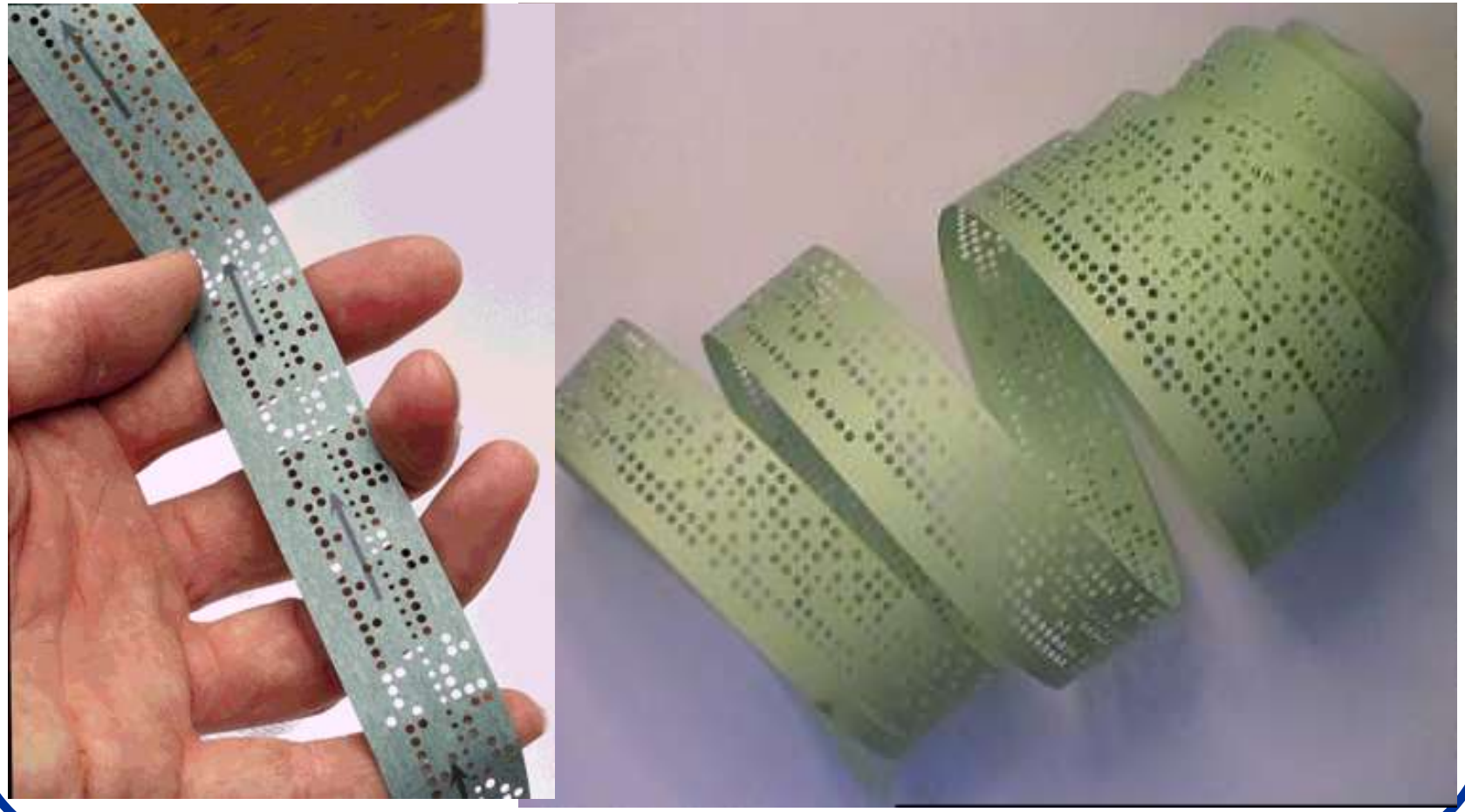
Istorija operativnih sistema

Radna stanica



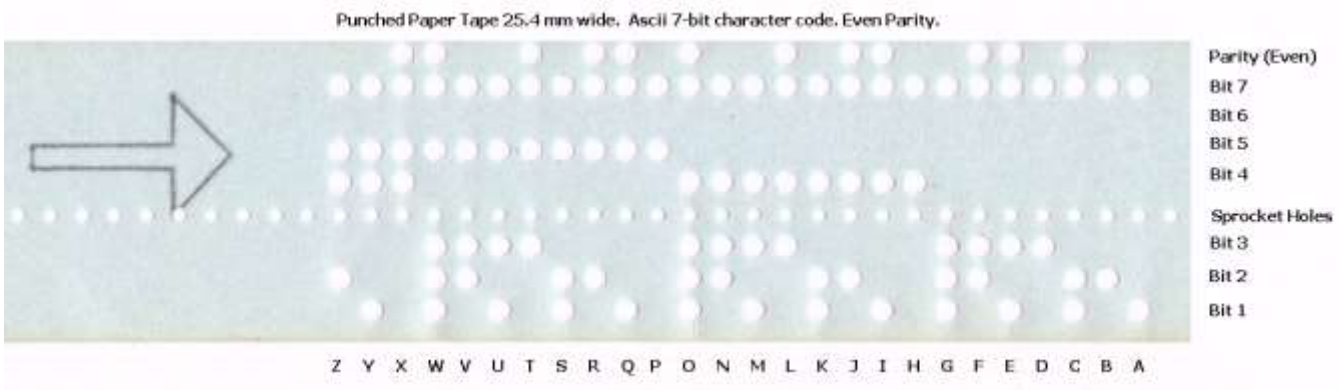


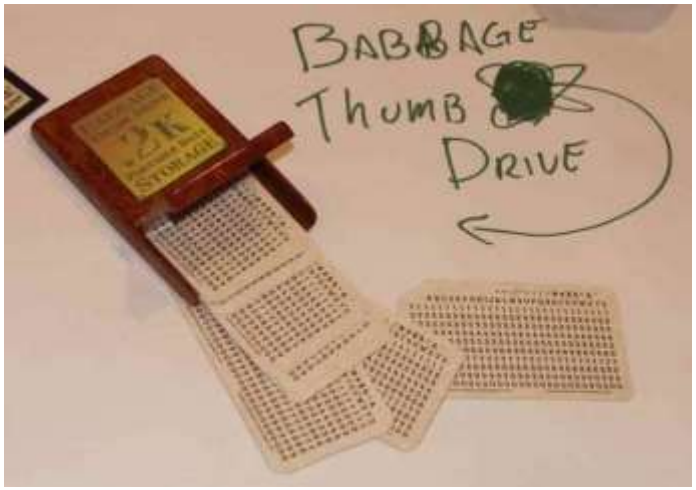
Istorija operativnih sistema



Bušena papirna traka

Istorija operativnih sistema





Bušena papirna traka

USB – fleš preteča ☺

Istorija operativnih sistema

Lola 8K

ZX spectrum 48K

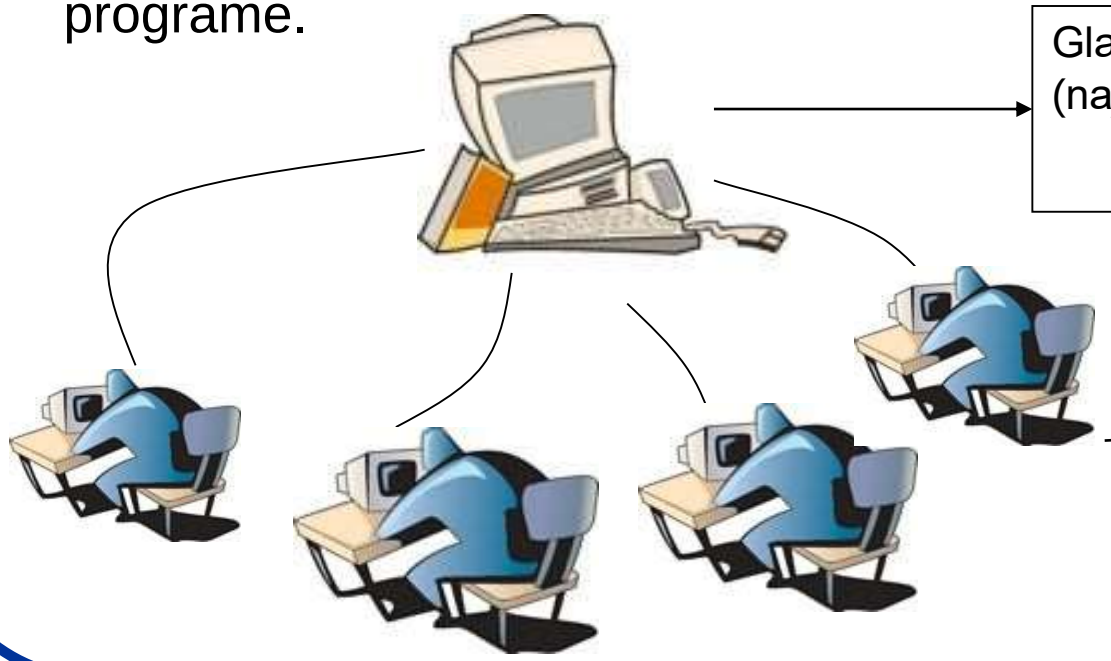
Atari 16K

Commodore 64K

K = Kb ROM

Istorija operativnih sistema

- 1970-tih, pojavila se ideja o novoj upotrebi računara koja je nazvana “**time-sharing”.**
- Ideja je bila da više korisnika ima terminale (ne kompjutere) koji su povezani sa glavnim računarom na kome izvršavaju svoje programe.

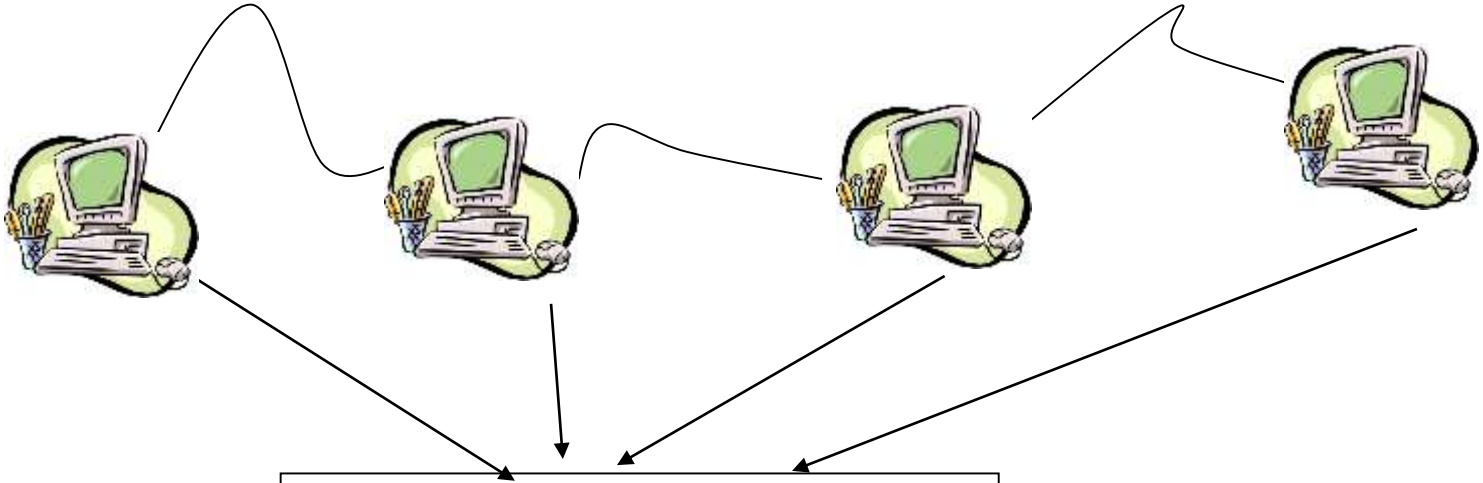


Glavni računar; ima CPU i OS, (najčešće UNIX tipa).

Terminali su povezani za glavni računar (terminali nemaju CPU).

Istorija operativnih sistema

- Još jedan tip računarskih sistema su **multiprocesorski sistemi**.
- Ovde imamo više procesora na jednoj ploči koji dele memoriju i periferije.
- Ovakvi sistemi imaju veliku snagu.
- Na kraju imamo i **mrežne sisteme** gde se svi procesi odvijaju na posebnim računarima ali (pomoću adekvatnog OS) svaka mašina ima pristup jedna drugoj.
- Na ovaj način, lako se mogu deliti fajlovi i svake druge informacije.

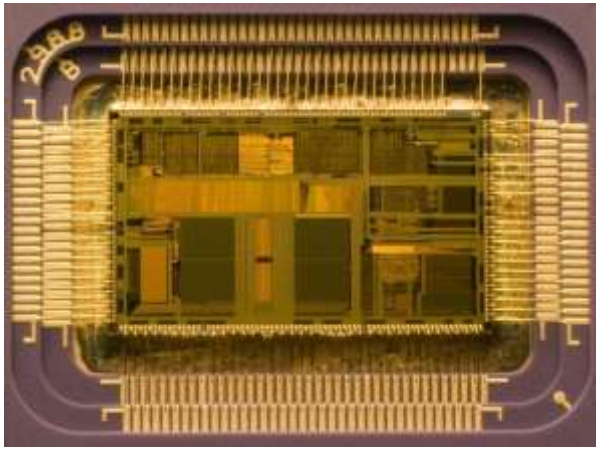


Svaki računar ima svoj sopstveni CPU, RAM, itd. Instalisan je OS koji podržava intenzivnu mrežnu razmenu.

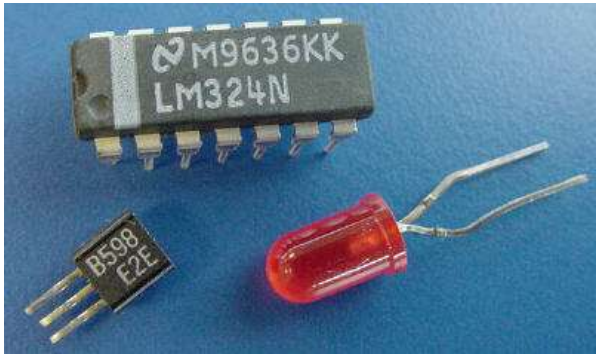
Istorija operativnih sistema

Centralni procesor:

- Aritmetičko-logičko-upravljačka jedinica koja se koristi za obradu podataka.
- CPU koji se pravi kao jedno integrisano kolo **(IC)** zove se **mikroprocesor**.
- Do sredine 90-tih mikroprocesori se prave od silicijumskih kristala **(poluprovodnika)** u koji se urezuju tranzistori (Si dolina).
- Tehnološko ograničenje: debljina krtih Si ploča.
- Danas se koriste novi materijali.
- Ali prvo terminologija ...



Mikroprocesor



Čip, LED, tranzistor

Silicijumska dolina

Silicon Valley, San Francisco, Northern California, United States.
Tu su predstavništva ...

Adobe Systems

AMD

Apple Inc.

eBay

Google

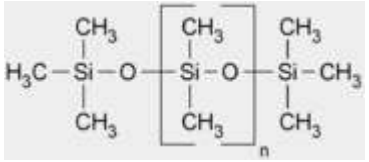
Hewlett Packard

Intel

Oracle

Yahoo!

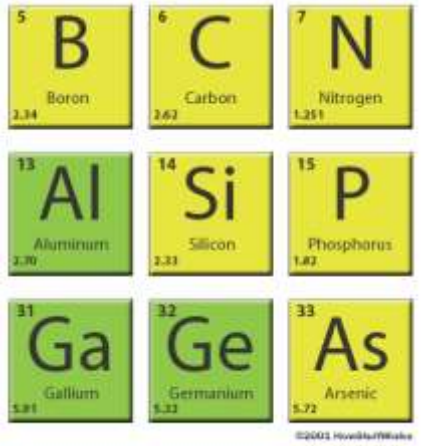
- Silicon je engleski nazi za silicijum **Si** (atomski broj 14).
- Jedinjenja Si sa O, H i C zovu se **silikoni** (silicones).

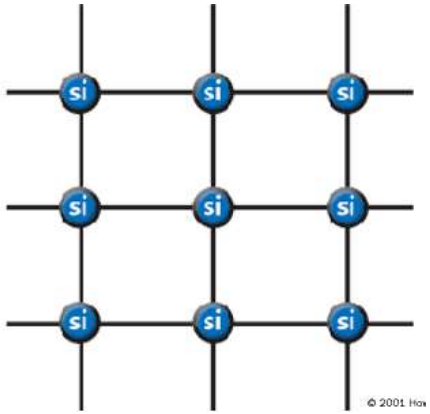


Poli(dimetillsiloksan)

Poluprovodnici:

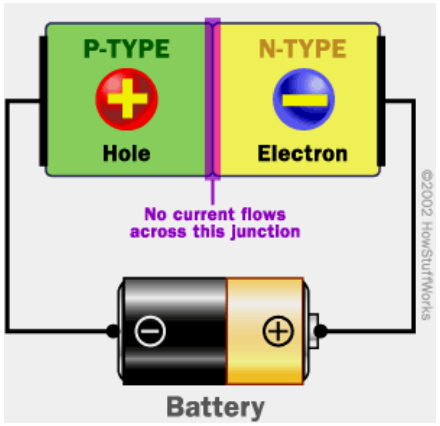
- Osnovni element od koga su napravljeni poluprovodnici je **Si** (koji kao **Ge** i **C** ima $4e^-$ u valentnoj orbitali.
- U svojoj kristalnoj stukturi formira čvrste kovalentne veze ($4e^-$ sa $4e^-$ suseda).
- Zbog ovoga praktično nema slobodnih e^- te je Si zbog toga skoro potpuni izolator.
- Osobine Si mogu se menjati njegovim onečišćavanjem (tj.dopiranjem) nečistoćama **N**-tipa i **P**-tipa.
- **N -** (negative) tip nečistoća podrazumeva dodavanje **P** ili **As** koji imaju $5e^-$ u spoljašnjoj ljusci. Dodatni e^- nije vezan i čini sistem provodnikom struje.
- **P -** (positive) tip nečistoća podrazumeva dodavanje **B** ili **Ga** koji imaju samo $3e^-$ u spoljašnjoj ljusci. Sada postoji višak šupljina koje takođe dobro provode struju (prihvatanjem e^- i prenošenjem šupljina kroz provodnik).
- Malo dodavanje nečistoća čine Si dobrim (ne odličnim) provodnikom, tj. poluprovodnikom. Šta se dobija kada se spoje **N** i **P**-tip **Si**?

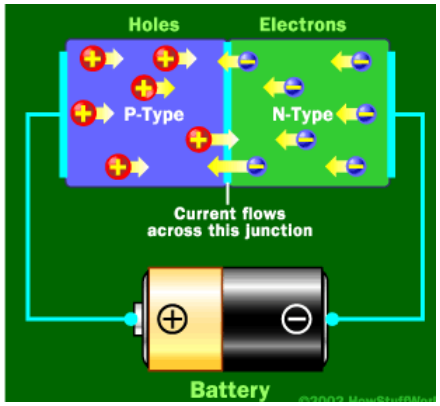




Dioda:

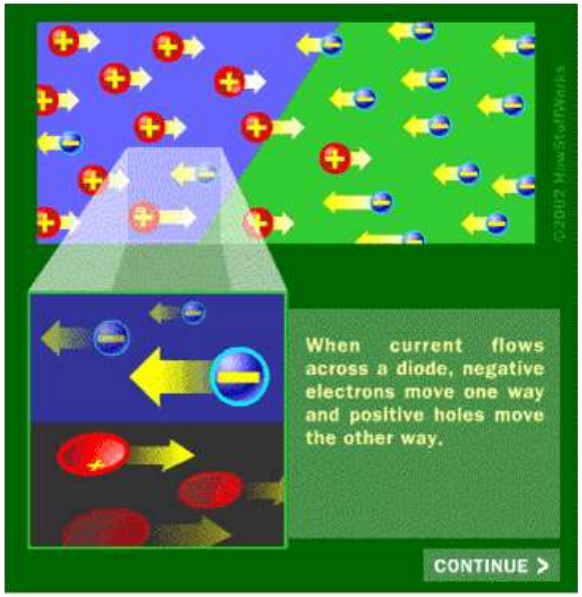
- **Dioda** je najjednostavniji poluprovodnički uređaj. Ona dozvoljava struji da se kreće samo u jednom smeru.
- Dioda se dobija se kada se spoji jedan poluprovodnik N i P-tipa.
- Ukoliko se dioda sa izvorom struje spoji kao na slici - nema provođenja struje jer se e^- i šupljine kreću u suprotnom smeru (šupljine privlači "-" a e^- "+").
- Ako se baterija okrene dioda će provoditi struju. Na spoju između N i P, susreću se e^- i šupljine (e^- popunjavaju šupljine) te oba entiteta prestaju da postoje, a njihovo mesto zauzimaju novi e^- i šupljine... Ovo se zove - STRUJA !!!
- Znači dioda služi da dopusti tok struje u jednom a da zabrani njen tok u drugom smeru.
- Čemu to služi? Npr. uređaji koji koriste baterije imaju diodu da spreče kvar ako obrnuto ubacite bateriju. Ispravljači naizmenične struje ...





LED (Light Emitting Diode):

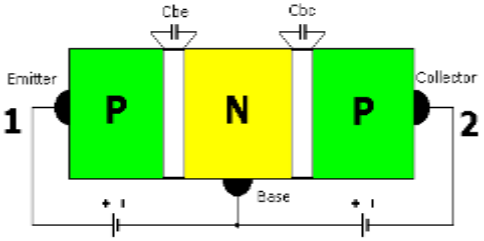
- Digitalni satovi, daljinski upravljači, jumbo TV ... LED ekrani!
- Ne pregorevaju, ne greju se puno jer svetle samo pomoću e^- prelaza u poluprovodniku.
- Za LED koristi se dopirani **AlGaAs** (alumunujum-galijum-arsenid)
- Kad e^- (iz N-oblast) upadne u šupljinu (P-oblast) prelazi iz provodne trake u oblast niže E, a razlika ΔE se oslobađa kao $h\nu$.
- U zavisnosti od tipa poluprovodnika nastali fotoni mogu biti u IC oblasti (Si dioda), ili da svetle u VIS oblasti kao **VLED** (visible light emitting diode) čija boja zavisi od ΔE provodne trake i nižih orbitala.
- Imaju voma veliko iskorišćenje jer se ne greju i struja se troši samo na svetljenje.
- Zbog skupoće materijala i procesa pravljanja i dalje koristimo sijalice.



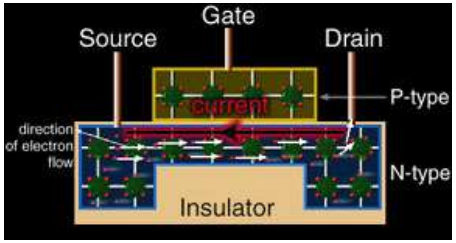
Princip rada LED

Tranzistori:

- Tranzistor se sastoji iz tri sloja (za razliku od diode koja ima samo 2).
- Mogu se formirati ili **NPN** ili **PNP** sendviči.
- Tranzistor može izigravati prekidač ili pojačavač. Izgleda kao dve obrnuto slepljene diode (bipolarni) ili **FET** (Field Effect Transistor).
- Kroz tranzistor ne bi zato trebala da teče struja (diode blokiraju struju u oba smera).
- Ipak, ako se pored spoljašnje struje, na središnji sloj dovede mala struja, kroz tranzistor može teći mnogo jača struja.
- Ovo od tranzistora čini pojačavač jer mala struja (u zavisnosti od struje koja se dovodi na bazu) može dati nekoliko stotina puta jaču struju.



Bipolarni tranzistori



FET tranzistori

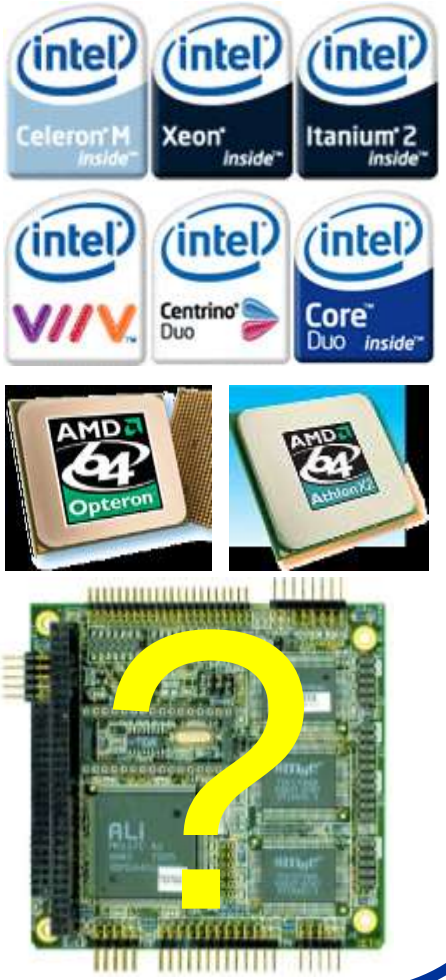
Čipovi:

- Silicijumski čip je pločica Si koja sadrži hiljade ili milione međusobno spojenih tranzistora.
- Brzina rada čipa zavisi od broja tranzistora na njemu, brzine svakog od tranzistora i kašnjenja pri prenosu el. impulsa kroz provodnik između dva tranzistora. Dokle mogu procesori da se ubrzavaju?
- Tipovi čipova:

  a) **Mikroprocesorski** (čip koji sadrži CPU i malu količinu memorije za specijalne namene tj. KEŠ). Ahitekture: Intel 80x86; (x=2,3,4); Pentium MMX, Celeron, Pentum Xeon, Pentium III, Pentium IV, Pentium V, Pentium iX, AMD... Do sada, u glavnom, 32-bitna arhitektura (po 32-bitnoj reči koju procesori obrađuju). Danas, 64-bitna kao i spajanje procesora sa 2, 4, 8 ili više jezgara.

  b) **Memorijski** (za pamćenje)

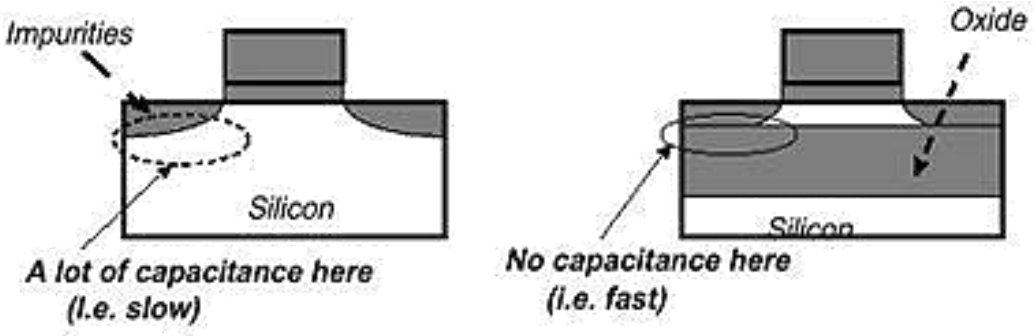
  c) **Logički** (za kontrolu rada magistrala, diskova...)



Čipovi – sastav i proizvodnja

1) Silicijum na izolatoru (SOI)

- SOI je način proizvodnje gde se sloj izolatora pravi na silicijumskoj osnovi izolujući gornji sloj silicijuma. Na taj način se aktivni tranzistori prave od ostatka silicijumske ploče (matrice).

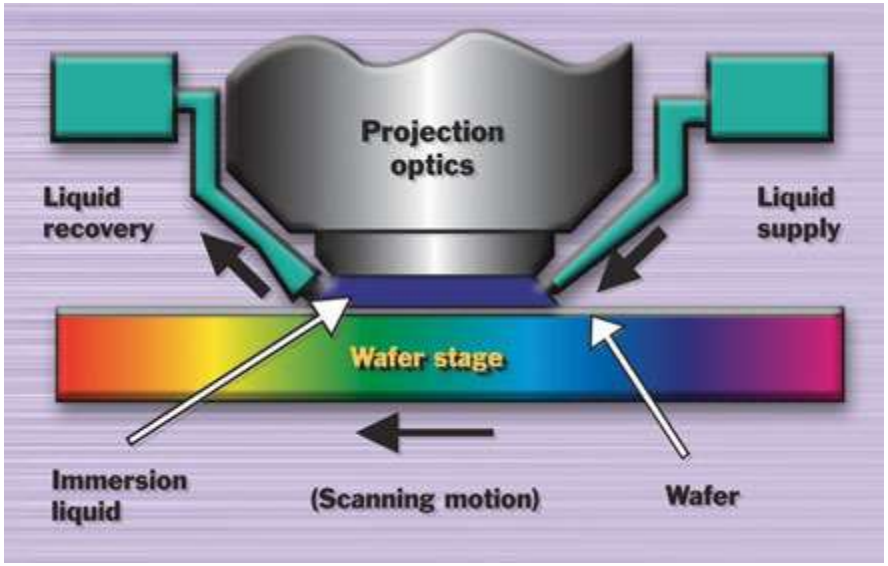


- Sloj oksida igra ulogu barijere koja sprečava „curenje“ struje iz tranzistora taklo da se na ovaj način dobijaju poluprovodnički uređaji koji su mali potrošači struje.
- Ovi čipovi se koriste za servere, radne stanice, laptop i desktop kompjutere, uređaje za bežičnu komunikaciju, integrisane optičke komponente itd.

Čipovi – sastav i proizvodnja

2) Imerziona litografija

- To je princip proizvodnje gde se stavlja sloj tečnosti između sočiva i osnove. To se radi zato da bi se postiglo dodatno prelamanje svetlosti dok prolazi kroz sloj tečnosti. Na ovaj način se povećeva rezolucija „štampanja“ procesorske sheme.
- Prema podacima koje daje IBM, imerzioni lito-proces može postići da se npr. pomoću svetlosti od 193 nm postiže kreiranje silicijumskih komponenti od 45 nm ili čak manje.



Čipovi – sastav i proizvodnja

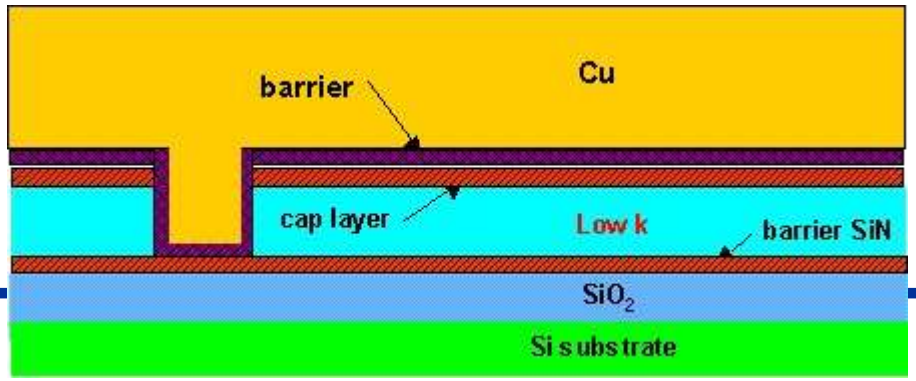
3) Bakar

- Bakar je kao materijal danas u širokoj upotrebi za kreiranje interkonektora. Ranije se koristio aluminijum.
- Naime, čipovi se na neki način moraju snabdeti strujom i to se radi putem bakarnih provodnika. S obzirom da je bakar dobar provodnik on dozvoljava snabdevanje procesora uz malu potrošnju stuje.
- Bakar se nanosi elektrodepozicijom iz rastvora $CuSO_4$ a uvek su u igri novi materijali koji bi bili jeftiniji ali i bolji prvodnici.

Čipovi – sastav i proizvodnja

4) Dielektrici sa niskom konstantom k

- Dielektrici koji imaju malu dielektričnu konstantu (k) u odnosu na silicijum-dioksid se koriste da bi formirali izolaciju između interkonektora u integrisanim kolima.
- Što je manja vrednost konstante „k“ to je izolacija bolja. Idealni izolator je vazduh koji ima konstantu k=1. Međutim, još uvek se ne prave kompleksna integrisana kola samo sa vazduhom između žica (tehnika koja se zove SON - „Silicon On Nothing“) jer je tehnika isuviše komplikovana za širu proizvodnju.
- Danas se istraživanja odvijaju u pravcu pronalaženja dielektričnih materijala koji su što porozniji (da sadrže maksimalnu količinu vazduha) ali da su dovoljno izdržlivi da bi podneli proces proizvodnje čipova.
- Većina dielektričnih materijala se postavljaju na čipove ili „spin-on“ procesom ili procesom naparavanja (CVD - „Chemical Vapor Deposition“).



Čipovi – sastav i proizvodnja

5) Napregnut (usmeren) germanijum

- Germanijum se kao dopant koristi u procesu proizvodnje čipova već duže vreme. Naime, ova tehnika koja se zove „prednapregnut silicijum“ (Strained Silicon). U ovoj tehnici mešavina silicijuma i germanijuma se stavlja na sloj čistog silicijuma pri čemu se silicijumovi atomi „rastežu“ tj. preuređuju duž kristalne rešetke germanijuma.
- Na ovaj način se prave „putevi“ u kristalnoj rešetki kroz koje je omogućen protok većeg broja elektrona kroz integrisano kolo.
- Dugo je već poznato da je germanijum bolji provodnik od silicijuma ali još uvek nije pronađen metod kako „ugraditi“ veću količinu germanijuma u čipove pomoću postojećih konvencionalnih tehnika.
- IBM puno ulaže na usavršavanju ove tehnike i već imaju dobrih rezultata u kreiranju „usmerenog germanijuma“ (Strained Germanium) i povećavanju performansi čipova.
- Germanijum, koji se inače dobija kao nus-proizvod u procesu proizvodnje u preradi rude cinka, je element koji ima istu kristalnu strukturu kao dijamant. On je po svojoj prirodi poluprovodnik.

Čipovi – sastav i proizvodnja

6) EUV tehnika (Extreme Ultra Violet)

- To je jednostavno tehnika gde se za litografiju koristi svetlost male talasne dužine (13nm). Ukoliko se naprave adekvatne „maske“ mogu se napraviti male komponente na čipu. Danas se radi na tzv. X-ray litografiji gde bi se koristile još manje talasne dužine.

7) Separacija implantiranjem kiseonika (SIMOX)

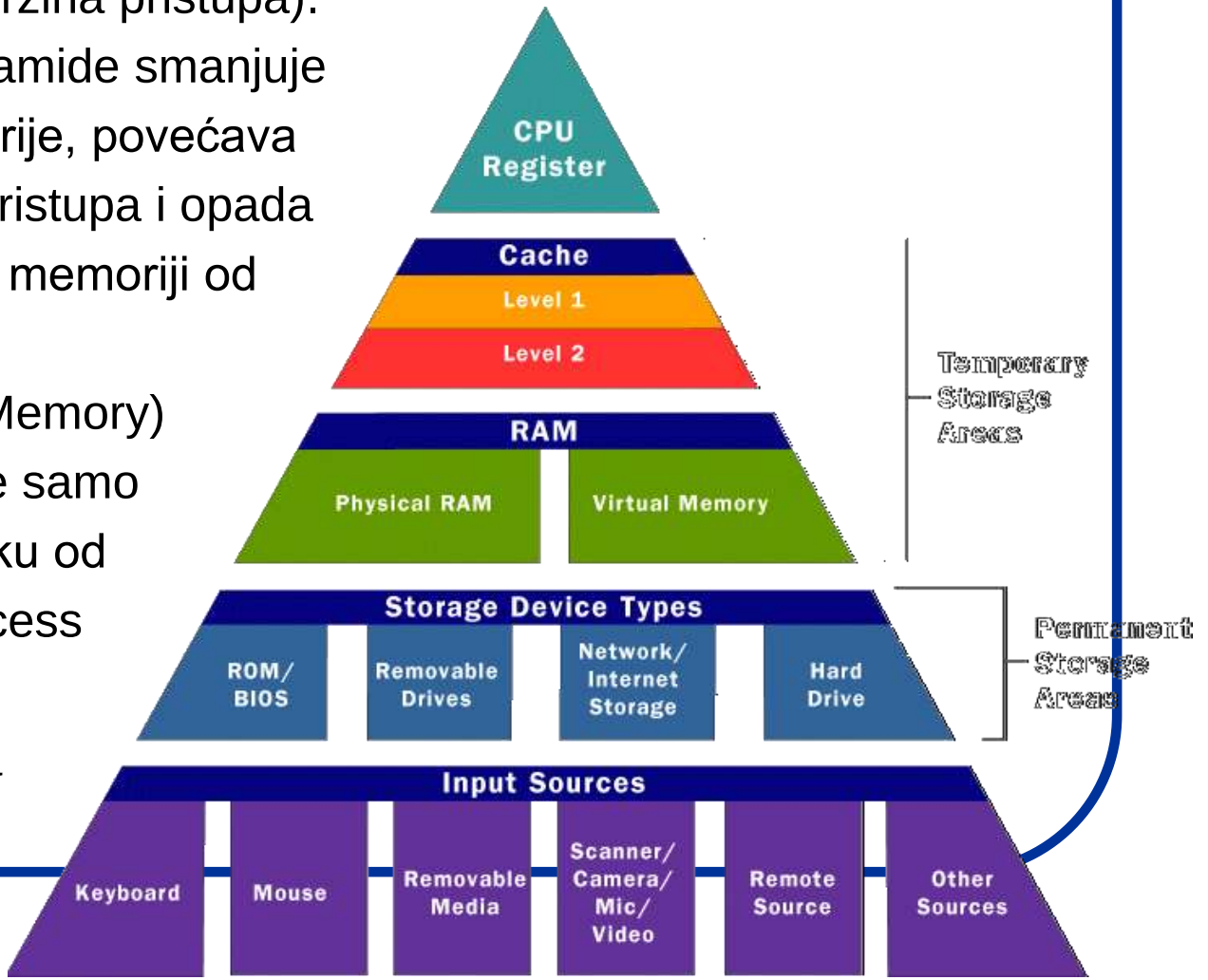
- *Separation by Implantation of Oxygen* (SIMOX) je radikalno nova tehnika proizvodnje čipova.
- Ova tehnika se zasniva na pravljenju savršeno glatkog sloja silicijum-oksida (debljine 0.15 mikrona). Ovaj tanki sloj silicijuma praktično nema nesavršenosti ili nečistoća.
- SIMOX proces proizvodnje uključuje direktno ubrizgavanje prečišćenog kiseonika u silicijumski matriks na ekstremno visokim temperaturama na kojima se se kiseonik vezuje za silicijum praveći tanak sloj (film) silcijum-oksida. Ovaj novoformirani sloj je idealno vezan za podlogu od čistog silicijuma.

Memorija računara:

- Deli se na **spoljašnju** i **unutrašnju** memoriju.
- **Unutrašnja** memorija sastoji se iz:
  - *glavne* memorije
  - *registra* u procesoru
  - *keš* memorije procesora
- **Spoljašnja** memorija nalazi se na različitim memorijskim uređajima: diskovi, trake, magnetno-optički diskovi, itd.
- Memorije možemo podeliti i **prema stalnosti zapisa** informacija koje čuvaju:
  - memorije sa **privremenim** zapisom (gube informacije po prestanku napajanja)
  - memorije sa **stalnim** zapisom (ne gube informaciju po nestanku struje)
- Memorije takođe možemo podeliti i **prema tehnologiji zapisa** (poluprovodnička, optička, magnetna).
- Memorije delimo i **prema kapacitetu** (količini informacija koju memorija može da sadrži). Izražava se u **bajtovima** ili rečima. Dužina reči zavisi od tipa procesora (najčešće dužine reči su 8,16,32 ili 64 **bita** tj. 1,2,4 ili 8 **bajta**).
- **1bajt=8 bitova** (binarnih cifara)

Hijerarhija memorija:

- Memorije u računarskom sistemu mogu se prikazati u obliku hijerarhije (cena i brzina pristupa).
- Od vrha ka dnu piramide smanjuje se cena bita memorije, povećava kapacitet i vreme pristupa i opada učestalost pristupa memoriji od strane CPU.
- **ROM** (Read Only Memory) je memorija koja se samo može čitati za razliku od **RAM** (Random Access Memory).
- Danas se ta razlika sve više gubi.



ROM memorija:

- **Glavna memorija** - pravi se od poluprovodnika i sastoji se iz *memorijskih ćelija* koje poseduju 2 stabilna stanja koja predstavljaju binarnu **0** i **1**.
- **Memorijske ćelije** - pakuju se u **čip** veličine 1, 4, 16, 32, 64, 128, 256, 512, 1G, 2G, 4G, 8G, 16G ... ćelija.
- Tipovi memorija:

  - **ROM** (**R**ead **O**nly **M**emory) je memorija čiji je sadržaj stalan i ne može da se menja (koristi se za čuvanje sistemskih programa).

  Prednosti: Brz pristup podacima

  Mane: Greška (kvar) traži zamenu celog čipa

Verzije ROM:

***PROM** (**P**rogarammable **ROM**) - ROM koji se isporučuje prazan (može se upisati samo jednom)*
***EPROM** (**E**rasable **PROM**) - to je PROM koji se može brisati pomoću **UV** zraka (sporo i glupavo)*
***EEPROM** (**E**lectically **EPROM**) - to je EPROM koji se može brisati strujom (brzo i skupo)*
***FLASH** memorija - slična je EEPROM memoriji, ali brisanje je duže (par sekundi, ali jeftino)*

RAM memorija:

- **RAM** (**R**andom **A**ccess **M**emory) je memorija koja se može proizvoljni broj puta upisivati i brisati.

  - pisanje i brisanje se vrši pomoću električnih signala
  - RAM je nestalna memorija (gubi svoj sadržaj po prestanku napajanja)

- Postoje 2 vrste RAM-a

  1. **S-RAM** (**S**tatički **RAM**) - koristi se za čuvanje podataka putem flip-flop kombinatorne mreže i ne mora se osvežavati u toku vremena (brža memorija ali ne preterano velika)

  2. **D-RAM** (**D**inamički **RAM**) - čuva podatke kao naelektrisanja u kondenzatorima koji se vremenom prazne te se mora periodično izvšti osvežavanje zapisa (sporija od SRAM ali gušći zapis dozvoljava veliku količinu memorije).

Postoje podvrste **DRAM** memorije u zavisnosti od tehnologija pravljenja:

FPM (Fast Page Mode); **EDO** (Enhanced Data Out), **BEDO** (Burst EDO), **CDRAM** (Cache DRAM), **SDRAM** (najčešće se koristi u PC-računarima kao PC-66, PC-100, PC-133), **ESDRAM** (Enhanced SDRAM), **DDR SDRAM** (Double Data Rate SDRAM - 2x brža od SDRAM: DDR1, DDR2, DDR3, DDR4), **SGRAM** (Synchronous Graphics RAM, za grafiku), **RDRAM** (Rambus DRAM), **SLDRAM** (Synchronous Link DRAM).

Keš memorija:

Keš memorija

- **Osnovna funkcija:** Povećanje performansi računarskog sistema.
- **Svrha:** Da premosti razlike u brzini između procesora i glavne memorije.
- **Kako to sve radi?** Keš memorija je znatno brža od glavne memorije. Ako procesor često zahteva neki podatak iz memorije, a on je privremeno smešten u keš, tada će brzina prenosa podataka biti znatno veća.
- **I zato u praksi ...** kupovati računare sa procesorom koji ima što više keša.
- Danas se keš memorija ne nalazi samo uz procesor već sa stavlja svuda (disk uređaji, razni kontroleri, grafičke kartice ...)

Bafer memorija

- Slična kešu samo što se u kešu nalaze kopije podataka koji se nalaze u glavnoj memoriji dok su baferu podaci originalno smešteni.

Spoljašnja memorija

kratko podsećanje

Spoljašnja memorija:

Šta je spoljašnja memorija ?

- **Sadrži podatke i programe koji se ne koriste aktivno u određenom trenutku.**
- **Sadržaj spoljašnje memorije je stalan i ne nestaje sa prestakom napajanja.**
- **Sporija je od unutrašnje memorije ali zato ima veći kapacitet.**
- **Tipovi spoljašnje memorije:**

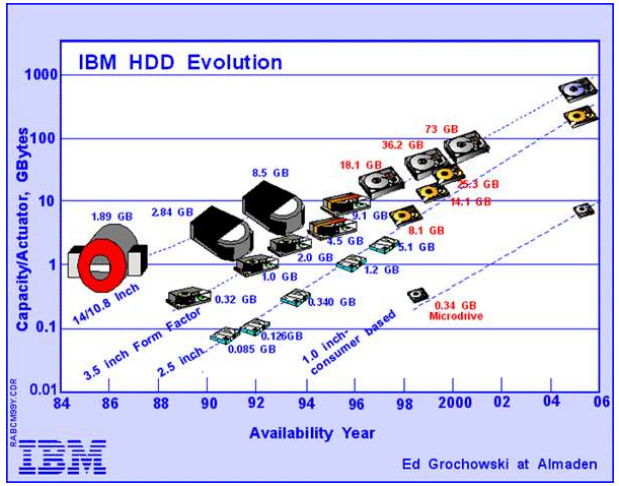
  - Magnetni diskovi (hard-diskovi)

  - Optički diskovi (CD-R, CD-RW, DVD-R, DVD-RW, HD-DVD, Blue-Ray...)

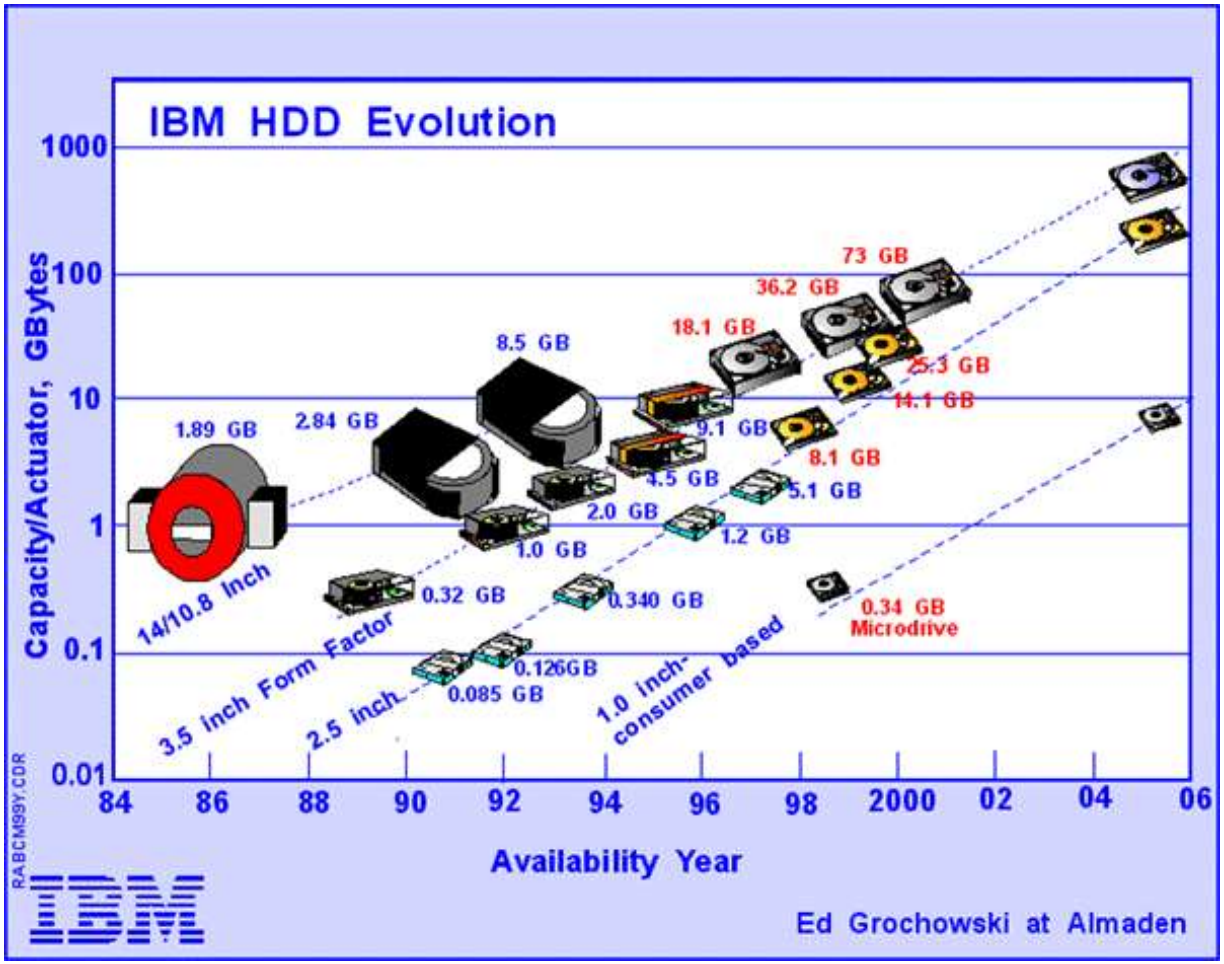
  - USB flash memorije, diskete, magnetne trake ...

Magnetni disk (hard-disk):

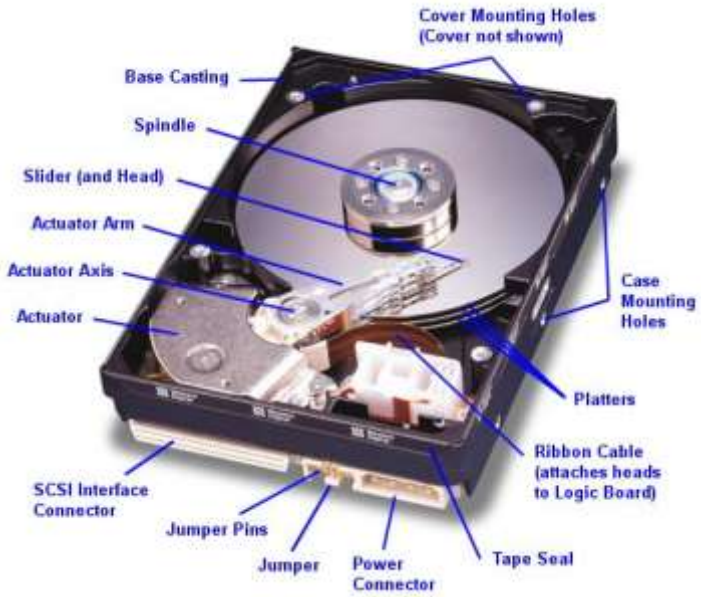
- Hard-disk se sastoji iz kružnih ploča veličina između 1.8 i 5.25 inča koje se prave od metala ili plastike i prevlače materijalom koji ima magnetna svojstva.
- Podaci se zapisuju i čitaju pomoću posebnog provodnika sa kalemom (upisno-čitajuća glava).
- Za vreme upisa glava je nepomična a rotira ploča ispod nje.
- Trend razvoja hard-diskova do danas.



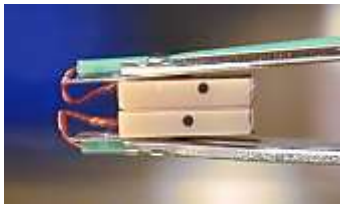
Magnetni disk (hard-disk):



Osnovni delovi hard-diska:



- Kućište diska
- Ploče (jedna ili više)
- Ruka čitača
- Upisno-čitajuća glava
- Osovina diska
- Mehanizam za pozicioniranje sa elektromotorom
- Konektori



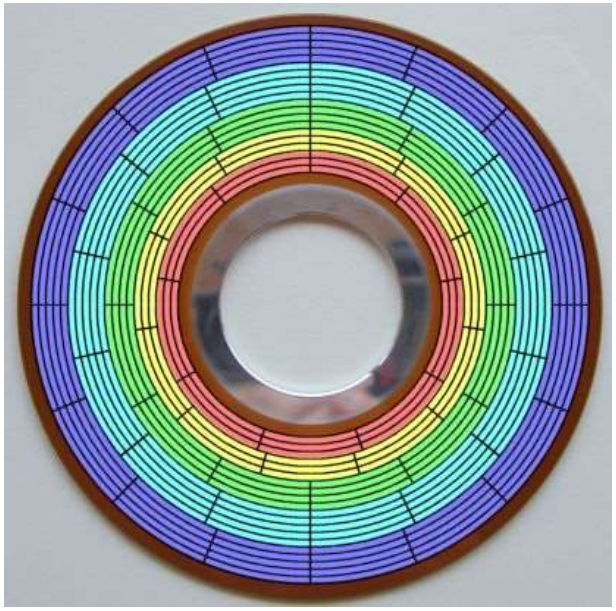
- Ovo su osnovni delovi svakog hard-diska bez obzira kojih je dimenzija

Stari IBM HDD

HDD za notebook

Micro HDD za iPod i DVD kamere

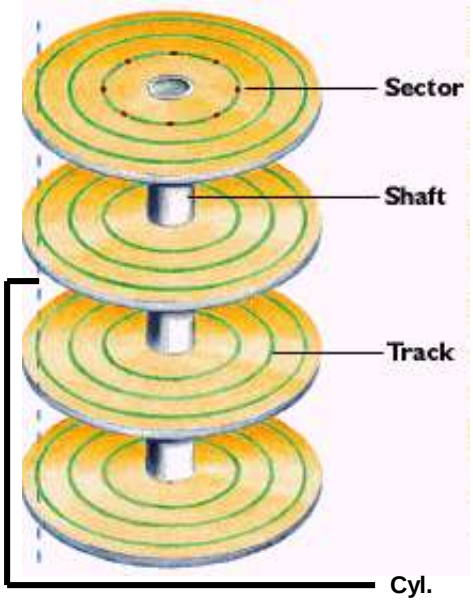
Struktura hard-diska:



U strukturi HDD ploče možemo razlikovati:

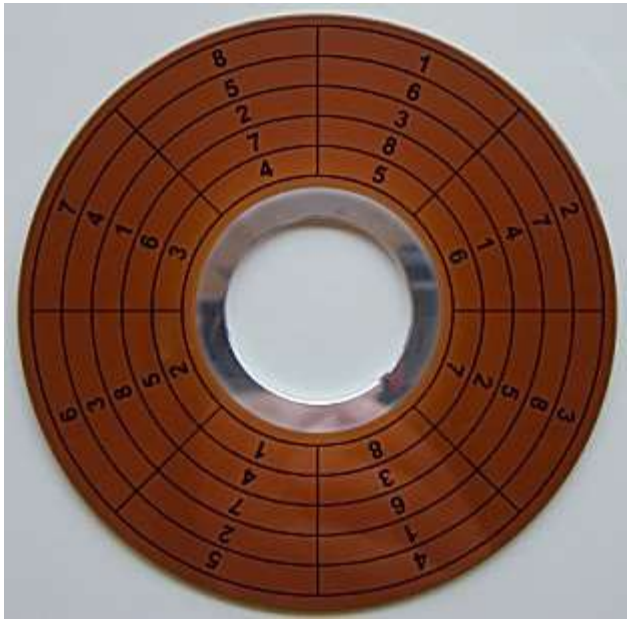
- **sektore**
- **staze (trake)**
- **cilindre**

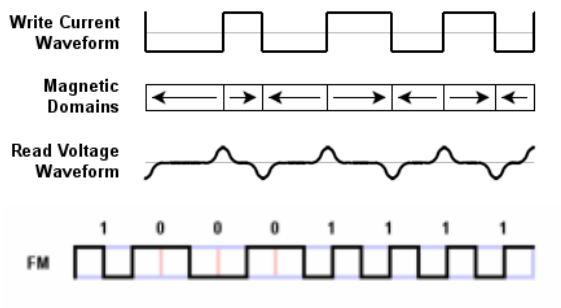
- Ovaj model diska ima 20 **traka**.
- Trake su podeljene na 5 zona različitih boja.
- Plava zona npr. ima 5 traka od kojih svaka ima 16 **sektora**, cian zona ima 5 traka sa 14 sektora
- Sektori mogu biti fiksne ili promenljive dužine.



- Dijagram desno ilustruje šta su to "**cilindri**".
- Označeni cilindar se sastoji npr. iz 8 traka (2 trake po jednoj ploči).
- Između staza je prazan prostor koji spečava interferenciju između magnetnih polja.
- Susedni sektori su razdvojeni prazninama koje se koriste kao oznake za početak i kraj sektora.

Način zapisivanja podataka:





• Podaci se upisuju u sektore u okviru staza (**sektorski metod**) ili po vertikali u okviru cilindra (**cilindrični metod**) pošto su čitači istovremeno iznad paralelnih sektora.

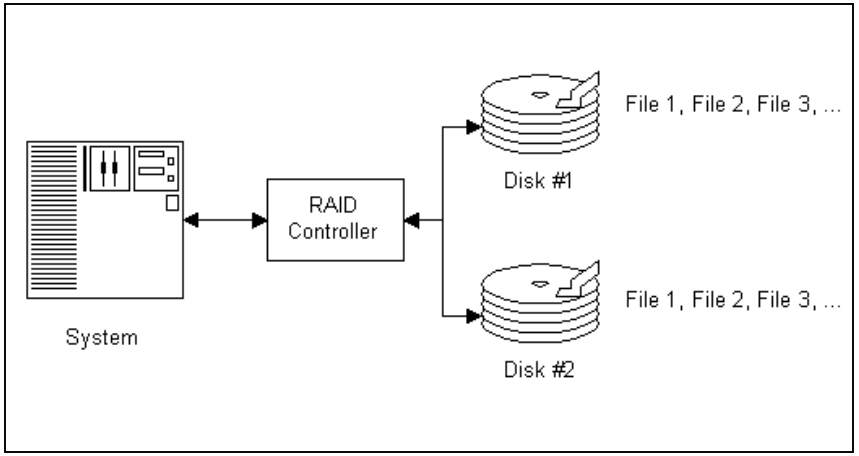
• Pre čuvanja podataka, ploča presvučena magnetnim materijalom mora biti podeljena u sektore procesom **formatiranja** niskog nivoa (low level formatting).

• Disk se može i **particionisati**, što znači da se odvajaju pojedine grupe cilindara koje operativni sistem može da posmatra kao pojedinačne diskove.

• Upis podataka na disk se vrši pomoću magnetnog polja koja formira struja u pisaču. Podaci se čitaju indukovanjem struje magnetnim poljem koje se pomera ispod čitača.

RAID tehnologija:

- **RAID** (**R**edundant **A**rray of **I**ndependent **D**isks) tehnologija napravljena je da podržava veliki broj diskova sa kontrolerskim čipom i ugrađenim specijalizovanim softverom.
- Koji je princip RAID tehnologije ? Umesto da smešta podatke na jedinicu diska jednim putem, RAID istovremeno razmešta podatke preko više paralelnih puteva i na taj način se postiže kraće vreme odziva.
- Neki računarski sistemi (npr. serveri) traže mnogo brži transfer podataka koji im današnji drajvovi mogu da pruže i to im obezbeđuje RAID-0 tehnologija komadanja podataka na više različitih diskova.
- Neki korisnici traže zaštitu podataka i to obezbedjuje RAID-1 tehnologija istovremenog zapisa podataka na dva diska (data mirroring).
- RAID-(2,3,4,5,6,7,10,53,1+0).



Blok dijagram RAID konfiguracije. RAID kontroler duplira istu informaciju na svaki od hard-diskova.

Solid-state hard diskovi:

- Korste flash memoriju (ROM) ili DRAM tj. SRAM memoriju (RAM).
- Prednosti:

  Nemaju pokretnih delova, brži, tiši, manje podložni kvaru, manje troše struju, manje se greju, podnose udarce.
- Mane:

  Visoka cena (10-15 $ po GB u odnosu na ispod 1 $ za mehaničke HD), za sada manji kapacitet - standardno 1.6 TB na 3,5" SSD, osetljiviji na nestanak stuje i uticaj spoljašnjeg magnetnog polja.

Tradicionalni 2.5" HD

V.S.

Solid-state HD

Optički diskovi:

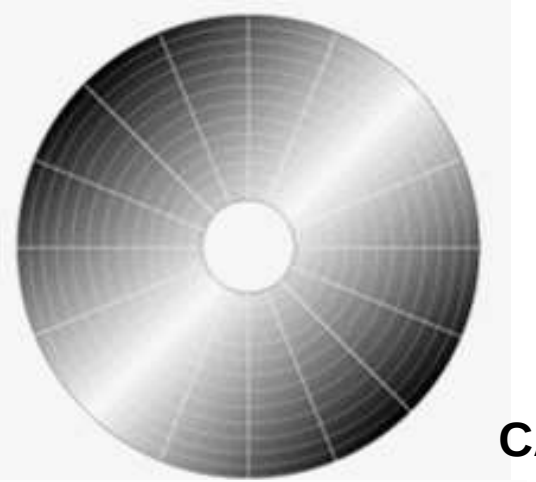
- Jedan od najviše korišćenih medijuma za arhiviranje podataka.
- Najviše koristi od tehnologije optičkih diskova imaju memorijski-zahtevne aplikacije (multimedija i obrada slika).
- Prvi **CD** (**C**ompact **D**isc) 1983.god (naravno, muzički :)
- Slično kao i kod magnetnih diskova, kod optičkih diskova podaci se čitaju tako što što ploča diska rotira ispod mehanizma za čitanje.
- Danas razni tipovi: CD/R/RW DVD+/-R/RW; DVD/DL; BlueRay; HD...

Mehanizmi zapisa:

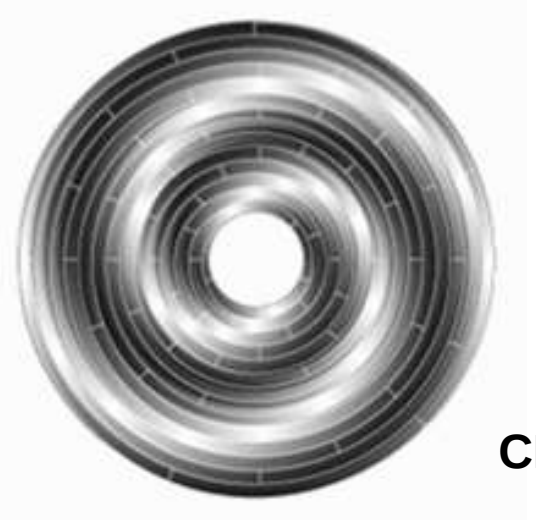
- Postoje 2 osnovna mehanizma za pisanje i čitanje podataka:

  1. **CAV** (**C**onstant **A**ngular **V**elocity) - disk se obrće konstantnom brzinom pa podaci po obodu diska prolaze većom brzinom nego podaci koji su bliže centru. Ova razlika se nadoknađuje povećanjem prostora među bitovima na delovima diska bližem obodu (nedostatak: neekonomično korišćenje prostora)

  2. **CLV** (**C**onstant **L**inear **V**elocity) - podaci se zapisuju na celom disku u segmente jednake veličine. Zbog toga, disk se rotira sporije kada se čitaju podaci bliže obodu a brže kada čita podatke koji su bliže centru diska. Ovako se čitanje obavlja konstantnom linearnom brzinom (često se umesto više koncentričnih, postavlja jedna spiralna staza).



CAV



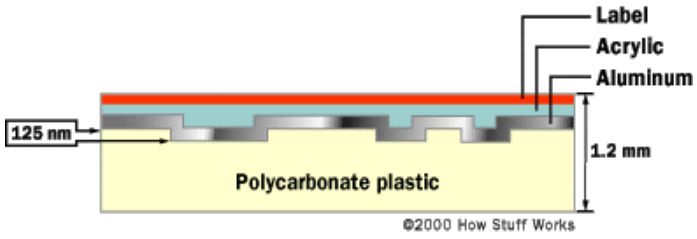
CLV

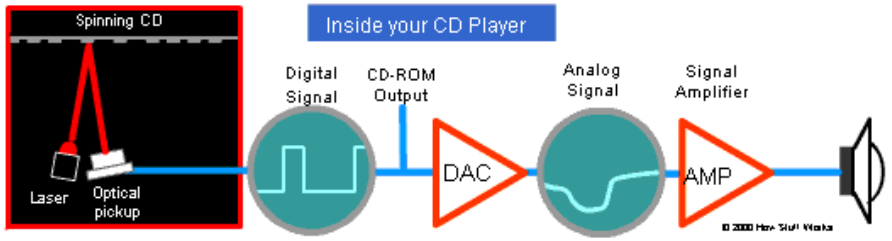
CD-ROM:

- **CD**-**ROM** (**C**ompact **D**isc-**R**ead **O**nly **M**emory)
- Obuhvataju **CD**-**R** (**CD**-**R**ecordable); **CD**-**RW** (**CD**-**R**e**W**ritable) ... diskove
- Postoje 2 načina zapisa:
  - za računarske podatke
  - za druge tipove podataka (audio, video ...)
- Osnovni formati za zapisivanje sistema datoteka su:
  - **ISO 9660** (malo zastareo, ograničen imenom datoteke do 31 karaktera)
  - **Joliet** (podržava dugačka imena datoteka za Win 95,98,NT,2000,XP,Vista,7)
  - **Rock Ridge** (poboljšani ISO 9660)
  - **UDF** (**U**niversal **D**igital **F**ormat) – rešen problem veličine fajlova
- Prečnik normalnog diska je standardno 120 mm a debljina 1.2 mm), ima manji
- Podaci se smeštaju sekvencijalno u sektroma veličine 2KB.
- Za zapis i čitanje podataka koristi se infracrveni laser od 780 nm.
- Kapacitet - 700 MB.
- Postoje i CD-DA (CD-DigitalAudio), CD-WO (CD-WriteOnly) ...

CD-ROM (tehnologija čitanja):

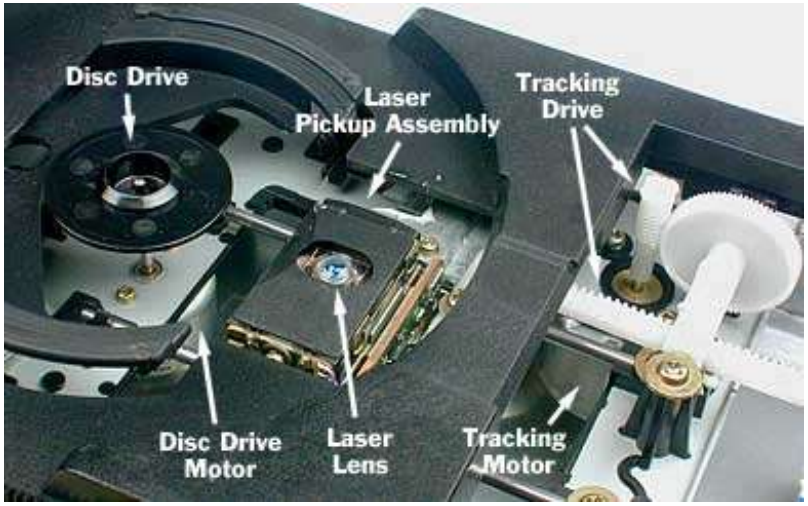
- Diskovi se prave od polikarbonata (plastika) u koji se u toku proizvodnje utiskuju rupice u obliku spirale od unutra ka spolja, zatim se preko stavlja tanka visoko-reflektivna površina (obično od aluminijuma) i zatim zaštini sloj. Ovako se fabrički prave diskovi (npr. muzički).
- Podaci se čitaju na sledeći način: laser se normalno reflektuje o Al sloj, ali kada naiđe na brdašce (koja je dubine oko 1/4-1/6 talasne dužine lasera),optički senzor detektuje promenu refleksije...

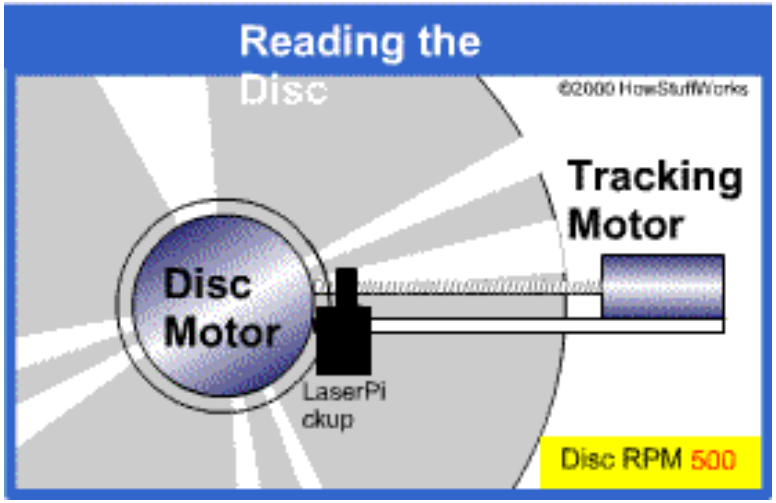




CD-ROM (tehnologija čitanja):

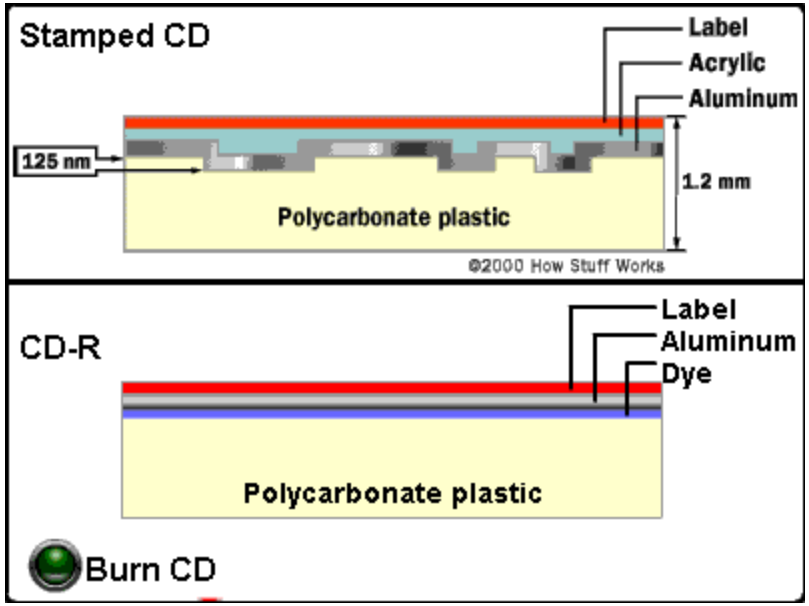
- Podaci se čitaju pomoću CD-drajva koji radi po sledećem principu:
  - 1 elektromotor vrti disk promenljivom brzinom
  - 2 elektromotor pomera laser od unutrašnjosti ka spolja
  - podaci se dekodiraju pomoću računara
- Motor CD-čitača usporava rotaciju diska kako laser ide ka obodu zbog pomenute konstantne brzine čitanja.





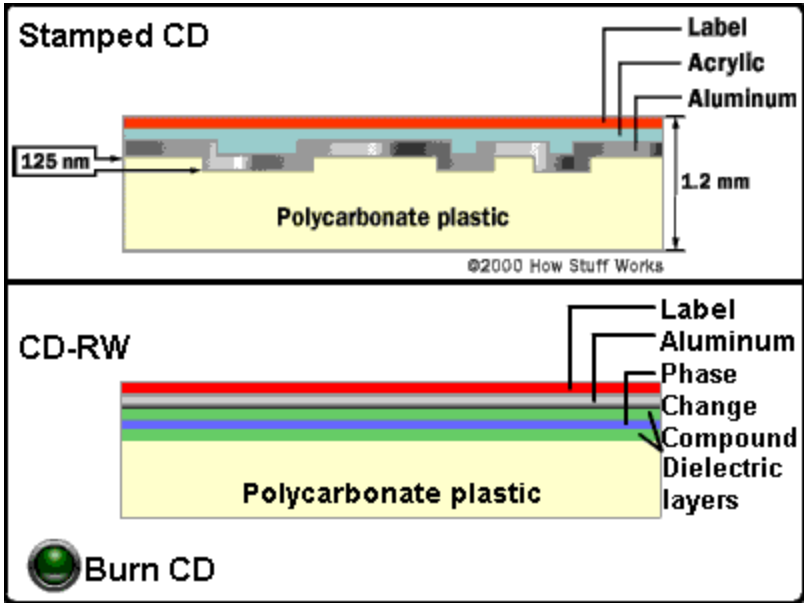
CD-R (tehnologija pisanja):

- Objašnjen princip upisivanja podataka nije praktičan za kućnu upotrebu, zato se koriste CD-R diskovi.
- CD-R diskovi nemaju brdašca kao "štampani" CD-ovi već su ravni:
- Međutim, iznad sloja plastike postoji sloj boje iznad koga je sloj Al, pa sloj zaštite.
- Kada se boja zagreje (u rezačima postoji pored slabijeg lasera za čitanje i "jači" laser koji služi za rezanje), ona potamni i svetlost ne može proći kroz nju.
- Ovako se (i pored toga što nema rupica) na CD-R može čitati (tamno je 0, a svetlo 1).



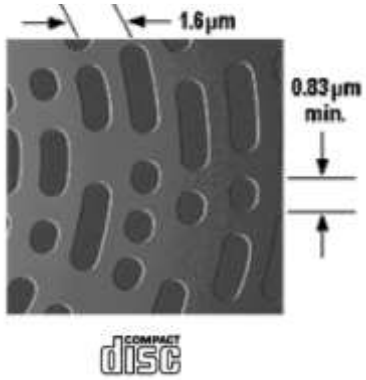
CD-RW (tehnologija pisanja):

- CD-RW diskovi se za razliku od CD-R diskova mogu brisati i na njih se podaci mogu upisivati više puta. Kako sad to radi ?
- Ovi diskovi bazirani su na tehnologiji promene faza. Element čija se faza menja je jedinjenje sastavljeno od srebra, antimona, telura i indijuma. Kada se ovo jedinjenje zagreje do 600°C postaje tečnost i ako se brzo ohladi ostaje amorfnog oblika (koji je netransparentan za svetlost čitajućeg lasera "0"). Ako se ovo jedinjenje zagreje samo do 200°C omekša, ali može potom brzo zauzeti kristalnu formu koja propušta svetlost "1".
- Ovde sada znači postoje 3 lasera:
  - jedan koji samo čita (read)
  - jedan koji topi do 600°C (write)
  - jedan koji topi do 200°C (format)
- CD-RW diskovi se isporučuju u kristalnoj formi, spremni za upis.
- Problem: CD-RW ne reflektuju dovoljno svetlosti, te ih stari CD-R čitači ne mogu prepoznati.

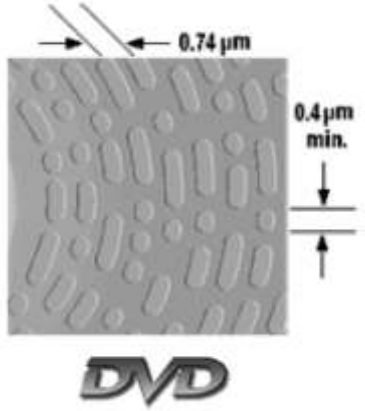


DVD-diskovi:

- **DVD** (**D**igital **V**ideo **D**isc, **D**igital **V**ersatile (svestran) **D**isc) su optički diskovi većeg kapaciteta, ali fizički iste veličine kao CD-ROM diskovi.
- Kapacitet 4.7 GB po jednom nivou zapisa na disku.
- Princip rada ovih diskova je isti kao kod CD-ROM diskova ali se ovde umesto lasera od 780 nm koristi laser od 650 nm.
- Zbog ovog, gustina zapisa je veća. Ali to nije sve...



(laser od 780 nm)



(laser od 650 nm)

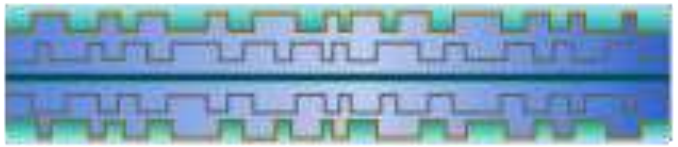
DVD-diskovi (princip rada):

- Sastav kao kod CD-ROM (plastika, rupice, Al).
- Međutim, postoje i višeslojni diskovi koji imaju unutrašnji refleksioni sloj od Al ali su spoljašnji slojevi od semi-transparentnog Au.
- Skupo.
- Isti je princip spiralnog čitanja od unutra ka spolja.
- Dužina trake je 12 km na single-layer DVD disku...
- Ništa novo, samo sitnije.

Single-sided, single layer (4.7GB)

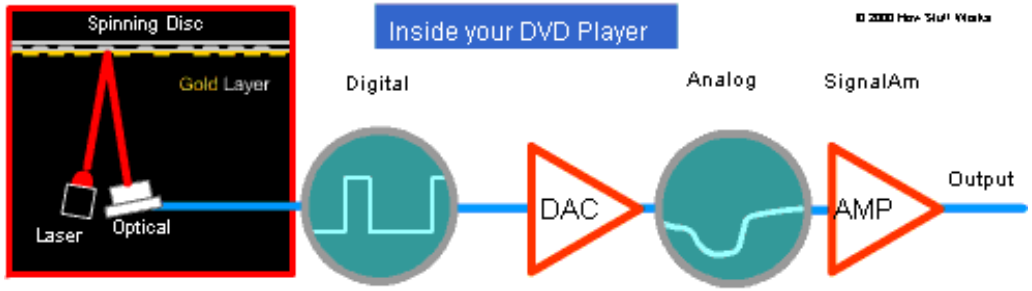
Single-sided, double layer (8.5GB)

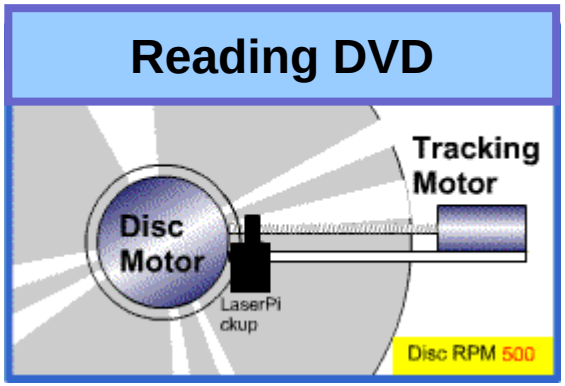
Double-sided, double layer (17GB)



DVD-diskovi (princip rada):

- Princip čitanja je kao kod CD-ROM diskova osim i slučaju kada se čitaju double-layer diskovi.
- Zapis na drugom sloju ne počinje sa unutrašnje već ide od spoljašnje strane ka unutra (laser samo na kraju prvog sloja promeni fokus i nastavi da čita). Na ovaj način skraćuje se vreme pristupa podacima.
- I ovde postoji optimizacija brzine rotacije diska zbog potrebe za konstantnom brzinom čitanja





DVD/R+ i DVD/R- diskovi:

- Koja je razlika između DVD+R/RW i DVD-R/RW diskova ?
- U tehničkom smislu, jedina je razlika što DVD + diskovi dozvoljavaju višeslojni zapis podataka dok DVD - dozvoljava samo jedan sloj (mada se danas redovno sreću DVD + diskovi sa jednoslojnim zapisom).
- Suština je u bivšem ratu između ova dva formata:

  - **DVD-R/RW** je razvijen od firme *Pioneer* i bazira se na CD-R tehnologiji (koristi heliksni zapis i kontrolu brzine rotacje diska). Ovaj format bio je podržavan od strane tzv. DVD-Foruma tj. određenih firmi za proizvodnju hardvera i softvera.

  - **DVD+R/RW** je takođe baziran na CD-R tehnologiji i podržavaju ga *Sony, Philips, HP, Dell, Yamaha i Microsoft*.
- Pošto su počeli da se komercijano prave CD-pisači i čitači koji podržavaju oba formata rat je utihnuo i sada je svejedno koji format se koristi. Drugačiji hemijski postupak i transparentnost svetlih i tamnih polja koji imaju ova dva formata danas nije problem za većinu drajvova.

HD (High Density) diskovi:

- Pojava novih diskova koji mogu čuvati velike količine podataka. To su: HD-DVD i Blue-ray (BD) diskovi.
- Trenutno na tržištu vidimo posledice rata između ova dva formata a potrošači su kolateralana šteta.
- **Blue-ray** koristi plavo-ljubičasti laser od 405 nm i single-layer diskovi imaju kapacitet od 25 GB (double-layer 50 GB).
- **HD-DVD** takođe koristi laser od 405 nm ali mu je kapacitet znatno manji (15 GB single-layer i 30 GB double-layer).
- Suština je zaštita od kopiranja HD filmova i igrica za konzole.
- Postoje i THD (Total Hi Def) diskovi koji su sa jedne strane Blue-ray a sa druge HD-DVD

SONY

Blu-ray Disc

VS

HD DVD

TOSHIBA

Diskete :)

- Zastareo princip zapisivanja podataka koji se od 1970. do danas relativno uspešno koristi (diskete od 8¨, 5,25" pa sve do 3,5").
- **FD** (**F**loppy **D**isc) ili savitljiva disketa, koristi magnetni zapis podataka (plastika prekrivena magnetnim materijalom).
- Podaci se beleže pomoću sektorske metode zapisa. Na disketi postoji 40 do 80 staza koje se dele u sektore. Podela staza u sektore vrši se:

  - na IBM PC računarima tako da svaka staza ima isti broj sektora tako da su podaci gušće spakovani u stazama sa manjim poluprečnikom.

  - na Apple Macintosh računarima svaka staza ima konstantnu gustinu zapisa tako da staze sa većim poluprečnikom imaju veći broj sektora.
- Postoje **SS**,**DS** (**S**ingle i **D**ouble **S**ide) kao i **SD**,**DD**,**HD** i **QD** (**S**ingle, **D**ouble, **H**igh i **Q**uad **D**ensity) diskete.
- Postoje i **ZIP** diskete (kao disketa od 3,5" ali deblja i većeg kapaciteta od 100-750 MB). Sporo i nepouzdano.
- Kapaciteti različitih disketa:

	DS/DD	DS/HD	DS/QD
5.25"	360K	1.2MB	2.4MB
3.5"	720K	1.44MB	2.88MB

Ostali oblici spoljašnje memorije:

- **BM** (**B**ubble **M**emory) uređaji (za zapis podataka koriste se magnetne površine u obliku mehurova na poluprovodničkom čipu). Ima mehurić "1", nema ga "0". Za razliku od RAM čipova ne gubi sadržaj po prestnku struje. Otporna na T°, prljavštinu i udarce (koristi se u vojnoj industriji, za robote, skupa).
- **PCMCIA** kartični memorijski uređaji (**P**ersonal **C**omputer **M**emory **C**ard **I**nternational **A**ssociation). Koriste se u Notebook računarima za povezivanje sa različitim tipovima eksternih memorija.
- **SC** kartice (**S**mart **C**ards) - sadrže mikroprocesor i memorijski čip na površini veličine kreditne kartice. Kartica pamti do 100KB podataka. To su razne kreditne, telefonske ... kartice.
- **USB** fleš memorije (**U**niversal **S**erial **B**us). Princip rada je već objašnjen. Ovakav tip memorijskih uređaja sada se sve više koristi zbog jednostavnosti povezivanja sa računarom preko USB ulaza.
- **Memorijska dugmad** (liče na baterije za sat) - sadrže EEPROM mikročip (osnovna memorija u mobilnim telefonima).
- **Memorijske kocke** - liče na staklenu kocku i imaju 3D optički zapis (6.5 TB). Jedan laser piše a drugi briše. Eksp. faza.

Ulazno/izlazni uređaji:

- Predstavljaju "vezu" okoline sa računarom, bez njih bi računar bio beskoristan.
- Ovi uređaji prikupljaju informacije iz okoline i pretvaraju ih u oblik koji računar "razume". Takođe, preuzimaju podatka sa računara i prikazuju ih u obliku upotrebljivom za ljude.
- Primeri: tastatura, miš, modem, mikrofon, *touchpad*, mrežna kartica, muzička kartica, monitor, štampač, skener, ... razne A/D i D/A kartice ...
- Postoji veliki broj ovakvih uređaja i upoznavanje sa osnivnim principom rada svakog od njih znatno prevazilazi okvire ovog kursa.
- Svi oni se priključuju na računar pomoću **U/I** (**U**lazno/**I**zlaznog) modula tj. interfejsa koji komunicira direktno sa procesorom i obrnuto (procesor preko njega obavlja komunikaciju sa perifernim uređajima).

U/I moduli:

- Direktno su priključeni za sistemsku magistralu i obezbeđuju razmenu informacija sa **U/I** uređajima i kontrolu njihovog rada na način koji najmanje utiče na performanse rač. sistema (U/I uređaji su sporiji od procesora i memorije).
- Osnovne funkcije U/I modula su:

  - kontrola i usklađivanje saobraćaja (između periferija i računara)

  - komunikacija sa procesorom (slanje signala i podataka preko magistrala)

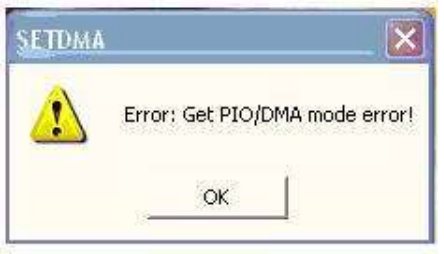
  - komunikacija sa uređajima (isto to samo sa uređajima)

  - prihvatanje i baferisanje podataka (procesor i memorija podatke preuzimaju isključivo iz U/I bafera).

  - otkrivanje grešaka (greške u prenosu podataka, stanju uređaja npr. nema papira u štampaču ...).
- Tačka priključenja perifernog uređaja sa računarom zove se **U/I** tj. **I/O** (**I**nternal/**O**uternal) **port** koji se najčešće sastoji od 4 registra (status, kontrola, data-in i data-out).
- Da bi komunikacja sa procesorom bila moguća, potrebni su komunikacioni programi koji će procesor izvršavati. To se zovu **drajveri**.

DMA (Direct Memory Access):

- Problem sa izvršavanjem U/I operacija je što one zahtevaju intervenciju CPU pri prenosu podataka između memorije i U/I modula.
- Prilikom izvršavanja svake U/I operacije ostali programi (koji ne koriste U/I uređaje) moraju da čekaju što je poseban nedostatak kada dolazi do prenosa veće količine podataka (zantno se usporava rad računara).
- U ovim slučajevima efikasnije je primeniti tehniku prenosa podataka nazvanu **D**irect **M**emory **A**ccess (**DMA**).
- DMA zahteva dodatni modul priključen na sistemsku magistralu koji se naziva DMA kontroler (to je u stvari specijalizovani procesor koji može da izvršava programe iz U/I modula).
- Nekad ima problema (korišćenje istih magistrala).



BIOS računara:

- Nakon što je računarska konfiguracija uspešno sastavljena, računar i dalje neće moći da funkcioniše (boot error).
- Jedina funkcija kojoj sada uspešno može da se pristupi je **BIOS** računara.
- **BIOS** (**B**asic **I**ntegrated **O**perating **S**ystem) je program ugrađen u čip koji prepoznaje i kontroliše različite komponente računarske konfiguracije (hard-disk, flopi, CD, memoriju ..)
- Zadatak BIOS-a je da pripremi i upozna računar sa osnovnim periferijama koje su mu na raspolaganju.
- BIOS je specifičan za različite proizvođače računara (prvenstveno matičnih ploča) i može se uspešno menjati sa novijim verzijama (BIOS update).
- BIOS-u se obično pristupa pritiskom na taster Delete. Greška u BIOS-u .....pravac majstor.

Gnothi Seauton

Formatiranje hard-diska:

- Formatiranje hard diska predstavlja pripremu hard-diska da bi na njega mogli da se upisuju novi podaci, tj. pravljanje praznog fajl-sistema. Postoje dva osnovna tipa formatiranja:

  - ***Low level formatting*** (predstavlja formatiranje ploča i instalaciju osnovnih karakteristika diska kao npr. broja sektora). Ovaj vid formatiranja se već obavlja u fabrici tako da ovakvo formatiranje najčešće nije ni potrebno. Povratak podataka sa diska nakon ovakvog vida formatiranja diska nije više moguć.

  - ***High level formatting*** (predstavlja pripremu diska za upis podataka prema specifičnim sistemima fajlova koji su karakteristika određenog operativnog sistema. Ovaj proces uključuje i formatiranje *boot sectora* na hard disku. *Boot sector* predstavlja mesto na hard-disku na kome je smešten jednostavan program koji obavlja inicijalizaciju operativnog sistema i bez njega OS ne može početi svoj rad. Povratak podataka posle ovakvog vida formatiranja je moguć ukoliko preko njih nisu upisivani novi podaci.

Particionisanje hard-diska:

- Predstavlja kreiranje logički-odvojenih celina na jednom hard-disku koje računar posmatra kao posebne disk-jedinice.
- U DOS i Windows operativnom sistemu postoje dva osnovna tipa particija:

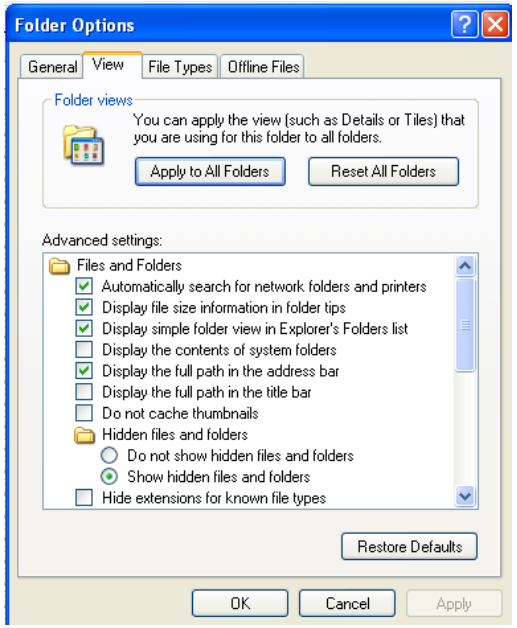
  - ***Primary partition*** (je particija na hard-disku koja se ponaša kao da je poseban disk). Primarnih particija može biti najviše četiri od kojih samo jedna može biti aktivna. Primarna particija može (a ne mora) sadržati OS i obično je to disk označen slovom C:

  - ***Extended partition*** (je particija u okviru jedne primarne particije koja nije aktivna i koja ne sadrži OS. Može biti podeljena u više logičkih particija npr: D: E: F: ...).
- **Primer**: **Disk koji ima C: D: E: i F: particije (i Windows OS):**

  Korste se dve primarne particije. Prva je aktivna DOS primarna particija i to je C: disk na kome se nalazi OS. Druga je *extended* DOS particija koja je podeljena na tri logičke particije D: E: i F:
- DOS će prepoznavati samo aktivnu primarnu particiju i to treba imati u vidu prilkom rada sa ovim OS.

Organizacija fajlova:

- Fajlovi su organizovani po sledećem principu:
  - Particija (Partition)
  - Folderi (Folders)
  - Fajlovi (Files)
- Primer:
  C:\WINDOWS\file.txt
- Treba znati:
  - **" \"** je backslash (koristi ga u glavnom Windows)
  - **"/"** je slash (koristi ga Linux i Internet)
- Fajlovi i foderi mogu biti:
  - hidden (nevidljivi)
  - vidljivi
- Setovanje:
  Tools - Folder Options
- Sitemske fajlove najbolje je ostaviti sakrivene



Sistemi fajlova (FAT):

- ***File systems*** (sistemi fajlova) su metoda čuvanja i organizovanja kompjuterskih fajlova (tj. podataka koji oni sadrže) i čine manjom ili većom njihovu raspoloživost korisniku računara.
- U okviru različitih sistema organizacije fajlova postoji tačno definisana hijerarhija važnosti, dostupnosti i manipulacije različitim podacima, i u zavisnosti od potrebe korisnik se opredeljuje za željeni sistem.
- Na ovom kursu biće razmatrani sistemi fajlova koji se mogu pojaviti u okviru Windows i LINUX okruženja, mada treba imati u vidu da je broj načina organizacije daleko veći (Mac OS, UNIX-like OS ...).
- *Microsoft* je po uzoru na svoj prvi OS (MS-DOS) razvio korišćenje **FAT** (**F**ile **A**llocation **T**able) i **NTFS** (**N**ew **T**echnology **F**ile **S**ystem) tehnologiju zapisivanja fajlova. Prethodne verzije FAT sistema (FAT12 i FAT16) imale su ograničenja koja su se ticala dužine broja karaktera u imenu, brojem unosa u osnovni direkorijum (**root**) i naročito u veličini diskova tj. particija na kojima su podaci bili čuvani (FAT12 i FAT16 su ograničavali broj karaktera u nazivu fajla na 8, i 3 za **ekstenziju** - tzv. 8.3 ograničenje).
- **FAT32** fajl sistem ukinuo je ovakav tip ograničenja ali je i dalje ostao limitiran u odnosu na NTFS fajl sistem.

Sistemi fajlova (NTFS):

- **NTFS** (**N**ew **T**echnology **F**ile **S**ystem) je uveden pojavom novog operativnog sistema Windows NT i postao je osnova kasnijih Windows operativnih sistema kao što su Windows 2000, XP, Server 2003, Vista i Windows 7.
- Ovaj fajl-sistem uvodi pojavu entiteta koji se označava kao ***metadata*** u kome se nalazi sve što ima veze sa osobinama fajla (ime, datum kreacije, dozvola o pristupnosti, čak i komentari).
- Ovakav način indeksiranja dozvoljava brži pristup podacima i čini sistem stabilnijim u toku rada.
- NTFS dodatno obezbeđuje sigurnost fajlovima primenom enkripcije (mogućnost skrivanja fajlova), mogućnost kompresije fajlova, foldera ili celih drajvova ukoliko je prostor problem, kao i povratak slučajno obrisanih podataka.
- Treba imati u vidu da FAT32 particije nemaju mogućnost da "vide" NTFS particije što obratno nije slučaj. Takođe, NTFS nije "savršen" i ima svojih nedostataka o kojima za sada neće biti reči.
- U principu je preporuka koristiti NTFS ukoliko ne postoji značajan razlog za korišćenjem FAT opcije (kao što je upotreba specifičnog aplikativnog softvera starije generacije koji ne funkcioniše pod NTFS sistemom).

Linux OS (organizacija):

- Linux je **besplatan** *Unix*-tip operativnog sistema koji je originalno napravljen od strane Linusa Torvaldsa uz asistenciju velikog broja programera širom sveta.
- "Source-code" za Linux OS je dostupan svakome (razlika od Windows-a). Neke od verzija Linuxa su: Debian, CentOS, Fedora Core, SUSE Linux, RedHat, Mandriva, Linspire, Xandros, MEPIS, Ubuntu, Knoppix ...
- Za razliku od Windows-a, Linux ne koristi slova da označi particije na hard-disku (npr. C:\). Ovde je primarna particija označena sa "**/**" dok postoji još mnoštvo particija kao što su "**/boot**" (sadrže kernel i boot loader), "**/home**" (sadrži korisničke fajlove), "**/var**" (sadrži programske konfiguracije)...
- Još jedna od važnih particija zove se "**swap**" particija. Njena veličina iznosi obično dvostruko od veličine memorije koju ima računar i predstavlja njenu dopunu u slučaju da je cela sistemska memorija u upotrebi.
- Instalacija aplikativnog softvera kao i ostale intervencije u okviru sistema pod Linuxom obavlja se iz "**Konzole**" (slično kao DOS shell). Njene komande možete upoznati sa: http://www.linuxcommand.org/index.php

Instalacija Linux OS:

- Kao primer instalacije Linux-a uzećemo instalaciju **SUSE 11.2** Linux OS, mada je postupak sličan i kod ostalix Linux OS.
- Prvo treba napomenuti da ukoliko želite da paralelno koristite dva OS (Windows i Linux) potrebno je prvo instalisati Windows OS a zatim formitati jednu particiju od najmanje 10 GB (za ozbiljne Linux OS) i formatirati je kao Linux particiju (mada to nije sasvim neophodno).
- Nakon toga, treba ubaciti Linux instalacioni CD i restartovati računar (podrazumeva se da je BIOS-u i dalje nameštено da je primarni boot sa CD-a).
- Kada se pojavi meni za instalaciju odabrati da se sve Linux particije formiraju na mestu koje je ostavljeno za Linux OS (onih 10 GB) i odbrati **grub** (boot loader) koji omogućava opciju za butovanje više različitih OS.
- Zatim izabrati jedan od dva moguća Linux okruženja (**GNOME** ili **KDE**), što je od sporedne važnosti, i uzeti ponuđene opcije u vezi sa organizacijom količine prostora za različite neophodne Linux particije. Ostalo je sve isto kao i u Windows XP instalaciji.

Linux vs. Windows:

Karakteristike	Windows	Linux	Comm.
Cena	45-450 $	0-350 $	jeftiniji L
Antivirusi	oko 100 $ godišnje	0 $	nema za L
Drajveri	uglavnom ima	mora se pomučiti	lakše za W
Sigurnost	puno se govori	probaju entuzijasti	serveri su L
Particije	može više OS	može više OS	L je bolji
Softver	kupuje se	open source	$$$
Stabilnost OS	od WIN 2000-bolje	stabilan	bitna stvar
Stabilnost aplik.	CTRL+ALT+DEL	CTRL + C ali veoma retko	šta reći
Podrška aplik.	puno apl.	puno apl.	stvar navike

Setovanje radnog okruženja:

- Pošto konačno funkcioniše svaki hardverski deo računara, vreme je da se pozabavimo sa setovanjem radnog okruženja pod Windowsom.
- Započeti stvaranjem svog korisničkog naloga u okviru kojeg će ostati svaka promena u radnom okruženju. Svaki korisnik svoje radno okruženje i podatke može zaštititi lozinkom (**password**).
- Setovanje počinje nameštanjem odgovarajućeg grafičkog okruženja (rezolucije ekrana i kvaliteta boja), multimedije, prečica i zaštite ekrana (ovde o tome neće biti reči već je to deo eksperimentalnih vežbi).
- Treba imati u vidu da je grafičko okruženje sporedna stvar koja treba da ima funkciju da ubrza korisnički rad sa aplikativnim softverom. Ukoliko je grafičko okruženje hardverski zahtevno (virtual ili 3D desktop) to može znatno usporiti rad računara jer procesor koristi isuviše svog vremena na sporedne stvari i može često doći da preopterećenja i **pada sistema**.
- Pad sistema predstavlja stanje kada program (ili ceo operativni sistem) prestane da obavlja svoju funkciju tj. prestane da komunicira sa drugim delovim sistema. Najčešće dolazi do "zamrzavanja" programa što može onemogućiti korisnika da sačuva svoj dotadašnja rad na računaru te su novi podaci bespovratno izgubljeni.
- Do ovoga dolazi usled nesavršenosti u komunikaciji između hardvera, OS i korisničkih programa. Korisnicima se zato preporučuje da prilikom rada sa računarom češće sačuvaju dotadašnji rad upisom na hard-disk ili neki vid eksterne memorije.

Instalacija aplikativnog softvera:

- Sledeći korak u pripremi računa ra za rad je instalacija aplikativnog softvera.
- **Instalacija (**ili **setup)** predstavlja proces implementacije programa u operativni sistem. Bez instalacije, operativni sistem nema informacije o svrsi određenog aplikativnog softvera i posmatra ga kao nezavisni niz fajlova te aplikativni softver neće moći da funkcioniše. Bez instalacije, sam aplikativni softver nema nikakvih podataka o računarskim resursima u konkretnom slučaju (jer su sve informacije o računaru zapisane u okviru OS).
- Većina programa se od proizvođača isporučuje u kondenzovanoj formi i da bi bili korišćeni moraju se prethodno raspakovati (o ovome će biti reči kasnije).
- Prilkom instalacije, operativni sistem pita korisnika gde želi da mu softver bude zapisan (standardno mesto je folder: C:\Program Files\...) ali i kontroliše kompatibilnost, hardversku i softversku zahtevnost datog programskog paketa.
- Aplikativni softver od OS dobija povratnu informaciju o stanju računara i periferijama koje su mu na raspolaganju, grafičkom okruženju, brzini procesora i raspoloživoj memoriji, kao i o već instalisanom softveru u koji se potencijano može implementirati (npr. **Chem Office** u okviru **Microsoft Office** paketa).
- Neki programi ne zahtevaju instalaciju (stariji, kopiraju se direktno na disk ali su sa njima česti problemi), dok neki zahtevaju kompajliranje (Linux). **Portable programi**.

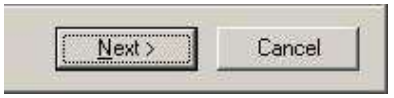
Postupak instalacije softvera:

- Primer instalacije aplikativnog softvera biće demonstriran u slučaju paketa Microsoft Office 2007 i Open Office (vežbe).
- Generalno, instalacija većine aplikativnog softvera zasniva se na istom principu:

  - na instalacionom CD-u ili Folderu u kome se nalazi program, pronaći fajl koji ima naziv **install.exe** (ili **setup.exe**)

  - pokrenuti izvršenje fajla i pratiti dalja uputstva (ukucati serijski broj i odrediti folder gde će program biti smešten ...)

  - instalacija je inače napravljena da na svako postavljeno pitanje u velikom broju slučajeva treba odgovoriti sa **Next >**.
- Bitno je zapamtiti da će se posle instalacije program moći pokrenuti iz Foldera u koji je program instalisan (mada se često koristi i prečica na destopu ili Start-Programs meni).
- Ukoliko više ne želite da koristite određeni softverski paket treba ga deinstalisati (Control Panel - Add or Remove Programs - Remove) da ne bi zauzimao mesta na disku.
- Prilikom instalacije programa treba izaći iz ostalih aplikacija da ne bi došlo do neočekivanih problema.



A kako se vrši instalacija softvera pod LINUX OS?

Korisni linkovi za Linux:

- Innstalacija programskih paketa se u Linuxu vrši iz konzole u tri koraka:
- 1) ***./ configure*** (proverava se da li je OS spreman za uspešno dodavanje novog programa)
  2) ***make*** (kompajlira program tj. prevodi ga iz izvornog koda u binarni)
  3) ***make install*** (uklapa kompajlirane datoteke na svoja mesta u OS)

- **Linux Software Encyclopedia**
  http://stommel.tamu.edu/~baum/linuxlist/linuxlist/linuxlist.html
- **Linux Software Map**
  http://www.boutell.com/lsm/lsmsubject.html
- **The table of analogs of Windows software in Linux**
  http://www.linuxrsp.ru/win-lin-soft/table-eng.html
- **Linux Software: Scientific Applications**
  http://linux.about.com/od/softscience/Linux_Software_Scientific_Applications.htm
- **Linux math software**
  http://www.usinglinux.org/math/
- **Linux games**
  http://www.linux-games.com/
- **Books about Linux**
  http://www.linux.org/docs/online_books.html
- **Chemistry and LINUX**
  http://www.redbrick.dcu.ie/~noel/linux4chemistry/

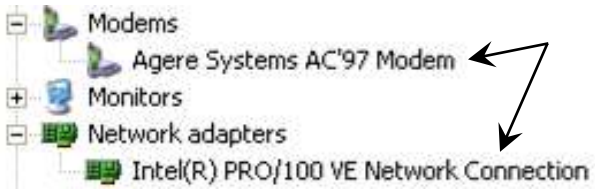
Mreža i mrežni parametri

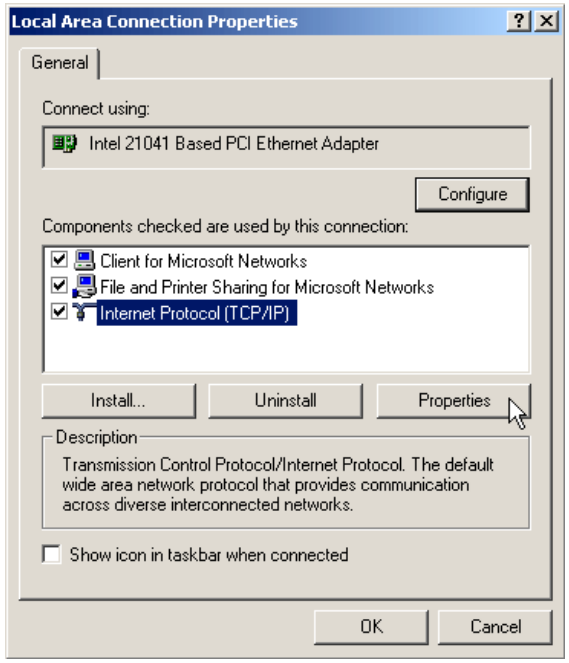
eduroam

KOBSON

TCP/IP protokoli:

- Računar Internet najčešće pristupa preko **mrežne karte** (nekada modema). Ukoliko nisu instalisani drajveri za njih, oni se moraju naći na Internetu pomoću nekog drugog računara koji je na njega već priključen.
- Ukoliko drajveri postoje, potrebno je pristupiti setovanju Internet protokola (**TCP/IP**) **T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol. Da objasnimo:
- Svaki računar koji je priključen na Internet poseduje jedinstvenu **IP adresu** (statičku ili dinamičku). IP adresa je 32-bitni ceo broj koji se radi lakše obrade zapisuje kao niz od četiri celobrojne vrednosti u intervalu od 0 od 255 koje su razdvojene tačkama.
- Prema IP adresi, koju je računar imao u datom vremenskom trenutku pristupanja mreži, može se lako detektovati sa kog računara je vršena mrežna komunikacija.
- Ukoliko je IP adresa računara npr: 147.91.71.53 može se za određeni računar zaključiti sledeće:





IP adresa 147.91.71.53:

- 147 (adresa **A klase** koja se dodeljuje zemljama) - znači da se računar nalazi u Srbiji.
- 91 (adresa **B klase** koja se dodeljuje firmama) - znači da je računar priključen na univerzitetsku mrežu.
- 71 (adresa **C klase** koja se dodeljuje manjim organizacijama) - znači da se računar nalazi na Fakultetu za Fizičku Hemiju.
- 53 (adresa **D klase** koja se dodeljuje konkretnom koriskinu) - znači da je to baš vaš računar.
- U poslednje vreme, zbog velikog broja računara koji su priključeni na Internet, javio se nedostatak dovoljnog broja IP adresa zbog čega je počela distribicija šestodelnih adresa (odnosno migracija na 48-bitne cele brojeve).
- **M**edia **A**ccess **C**ontrol address (**MAC** adresa) is je jedinstven broj pripisan mrežnom interfejsu i dodeljuje ga proizvođeč mrežne **N**etwork **I**nterface **C**ard (tj. **NIC**) karte. Zove se i "burned in adress" jer je locirana u ROM-u mrežne karte.
- S obzirom da je potreban efikasan način komunikacije između razlličitih IP adresa, napravljen je softver koji se zove **DNS** (**D**omain **N**ame **S**ystem) koji svaka lokalna mreža ima da bi imala distribuiranu bazu podataka za preslikavanje imena domena u **IP** adrese.

DHCP: 192.168.1.XXX

• Dynamic Host Configuration Protocol (DHCP) je mrežni protokol koji omogućava serveru da automatski dodeli IP adresu nekom računaru (iz definisanog opsega brojeva) konfigurisanih datom mrežom.

• DHCP dodeljuje IP adresu kada se sistem startuje
• Generalno, ovaj proces teče ovako:
- Računar po startovanju pokuša da se poveže na internet
- Mreža zahteva IP adresu
- DHCP server **privremeno** dodeljuje IP adresu za nov mrežni uređaj koja se prosleđuje ma mrezu pomoću rutera.
- DHCP pokreće odgovarajuće mrežne servise postojećom IP adresom i drugim konfiguracionim parametrima.
- Mrežni servis prihvata IP adresu

Figure 1: Standard DHCP provides IP addresses to all machines

• Kada se računar isključi:
- Privremeno dodeljenoj IP adresi ističe rok.
- DHCP može dodeliti istu IP adresu novom klijentu.

• Da bi se setovao DHCP potreban je: DHCP klijent, ruter i DHCP server.

Ukratko o Internetu:

- Internet je najveća globalna računarska mreža koja povezuje stotine hiljada individualnih računarskih mreža širom sveta.
- Razvio se iz naučno-istraživačke mreže istraživačkih centara pod okriljem ministarstva odbrane USA (Arpa-net).
- Kako pristupiti Internetu?

  Internetu pristupamo pojedinačno ili preko neke institucije. Pojedinačni pristup je moguć preko davalaca usluga (provajdera). Institucije pored provajdera mogu koristiti i direktan pristup.
- Internet je zasnovan na klijent-server tehnologiji. Svi podaci (baze podataka, elektronske pošte i Web stranice) nalaze se na serverima koje grubo delimo na **servere za komunikaciju na mreži** i **servere za pretraživanje informacija**.
- Postoji veliki broj alata i servisa koji omogućuju komunikaciju na mreži:

  - elektronska pošta: **ime osobe@ime domena**

  - diskusione grupe (**newsgroups**)

  - chat (najčešće preko **IRC** tj. **I**nterner **R**elay **C**hat servera)

  - **Listserv** (razmena informacija preko grupa) i **Telnet** (rad na udaljenom računaru).

Parametari za pristup Internetu:

- Da bi korisnik mogao da se "priključi" na Internet nije dovoljno samo imati IP adresu i broj DNS servera (upisuje ih sam korisnik ili se automatski dodeljuju), već je nekada potrebno namestiti i **Proxy** konfiguraciju.
- **Proxy server** je kompjuter koji omogućava klijentima, korisnicima njegovog mrežnog servisa, da prave indirektnu mrežnu komunikaciju sa drugim servisima.
- Proxy server se setuje u okviru programa za pretragu Interneta, obično u meniju Options (ili Properties)-Settings-Proxy Configuration.
- Proxy se ili dodeljuje automatski od strane Internet provajdera ili se mora ručno uneti za svaki tip Internet servisa (biće reči kasnije)
- Naš proxy je: proxy.rcub.bg.ac.rs ili u obliku brojeva: 147.91.1.43

Modemski pristup Internetu:

- Dok se za pristup Internetu preko mrežna karte, osim instalacije drajvera, ne zahtevaju dodatna podešavanja, to nije slučaj za modemski pristup.
- Modemski pristup je pre desetak godina bio najčešći način na koji su se na Internet povezivali mnogi korisnici. Brzina prenosa podataka koja se ovim putem može postići ograničena je kvalitetom telefonske veze (tj.centrale), ali svakako nije mogla da prelazi 56 kilobita u sekundi (što je u odnosu na današnjih 1-100 Mbita u sekundi za (LAN, kablovsku ili ADSL konekciju) mnogo sporije.
- Da se podsetimo: (byte = 1 slovo = 8 bitova (binarnih cifara)).
- Modem prvo treba setovati da funkcioniše u pulsnom ili tonskom režimu (u zavisnosti od telefonske centrale na koju ste priključeni). Control Panel-Phone and Modem Options-Dialing Rules-Edit.
- Zatim treba uspostaviti novu konekciju: Control Panel-Network Connections-Create a new connection-Connect to Internet-Set up my connection manually-Connect using a dial-up modem-ukucati naziv konekcije, broj telefona, username i password (detaljnije će biti objašnjeno na vežbama).
- Ostala setovanja su ista kao i u slučaju mrežnog pristupa.
- Ukoliko je moguće, modem treba namestiti na tonski režim jer je biranje znatno brže a oštećenja modema usled biranja brojeva su znatno manja.

Osnovni Internet servisi:

- Neki od alata koji se upotrebljavaju za pristup i dobijanje informacija sa Interneta su:

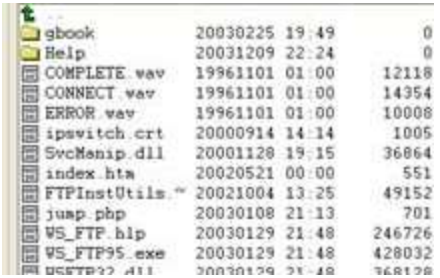
  - **FTP** (**F**ile **T**ransfer **P**rotocol) koji je relativno jednostavan i brz način dobijanja informacija i podataka. Nema grafičko okruženje ali u tome i jeste njegova prednost.

  - **Gopher** (softverska aplikacija koja pomaže nalaženje tekstualnih informacija na Internet Gopher serverima) - sve manje se koristi jer nudi samo tekstualne podatke.

  - **WWW** (**W**orld **W**ide **W**eb) trenutno najpopularniji način da dobijanje podataka. Podržava multimediju.

  - **BitTorrent** (**P2P P**eer to **P**eer protocol) gde ne postoji originalni distributor već su fajlovi dostupni na različitim serverima.

  - **eMule** (oblik **P2P**) sličan **EDonkey2000 i** BitTorrentu



Traženje drajvera na Internetu:

- Pošto je računar povezan na Internet može se započeti potraga za drajverima koji nedostaju. Najpre treba ponuditi samom Windowsu da ih sam potraži: Start - Settings - Control - Panel - System - Hardware - Device Manager - Update driver
- Ukoliko je pretraga bila bezuspešna potražiti drajvere preko nekih od pretraživačkih servisa.
- Ovakvih servisa ima mnogo i na korisniku je da odluči koji će servis koristiti (preporuka: Google).
- Postoje i specijalizovani sajtovi na kojima se može naći najveći broj drajvera, mada je najbolje prvo otići na sajt proizvođača hardvera za koji nedostaje drajver (ukoliko je uopšte poznat).
- Ukoliko postoji problem sa nazivom proizvođača ili tipom hardvera, treba pažljivo pročitati svaku oznaku direktno na njemu i to upisati u pretraživač.
- Važno je upoznati se sa načinima pretraživanja u cilju dobijanja potrebnih informacija.

Google

LYCOS msn

YAHOO!

alltheweb

altavista
THE SEARCH COMPANY

Kompresija fajlova:

- Ukoliko postoji potreba da nekome pošaljete neke fajlove mail-om, potrebno je obratiti pažnju na veličinu poslatog fajla. Brzina kojom korisnici mogu "skidati" podatke sa Interneta može biti veoma mala (naročito u slučaju modemskog pristupa) tako da velike fajlove (veće od 10 MB) nikada ne treba bez dogovora slati mail-om.
- Da bi veličinu fajlova (ili grupe fajlova) maksimalno smanjili i omogućili njihov prenos često se koristi njihova **kompresija** ili "pakovanje".
- Kompresija (ili source coding) je proces enkodiranja informacija u manji broj bitova nego original, korišćenjem specifičnih algoritama kodiranja.
- Postoje dva osnovna vida kompresije:

  - **koja ne gubi podatke** (lossless) kao što je ZIP, RAR, ARJ, TAR ...

  - **koja gubi podatke** (lossy compression) kao što je DIVX, MP3, JPG ...

- Kompresija koja ne gubi podatke (lossless) može se dekomprimovati u originalni fajl bez ikakvih gubitaka, što ne važi i za lossy jer se njen princip zasniva na namernoj eliminaciji podataka za koje se smatra da nisu od suštinske neophodnosti.
- Najbolji primer za razlikovanje ova dva vida kompresije je pisanje broja (stringa): 888883333333. U lossless vidu kompresije on će se pisati kao 8(5)3(7) dok će lossy posati isti broj samo kao 83. Osoba koja dekodira enkodirane fajlove mora imati identičan kompresioni alogoritam ili računar neće razumeti pakovanu arhivu.

Virusi i antivirusni programi:

- Pre nego što korisnik počne sa radom, neophodno je da se zaštiti od **kompjuterskih virusa**.
- Kompjuterski virus je kompjuterski program koji kopira samog sebe bez znanja korisnika računara. Kopije se mogu menjati (virus mutira) što značajno otežava izlečenje.
- Virus se može širiti sa zaraženog na nezaraženi računar ukoliko se uspostavi kontakt između njih (putem mreže ili kopiranjem fajlova).
- Viruse treba razlikovati od **crva** i **trojanskih konja.** Crvi se takođe samorepliciraju ali (za razliku od virusa) nemaju potrebu da se implementišu u neki određeni program i unište određene fajlove, već oštećuju isključivo mrežu. Sa druge strane, trojanski konji su programi koji za razliku od virusa i crva nemaju mogućnost sami da funkcionišu i zavise od postupaka korisnika (to su često nekada bili korisni programi koji su izmenjeni tako da komplikuju rad sa računarom i omogućavaju pristup podacima koji inače ne bi smeli da budu svima dostupni).
- Uspešna zaštita od ovih programa, u većini slučajeva, može biti instalacija različitih **antivirusnih programa** uz redovno ažuriranje baze podataka o novim virusima koji se svakodnevno pojavljuju.
- Drugi način na koji se korisnik može zaštititi od virusa je rad u drugim operativnim sistemima (kao što je je npr. **Linux**).

Prikupljanje informacija:

- Prvi korak u pisanju nekog rada je prikupljanje informacija. Ovde će biti reči o prikupljanju informacija (literature, radova i knjiga) sa Interneta.
- Jedan od načina na koji se literatura može nabaviti je korišćenje **Internet pretraživača** (bilo je već reči u delu o drajverima), dok je drugi korišćenje specijalizovanih univerzitetskih baza podataka kao što je **SciFinder** i **Kobson**.
- Razlika u ova dva načina pretraživanja je što pomoću prvog niste ograničeni samo na naučne radove i časopise (što je slučaj sa druga dva) već postoji mnogo širi pristup informacijama koje mogu biti od koristi, ali je ovo i mnogo teži put.
- Pre nego što se krene u pretragu, treba razmisliti o tome **ŠTA JE POTREBNO PRONAĆI** (da li je to naučni rad, knjiga, slika, neka animacija, kompjuterski program...), jer od toga zavisi koji metod pretrage može biti efikasniji.
- Sledeći korak je znati **NAZIV TOG FAJLA** (ili deo naziva). Za naučne radove vrlo je korisno znati naslov rada (ili knjige) ili bar imena autora, mada ni to nije neophodno.
- Sledeći korak je pretpostaviti **FORMAT FAJLA** koji se traži (.pdf, .avi, .zip ...). Mnogi tipovi fajlova se uobičajeno koriste za određene vrste podataka. Tako se za knjige i dokumenta često koriste: .PDF, .CHM, .PDB, .DJVU (kao i .ZIP i .RAR arhive) što često može biti od velike pomoći.
- Poslednji korak je pristupiti pretrazi i obezbediti dosta vremena i živaca.

Nešto o formatima fajlova:

- Kratka priča o formatima (ekstenzijama) koji se najčešće koriste za različite tipove fajlova u cilju njihovog naleženja na internetu:
- Dokumenti: **DOC, RTF** (Word dokumenti, retko se koriste za dokumente na Internetu), **PDF** (Adobe Acrobat dokumenti, najčešći format za radove i knjge na Internetu), **DJV** ili **DJVU** (Dežavu format, alternativa PDF formatu ali na žalost lošija alternativa), **CHM** (Microsoftov komprimovani HTML help format koji se ponekad koristi), **TXT** (jednostavan ali uvek aktuelan), **PRC, DNL, OPF, TEX, PS** (formati koji se isto koriste, ali ređe, u glavnom za elektronske knjige.
- Slike: **BMP, JPG, GIF, TIF, TGA, EMF, WMF, CDR** (Corel Draw format), **PSD** (Photoshop format) - razlikovati vektorske o nevektorskih formata (biće reči kasnije).
- Video fajlovi: **AVI, MPG, MPEG, MOV** (Quicktime), **WAV** (Microsoft), **RM** (Real Payer format) i mnogi drugi što zavisi od codeca koje se koristi.
- Audio formati: **CDA** (audio CD format), **MP3, WMV** (Windows media), **MID** i mnogi drugi u zavisnosti od audio codeca koji se koristi.
- Mulitimedija: **SWF, FLV** (Macromedia tip formata koji su univerzalnog karaktera i mogu sadržati i tekst, video, slike ili audio).
- Formati za arhivirane atoteke: **RAR, ZIP, ARJ, TAR, CAB, LZH, TAR, GZip, JAR**...
- Izvršni formati: **EXE, BAT** (Oprezno ih pokretati jer mogu sadržati viruse).

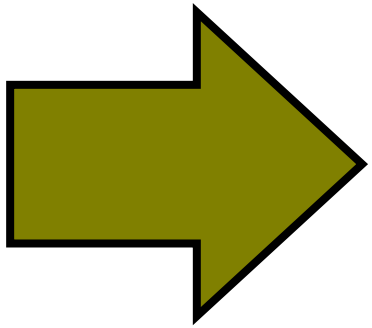
Google pretraživanje:

- Google pretraživanje u cilju dobijanja određenoj fajla može biti mukotrpan posao s obzirom na ogromnu bazu podataka koja se u njemu nalazi.
- Standardna pretraga koja se sastoji od ubacivanja naziva ili formata fajla u prostor za pretragu često će naći ogroman broj sajtova koji su manje-više povezani sa kontekstom pretrage (ovo nije preterano informativno), i zato treba koristiti **Advanced Search** opciju koja znatno može da suzi izbor fajlova.
- Međutim, često ni ovo nije dovoljno, i za korisniju pretragu je potrebno poznavati princip skladištenja podataka u Google bazi. Podaci se najčešće pakuju u tipske Foldere, što znači da će najveći broj npr. knjiga biti smešten u Folderima tipa: "parent directory " Books ili za MP3 ukucati: MP3 search link
- Znaci navoda u Googlu znače da se traže samo fajlovi koji se nalaze pod ovim specifičnim nazivom i vrlo ih je korisno upotrebljavati. Tu je i opcija traženja specifičnih formata fajlova kao i domena, jezika, datuma, zatim posebna opcija za traženje slika, videa, newsgrupa i korisnika sa sličnim problemima i pitanjima (lepo je znati da još neko ima problema kao i vi).

KoBSON:

KoBSON

Konzorcijum biblioteka Srbije za objedinjenu nabavku



Još neki servisi ...

VPN (tunelovanje)

Google Scholar:

• Najveći broj istraživača i studenata kao prvi izvor za pretraživanje naučnih informacija koristi Google koji je razvio poseban servis ***Google Scholar*** (http://scholar.google.com) namenjen pretraživanju i lociranju naučne literature.

• KoBSON je uspostavio saradnju sa *Google Scholar*-om, tako da je svim korisnicima **KoBSONa** omogućeno linkovanje na dostupne izvore direktno sa *Google* stranice za pretraživanje i prikazivanje rezultata. Kao prvi korak neophodno je na glavnoj stranici izabrati *Scholar Preferences.*

• U predviđeno polje uneti KoBSON, kliknuti na *Find Library,* i sačuvati podešavanje klikom na *Save preferences.* Ovo je potrebno uraditi samo jedan put. Nakon obavljenog traženja u *Google Scholar*-u rezultati će biti prikazani u prikazanoj formi.Ukoliko ste na akademskoj mreži (ili ste prijavljeni preko lične licence).

Google Scholar:

- Ukoliko je KoBSON pretplaćen na izdavača časopisa u kome je članak objavljen, klikom na naslov prilazite punom tekstu.
- Ukoliko je članak dostupan preko nekog od pretplaćenih agregatora (EBSCO, PROQUEST, HINARI) potrebno je izabrati KoBSON, nakon čega će se izlistati svi neophodni podaci vezani za časopis u kome je članak objavljen, npr:
- Ukoliko je članak elektronski dostupan, klikom na reč *članak,* na ekranu će se prikazati pun tekst članka (kao podsetnik, u levom gornjem uglu ekrana, prikazani su svi bibliografski podaci o traženom članku).

Čuvanje podataka dobijenih pretraživanjem Interneta:

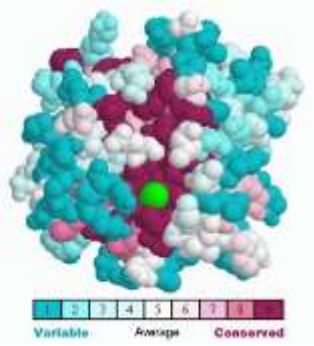
- Mučili ste se da dođete do podataka preko Interneta i sada je red da ih sačuvate. Ukoliko su dobijeni podaci u obliku PDF dokumenta ili nekog vida arhive (**RAR** ili **ZIP** - bez lozinke), to je lako - ali to često nije slučaj !!!
- U cilju zaštite podataka, mogućnost da se sačuva ono što se vidi na ekranu može biti onemogućena. Šta raditi u ovim slučajevima?

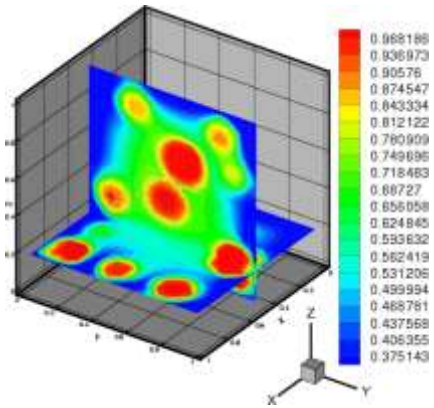
  - Najjednostavnije je pritisnuti taster "**Print Screen**" koji automatski u keš stavlja ono što se nalazi na ekranu. Ovako zapamćenu stranicu možete "zalepiti" u bilo kom programu za obradu slika "npr. **Paint**". Ovo funkcioniše često ali ne uvek.

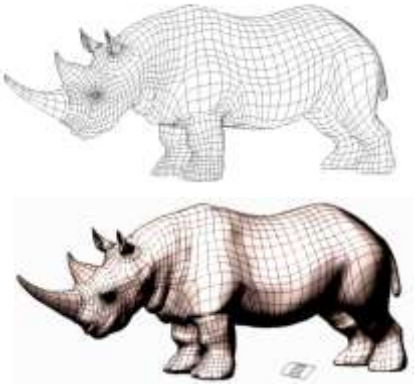
  - Ovakav pristup nije od pomoći ukoliko želite da sačuvate neku animaciju ili multimedijalni fajl. U tom slučaju treba poznavati princip po kome se ovakav tip fajlova, koji se mogu naći na Internetu, reprodukuju na računaru.

- Fajlovi se pre reprodukcije uvek prvo smeštaju u keš (ne memorijski, već jedan "skriveni" Folder na hard-disku).

Uvod u računarske simulacije:







Teorijske osnove računarskih simulacija:

- Napredak računarskih sistema doprineo je da računare možemo koristiti za simuliranje različitih fizičkohemijskih procesa.
- Karakterizacija nekog fizičkohemijskog sistema se, u principu, zasniva na numeričkim proračunima nekakavih fizičkih veličina prema zakonitostima koje određuje teorija.
- Rezultati dobijeni izračunavanjem nekih formula moraju imati i svoju eksperimentalnu potvrdu.
- Ukoliko se teorijski proračuni i praktično dobijeni rezultati poklope, teorija se može smatrati ispravnom (i obrnuto, eksperiment se može smatrati ispravno izvedenim).
- Međutim, između teorije i eksperimenta može se postaviti **SIMULACIJA** kao npr. sintetički izveden eksperiment koji se bazira na teorijskim konceptima.
- Proračuni koji se izvode na jednostavnim, idealnim, sistemima su relativno prosti i često ne predstavljaju problem. Međutim, ukoliko sistem ima puno parametara (tj. elemenata sistema), situacija se znatno usložnjava.

Tipovi računarskih simulacija:

- Realni sistemi, sa kojima se susrećemo u svakodnevnom životu, odnose se najvećim delom na veliki broj čestica sa još većim brojem interakcija između njih.
- Ponašanje ovih sistema najčešće sa ispituje putem **SIMULACIJA** preko konstrukcije **MODELA** koji predstavlja uprošćenu reprezentaciju realnog sistema.
- Ukoliko je model sličniji realnom sistemu simulacija će biti realnija (ali to često zahteva veoma jake računarske sisteme).
- Modeli mogu biti: eksperimentalni (npr. neka maketa u vakuumskoj komori) ili kompjuterski (danas se sve više koriste).
- Kompjuterske simulacije se generalno mogu podeliti na:

  - one koje koriste čestične sisteme (mikroskopski pristup) - koordinate, brzina čestica ...

  - one koji koriste kontinualne sisteme (makroskopski pristup) - pritisak, temperatura, gustina ...

Prototip i model:

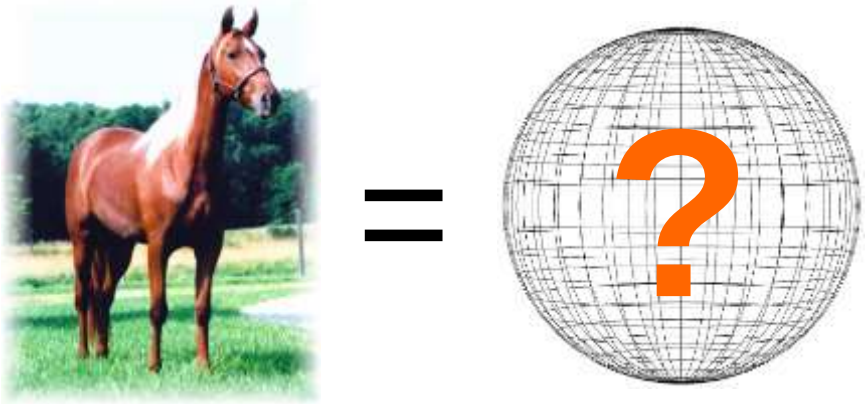
- **Prototip** predstavlja deo (podsistem) realnog sistema čiji model želimo da simuliramo. On obično predstavlja kompleksni sistem koji se sastoji od više odvojenih podsistema koji međusobno interaguju.
- Naš model bi trebao da predstavlja sve podsisteme, ali ponekad je korisno da se neki od njih smatraju neaktivnim (ili konstantama pod datim uslovima).
- Predstavljanje modela pomoću računara bi takođe trebalo da prati ovakvu strukturu, što znači da bi program trebao da se sastoji iz više delova (potprograma ili podrutina) koje se po potrebi pozivaju u glavni program.
- Nakon izvršenih proračunavanja kao rezultat imamo neke podatke (ili najčešće velike grupe podataka) koje treba nekako jasno predstaviti (a to je uvek najbolje uraditi putem grafičkog prikaza). Iz tog razloga, rad sa računarskim simulacijama usko je povezan sa kompjuterskom grafikom.

Validacija modela:

- Da bi bili sigurni da model ispravno reprezentuje eksperiment koji želimo da izvršimo, mora se izvršiti **validacija modela**. Ovaj postupak se sastoji iz nekoliko koraka:

  - provera grešaka na nivou programa (da se vidi da li program radi onaj posao za koji je napisan). Ovo se radi tako što se programu zadaju ulazni parametri za koje je rezultatat već poznat (mukotrpan posao koji je toliko uspešniji što je broj provera veći i raznovrsniji).

  - provera ispravnosti funkcionisanja modela (da li zaista predstavlja ono što smo želeli da simuliramo).

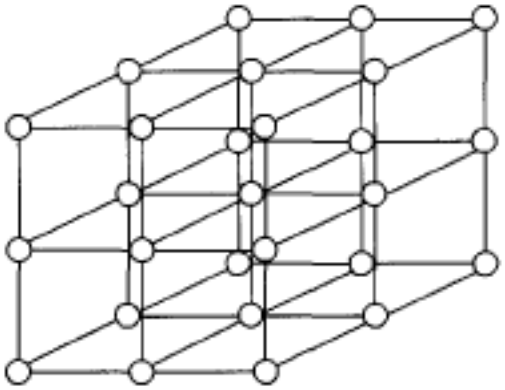


Model i simulacija:

- Suština **modela** je to da on predstavlja uprošćenu reprezentaciju nekog realnog objekta ili fizičkog procesa u cilju reševanja nekog (makar ograničenog) problema.
- Modeli mogu opisivati realnu situaciju sa manjom ili većom preciznošću i treba imati u vidu koliko precizan model je potrebno simulirati da bi se dobili zadovoljavajući rezultati.
- Jedan od modela svima poznat je napravio Nils Bor 1913. (Borov model atoma) gde su elektroni predstavljani kao male naelektrisane čestice koje se vrte oko pozitivno naelektrisanog jezgra (planetarni model). Iako je on daleko od preciznog opisa realne situacije, on je do izvesnog stepena sasvim dobro mogao da nam pojasni neke atomske interakcije.
- Problem kod ovog modela je to što je ubrzano kretanje elektrona trebalo da uslovljava emitovanje EM zračenja, ali je Bor zato uveo postulate o postojanju samo diskretnih E nivoa (i prelazima između njih).
- Naime, Borov model sasvim lepo funkcioniše za H-atom, ali je za potpuno opisivanje složenijih atoma potrebno uvesti izvesne korekcije u vidu kvantno-mehaničkih modela Šredingera i Hajzenberga (1926. godina).
- Ovde se vidi prednost matematičkog u odnosu na fizički model. Fizički model može biti realno konstruisan (od kuglica i žica) ali neće biti svrsishodan kao matematički model koji može pokazati prednosti i mane nekog modela.

Primeri modelovanja:

- Još jedan od primera kada je korisno praviti modele je kada proučavamo kristalnu strukturu materijala. Na slici je primer kubne rešetke sa atomima koji su postavljeni u rogljevima elementarne ćelije.



  - U ovom klasičnom modelu smatra se da su svi atomi u stanju mirovanja iako u realnosti oni vibriraju oko ravnotežnih položaja.
  - Ukoliko posmatramo sistem koji pravi male oscilacije možemo primeniti harmoničnu aproksimaciju, ali ukoliko atomi počnu da, pod određenim uslovima, više odstupaju od ravnotežnog položaja, neophodno je uključiti i jednačine za anharmonične vibracije.

- Koji model ćemo primeniti u našoj simulaciji, dakle, zavisi od vrste problema koji treba rešiti i od eksperimentalnih uslova pod kojima želimo da se ispitivani sistem nalazi. Kako to drugi rade?

Tipovi simulacija i njihove primene:

- Postoji nekoliko osnovnih načina na koje možemo izvršiti simulacije različitih tipova problema.

1) Monte Karlo metoda

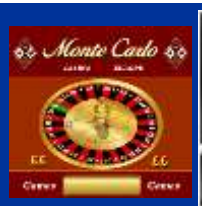
2) Metoda čestica

3) Metoda konačnih promena

4) Metoda konačnih elemenata

- Ovo su metode koje se najčešće koriste i biće ukratko objašnjeno na kom principu svaka od njih funkcioniše, kao i na kojim tipovima problema se svaka od njih može primeniti.
- Pored ovih, postoji još puno drugih metoda koje su specifične za simulacije određenih fizičkohemijskih problema.

Monte Karlo metoda:

1) Monte Karlo metoda

Tokom II svetskog rata američki naučnici su intenzivno radili na dizajniranju atomske bombe. U svom radu često su se susretali sa problemom brzog rešavanja zadataka sa kojima do sada nikada nisu imali nikakvih iskustava (npr. problem kako će neutroni prolaziti kroz nekakvu barijeru). Kao što je poznato, neutroni sa nekom sredinom ili: ne interaguju, elestično ili neelastično bivaju odbijeni, ili mogu biti absorbovani. Verovatnoća svakog od ovih procesa zavisi od energije neutrona. Iako je zavisnost verovatnoće svakog od ovih procesa dobro poznata, ne postoji nikakav način da se ovaj problem reši konvencionalnom matematičkom analizom.

Ulam i **Fon Nojman** su ovaj problem uspeli da reše na sledeći način: Posmatrali su prolazak jednog neutrona kroz barijeru. Da bi odlučili šta će se sa njim dalje dešavati bacali su kockicu (tj. okretali točak na ruletu). Prateći kretanje neutrona i praveći Monte Karlo odluku o njihovoj daljoj interakciji sa okolnom sredinom oni su uspeli da, ponavljajući ovaj postupak veoma veliki broj puta, veoma uspešno predvide ponašanje realnog fizičkog sistema (tj. da ustanove koliki broj neutrona je prošlo ili nije prošlo barijeru).

Metoda čestica:

2) Metoda čestica

Često je potrebno simulirati sisteme sa veoma velikim brojem čestica (npr. galaksija koja sadrži 10^{11} zvezda ili interkcije u tečnosti koja sadrži 10^{24} čestica). Današnje simulacije iz molekulske dinamike mogu obuhvatati 10^{7}-10^{8} čestica i to ako su uključene samo interakcije kratkog dometa (izm. najbližih suseda). Postoji jedan bazični pristup koji znatno smanjuje broj čestica koji treba posmatrati, a to je da sistem posmatramo preko tzv. **superčestica** pri čemu svaka predstavlja veliki broj realnih čestica.

Dalja aproksimacija može se izvršiti ukoliko se posmatra da se grupacije čestica nalaze u paralelopipedima (često, ali ne uvek u kocki). Uticaj svih čestica u okviru te kocke na neku udaljenu česticu posmatra se kao da se sve čestice nalaze u centru kocke.

Metoda konačnih promena:

3) Metoda konačnih promena

Postoji potreba da se simuliraju fizički sistemi u kojima dolazi do kontinualne promene neke od fizičkih veličina (zapremine, temperature ...) u vremenu (ili prostoru) i u kojima se brzina promene tih veličina može opisati pomoću parcijalnih diferencijalnih jednačina.

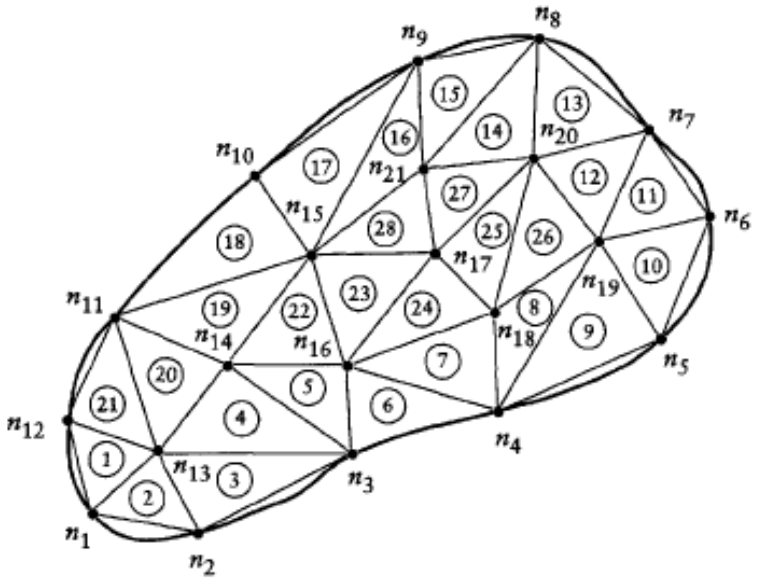
Ova metoda se sastoji u tome da se promena osobine koja se ispituje (neka je to npr. koncentracija *n*), definiše samo u okviru regije od interesa.

Parcijalni diferencijali koji određuju promenu ovih fizičkih veličina ($\partial^2 n/\partial x^2$ i $\partial n/\partial t$) se aproksimativno uvek mogu prikazati kao linearna kombinacija vrednosti poznatih parametara u jednom trenutku i nepoznatih parametara u trenutku koji sledi nešto kasnije.

Metoda konačnih elemenata:

4) Metoda konačnih elemenata

Koristi se najviše za 2D i 3D simulacije ravnotežnih sistema sa malom simetrijom. Specifični region koji treba simulirati se definiše setom tačaka koje se zovu **NODOVI.** Njihovim povezivanjem se dobijaju **ELEMENTI** i i **GRANICE** simuliranog sistema.



Sistem kompleksnog oblika koji je predstavljen od 21 noda i 28 trouglasta elementa.

Ovako prezentovani sistem je mnogo lakše posmatrati i uključiti u matematičke operacije (integracija površina i funkcionalna zavisnost nodalnih vrednosti.

Simulacije i aplikativni softver

- Kompjuterski programi danas mogu u trenutku rešiti komplikovane matematičke i fizičkohemijske probleme za čije je rešavanje ranije trebalo mnogo više vremena.
- Danas postoji veliki broj kompjuterskih programa koji su napisani za ovu svrhu i stoga prosečnom korisniku nije potrebno detaljno poznavanje programiranja da bi rešio svoj specifični problem. Međutim, korisnik mora da poznaje princip rada ovih programa i način na koji može da izvrši njihovu pripremu za rešavanje svog specifičnog problema.
- Treba se suočiti sa time da kompjuterski programi nisu savršeni niti da mogu rešiti bez greške svaki postavljeni problem.
- Programi o kojima će u ovom kursu u daljem izlaganju biti reči su:

  Excel (dopunske opcije), **Matlab**, **Mathematica**, **Chem Office**, **Lab View** i **SRIM**.
- Naravno, treba imati u vidu da je ovo je samo mali segment programa koji se mogu iskoristiti za ove svrhe.

Poznavanje problema:

- Računar može relativno brzo (i uspešno) rešiti zadati problem. Veći problem je zadati (postaviti) zadatak na onaj način kako ga računar može prepoznati.
- Da bi se to uspešno moglo izvesti neophodno je poznavati fizički fenomen koji treba reštiti i postaviti ga u matematički kontekst koji računar može rešiti.
- Problem najčešće može biti rešen na nekoliko načina, upotrebom različitih matematičkih programa od kojih svaki može imati svoje prednosti i nedostatke.
- Na korisniku je da izabere na koji način će problem rešiti i pri tome treba da ima u vidu i način na koji softver koji primenjuje može komunicirati sa drugim programima.
- Poćićemo prvo od rešavanja najjednostavnijih jednačina sa jednom nepoznatom pomoću računara.

Rešavanje jednačina u programu Excel:

- Primer kako možemo rešiti prostu jednačinu pomoću računara u programu Excel: Jednačina sa jednom nepoznatom

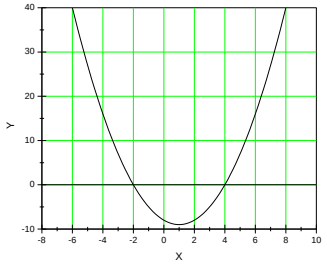
$$f(x) = x^2 - 2x - 8$$

- U Excelu se jednačine mogu rešiti pomoću opcije "**Goal Seek**". Otvori se nova radna sveska i u B1 upiše zadata jednačina:

```
=A1*A1-2*A1-8
```

- Cilj je pronaći "nulu" jednačine. Ići na Tools-Goal Seek i u prikazanom meniju otkucati:

Setcell	B1
To value	0.
By changing cell	A1



X1=-2
X2=4

Origin: File-New-Function

Rešavanje jednačina u programu Excel:

-2.000007 4.1137E-06

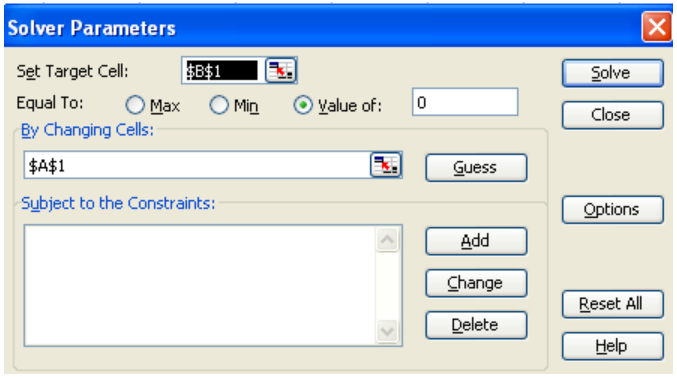
- Kao što se vidi, računar je pronašao rešenje, ali sa malom greškom. Takođe, ni rešenje koje je računar pronašao nije baš tačno nula, već broj koji je veoma blizu nje (ova tačnost je najčešće dovoljna za većinu zadataka).
- Ukoliko želimo da budemo manje "tolerantni" prema greški koju računar pravi, to možemo regulisati u Tools-Option-Calculation-Maximum, pa dodati još jednu 0 u maksimalni broj iteracija i tri nule u Maximum change.
- Sada će novo rešenje istog problema biti:

-2 -1.376E-8

- Da bi našli drugi koren jednačine, u A1 ukucati npr. 3 pa zatim ponovo uraditi Goal Seek. Šta se događa ako u A1 ukucamo tačno 4? Šta smo iz ovoga zaključili, kako računar rešava jednačine?

Drugi način rešavanja jednačine u Excelu:

- Isti problem se u Excelu može rešti na još jedan način, upotrebom programa **Solver**.
- Ići na: Tools-Solver. Ukoliko tu ne postoji Sover, to znači da u toku instalacije ova opcija Excela nije instalisana, tako da je treba potražiti na Tools-Add Ins-Solver (iz Analysis Tool paketa).



- Isti princip se primenjije i ovde, samo što se sve opcije koje se tiču preciznosti dobijenog rezultata nalaze u Options meniju.
- Drugo rešenje se dobija na identičan način kao i u prethodnom primeru.

Primena u fizičkoj hemiji:

- Isti princip rešavanja problema možemo primeniti za rešavanje nekih fizičkohemijskih problema.
- Npr. zadatak je <u>naći specifičnu zapreminu</u> n-butana na 500K i 18 atmosfera korišćenjem Redlih Kwong-ove jednačine realnog gasnog stanja.

$$\widehat{v}^3(p) - \widehat{v}^2(RT) + \widehat{v}\,(a - pb^2 - RTb) - ab = 0$$

$$a = 0.42748\left(\frac{R^2T_c^2}{p_c}\right)\alpha, \quad b = 0.08664\left(\frac{RT_c}{p_c}\right), \qquad T_r = \frac{T}{T_c}, \quad \alpha = \frac{1}{T_r^{0.5}}$$

- Prvi korak je pronaći kritičnu temperaturu i pritisak u literaturi: T_c=245.2K i p_c=37.5 atm (R=0.08206 l atm/g mol K).
- Drugi korak je izračunati vrednosti za koeficijenate a i b.

Rešavanje problema u Excelu:

- Napraviti nov dokument u Excel-u i upisati sve poznate parametre i jednačine u odgovarajuća mesta:

	A	B	C	D	E	F	G
18							
19			*n*-Butane				
20							
21							
22		Parameters				Data	
23	Tc	425.2	K		T	500	K
24	pc	37.5	atm		p	18	atm
25	R	0.08206	liter-atm/g mol K				
26							
27		Formulas				Results	
28	Tr	F23/B23			Tr	1.1759	
29	a	0.42748*(B25^2*B23^2/B24)^(B23/F23)^0.5			a	12.7981	
30	b	0.08664*(B25*B23/B24)			b	0.0806	
31	f(v)	F24*F31^3-B25*F23*F31^2			v	2.0377	liter/g mol
32		+(F29-B25*F23*F30-F24*F30^2)*F31-F29*F30			f(v)	1.957E-07	
33	ideal gas	volume	B25*F23/F24		ideal gas	2.2794	

- Pravilno obeležiti poznate i nepoznate veličine i odvojiti ih radi preglednosti. Iskoristiti **Goal Seek** komandu da bi izraz f(v) izjednačili sa nulom i našli rešenje variranjem parametra "v".
- Dobija se rešenje 2.0333 l/gmol (koje možemo porediti sa jedn. ideal gasnog stanja RT/p) .... ali..... kako možemo proveriti da li je ovo tačno?

Provera dobijenog rezultata:

- Kao i kada nešto računate na digitronu, niste nikada sigurni da li ste dobro otkucali sve brojeve i računske operacije. I ovde treba proveriti dobijeni rezultatat :)
- Prvo proveriti da li su sve formule dobro unesene, zatim proveriti da li su svuda usklađene jedinice.
- Znači, provera dobijenog rezultata počiva na prethodnom korišćenju papira i olovke, proveri jedinica i konačnog izraza formule koju kasnije samo treba pažljivo uneti u računar:

Pa zar onda nije lakše i izračunati tu vrednost pomoću digitrona ?

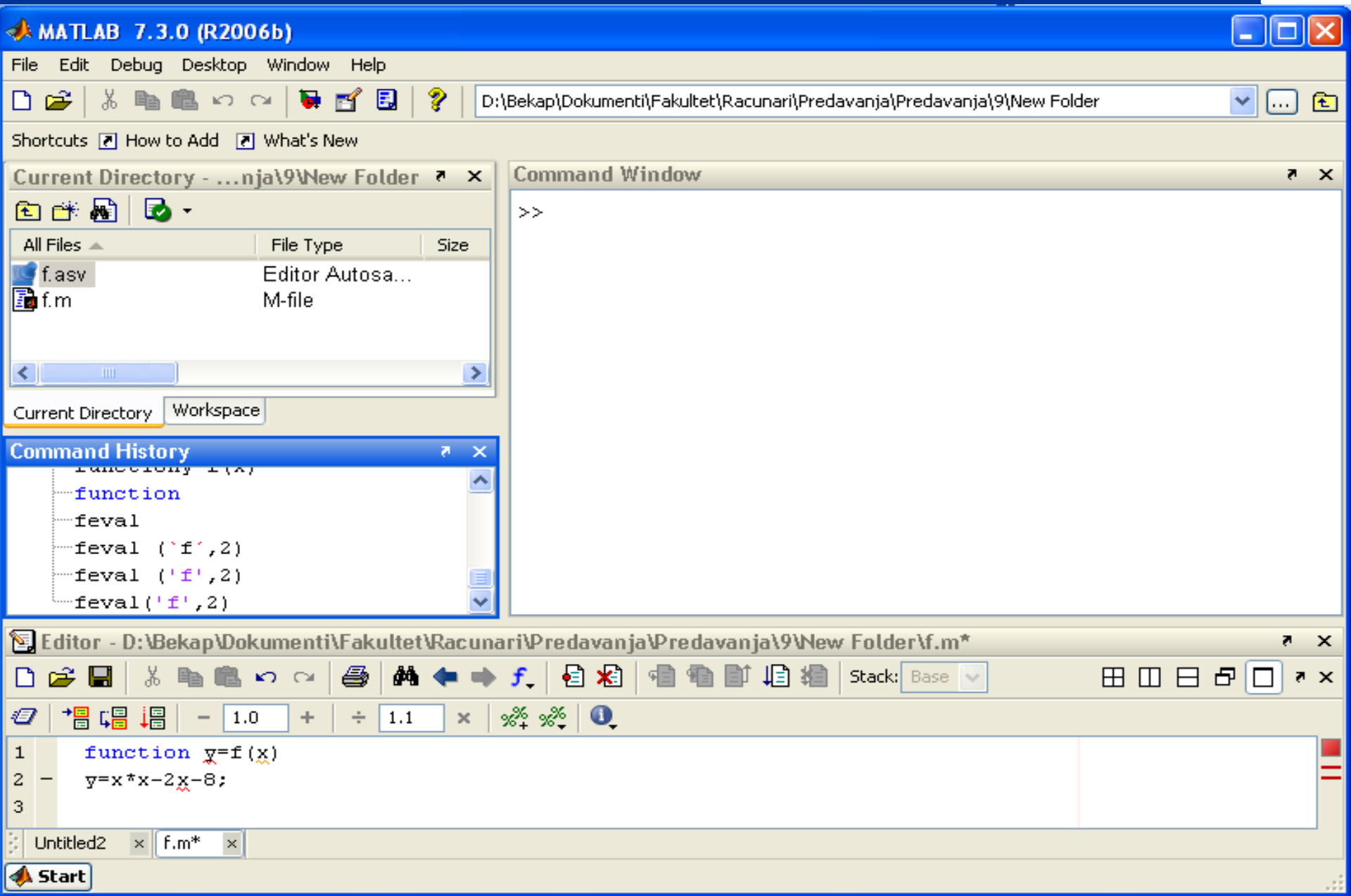
Odgovor: Jeste možda jednom, ali univerzalnost postavke izraza dozvoljava i obrnut račun.

Rešavanje jednačina korišćenjem programa MATLAB:

- Nelinearne jednačine mogu biti rešene i korišćenjem programa **MATLAB**. Prvo šta treba uraditi je definisati problem koji treba rešiti pravljenjem tzv. "m-fajla".
- Sledeći korak je proveriti postavku zadatka a zatim treba pronaći opciju za njegovo rešavanje unutar programa (ovo su koraci analogni onima koji se izvode u Excel-u).
- Ali da bi napravili m-fajl potrebno je znati raditi u komandnom prozoru MATLAB-a, i zato prvo ... **upoznavanje sa MATLAB programom**.
- Počinjemo: Otvoriti **MATLAB,** ići na File-New-M-file. Radna površina se sastoji iz različitih prozora. Direktorijum u kome se radi prikazan je u gornjem meniju.
- Ici na Desktop-Desktop Layout-Default da bi dobili standardni meni u radu sa programom.
- Na ekranu se mogu prepoznati sledeći osnovni prozori:

  Current Directory (koji se može promeniti u Wokspace prozor), Command History, Command Window i Editor.

Izgled MATLAB prozora:



MATLAB - neke osnovne stvari:

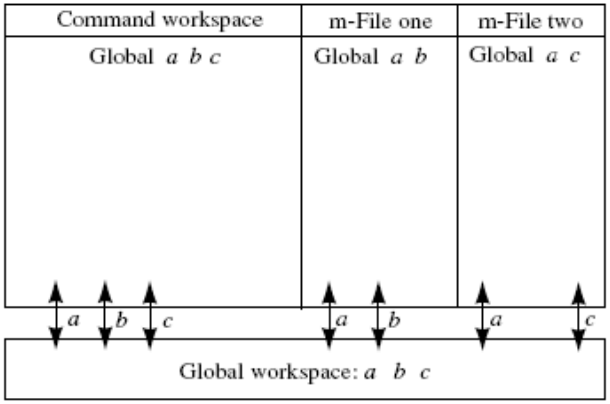
- U **Workspace** prozoru nalazi se lista promenjivih, njihov opis (u smislu da li su globalne ili ne).
- U **Comand History** nalazi se spisak komandi koje ste izdavali ranije.
- Command window je mesto gde se komande zadaju i gde se dobija rezultat računa.
- Odabiranjem raznih opcija u Meniju **Desktop**, mogu se prikazati ili sakriti svi ovi prozori.
- Sve komande koje ste ukucavali ranije možete povratiti pritiskom na gornju strelicu "↑". Pojavljuju se jedna po jedna. Ovo je veoma korisna opcija i olakšava proces ukucavnja izraza, pogotovu ukoliko se tokom upisa potkrala neka greška.
- Pritiskom na strelicu na dole, komande se pojavljuju obrnutim redom.
- Rad (m-fajl) se mora po završetku rada sačuvati da bi nam, pri sledećem radu u programu, sve informacije o svim numeričkim vrednostima koje smo prethodno izračunali bile na raspolaganju.
- Ako proračun (simulacija) ponekad potraje suviše dugo, i želimo da je prekinemo, to se radi pritiskom na **Control+C** (Forced quit).

m-fajlovi:

- U progamu **MATLAB** pišu se kompjuterski programi koji se zovu **m-fajlovi** i oni se mogu sačuvati na disku. Ove fajlove možemo koristiti svaki put ako ukucamo u komandnom prozoru naziv fajla.
- Takođe, jedan m-fajl može koristiti drugi m-fajl.
- Za svaki m-fajl treba ukucati komentar šta on u stvari radi, i to se postiže tako što se ispred onog što se ukucava upise % (sve što se nalazi iza % smatra se komentarom).
- Svaki m-fajl ima svoj jedinstveni **Workspace** i oni međusobno ne kominiciraju osim ako to eksplicitno ne želimo (ovo možemo da poistovetimo sa dva druga koji su se međusobno naljutili i neće da komuniciraju osim ako neko treći (vi) to ne organizuje).
- Postoje dva načina da organizujete komunikaciju između njih:
  - korišćenjem globalne komande
  - pozivanjem argumenata iz m-fajlova

Globalna komanda:

- **Globalna komanda** može se predstaviti preko Komandnog prozora i dva m-fajla:



Ukucajte (ukucavanja se vrše u Command prozoru):

```
>>global a b c
>>a=1
>>b=2
>>c=3
```

- Pitanje je sada kako vrednosti a, b i c dodeliti m-fajlovima ??? To se radi ubacivanjem globalne komande u m-fajl.
- Na slici su parametri a i b raspoloživi u m-fajlu 1 dok su a i c raspoloživi u m-fajlu 2. Ako se promenljiva "a" promeni u fajlu 1, promeniće se i u fajlu 2 (promena je izvršena na globalnom nivou). Da bi se promenjiva "c" iskoristila u m-fajlu 1, ona mora biti posebno definisana pošto je to promenljiva iz m-fajla 2 ("c" nije definisana na listi globalnih komandi fajla 1). Promena "c" parametra u m-falju 2 promenila bi i njenu vrednost na globalnom nivou.

Komande "disp", "clear" i "input":

- Display komanda "disp" se u MATLAB-u koristi da na ekranu prikaže željenu vrednost promenljive, parametri ili tekst: Sintaksa:

  >> disp ('Ovo je primer prikaza na ekranu')

Obratiti pažnju na zagrade i znake "polunavoda". Pitanje: Šta je važnije staviti ???

- Komanda "clear all" briše sve zadate promenjive: Sintaksa:

  >> clear all

- Komanda "input" omogućava korisniku unos novih promenljivih: Sinataksa:

  >> brzina = input('Koja je brzina (m/s)?')

Još nešto ...

- Prikazivanje najvećeg i najmanjeg realnog broja koji se mogu obraditi u MATLAB-u:

```
>> realmin

ans =

  2.2251e-308

>> realmax

ans =

  1.7977e+308
```

- Kompleksni brojevi u MATLAB-u:

```
>> i

ans =

        0 + 1.0000i
```

- Znakovi deljenja u MATLAB-u: (postoji "/" i "\") i ne znače isto tj. "\" predstavlja inverziju tj. INV (A)*B

```
>> 10/2

ans =

     5

>> 10\2

ans =

    0.2000
```

- Ukucavanje dugačkih komandnih linija se vrši pomoću "..."

```
>> x = sin(1) - sin(2) + sin(3) - sin(4) + sin(5) -...
sin(6) + sin(7) - sin(8) + sin(9) - sin(10)

x =

    0.7744
```

Petlje u MATLAB-u:

- U programu MATLAB možemo zadavati petlje koje služe da se neke komende rade stalno iznova. Sintaksa je sledeća:

```
>> for i=1:5
y(i)=0+i
end
```

Petlja

```
y =
1
y =
1    2
y =
1    2    3
y =
1    2    3    4
y =
1    2    3    4    5
```

Rezultat petlje

Pitanje: Šta se događa ako je u Worspace prozoru ostala neka od promenljivih? Kako to reštiti? Napraviti m-fajl za ovu petlju. Ubaciti comandu clear all na pocetku programa (m-fajla). Koja je razlika?

Uslovne rutine:

- Uslovne rutine u MATLAB-u unosimo ukoliko želimo da nam program "odluči" da li je neki uslov ispunjen a zatim uradi određenu operaciju:
- Napravimo mali programčić uz uslov u MATLAB-u:

  - ponovimo da se programi ukucavaju u **Editor** prozoru i da se nakon toga moraju sačuvati pod nekim nazivom (m-fajl). Zatim se u Command prozoru sacuvani m-fajl poziva (pokreće program) ukucavanjem naziva fajla.

```
a=input('Kolika je prva vrednost ')

b=input('Kolika je druga vrednost ')

if (a<b) disp('Prva vrednost je manja od druge')

else disp ('Prva vrednost je veca od druge')

end
```

Rutina koja traži od korisnika da unese dve vrednosti i kaže koja je veća

Šta se dešava ako ukucamo”;”? Šta je rezultat ako upišemo jednake brojeve ?

Povećajmo preciznost ovog programa:

- Uvedimo i komandu: "elseif" koja dopušta dopunski uslov poređenja:

```
a = input('Kolika je prva vrednost ');
b = input('Kolika je druga vrednost ');
if (a<b)
    disp('Prva vrednost je manja od druge')
elseif (a==b)
    disp ('Prva vrednost je jednaka drugoj')
else disp ('Prva vrednost je veca od druge')
end
```

Program: "manjevece"

- Naglasimo ovde dve stvari:

  - Iza zagrade upisujemo znak ";" da bi odredili obim izlaznog signala (u ovom slučaju program bi funkcioniso ispravno i bez toga).

  - Znak jednakosti se upisuje kao "==" a ne kao "=" jer ima za zadatak upoređivanje dva broja a ne predstavlja deo nekog izraza ili funkcije.

Funkcije u MATLAB-u

- U MATLAB-u je u komandnom prozoru korisno definisati funkciju koja će se koristiti. To se radi komandom "inline"

$$f(x,y) = \sqrt{x^2 + y^2}$$

- Vrednost funkcije se poziva na sledeći način:

```
>> f = inline('sqrt(x.^2+y.^2)','x','y')

f =

     Inline function:
     f(x,y) = sqrt(x.^2+y.^2)
```

ili

```
>> f =@(x,y) sqrt(x.^2+y.^2);

Na ovan način definišemo „function handle“ tip promenljivih. Zapaziti da nema potvrde ispisa na ekranu jer smo upisali ;
```

```
>> f(3,4)

ans =

     5
```

- Ovo radi i sa nizovima. Njih pravimo sledećom komandom:

```
>> A = [1 2;3 4]

A =

     1     2
     3     4
```

```
>> B = ones(2)

B =

     1     1
     1     1
```

```
>> C = f(A, B)

C =

    1.4142    2.2361
    3.1623    4.1231
```

Rešavanje funcija u komandnom prozoru

- Sada se možemo vratiti na naš primer rešavanja nelinearne jednačine koristeći program MATLAB:
- Jednačina koju je trebalo rešiti je: $y=x^2-2x-8$; Prvi korak je u komandnom prozoru definisati funkciju:

```
>> f = inline ('x^2-2*x-8','x')

f =

     Inline function:
     f(x) = x^2-2*x-8
```

- Uvedimo "feval" komandu pomoću koje dobijamo rešenje funkcije u određenoj tački (npr. za x=7)

```
>> feval (f,7)

ans =

    27
```

- Komandom "fzero" možemo naći nulu funkcije (za y=0), variranjem rešenja oko 0 i 3;

```
>> fzero (f,0)

ans =

    -2
```

```
>> fzero ('f',3)

ans =

     4
```

Rešavanje funcija korišćenjem Editor prozora u MATLAB-u:

- Sada možemo pokušati da za ovakav način rešavanja nelinearne jednačine napravimo programčić u Editor prozoru.
- Jednačina koju je trebalo rešiti je: $y=x^2-2x-8$; Prvi korak je u Editor prozoru definisati funkciju:

```
function y=f(x)
y=x*x-2*x-8;
```

- Zatim definisanu funkciju treba sačuvati kao m-fajl (recimo da je nazovemo "f.m").
- U Command prozor-u zatim treba pozvati naš m-fajl uz sledeću komandu:

```
>>feval('f',2)
```

- Ova komanda traži od MATLAB-a da nam da brojevnu vrednost funkcije f(x) ako x iznosi 2 … i kao rezultat, naravno, dobjamo:

```
ans=-8.
```

- Na ovaj način smo videli da smo funkciju ispravno uneli i da program funkcioniše.

... nastavak:

- Sledeći zadatak je pronaći "nulu" funkcije. Ona se u programu MATLAB, kao što smo videli, pronalazi pomoću komande "fzero".
- Ukucajmo u Command prozoru sledeću naredbu:

```
>>fzero('f',0)
ans=-2
```

- Znači sada je program (za razliku od malopre) izračunao x za y=0. Ukoliko želimo da proverimo rešenje, ukucamo:

```
>>feval('f',ans)
```

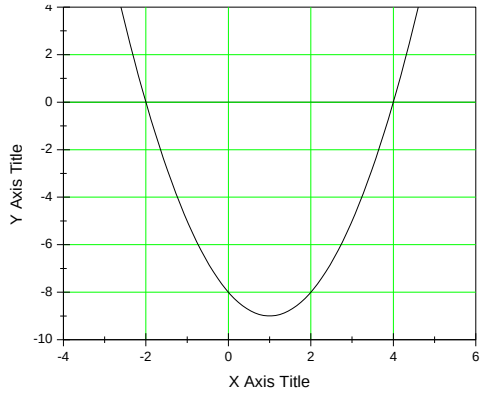
- Ukoliko želimo da dobijemo i drugu "nulu", moramo MATLAB-u naložiti da varira druge vrednosti za "x", različite od onih malopre. Upišemo:

```
>>fzero('f',3)
ans= 4
```

Kako dobiti sva rešenja jednačine?

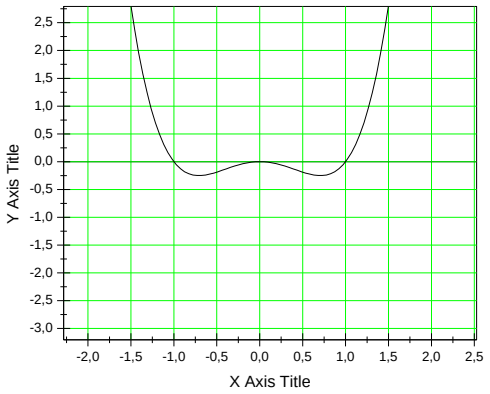
- Ipak, MATLAB ima komandu pomoću koje možemo jednostavno dobiti sve nule neke funkcije:

```
>> solve('x*x-2*x-8=0')

ans =

 -2
  4
```



ili:

```
>> solve('x^4-x^2=0')

ans =

 -1
  0
  0
  1
```



Rešavanje jednačine s opštim brojevima:

- MATLAB može raditi ne samo sa pravim već i sa opštim brojevima (simbolički objekti). Rešimo sledeću jednačinu: $y=ax^2+bx+c$

```
>> syms a b c x;
>> solve('a*x^2 + b*x + c')

ans =

 -(b + (b^2 - 4*a*c)^(1/2))/(2*a)
 -(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

- U prvom koraku komandom “syms” definišemo simboličke objekte: a,b,c,x; pri čemu se kreiraju "simboličke" promenljive (pogledati Workspace prozor).
- U drugom koraku se traži rešenje jednačine komandom “solve” (podrazumeva se da je x nezavisno promenljiva).

- Ukoliko želimo da rešimo ovu jednačinu po "b“ dopunimo komandu “solve” na sledeći način:

```
>> syms a b c x;
>> solve('a*x^2 + b*x + c','b')

ans =

-(a*x^2 + c)/x
```

- Zadatak: Rešiti nekoliko jednačina 2,3,4 stepena po različitim promenljivama.

Rešavanje sistema jednačina sa opštim brojevima:

- MATLAB može rešavati i sisteme jednačina sa pravim i opštim brojevima. Pokažimo to na sledećem primeru:

```
>> [x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0')

x =

 1
 3

y =

   1
-3/2
```

- Izrazom u velikoj zagradi smo najpre definisali promenljive (što se može videti u “Workspace” prozoru).

- Na isti način rešavamo i sisteme jednačina sa opštim brojevima:

```
>> [u,v] = solve('a*u^2 + v^2 = 0','u - v = 1')

u =

 1 - (a + (-a)^(1/2))/(a + 1)
 1 - (a - (-a)^(1/2))/(a + 1)

v =

 -(a + (-a)^(1/2))/(a + 1)
 -(a - (-a)^(1/2))/(a + 1)
```

- Zadatak: Rešiti nekoliko sistema jednačina sa pravim i opštim brojevima.

Rešavanje diferencijalnih jednačina:

- Prilikom rešavanja diferencijalnih jednačina u MATLAB-u treba obratiti pažnju na sintaksu.
- Na primer, ukoliko želimo da rešimo diferencijalnu jednačinu:

$$\frac{dx}{dt} = -ax$$

Sintaksa bi bila:

```
>> dsolve('Dx = -a*x')

ans =

C2/exp(a*t)
```

- Kao što vidimo, diferencijalne jednačine se rešavaju komandom “dsolve”.
- Slovo “D” označava diferencijal u odnosu na nezavisno promenljivu (podrazumevano je: d/dt) a rešenje se daje za zavisno promenljivu (u ovom slučaju je to “x”).

- Ukoliko želimo da promenimo oznake za nezavisno i zavisno promenljve to moramo posebno navesti koristići sledeću sintaksu.
- Znači, samo pod apostrofom navedemo šta nam je nezavisno promenljiva.

```
>> dsolve('Dy = -a*y','x')

ans =

C6/exp(a*x)
```

- Zadatak: Rešiti nekoliko diferencijalnih jednačina prvog reda sa različitim promenljvama.

Rešavanje diferencijalnih jednačina višeg reda i jednačina sa uslovom

- Prilikom rešavanja diferencijalnih jednačina višeg reda potrebno je upotrebiti sledeću sintaksu. Na primer, ukoliko želimo da rešimo diferencijalnu jednačinu:

$$\frac{d^2y}{dx^2} = -ay$$

pišemo

```
>> dsolve('D2y = -a*y','x')

ans =

C8*exp((-a)^(1/2)*x) + C9/exp((-a)^(1/2)*x)
```

- Znači, jedino je potrebno iza slova “D” dopisati broj (red izvoda) a sve ostalo je isto.
- Obratiti pažnju da, ukoliko koristimo opšte brojeve u svom izrazu, ne upotrebljavamo slovo D jer ono označava diferencijal.
- Ukoliko imamo dodatne uslove, to možemo navesti u zagradi pre definisanja nezavisno promenljive.

$$\frac{dy}{dx} = -ay$$

```
>> y = dsolve('Dy = -a*y','x')

y =

C6/exp(a*x)
```

bez uslova

```
>> y = dsolve('Dy = -a*y','y(0) = 1','x')

y =

1/exp(a*x)
```

uslov

- Zadatak: Rešiti nekoliko diferencijalnih jednačina višeg reda sa različitim promenljvama i uslovima.

Zamena opštih brojeva u rešenje jednačine

- Ukoliko smo dobili rešenja neke jednačine (npr. diferncijalne) u opštim brojevima i sada želimo da to rešenje primenimo za neke određena brojevne vrednosti, to se radi pomoću komande "subs". Na primer:

```
>> y = dsolve('Dy = -a*y')

y =

C2/exp(a*t)

>> a = 980

a =

   980

>> C2=3

C2 =

     3

>> subs(y)

ans =

3/exp(980*t)
```

- Znači, nakon dobijenog rešenja potrebno je da navedemo koje brojevne vrednosti pridružujemo opštim bojevima a zatim koristimo komandu "subs".
- Pri tom treba imati u vidu da se aktuelne promenljive uvek mogu pogledati u "Workspace" prozoru.
- Ukoliko su tu prethodno postojale promenljive sa istom oznakom, one će biti zamenjene novim.

- Zadatak: Rešiti nekoliko diferencijalnih jednačina prvog reda u opštim brojevima nakon čega zadati neke brojevne vrednosti promenljivima i pomoću komande "subs" dobiti rezultat kao brojevnu vrednost.

Rešavanje sistema diferencijalnih jednačina

- Rešavanje sistema difencijalnih jednačina odvija se po istom pricipu prilikom čega se mogu primeniti sledeće sintakse:

```
>> z = dsolve('Dx = y', 'Dy = -x')

z =

    y: [1x1 sym]
    x: [1x1 sym]

>> z.x

ans =



>> z.y

ans =
```

- Sintaksa levo pridružuje broju "z" vrednost rešenja pri čemu se formira strukturni niz "z" simboličkih promenljivih "x" i "y".
- Da bi videli njihove vrednosti potrebno je ukucati: "z.x" i "z.y".
- Donja sintaksa je jednostavnija i odmah daje rešenja ali se pri tom ne formira strukturni niz kao posebna promenljiva (što može uticati na kasniju strukturu i upotrebljivost programa npr. zamenu vrednosti "subs" komandom).

```
>> [x,y] = dsolve('Dx = y', 'Dy = -x')

x =



y =
```

- Zadatak: Rešiti nekoliko sistema diferencijalnih jednačina upotrebom obe vrste sintaksa.

Računanje izvoda

```
syms x %definisanje simboličkih objekata
f(x) = x^2 %definisanje fje
y = diff(f) %definisanje izvoda fje

%crtanje grafika

subplot(2,1,1); %definisanje položaja I slike
ezplot (f(x)) %crta fju
subplot(2,1,2); %definisanje položaja II slike
ezplot (y) %crta izvod
```

Rešavanje neodređenih integrala

- Rešavanje integrala se u MATLAB-u izvodi komandom "int". Možemo rešavati neodređene i određene integrale i sintaksa je sledeća:

```
>> int(x^2)
??? Undefined function or variable 'x'.
```

```
>> syms x
>> int(x^2)

ans =

x^3/3
```

```
>> syms x z;

>> int(x/(1+z^2),z)

ans =

x*atan(z)

>> int(x/(1+z^2),x)

ans =

x^2/(2*(z^2 + 1))
```

- Pokušali smo da rešimo jedan jednostavan neodređen integral: $\int x^2 dx$ direktnom upotrebom komande "int", ali prijavljena je greška.
- Razlog za grešku je zbog toga što prethodno nije definisana koja simbolička promenljiva je u pitanju.
- Nakon što definišemo simboličku promenljivu (pomoću "syms" komande) greška je otklonjena.
- Ukoliko želimo da naglasimo po kojoj promenljivoj integral treba da bude izračunat (po "z" ili po "x"), to radimo sledećom sintaksom (pre toga, naravno, moramo definisati sve simboličke promenljive).

- Zadatak: Rešiti nekoliko neodređenih integrala u odnosu na različite promenljive.

Rešavanje određenih integrala

```
>> syms x
>> int (x*log(1+x))

ans =

x/2 - x^2/4 + (log(x + 1)*(x^2 - 1))/2
```

```
>> syms x
>> int(x*log(1+x),0,1)

ans =

1/4
```

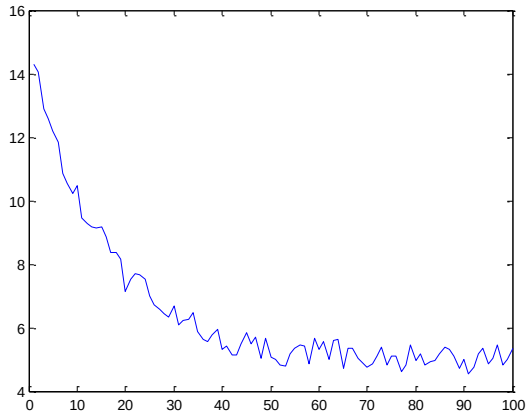
```
>> syms x z;

>> int(x/(1+z^2),z,0,1)

ans =

(pi*x)/4
```

- Rešavanje određenih integrala se u MATLAB-u takođe izvodi komandom "int" ali uz nešto dopunjenu sintaksu. Rešimo prvo jedan neodređeni integral po x.
- Zatim, zadajmo programu da nam izračuna vrednost tog integrala u intervalu od 0 do 1. Sintaksa je sledeća:
- Primetimo da je pri ovom kao promenljiva uzeta "x" (s obzirom da je to jedina definisana promenljiva).
- Ipak, ukoliko želimo da izračunamo određeni integral (od 0 do 1) po nekoj drugoj promenljivoj (npr. "z") primenićeno sledeću sintaksu:
- Naravno, ukoliko prethodno ne definišemo i promenljivu "z", program će nam prijaviti grešku.

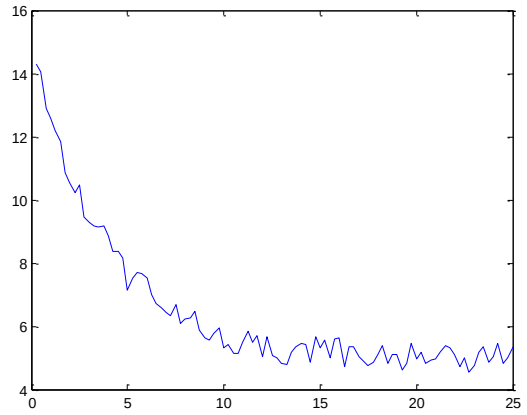
- Zadatak: Rešiti nekoliko neodređenih integrala u odnosu na različite promenljive. Probati računanje integrala bez definisanja promenljivih (ali prethodno obrisati postojeće promenljive komandom "clear all".

Crtanje 2D grafika u MATLAB-u

```
>> plot (y)
```




```
>> plot (x,y)
```

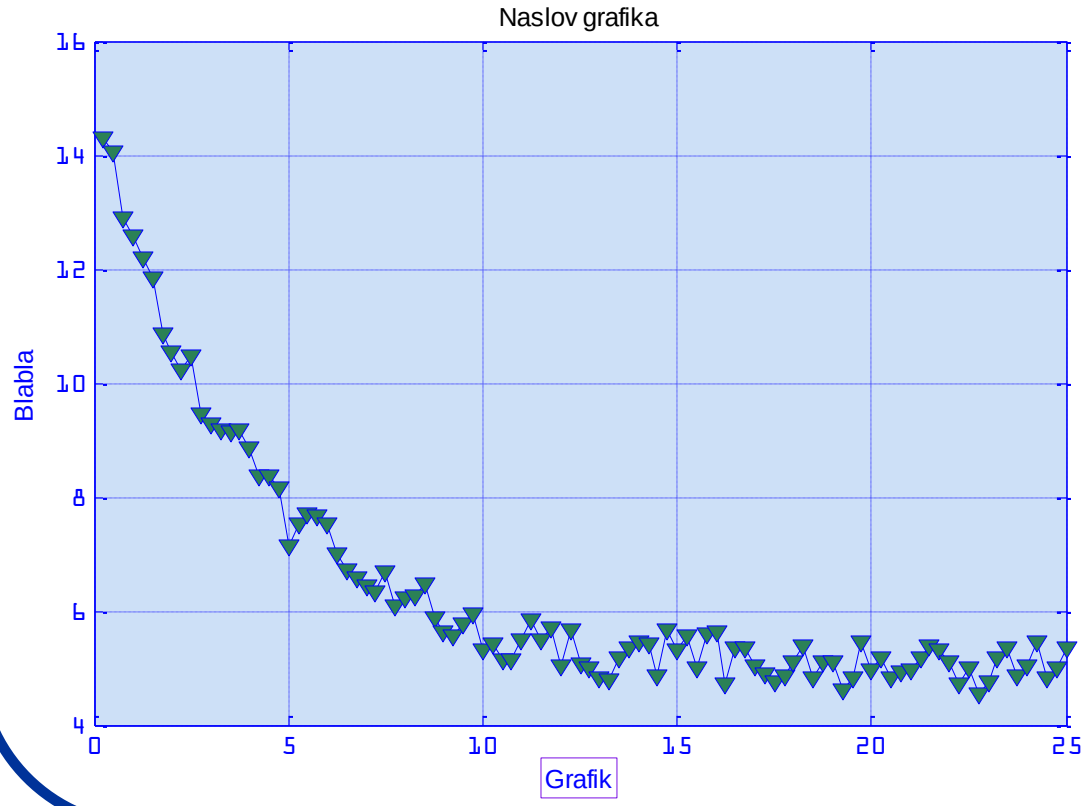


- Najpre naučimo kako nacrtati grafik ukoliko imamo zadate tačke samo na y-osi.
- U "Workspace" prozoru desnim klikom izaberemo opciju "New" pri cemu formiramo novu promenljivu (nazovimo je "y").
- Dvoklikom na promenljivu "y" otvoriće se "Variable editor" prozor (matrica) u kome možemo upisati tačke (ili ih zalepiti iz Origin fajla). Mi ćemo uvesti promenljive "x" i "y".
- Grafik se dobija ukucavanjem komande "plot" sledećom sintaksom:
- Za x-vrednosti se uzimaju celi brojevi od 1 do N (N=boj tačaka na grafiku).
- Ukoliko želimo da nacrtamo grafik za (x,y) parove to se radi sledećom sintaksom:
- Treba samo obratiti pažnju da svakoj vrednosti "x" odgovara po jedna vrednost "y".

- Zadatak: Nacrtati nekoliko 2D grafika koristeći proizvoljne tačke (prvo samo "y" vrednosti a zatim (x,y) parove).

Crtanje 2D grafika u MATLAB-u

- Nakon što smo nacrtali grafik pokušajmo da na njega dodamo nekoliko dodatnih oznaka. Grafik može biti prikazan sa: određenom širinom linije, veličinom, oblikom i bojom i tačaka, nazivima za x i y ose, naslovom ...

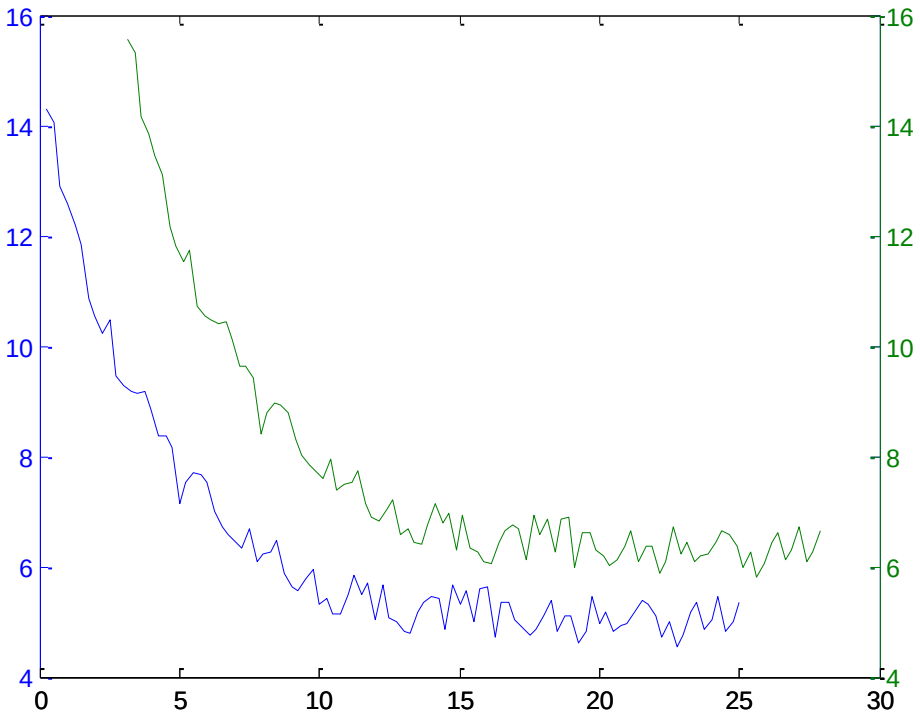


- Jednostavno na grafiku u gornjem meniju ići na. View-Proprety Editor
- Ne zaboraviti opciju: More Properties u okviru ovog menija.
- Upotrebiti i opciju: Tools za Zoom, 3D prikaz, markiranje i brisanje tačaka ...

- Zadatak: Na što veći broj načina prikazati dobijene 2D grafike koristeći što više opcija koje nudi MATLAB. Pregledati sve dodatne opcije koje nudi grafički meni i upotrebiti ih (ubacivanje teksta, linija ...).

Prikaz više funkcija na istom grafiku

```
>> plotyy (x,y,x1,y1)
```



- Za crtanje više različitih setova tačaka ili funkcija na istom grafiku koristi se komanda "plotyy" po sledećoj sintaksi:
- Da bi ova komanda funkcionisala, moramo prethodno napraviti setove tačaka koje ćemo dodeliti određenim promenljivima.
- Manipulisanje sa ovakvim grafičkim prikazom se izvodi na isti način kao i kod prikaza samo jednog seta tačaka.

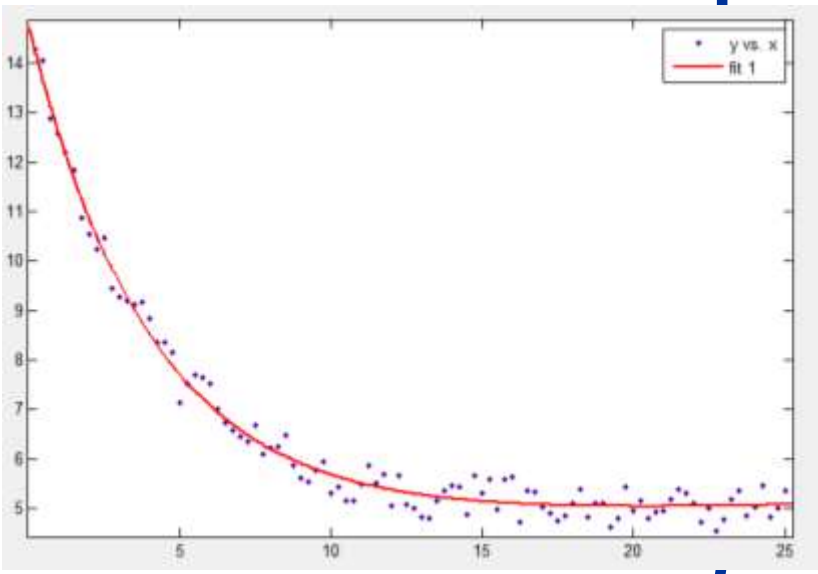
- Zadatak: Na što veći broj načina prikazati dobijene 2D grafike sa više setova tačaka. Upotrebiti što više opcija koje nudi MATLAB. U "Help" meniju ukucati "plot" i "plotyy" pa pročitati i upotrebiti ostale sintakse koje su na raspolaganju.

Fitovanje grafika u MATLAB-u

- MATLAB ima posebnu rutinu koja je zadužena za fitovanje različitih setova tačaka.
- Da bi primenili ovu rutinu najpre moramo u "Workspace" prozor uvesti (ili napraviti) promenljive koje predstavljaju setove tačaka tj. njihove (x,y) vrednosti.
- Upotrebimo dosadašnje setove tačaka "x" i "y". U "Command" prozoru zatim ukucati komandu "cftool".

```
>> cftool
```

- Ići na opciju "Data" pa izabrati za "X Data" i "Y Data" promenljive iz "Workspace" prozora. Kliknuti na "Create data set" da bi tačke bile prikazane.
- Ići na "Fitting" pa "New fit", i izabrati tip fita iz padajućeg menija "Type of fit". Odatle izabrati podtip funkcije za koju se pretpostavlja da bi bila najbolja za zadati set tačaka.
- Dopunske korekcije funkcije i parametara se mogu uraditi opcijom "Fit options". Rezultat fitovanja je prikazan grafički i brojevno (u prozoru "Resuts".



- Zadatak: Isfitovati zadate setove tačaka (zadaje ih asistent) upotrebom "cftool" rutine.

Kako nacrtati funkciju?

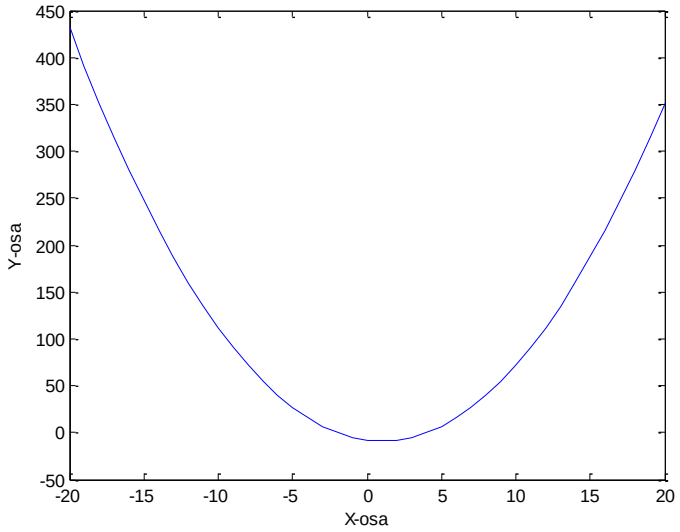
• Ukoliko definišemo našu funkciju kao:

Fajl: “fplot.m”

```
x=-20:1:20;
y=x.*x-2*x-8;
plot(x,y);
xlabel('X-osa')
ylabel('Y-osa')
```

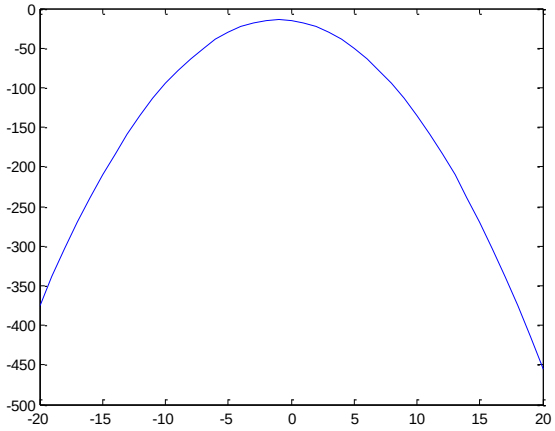
Definicija neke funkcije, određivanja opsega za x-vrednost sa korakom 1, crtanje funkcije i obeležavanje osa.

• Zapazimo tačku koja se nalazi iza x (u definiciju funkcije).
• Ova komanda nam govori da se vrednosti y računaju za svaki broj x.
• Izbrisati tačku iza x i videti da li će program raditi. Promeniti korak za x.
• Nacrtati nekoliko različitih funkcija kao npr.:sin(x), e^x, log (x) ....



Crtanje fje u intervalu i ispitivanje prevojnih tačaka

```
x=-20:1:20;              % definisanje intervala
y=-x.^2-2*x-15;          % definisanje funkcije
plot(x,y);               % crtanje funkcije

clear                    % brisanje promenljivih
x = sym('x');            % pravljenje simboličkog objekta 'x'
y=-x.^2-2*x-15;          % definisanje funkcije

g = diff(y,x)            % diferenciranje funkcije
g = - 2*x – 2            % rezultat diferenciranja

s = solve(g)             % traženje nule diferencijala
s = -1                   % rezultat
```



Zadatak:
Najpre pronaći prevojne tačke funkcije (y = 2*x.^3 + 3*x.^2 - 12*x + 17) a zatim pronaći još neku funkciju sa nekoliko prevojnih tačaka i pronađi njihove vrednosti.

Kako još možemo pronaći minimum funkcije u intervalu?

```
>> f = @(x)x.^3-2*x-5;

>> x = fminbnd(f, 0, 2)

x =

    0.8165

>> y = f(x)

y =
   -6.0887
```

Definisanje funkcije pomoću „function handle“

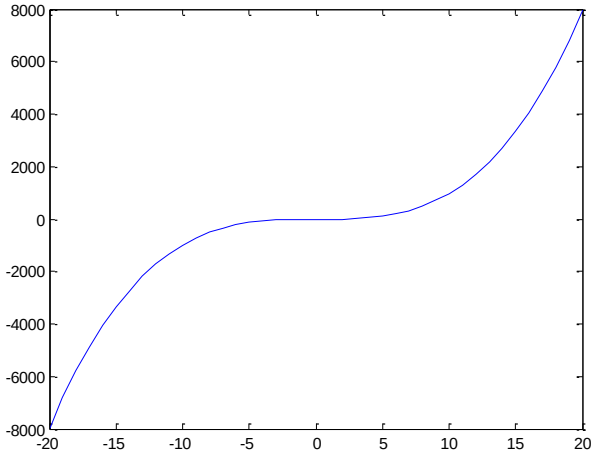
Definisanje intervala i kome želimo da izračunamo minimum funkcije

Rezultat

Tražimo vrednost fje u minimumu

Rezultat

f =x^3-2*x-5



Primena u fizičkoj hemiji:

- Primenimo sada MATLAB za rešavanje istog fizičkohemijskog problema kao što smo radili pod programom Excel: Naći specifičnu zapreminu n-butana na 500 K i 18 atmosfera korićšenjem Redlih-Kwong-ove jednačine realnog gasnog stanja.
- Naravno, potrebno je prvo pripremiti m-fajl koji će nam omogućiti da definišemo zavisno i nezavisno promenljive, upišemo konstante i postavku problema.

```
function y=specvol(v)
% in K atm l/gmol
% parameters for n-butane
Tc=425.2
pc=37.5
T=500
p=18
R=0.08206
aRK=0.42748*(R*Tc)^2/pc
aRK=aRK*(Tc/T)^0.5
bRK=0.08664*(R*Tc/pc)
y=p*v^3-R*T*v^2+(aRK-p*bRK^2-R*T*bRK)*v-aRK*bRK;
```

Definisanje funkcije i promenljivih.

Dopuske tekstualne informacije koje nemaju uticaja na račun.

Definisanje konstanti tj. nezavisno promenljivih. Definisanje zavisno promenljivih.

Definisanje funkcije preko promenljivih.

Testiranje funkcije:

- Napisan m-fajl nazovimo npr. "specvol.m". Prvo je potrebno testirati ispravnost programa; Ukucati komandu:

```
feval('specvol',0.2)
ans=specvol(0.2)
```

- Ukoliko je rezultat ovakav, problem je dobro postavljen jer je "feval" funkcija izračunala vrednost promenljive "y" za vrednost v=0.2. Rezultat:

```
Tc=425.2000
pc=37.5000
T=500
p=18
R=0.08206
aRK=13.8782
aRK=12.7981
bRK=0.0806
y=-0.6542
```

Program je prikazao račun za sve tražene vrednosti i treba ga pažljivo proveriti, naročito aRK, bRK i y.

... nastavak:

- Dalje tražimo nulu funkcije, tj. za koje vrednosti v je izraz=0. Međutim, nezgodno je da se svaki put kao rezultat prikazuju sve vrednosti nezavisno promenljivih pa je zato korisno znati da se ovo može izbeći ukoliko se iza svake programske linije doda ";".
- Ovim postupkom kao rezultat se prikazuje samo konačno rešenje funkcije.
- Sada treba ukucati komandu za pronalaženje "nule" funkcije:

```
v=fzero('specvol',0.2)
v=2.0377
```

- U "feval" komandi, 0.2 je je bila vrednost v za koju je bilo potrebno pronaći vrednost y, dok je sada 0.2 upotrebljeno kao "pretpostavka" rešenja, tj. broj oko koga treba varirati rešenje. Da bi proverili koliko je rešenje blizu treba potražiti vrednost funkcije za zadato rešenje:

```
>>ans=specvol(v)
ans=-2.2204e-15
```

Ukoliko MATLAB ne može da pronađe rešenje, on će to reći.

Još jedan zadatak:

- Naći faktor kompresibilnosti za nekoliko vrednosti pritisaka. Faktor kompresibilnosti računa se po formuli:

$$Z = \frac{pv}{RT}$$

- Za niske pritiske (kada se gas smatra približno idealnim), faktor kompresibilnosti je blizak jedinici,ali kako se pritisak povećava,Z se menja.
- Sada želimo da napravimo program (simulaciju) koji će naći specifičnu zapreminu za pritiske od 1 do 31 atmosfera.
- Program zatim treba da izračuna odgovarajuće faktore kompresibilnosti.
- Na kraju, želimo da nam program grafički predstavi izračunatu zavisnost Z=f(p).
- Da bi ovo ostvarili, moramo primeniti sve što smo do sada naučili: uvesti globalne promenljive, upotrebiti petlju i pomenutu komandu "plot".

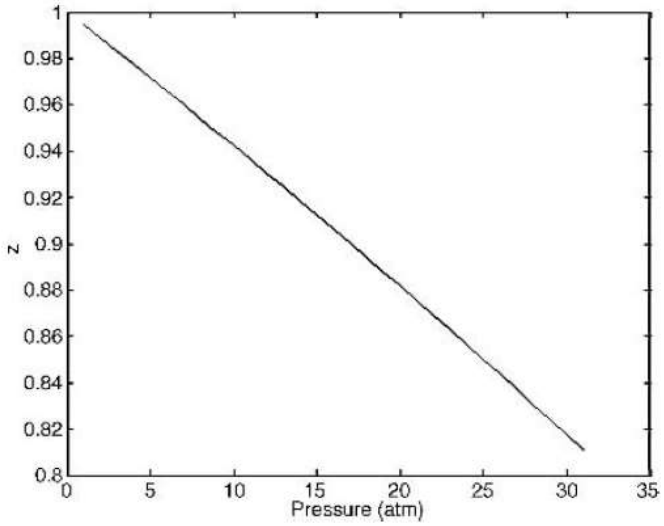
Programski kod (objašnjenje):

```
% run_volplot
global T p Tc pc R aRK bRK
% in K atm l/gmol
% parameters for n-butane
Tc=425.2
pc=37.5
T=500
R=0.08206
for i=1:31
  pres(i)=i;
  p=i;
  aRK=0.42748*(R*Tc)^2/pc;
  aRK=aRK*(Tc/T)^0.5;
  bRK=0.08664*(R*Tc/pc);
  vol(i)=fzero('specvol',0.2);
  Z(i)=pres(i)*(vol(i)/R*T;)
end
plot(pres,Z)
xlabel('pressure(atm) ')
ylabel('Z')
```

- Programski kod pod nazivom run_vplot računa specifične zapremine za pritiske od 1 do 31 atmosfere a zatim računa odgovarajući faktor kompresibilnosti.
- Druga linija programa uvodi globalne promenljive kojima se kasnije pripisuju poznate vrednosti.
- U programu "specvol" takođe su uvedene iste promenljive (kojima sada možemo direktno pristupiti jer smo ih definisali kao globalne).
- Sledeći deo programa je petlja od 31 koraka u kojoj se računavaju sve nepoznate vrednosti: a, b, aRK, bRK (dok se pritisak menja od 1 do 31 atmosfere).
- Zatim zovemo program "specvol" da izračuna specifičnu zapreminu čija vrednost se zatim čuva u promenljivoj "vol". Faktor kompresibilnosti se čuva u promenljivoj "Z".
- Konačno, komandom "plot" crtamo grafik čije ose obeležavamo komandom xlabel i ylabel.

Rezultat:

- Program sačuvajmo kao m-fajl pod nazivom run_vplot.m i to u direktorijumu u kome se nalazi i m-fajl pod nazivom spacevol.m



- Njegovim pokretanjem u Command-prozoru dobijamo sledeći grafik:
- m-fajl "spacevol" treba malo promeniti da bi koristio globalnu komandu:

```
function y=specvol(v)
global T p Tc pc R aRK bRK
y=p*v^3-R*T*v^2+(aRK-p*bRK^2-R*T*bRK)*v-aRK*bRK;
```

Male promene u programskom kodu:

- Navedeni programski kod **nije jedini način** da izračunamo željenu vrednost. Kao alternativa, nameće se mogućnost računanja zapremine drugom metodom, što se postiže ukoliko se umesto izraza u petlji:

```
Z(i)=pres(i)*vol(i)/(R*T);
```

van petlje upiše:

```
Z=pres.*vol/(R*T);
```

- Treba primetiti izraz "**.***" koji navodi program da izvršava proračune element po element.
- Naravno, dobijeni rezultat biće isti u oba programa.
- Svaka simulacija može biti izvedena na više načina i to treba imati u vidu prilikom pisanja programa i traženja grešaka unutar njega.

Zadatak: Napraviti obe verzije programa i videti koji od njih će brže izvršiti izračunavanje. Upotrebiti za ovu svrhu veći opseg pritisaka i povećati broj koraka. Fajlove priložiti kao: "ime_prezime_naziv fajla.m"

Simulacije procesa koji se javljaju u "realnom životu":

- Simulacija može predstavljati i kompjuterski eksperiment koji prikazuje neke aspekte "realnog života" koji se često zasnivaju na slučajnim događajima.
- Koji procesi se mogu smatrati slučajnim je već filozofsko pitanje i prevazilazi okvire ovog kursa, ali kao primer možemo navesti: radioaktivni raspad, bacanje kockice, bacanje novčića kao i deoba bakterija.
- Suština simulacije "realnog života" je da programer nikada ne može tačno da predvidi rezultat eksperimenta (s obzirom da ne poznaje vrednosti ulaznih parametara koji se generišu slučajno).
- Na primer, kada bacite novčić nikada sa sigurnošću ne možete da tvdite da li ćete dobiti krunu ili pismo.

Generisanje slučajnih brojeva u programu MATLAB:

- Slučajni događaji se lako mogu simulirati u MATLAB-u pomoću komande "rand".
- Ova komanda kao rezultat daje nasumično izabran broj koji je: 0≤ rand <1.
- Kompjuter, naravno, ne može da generiše pravi nasumični broj ali može generisati broj koji je praktično nepredvidljiv (pseudoslučajni broj).
- Ukucati u komandnom prozoru: "rand"

```
>> rand

ans =

    0.8147
```

- Matricu slučajnih brojeva sledećom generišemo sintaksom:
- Broj u zagradi označava kvadratnu matrucu "nxn" gde je n-broj koji stoji u zagradi

```
>> rand (2)

ans =

    0.1361    0.5797
    0.8693    0.5499
```

- Komandom: "rand (broj redova,broj kolona)" dobijamo redove i kolone koji se sastoje od nasumičnih brojeva.
- Pogledajmo primer:

```
>> rand (2,3)

ans =

    0.1450    0.6221    0.5132
    0.8530    0.3510    0.4018
```

Pitanje: Da li je "rand" broj zaista slučajan? Nakon prvog pokušaja tražiti još nekoliko "slučajnih" brojeva. Izbrisati Command Window, Command History i Workspace. Izaći iz programa MATLAB. Ponovo ući u program. Ukucati "r = rand(5,1)" ... ponoviti ovo više puta ... koji brojevi se dobijaju?

Generisanje slučajnih brojeva u programu MATLAB:

```
>> a=3

a =

     3

>> b=5

b =

     5

>> r = a + (b-a).*rand(10,1)

r =

    3.0308
    3.0860
    3.3380
    4.2982
    4.4634
    4.2955
    3.9018
    4.0940
    3.5926
    4.4894
```

- Znači komandom: "rand" dobijamo nasumične brojeve u intervalu od [0,1). Uradimo sledeći zadatak:
- **Zadatak:** Upotrebom komande "rand" dobiti 10 nasumično raspoređenih brojeva u intervalu od [a,b).
- Primer: Recimo da je to interval od [3,5). Sintaksa je sledeća (zapaziti tačku iza zagrade).
- Napravimo sada u "Editor" prozoru kompletan program koji kontroliše vrednost brojeva koji se unose kao interval.
- Zapazimo komandu "return" koja prekida program ukoliko je ispunjen određeni uslov.

randint.m

```
%Program koji pravi 10 nasumicnih brojeva u zadatom intervalu
clear all
a = input('Kolika je pocetna vrednost intervala? ');
b = input('Kolika je krajnja vrednost intervala? ');
if (a>b)
    disp('Prva vrednost mora biti manja a ne veca od druge!')
    return
elseif (a==b)
    disp ('Prva vrednost mora biti manja a ne jednaka drugoj!')
    return
else r = a + (b-a).*rand(10,1)
end
```

Zadatak: Vežbati upotrebu komande "rand" praveći varijacije napisanog programa uz zadavanje različitih intervala i uslova.

Napravimo umetničko delo :)

- Iskoristimo dosadašnje znanje da napravimo jedno umetničko delo.

```
x =rand(2,10000);
y =rand(2,10000);
plot (x,y)
```

ili

```
x = -1.0 : 0.01 : +1.0;
y = -1.0 : 0.01 : +1.0;
[ X, Y ] = meshgrid ( x, y );
Z = abs ( X .* X + Y .* Y - 0.75 );
surf ( X, Y, Z, ...
'FaceColor', 'Interp', 'EdgeColor', 'Interp' )
xlabel ( 'X' );
ylabel ( 'Y' );
zlabel ( 'Z=F(X,Y)' )
title ( 'Z = | X^2 + Y^2 - 3/4|' )
```

ili još lepše:

```
xyz = randn ( 3, 1000 )
for j = 1 : 1000
xyz(:,j) = xyz(:,j) ./ norm ( xyz(:,j) );
end
scatter3 ( xyz(1,:), xyz(2,:), xyz(3,:) )
```

... ali detaljnije o značenju ovog programskog koda ... nešto kasnije

Simulacija bacanja novčića:

- Kada se baca novčić, verovatnoća da će se dobiti željena strana je 50% (ili 0.5). Kako bacanje novčića predstaviti u mat. formi?
- Pošto je vrednost koja se dobija *rand* komandom [0,1) uzmimo da glavu (G) predstavlja broj manji od 0.5 a pismo (P) broj veći ili broj jednak 0.5.
- Kako bi izgledao model koji simulira bacanje novčića 50 puta pri čemu bi se svaki put zabeležio dobijeni rezultat?

```
for i=1:50
    r = rand;
    if r<0.5
        fprintf('G')
    else
        fprintf('P')
    end
end
```

bacinovcic1.m

ili lepše:

```
for i=1:50
    r = rand;
    if r<0.5
        fprintf('G')
    else
        fprintf('P')
    end
end
fprintf( '\n' ) %newline
```

bacinovcic.m

da bi kursor ostao u sledećem redu.

Pitanje:Zašto program oblika

```
if rand < 0.5 fprintf(  'G'  ), end
if rand >= 0.5 fprintf(  'P'  ), end
```

nije dobar za ovu simulaciju?

Odgovor:

Zato što je to onda model koji simulira 2 nezavisna dogadjaja!

Ne može se za jedno "vrtenje" novčića komanda *rand* pozivati dva puta.

Kako možemo izbrojati slučajne događaje?

- Slučajne događaje možemo izbrojati pomoću komande *sum* ili upotrebom brojača. Probajmo prvi način:
- Napravimo m-fajl "sumiranje" oblika:

```
r = rand (1,7)
sum ( r < 0.5 )
```

```
r =
    0.0942    0.5985    0.4709    0.6959    0.6999    0.6385    0.0336

ans =
     3
```

- Znači u ovom slučaju imamo odgovor da su tri nasumično generisana broja manji od 0.5.
- Šta će se desiti ako iza r=rand (1,7) stavimo ";"

Zadatak: Napraviti m-fajl "sumiranjesuma" koji broji koliko slučajno generisanih brojeva je < 0.5 a takođe i koliko je ≥ 0.5 a zatim daje sumu ove dve sume.

A drugi način?

- Slučajne događaje možemo izbrojati i upotrebom brojača.
- Napravimo m-fajl "sumbrojac" oblika:

```
% pocetak programa
a = 0;    % brojevi koji su >= 0.5, brojac se postavlja na 0
b = 0;    % brojevi koji su < 0.5, brojac se postavlja na 0
for n = 1:5000  % pocetak ciklusa
   r = rand;   % generise se jedan broj po ciklusu
   if r >= 0.5 % prvi uslov
      a = a + 1;  % brojac 1
   else
      b = b + 1;  % brojac 2
   end;    % kraj if uslova
end        % kraj for uslova
disp ( ['manje od 0.5: ' num2str(a)] )
disp ( ['vece ili jednako od 0.5: ' num2str(b)] )
```

program "sumbrojac"

```
manje od 0.5: 2485
vece ili jednako od 0.5: 2515
```

rezultat

Zadatak: Napraviti m-fajl "sumbrojact" koja će meriti vreme za koje se ovaj program izvršava. Statistički, pokretanjem barem 10 izvršenja programa, uporediti to sa vremenom koje je potrebno za obradu istog zadatka pomocu *sum* komande: program - "sumiranjet". Šta zaključujete?

Zaokruživanje brojeva u MATLAB-u

- MATLAB nam može zaokružiti ne-cele brojeve korišćenjem 4 osnovne komande: "floor" (zaokružuje na prvi manji ceo broj), "ceil" (zaokružuje na prvi veći ceo broj), "fix" (zaokružuje ka nuli), "round" (zaokružuje ka najbližem celom broju). Upotrebimo ove komande za rešavanje sledećeg zadatka:
- **Zadatak:** Napraviti program koji poredi svaki slučajno dobijeni broj u intervalu od 0 do 10. Ukoliko se taj broj poklopi sa zadatim uslovom on izvršava zadatu komandu: Komande su: Ako je x=1 ili 2, da se ispiše na ekranu: Verovatnoća 20%; za x=3 ili 4 ili 5 da se ispiše: Verovatnoća 30%; a za ostale brojeve da piše: Verovatnoća 50%.

```
% Skript fajl za upotrebu ceil, radn, switch, case i disp
komandi
x = ceil(10*rand) % Kreira nasumicni broj u opsegu (1-10)
switch x
   case {1,2}
      disp('Verovatnoca = 20%');
   case {3,4,5}
      disp('Verovatnoca = 30%');
   otherwise
      disp('Verovatnoca = 50%');
end
```

verovatnoca.m

- Komanda "ceil" zaokružuje na veći broj onaj koji je dobijen komandom "rand".
- Komanda "switch" stavlja broj "x" pod "prismotru".
- Naredba "case" poredi x (koji je pod prismorom zbog "switch" komande), sa brojevima koji stoje u velikoj zagradi.
- Naredba "disp" daje ispis onoga šta piše u zagradi u okviru apostrofa.
- Naredba "otherwise" primenjuje "disp" komandu za sve ostale slučajeve.

- Zadatak: Napraviti varijaciju zadatog zadatka (za ispis više ili manje brojeva uz različite uslove). Napraviti zatim rutinu koja pomoću petlje ponavlja program "verovatnoca.m" 10 puta.

A bacanje kockice ?

- Kao što je poznato, prilikom bacanja kockice može se dobiti bilo koji broj od 1 do 6

Naravno, ukoliko su kockice ispravne :)

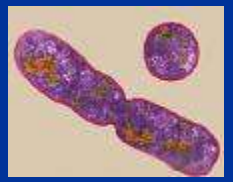
- Tako, ako je slučajno generisan broj **rand** u opsegu [0, 1), tada je **6 * rand** broj koji će biti u opsegu [0, 6), dok ce **6 * rand + 1** biti u opsgu [1, 7) ili preciznije od 1 do 6.999...
- Ukoliko odbacimo decimalni deo broja (tako što dobijenu vrednost zaokružimo na manji broj komandom **floor (x)** dobićemo broj koji se nalazi u željenom opsegu.
- Ukucajmo u komandni prozor:

```
>> d = floor ( 6 * rand (1,10) + 1 )

d =

    4    1    6    6    5    5    5    3    4    2
```

ZADATAK:

- 1. Napraviti program koji traži imput broja bacanja kockice a zatim broji broj šestica koji su dobijeni u ovom nizu bacanja komandom (d == 6) - fajl "kockica".
- 2. Proceniti verovatnoću da se u nizu od 10 bacanja dobije baš 6 (podeliti broj dobijenih šestica sa 10).
- 3. Posmatrati kako se ova verovatnoća menja sa povećanjem broja bacanja kockice i kako se postepeno približava teoretski očekivanoj vrednosti od 1/6 (tj. 0.1667).
- 4. Uočiti da ovde nije bilo moguće iskoristiti komandu **round** (koja zaokružuje vrednosti i ka dole i ka gore).
- 5. Primeniti još neki metod zaokruživanja i komandu "rand" da bi dobili isti rezultat.

Simulacija deobe bakterija:



- Pretpostavimo da se neka vrsta bakterija razmnožava prostom deobom, i to tako, da je verovatnoća za njenu deobu 0.75 (ili 75%). Ukoliko se bakterija ne podeli - ona umire. Naš zadatak je da napravimo program koji simulira ovakav događaj.
- Pošto je brojna vrednost koja se generiše komandom *rand* između 0 i 1, šansa da se dobije broj koji je manji od 0.75 je tačno 75%.
- U skladu s tim, zadatu situaciju možemo simulirati na sledeći način:

```
r = rand;
if r < 0.75
disp( 'Jos uvek postojim' )
else
disp( 'Nema me vise' )
end
```

Program "bakterija"

- Važno je napomenuti da se ovde za svaki događaj mora generisati posebni slučajni broj jer je život svake bakterije poseban događaj u zadatom vremenskom intervalu.

ZADATAK:
Napraviti simulaciju "kolonija" i "kolonijan" za preživljavanje kolonije od 1000 bakterija. Prvi program nam grafički predstavlja život kolinije a drugi nas na kraju obaveštava koliko bakterija je preživelo a koliko ne. Pogledati fajl “temp.m” iz foldera Hemotaksije.

Logičko adresiranje u MATLAB-u

- Upotrebimo MATLAB da bi izveli operaciju logičkog adresiranja.
- Logičko adresiranje je operacija u kojoj se programu zadaje da pronađe ne samo brojeve koji ispunjavaju određene uslove već i na kojim mestima (npr. u matrici) se oni nalaze.
- **Zadatak:** Napraviti matricu od 10 slučajnih celih brojeva u intervalu od 1 do 5 i naložiti programu da nam kao rezultat da novu matricu u kojoj će se nalaziti obeleženo na kojim mestima se u prethodnoj matrici nalaze brojevi koji ispunjavaju određene uslove (npr. uslov identičnosti da je neki broj =1). Rešenje je:

```
clear all
a=ceil(5*rand(1,10))
ind = find(a == 1)
b = a(ind)
```

logadr.m

- Komanda "ceil" zaokružuje na veći broj onaj koji je dobijen komandom "rand" (množi se sa 5 pošto nam treba broj od 1 do 5).
- Komanda "find" pronalazi koji brojevi ispunjavaju uslov u zagradi (znak identičnosti "==") i pravi novu promenljivu - "ind".
- Poslednji red prikazuje matricu a kao skup rešenja koji ispunjavaju uslov identičnosti.

- Zadatak: Napraviti varijaciju zadatog zadatka (za ispis više ili manje brojeva uz različite uslove). Napraviti zatim rutinu koja pomoću petlje ponavlja program "logadr.m" 10 puta.

Logičko adresiranje - dopuna

- Proširimo prethodni zadatak. Sada želimo da nam program pored postojećeg zahteva takođe izlista i novu matricu u kojoj se nalaze obeležena mesta brojeva koji nisu =1, kao i da nam nakon toga da matricu samo sa ovim brojevima (tj. matricu iz koje su izbačene sve jedinice).

```
clear all
a=ceil(5*rand(1,10))
ind = find(a == 1)
bin = find(a < 1 | a > 1)
b = a(ind)
c = a(bin)
```

logadr1.m

```
clear all
a=ceil(5*rand(1,10))
ind = find(a == 1)
bin = find(a ~= 1)
b = a(ind)
c = a(bin)
```

logadr2.m

- U četvrtom redu programa kreira se nova promenljiva "bin" koja predstavlja matricu na kojim mestima se iz "a" nalaze brojevi koji nisu == broju 1.
- **Za ovo je korišćen znak "|" koji u prevodu znači "ili".**
- Takođe, da bi nam se prokazala matrica "a" bez brojeva 1, se u šestom redu formira matrica "c".
- Ovaj problem se može rešiti i ako se u četvrtom redu napiše **komanda "~" koja znači "nije".**

- Zadatak: Rešiti sledeći zadatak (nakon obrade prethodnih primera). Generisati matricu 10 slučajnih brojeva u intervalu od 1 do 10. Izlistati mesta u matrici na kojima se nalaze brojevi koji su: veći od 1 a manji od 4, i prikazati tu matricu. Koristiti komandu "&" koja znači "i".

logadr3.m

Grafički prikaz jednačina kretanja (pojam pravljenja strukturnog plana programa)

- Nakon što smo apsolvirali simulacije koje se zasnivaju na slučajnim događajima, pokušajmo da napravimo prikaz "dobro poznatih" jednačina iz mehanike.
- Zadatak je da grafički prikažemo rastojajanje koje pređe kamen bačen u visinu u zavisnosti od vremena.
- Da bi to uradili, upotrebićemo jednačinu koja opisuje pređeni put u zavisnosti od ubrzanja i vremena.

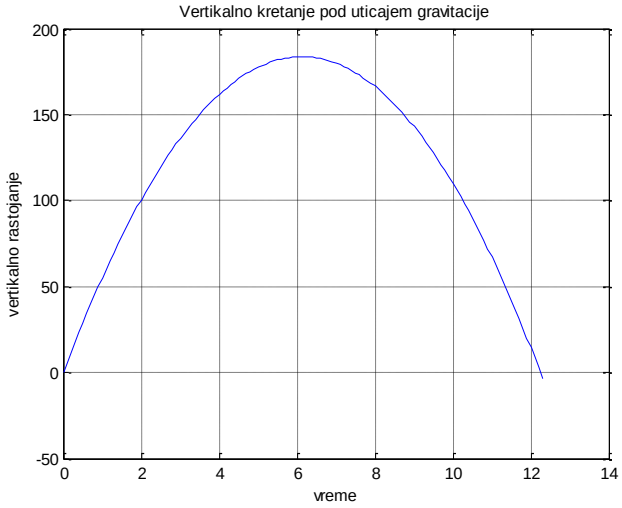
$$s=ut-gt^2/2$$

- Zanemarimo otpor vazduha, i kao zadatak uzmimo da je kamen leteo 12.3 s, sa u=60m/s, a pređeni put racunajmo u intervalima od 0.1 s.
- Napravimo prvo strukturni plan kako bi program trebalo da izgleda: (pravljenje strukturnog plana programa veoma je važan korak i znatno smanjuje mogućnost pojave grešaka i programu).

Grafički prikaz jednačina kretanja (pravljenje strukturnog plana programa)

1. % Pripisati podatke (*g*, *u*, *t*) MATLAB promenljivima
2. % Izračunati vrednost s pomoću poznate formule
3. % Nacrtati grafik s=f(t)
4. % Stop.

- lako ovo izgleda trivijalno, veliki broj početnika redovno prvo pristupa koraku broj 2 umesto 1.




```
% Vertikalno kretanje pod uticajem gravitacije
g = 9.8; % gravitaciono ubrzanje
u = 60; % pocetna brzina (m/s)
t = 0 : 0.1 : 12.3; % vreme u sekundama
s = u * t - g / 2 * t.^2; % predjeno rastojanje u metrima
plot(t, s), title( 'Vertikalno kretanje pod uticajem gravitacije' ), xlabel( 'vreme' ), ylabel( 'vertikalno rastojanje' ), grid
```

Program:"vkretanje.m"

Objasnimo malo ovaj program:

- Četvrta linija od vremena "t" pravi vektor (biće kasnije detaljnije objašnjeno kako)
- Peta linija uračunava svaki vektor "t", tj. pravi nov vektor "s". U okviru nje operacija t.^2 (sa tačkom iza t) diže na kvadrat svaki element od "t" (to je *array* operacija i razlikuje se od operacije dizanja vektora na kvadrat što je matrična operacija).

Pitanje: Sta se dešava ako iza t u petom redu programa ne stavimo tačku?
Zadatak: Napraviti program koji prvo traži da se unese "u" , "t" i korak (program nazvati "vkretanjevar.m")

Nađimo sada sa grafika vreme za koje će kamen pasti:

```
clear all
g = 9.8; % gravitaciono ubrzanje
u = 60; % pocetna brzina (m/s)
t = 0 : 0.1 : 12.3; % vreme u sekundama
s = u * t - g / 2 * t.^2; % predjeno rastojanje u metrima
plot(t, s), title( 'Vertikalno kretanje pod uticajem gravitacije' ), xlabel(
'vreme' ), ylabel( 'vertikalno rastojanje' ), grid

% Zatim pronalazimo koliko je vremena proslo da bi kamen pao (u
okolini tacke 12 procitanoj sa grafika)

f = @(x) u*x-g/2*x^2;
z = fzero (f,12);
disp ( ['vreme za koje ce kamen pasti je: ' num2str(z) '(s)']);
```

Program:"vkretanje1.m"

Vežba: U “help” meniju ukicati ”function_handle” i videti značenje komande sa “@”.

Vežba: Function handle

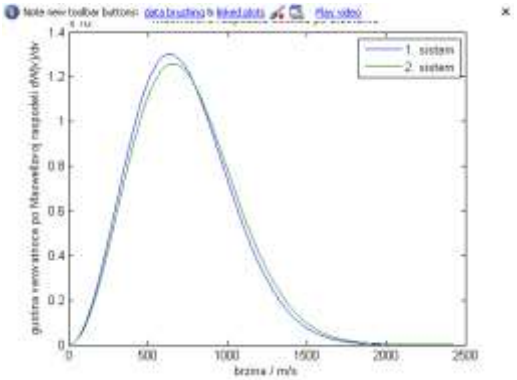
Zadatak: Igrati se malo sa upotrebom ove korisne sintakse.
Na primer:

```
>> sqr(4)
??? Undefined function or method 'sqr' for input arguments of type 'double'.

>> sqr=@(x) x.^2

sqr =

    @(x)x.^2

>> sqr(4)

ans =

    16
```

Simulacija Maksvelove raspodele :

- **Zadatak:** Napraviti program koji vrši izračunavanja i grafički prikazuje Maksvelovu funkciju raspodele čestica po brzinama na osnovu unete temperature sistema (u kelvinima) i mase jedne čestice sistema (u atomskim jedinicama mase).

```
Unesi broj sistema (0<s<5):          s = 2

PARAMETRI 1. SISTEMA:
  Unesi temperaturu sistema (T/K):   T = 295
  Unesi masu cestice (m/amu):        m = 12

PARAMETRI 2. SISTEMA:
  Unesi temperaturu sistema (T/K):   T = 315
  Unesi masu cestice (m/amu):        m = 12

KARAKTERISICNE BRZINE 1. SISTEMA (T=295K i m=12.00 amu):
  Najverovatnija brzina vmax iznosi:      637.55 m/s
  Srednja brzina vsr iznosi:              719.39 m/s
  Srednja kvadratna brzina vsk iznosi:    780.83 m/s

KARAKTERISICNE BRZINE 2. SISTEMA (T=315K i m=12.00 amu):
  Najverovatnija brzina vmax iznosi:      658.80 m/s
  Srednja brzina vsr iznosi:              743.38 m/s
  Srednja kvadratna brzina vsk iznosi:    806.87 m/s
```



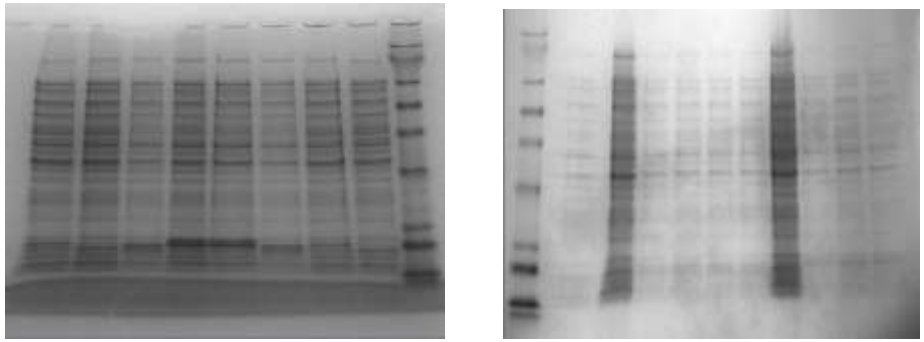
Program:
maxwell.m (sajt ffh)

Potprogrami:

unos.m	vmaxi.m
brzine.m	vsred.m
grafik.m	vsrkv.m
gvmaxi.m	
gustinaverovatnoce.m	

MATLAB - analiza slike:

- Cilj je upoznati se sa MATLAB okruženjem koje se koristi za matematičko modeliranje i obradu digitalnih slika.
- Šta će to nama fizikohemičarima? Kakvu to ima primenu u fizičkoj hemiji ?
- **Primer 1**: Obrada MRI slike. Snimite MRI sliku pacijenta i sada treba neko da protumači šta se to u stvari vidi. Da li je ovo samo zamućenje slike ili ne? Koja je i kolika ova regija? Koji deo zahvata?
- **Primer 2**: Obrada slika koje su dobijene razdvajanjem proteina putem elektroforeze.
- A **3D** slike? Kako odrediti region u prostoru?



Tipične slike dobijene 2D SDS gel-elektroforezom proteina

MRI slika sagitalnog preseka glave

Ali pre nego što počnemo...:)

- Moramo da se podsetimo nekih osnovnih stvari iz matematike tj. da ponovimo deo koji se odnosi na matrice.
- U MATLAB-u sve je matrica (stara kineska poslovica :) Struktura programa koje ćemo učiti da pišemo se zato mora zasnivati na matricama (za razliku od klasičnog "C" jezika ili JAVE o kojima neće biti reči na ovom kursu).
- Ne zaboravite da ukoliko vam nešto nije jasno, uvek možete da ukucate komandu: help *komanda* i dobijate objašnjenje.
- Ukucajmo komandu:

```
>> x = [ 1 2 3 4 5 6 7 8 9 10 ];
```

  i dobili smo matricu od 1 reda i 10 kolona (1x10).
- Uradimo to jednostavnije tako što ćemo da ukucamo komandu:

```
>> x = 1:10;
```

(Ovde je uneta samo prva i poslednja vrednost matrice, a korak je 1 po difoltu)

Analiza slike - matrice:

- Probajmo da promenimo korak u prethodnoj matrici:

```
>> z = 0 : 0.1 :20;
```

- Ukoliko 2x kliknemo na matricu z u "Workspace" prozoru videcemo matricu sa novim korakom (šta je MATLAB sada razumeo da nam treba ?)
- Ukoliko želimo da napravimo matricu (10x1) upotrebićemo komandu:

```
>> y = [1;2;3;4;5;6;7;8;9;10];
```

- Ili jednostavnije:

```
>> y=[1:10]';
```

ili, ako smo sačuvali prethodnu matricu:

```
>> y=x'
```

- Pokušajmo sada da pomnožimo matrice x (1x10) i y (10x1). Prvo da se podsetimo da je množenje matrica dozvoljeno samo ukoliko prva matrica ima isto redova koliko druga kolona. Tako proizvod x*y daje matricu (1x1) tj. broj, dok proizvod matrica y*x daje matricu (10x10).
- Kako beše ide ovaj postupak:

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1\times 3+0\times 2+2\times 1 & 1\times 1+0\times 1+2\times 0 \\ -1\times 3+3\times 2+1\times 1 & -1\times 1+3\times 1+1\times 0 \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}$$

Analiza slike - matrice:

- Mnozenje matrica x i y vršimo komandom:

```
>> x*y

ans =
   385
```

- Na isti način možemo pomnožiti matrice i komandom y*x (dobija se matrica 10x10).
- Šta će da se dogodi ako probamo da pomnožimo matrice x*x ili y*y ??? Naravno, dobijamo obaveštenje o grešci.

```
>> x*x
??? Error using ==> mtimes
Inner matrix dimensions must agree.
>> y*y
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

- **Važno da ponovimo:** Ukoliko želimo da množimo matrice postupkom "element po element" (što je često slučaj) ispred operatora samo upišemo tačku

Analiza slike - matrice:

- Evo kako to treba da izgleda:

```
>> x.*x
```

```
ans =
1   4   9   16   25   36   49   64   81   100
```

- Kako napisati matricu sa više redova i kolona (recimo 4x4)?
- Ukucajmo ovu matricu na sledeći način:

$$mat = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

```
>> mat = [1 2 3 4;2 3 4 5;3 4 5 6;4 5 6 7];
```

- Ukoliko želimo da "pozovemo" ili prikažemo određenu vrednost iz matrice (recimo element u drugom redu i drugoj koloni), ukucajmo komandu:

```
>> mat (2,2)
```

- Moguće je izvršiti neku matematičku operaciju nad svim elementima neke matrice. Recimo, pokušajmo da pripišemo vrednost funkcije sin(x) elementima matrice z.

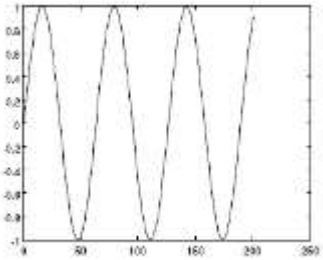
```
>> s1 = sin (z);
```

Probati: Ukucati i alternativni način množenja element po element: >> x.^2 ili >> y.^2
Zapaziti: Da li tačku možemo alternativno ukucati i iza drugog množioca?

Analiza slike - prikaz matrice:

- Da bi grafički prikazali novonastalu **1D** matricu **s1** upotrebimo komandu:

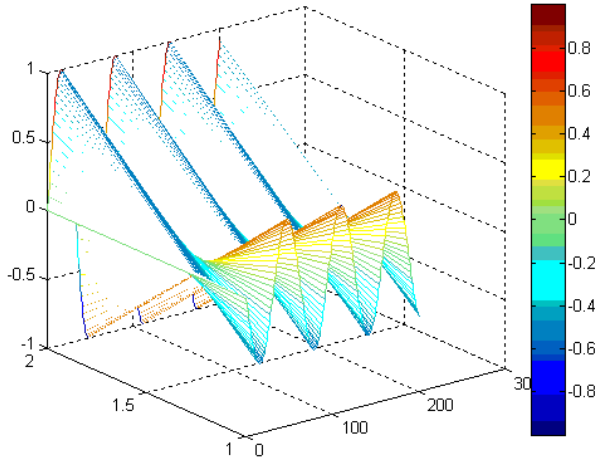
```
>> plot(s1);
```



- Možemo da iskoristimo redove i kolone ove matrice za elemente druge matrice, npr.

```
s2(1,:) = -s1/2;
s2(2,:) = s1;
s2(3,:) = s1 * 4;
```

%prvi red matrice s2
%drugi red matrice s2
%treci red matrice s2



- Za grafički prikaz matrice **s2** u **3D** okruženju koristimo komadu:

```
>> mesh(s2);
```

Analiza slike - osnovne stvari:

- Dobro, lepo smo ponovili matrice, videli smo da MATLAB može da ih generiše, računa, pa i da ih grafički prikaže. Ali kakve to ima veze sa obradom slika ???
- Ima i to velike. Zašto ???
- Zato što svaku sliku možemo da posmatramo kao funkciju **I=f(x,y)**, tj. za svaku koordinatu slike **(x,y)** postoji neka vrednost funkcije **I**.
- Ovo znači da svaku sliku možemo da posmatramo kao 2D matricu u kojoj svaki element matrice odgovara jednom elementu slike koji se naziva **"piksel"**.
- Tipična veličina slike tako može biti kvadrat dimenzija $\mathbf{2^n x 2^n}$ na primer: 128x128; 256x256; 512x512; 1024x1024 ... piksela.
- U opštem slučaju, slika može biti dimenzija $\mathbf{2^n x 2^m}$, kao sto je poznatih 800x600; 1024x768 .... piksela.
- Vrednost funcije koja se pripisuje pikselu može biti različita. Tako ona može na primer označavati: osvetljenost, kontrast, zamućenost, informaciju o dubini ....

Informacija o intenzitetu je od suštinskog značaja za segmentaciju slike

O.k. ovo je lako shvatljivo za crno-bele slike, ali na koji način možemo da predstavimo matricu slika koje sadrže različite boje?

Analiza slike - osnovne stvari:

- Slike u boji nose dodatnu informaciju koja je povezana sa percepcijom talasnih dužina koje su iz oblasti vidljivog dela elektromagnetnog spektra.
- Tako, informacije o boji u većini slučajeva možemo svesti na:

  1) podatke o prisutnosti: **crvene, zelene i plave** (zapaziti da nisu tri osnovne boje)

  2) ostali podaci: **obojenost, zasićenost i intenzitet**.

- Prikaz **intenziteta** nekog piksela je od posebne važnosti jer govori o tome koliko je svetlo ili tamno obojen (i da li ga na taj način možemo razlikovati od susednog piksela).
- Postoje dve osnovne kvalifikacije pomoću kojih prikazujemo broj koji reprezentuje osvetljenost piksela:

  1) "***double class"*** (dodeljuje broj između 0 i 1 svakom pikselu). Vrednost **"0"** odgovara **crnoj** a vrednost **"1"** odgovara **beloj** boji.

  2) "***unit 8"*** (pripisuje broj između 0 i 255 koji određuje osvetljenost piksela). Takođe, vrednost **"0"** odgovara **crnoj** a vrednost **"255"** odgovara **beloj** boji.

- **Unit 8** zauzima samo 1/8 prostora u poređenju sa **double class**. Treba napomenuti da većina mat. fja. podržava samo rad sa **double class** brojevima.

Analiza slike - prikaz slika:

- Pošto smo se upoznali sa osnovnim parametrima koji karakterišu neku sliku i o vezi slike sa matricama (koje možemo kasnije dodatno obrađivati u MATLAB-u), pokušajmo da uvezemo neku sliku u MATLAB, razdvojimo je po komponentama boje i pretvorimo u željenu matricu.
- Slika se uvozi komandom **"imread"** a prikazuje komandom **"imshow"**. Napravimo jegan m-fajl **"slike.m"** i ukucajmo sledeće komande:

```
im1 = imread ('cvet.tif');          %Čita sliku "cvet.tif" i smešta je u matricu im1
im2 = imread ('mesec.tif');         %Čita sliku "mesec.tif" i smešta je u matricu im2
im3 = imread ('mri.tif');           %Čita sliku "mri.tif" i smešta je u matricu im3
im4 = imread ('slika.tif');         %Čita sliku "slika.tif" i smešta je u matricu im4
figure(1)                           %Otvara prozor za sliku
imshow(im1);                        %Prikazuje sliku
```

- Znači dobili smo prikaz prve slike. Da bi prikazali svaku sliku posebno, moramo da u komandi "imshow" ukucamo naziv željene matrice.
- A kako prikazati sve 4 slike u jednoj ? To se radi komandom **"subplot"** koja ima zadatak da slike (tj. njihove matrice) rasporedi u prostoru. Dopunimo m-fajl ...

Analiza slike - razlaganje:

- Ukucamo dodatne linije programa i dobićemo m-fajl **"sliked.m"**.

```
figure(2)                          %Otvara novu sliku
subplot(2,2,1); imshow(im1);       %Ubacuje prvu sliku gore levo
subplot(2,2,2); imshow(im2);       %Ubacuje prvu sliku gore desno
subplot(2,2,3); imshow(im3);       %Ubacuje prvu sliku dole levo
subplot(2,2,4); imshow(im4);       %Ubacuje prvu sliku dole desno
```

2 (broj redova
2 (broj kolona)
1 (redni broj slike)

- Razdvojmo sada sliku na osnovne komponente (**crvenu, zelenu i plavu**):

```
im1 = imread ('cvet.tif');         %Cita sliku "cvet.tig" i smešta je u matricu im1
figure(1)                          %Otvara prozor za sliku
subplot(2,2,1)                     %Odredjuje mesto prikaza originalne slike
imshow(im1)                        %Prikazuje originalni sliku 1
subplot(2,2,2)                     %Odredjuje mesto prikaza slike
imshow(im1(:,:,1))                 %Prikazuje crvenu componentu slike
subplot(2,2,3)                     %Odredjuje mesto prikaza slike
imshow(im1(:,:,2))                 %Prikazuje zelenu componentu slike
subplot(2,2,4)                     %Odredjuje mesto prikaza slike
imshow(im1(:,:,3))                 %Prikazuje plavu componentu slike
```

Fajl:
"sliker.m"

ZADATAK:
Napraviti m-fajl u kome će i ostale slike biti razložene po bojama. Promenti redosled prikazivanja slika i rastera.

Analiza slike - *doubles:*

- Dalja manipulacija slikama nije moguća ukoliko su vrednosti osvetljenosti tipa **unit 8.** Zbog toga, za dalji rad, sve vrednosti treba prevesti u **doubles** formu (m-fajl **doublemat.m**).

```
im4=imread ('mri.gif');          %Cita sliku mri.gif
im5=double(im4);                 %Pretvara sliku u doubles matricu
surf(im5); shading flat;         %Primenjuje surf 3D prikaz slike
colormap(gray);                  %Mapira boje u slici kao nijanse sive
colormap(jet);                   %Mapira sliku u boji
rotate3d on;                     %Omogucava 3D rotaciju slike
figure(2)                        %Otvara drugu sliku
imshow(im4)                      %Prikazuje originalnu sliku radi poredjenja
```

- Da bi dobili grafički prikaz **doubles** matrice dovoljno je desnim klikom obeležiti im5 (u Workspace prozoru) i izabrati način prikaza. Da li ovo može da se uradi i za im4 formu ?
- Suština je u tome da kada se jednom slika nađe u **doubles** matričnoj formi, tada se na nju mogu primeniti sve operacije koje se mogu izvoditi sa matricama (sabiranje, oduzimanje, množenje, inverzija, transponovanje).
- Funkcije: **max, min, mean, std, sort** mogu biti veoma korisne za obradu slike u doubles matričnoj formi.
- Da bi mogli potpuno da manipulišemo slikom moramo imati u vidu da sliku moramo digitalizovati i u **prostornim** i u **Intensity** dimenzijama. Malo detaljnije...

ZADATAK:

1. Uraditi prevod u **doubles** formu i ostalih .gif slika. Šta se dobija ako se izostavi 5 linija programa? Da li je ona potrebna za prikaz u crno-beloj formi. Da li važi obrnuto (4 linija potrebna za prikaz u boji?). Da li je 6 linija suvušna?
2. Upotrebiti funkcije **max, min, mean, std, sort** na matricu im5. Za objašnjenje kako se ove komande koriste upotrebiti komandu: help (komanda)
3. Grafički prikazati dobijene rezultate (desni klik na novonastalu matricu i Workspace prozoru).

Analiza slike - digitalizacija:

- Digitalizacija po prostornim koordinatama **(x,y)** zove se ***image sampling*** dok se digitalizacija po amplitudi (tj. intenzitetu) naziva ***grey-level quantization***.
- Predstavimo originalnu digitalnu sliku funkcijom ***I=f(x,y).*** Neka ima dimenzije po redovima i kolonama $N_r \times N_c$.

  - Domen prostornih koordinata možemo obeležiti sa: ***Lr=1,2,3....Nr*** **i** ***Lc=1,2,3...Nc***

  - Domen intenziteta (nijanse sive boje) označavamo sa: $G=1,2,3...N_g$.
- Sliku onda možemo predstaviti kao funkciju koja pripisuje sivi ton svakom paru koordinata: $L_r \times L_c : I : L_r \times L_c \rightarrow G$.
- Napomenimo da je u velikom broju slučajeva moguće dimenzije slike predstaviti kao stepen broja 2, tako da je: $N_r = 2^{nr}$, $N_c = 2^{nc}$, $N_g = 2^{ng}$.
- U skladu sa prethodnim, vidimo da je količina bitova potrebnih za čuvanje slike onda: $b = N_r \times N_c \times N_g$
- Vidimo da od broja elemenata matrice kojom predstavljamo sliku zavisi detaljnost prikaza **(rezolucija)** slike, što utiče na veličinu fajla u kome se slika čuva.

Histogrami i segmentacija:

- Jedan od najpopularnijih tehnika za segmentaciju slike je **tzv.grey level thresholding.** Cilj ove metode je da razvrsta sive tonove slike koji su određeni zadatim parametrima (piksele sa nivoom "sivoće" X±ΔX pripadaju jednoj ili drugoj regiji). Na ovaj način delimo sliku na regije tipa: **objekat A/objekat B/../ pozadina.**
- Problem kod ove metode se javlja u slučajevima kada intenziteti nekih struktura nisu uniformni te se događa da se jedna struktura podeli u više regiona.
- Da bi uradili segmentaciju slike, primenjujemo komandu: **imshow (I,[min max])**
- Napravimo sledeći m-fajl **"segmentacija.m"**

```
im3=imread ('mesecgs.gif');                 %Ucitava sliku "mesecgs.gif"
im4=double(im3);                            %Pretvara sliku u doubles matricu
subplot(2,2,1);hist(im4);                   %Pravi histogram od slike
subplot(2,2,2);imshow(im3,[]);              %Prikazuje sliku sa celim opsegom intenziteta
subplot(2,2,3);imshow(im3,[0 200]);         %Prikazuje sliku u opsegu I od 0 do 200
subplot(2,2,4);imshow(im3,[100 300]);       %Prikazuje sliku u opsegu I od 100 do 300
```

- Pored histograma, ovaj program nam prikazuje kako se slika razlikuje u zavisnosti od min (minimalnog intenziteta ispod koga se sve prikazuje kao crno) i max (maksimalnog iznad koga je sve belo). Međuregije se prikazuju kao nijanse sive.

ZADATAK:

Napraviti m-fajl sa 6 prikaza različitih opsega koji najbolje opisuju segmentaciju neke slike (mri1, mesec, mozak ...).

Promena intenziteta:

- Ponekad je potrebno promenti vrednosti intenziteta slike (recimo da bi izbegli greške u segmentaciji nastalih usled njenog lošeg kvaliteta).
- Sintaksa za ovaj postupak je:

```
imadjust(I,[low high],[bottom top])
```

- Gde su **low-high** opsezi vrednosti intenziteta u početnoj slici koji prelaze u opsege intenziteta **bottom-top**. Napravimo m-fajl koji npr. posvetljuje sliku cvet.tif

```
im7 = imread('cvet.tif');                    %Cita sliku cvet.tif
im8 = imadjust(im7,[0 0.3],[0.5 1]);         %Prebacuje opsege intenziteta
subplot(1,2,1); imshow(im7)                  %Prikaz izvorne slike
subplot(1,2,2); imshow(im8)                  %Prikaz slike korigovanog I
```

"intenzitet.m"

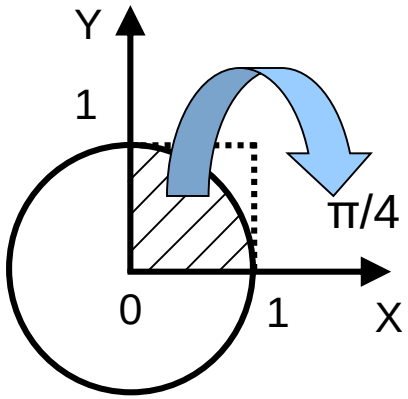
- Na sličan način možemo obrađivati i audio-signale ali ovo prevazilazi okvire ovog kursa (ili bar nema direktne veze sa fizičkom hemijom :)

ZADATAK:

1. Eksperimentisati sa različitim promenama opsega intenziteta na različitim slikama.
2. Videti da li možete postići poboljšanja na polju segmentacije.
3. Da li se mogu otkriti neki novi detalji na slici?

Monte Karlo simulacija i MATLAB:

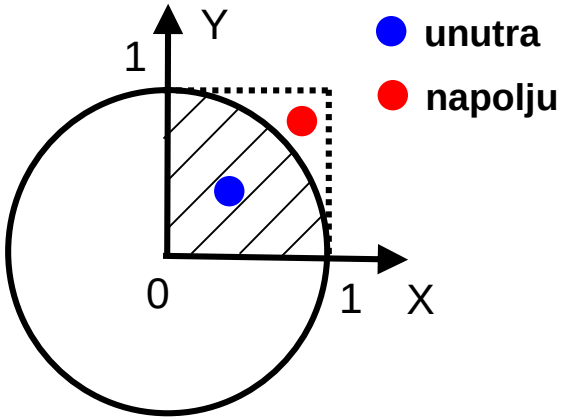
- Kao što je prethodno već bilo napomenuto Monte Karlo simulacije su imale veliki uticaj na razvoj računarskih simulacija u različitim oblastima nauke.
- Da ponovimo, ova tehnika koristi slučajne brojeve da bi modelovala različite vrste fizičkohemjskih procesa.
- Monte Karlo simulacija posebno dobro opisuje procese za koje se zna verovatnoća odigravanja ali ne i konačan ishod događaja.
- Da bi se upoznali sa principom rada ovog tipa programa, napravimo Monte Karlo simulaciju koja će nam omogućiti da izračunamo vrednost broja "π".
- Uradimo ovo na sledeći način: Zamislimo kvadrat dužine stranice 1, koji se nalazi u početku koordinatnog sistema.



- Očigledno je da kvadrat ima površinu 1.
- Posmatrajmo sada upisan krug dužine stranice 1. Lako se može izračunati da je površina četvrtine kruga = **π/4**.
- Napišimo program u MATALAB-u koji izvršava Monte Karlo simulaciju izračunavajući relativni odnos površine kvadrata i prikazane četvrtine kruga.

Monte Karlo simulacija: broj π

- Zašto to radimo? Logika koju pratimo je sledeća:
- Da bi pronašli površinu četvrtine kruga generišimo veliki broj nasumičnih (X,Y) pozicija (naravno da vrednosti X i Y treba da budu između 0 i 1). Pri tom treba odrediti koja od tih tačaka je unutar kruga radijusa 1, a tačka je unutar kruga ukoliko je vrednost korena (X^2+Y^2) **≤ 1.**
- Ukoliko je tačka unutar kruga, dodaćemo brojaču jedan. Nakon što generišemo veliki broj tačaka, odnos broja tačaka koje su unutar kruga prema ukupnom broju generisanih tačaka će približno biti jednak odnosu površine četvrtine kruga i površine kvadrata.
- U skladu s tim vrednost "π" biće:



π=4*(broj tačaka koje su u krugu)/(ukupan broj generisanih tačaka)

- Napišimo program pomoću koga ćemo moći da izvedemo simulaciju za izračunavanje broja π.

Monte Karlo simulacija: broj π

Program **"montecpi.m"**

```
%  Matlab Program koji daje vrednost Pi pomocu slucajnih brojeva
Nrand = input('Koliko zelite slucajnih brojava ');
NInside = 0;                                % Postavlja brojac na 0
for nloops=1:Nrand                          % Pocetak petlje
Xrand = rand;                               % Generise nasumicnu vrednost za X
Yrand = rand;                               % Generise nasumicnu vrednost za Y
Rrand = (Xrand^2 + Yrand^2)^(1/2);          % Nalazi rastojanje tacke od kkordinatnog pocetka
if (Rrand <= 1)                             % Uslov da li je R<=1
NInside = NInside + 1;                      % Ako jeste, brojac dodaje jedan
end                                         % Kraj uslova
end                                         % Kraj petlje
disp(['Ukupno generisanih tacaka je: ' num2str(Nrand) ', Tacaka unutar kruga je: ' ...
num2str(NInside)]);
piapprox = 4*NInside/Nrand;                 % Aproksimacija broja Pi
disp(['  Aproksimacija za broj Pi je = ' num2str(piapprox)]);
```

ZADATAK:

1. Eksperimentisati sa različitim brojem tacaka na osnivu kojih Monte Karlo simulacija aproksimira broj Pi.
2. Postepeno povecavati broj tacaka sve više. Kako se u skladu s tim povećava dužina vremena koje je računaru potrebno da bi izveo simulaciju (dodati programsku liniju za računanje vremena izvršenja programa).

I drugi rade u MATLABU :)

- Pod nazivom MATLAB Toolobox-es mogu da se nađu programi koji su drugi pisali u MATLAB-u.
- Većina njih su besplatni i mogu se ne samo koristiti već i modifikovati po potrebi. Linkovi:

  http://www.mathworks.com/matlabcentral/

  http://stommel.tamu.edu/~baum/toolboxes.html

  http://www.mathtools.net/MATLAB/Biotechnology/index.html

  http://www.sbtoolbox2.org/main.php
- Video tutorijali za ucenje MATLAB-a:

  http://www.mathworks.com/academia/student_center/tutorials/mltutorial_launchpad.html

Simulacije u programu SIMULINK

- Šta je to **SIMULINK**?

- SIMULINK je softverski paket koji se koristi za modelovanje, simuliranje i analiziranje dinamičkih sistema (linearnih i nelinearnih).
- Za modelovanje **SIMULINK** koristi **GUI** (**G**raphical **U**ser **I**nterface) što u mnogome olakšava kreiranje modela u vidu blok-dijagrama (kao da crtamo na papiru sto liči na programe tipa **LABVIEW**).
- Za rad u ovom programu neophodno je poznavanje programa **MATLAB** pa se **SIMULINK** paket često isporučuje zajedno sa njim.

Modelovanje - kako se to radi ?

- Nekoliko saveta kako da počnete sa pravljenjem modela:

- Kada se pravi model, najbolje ga je prvo skicirati na papiru (osmisliti kako bi trebalo da izgleda), pa tek onda sesti za računar.
- Kada se počne sa računarskom fazom izrade modela, prvo treba ubaciti u radni prostor sve neophodne komponente modela (tzv.blokove), pa tek onda pristupiti njihovom međusobnom povezivanju.
- Ukoliko se modeluje više modela koji su slični, treba pokušati da se napravi univerzalni model koji će posle moći da se lako modifikuje za različite situacije.

Najbolje je shvatiti na primeru:

Modelovanje jednačina:

- Otvoriti program **SIMULINK** pritiskom na ikonu:  ili ukucavanjem rutine "simulink" u MATLAB komandnom prozoru.
- ***Primer 1: Pretvaranje Celzijusa u Farenhajte***

  Kao što je poznato, formula za pretvaranje Celzijusovih u Farenhajtove stepene je:

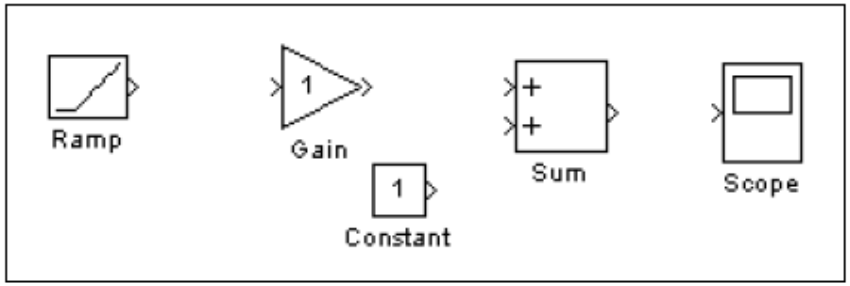
$$T_F = 9/5(T_C) + 32$$

Najpre da razmotrimo koji blokovi (tj. operacije) su nam potrebne da bi rešili ovu jednostavnu jednačinu:

Modelovanje jednačina: *Primer 1*

- Ulazni blok - takozvani "**Ramp block**" (za imput temperature u Celzijusovim stepenima)
- Konstanta - "**Constant block**" (da definiše konstantu tj. broj 32)
- Operacija množenja - "**Gain block**" (da pomnoži ulaznu vrednost sa 9/5"
- Operacija sabiranja - "**Sum block**" (da sabere dobijene vrednosti)
- Prikaz rezultata - "**Scope block**" (da prikaže dobijeni rezulatat)

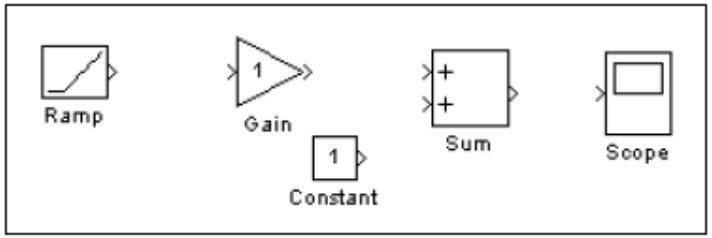
$$T_F = 9/5(T_C) + 32$$



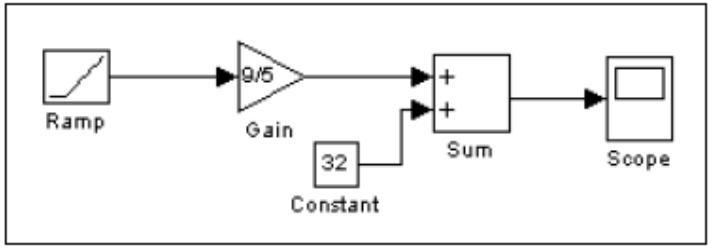
Realizacija Modela:

$T_F = 9/5(T_C) + 32$

- Prvo je potrebno otvoriti novi Model: u okviru **SIMULINK** programa ići na: File-New-Model
- Kao što je prethodno napomenuto prvo treba pronaći sve potrebne blokove i prevući ih na radnu površinu: (Math operations za **Sum (Add)** i **Gain** blokove, Sources za **Constant** i **Ramp** blokove i **Scope** blok iz Sinks foldera).
- Modifikovati **Gain** i **Constant** blokove dvoklikom, i u predviđena polja uneti vrednosti koje se nalaze u jednačini.
- Na kraju, povezati blokove kako je prikazano na slici.



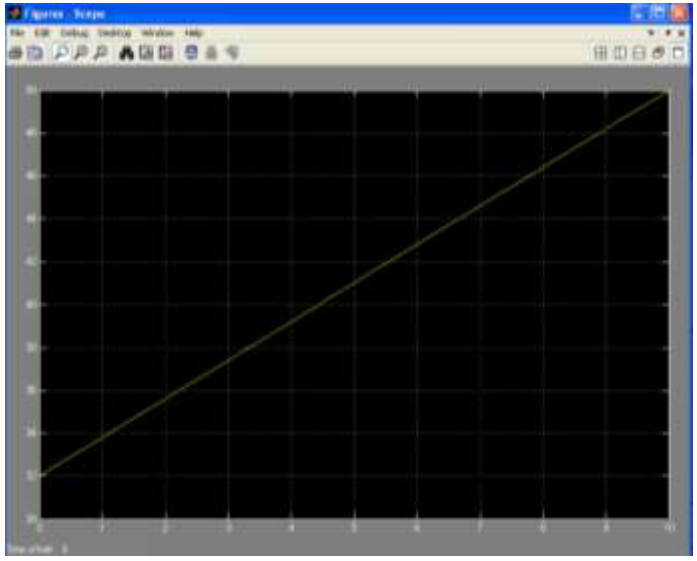
pre



posle

Pokretanje simulacije:

- Da bi dobili grafički prikaz rezultata potrebno je da najpre otvorimo **Scope** blok. U **Ramp** meni zatim unosimo za koji opseg temperatura želimo da nam model da prikaz.
- Simulacija se pokreće opcijom: **Simulation-Start**.



Dobili smo grafik prema kome se Celzijusova skala pretvara u Farenhajtovu. Opseg vednosti koji želimo da vidimo može se menjati opcijom: Simulation-Configuration parameters. Ukoliko želimo prikaz brojnih vrednosti, probati sa drugim prikazima iz "**Sinks**" menija (npr. **Display** opcija). Model sačuvati komandom: File-Save As "CeluFaren.mdl" i "CeluFarensadisp.mdl"

Zadatak: Napraviti program koji traži upis brojne vrednosti temperature u celzijusima i daje ispis kao brojna vrednost u farenhajtima (CeluFarenbroj.mdl). Napraviti modele i simulacije za 3 različite jednačine. Rezultate prikazati grafički i numerički

Simulacija invertora (AC/DC pretvarača):

- Kao sledeći primer simulacije, napravićemo model jednog klasičnog strujnog ispravljača koji naizmeničnu struju (AC) pretvara u pulsirajuću direktnu (DC) struju (setite se diode).
- Da bi ovo uradili moramo da se upoznamo sa još 2 pojma a to su:

  - Uslovno izvršavajući podsistemi

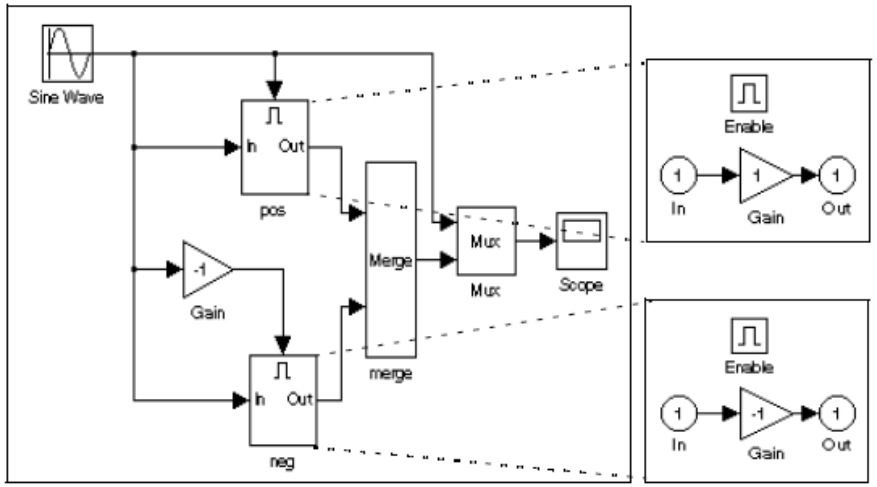
  - Merge (spajajući) blokovi
- **Podsistem (Subsystem)** predstavlja deo sistema čiji elementi međusobno interaguju nezavisno od sistema kao celine. **Uslovno izvršavajući podsistemi** (pod Ports & Subsystems), predstavljaju podsisteme čije izvršenje zavisi od vrednosti ulaznog signala (recimo, puštaju signal samo u slučaju da je ulazna vrednost pozitivna).
- **Merge** (spajajući) blok spaja ulazne signale u jedinstveni izlazni signal koji je određen postavljenim izlaznim prametrima.

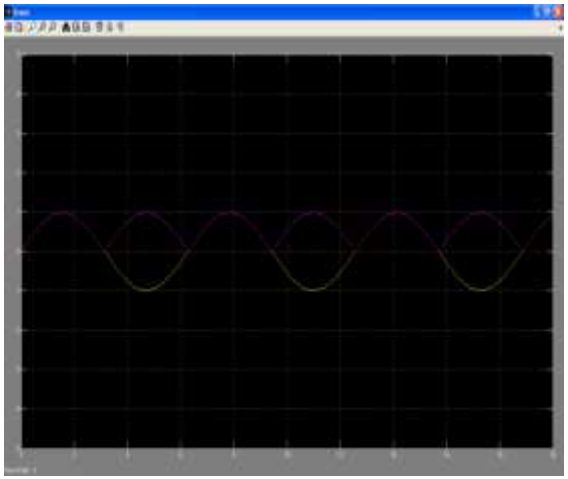
Opet, najbolje je ovo shvatiti na primeru:

Model AC/DC pretvarača:

- U ovom primeru podsistem označen sa "**pos**" je otvoren kada je AC signal pozitivan i strujni signal propušta neizmenjen ka izlazu.
- Podsistem označen sa "**neg**" je otvoren kada strujni signal ima negativnu vrednost i on na svom izlazu ima zadatak da da invertovanu formu signala.
- Blok "**Merge**" sprovodi aktivni izlazni signal do "**Mux**" bloka koji daje na prikaz izlazni signal uklopljen zajedno sa originalnim talasom.

"ACDCconvert.mdl"





Zadatak: Napraviti izlazni signal koji u svom sastavu neće imati i originalni AC već samo pulsirajući DC signal i sačuvati ga kao "ACDCconvertbezAC.mdl"

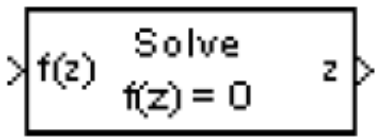
Rešavanje sistema algebarskih jednačina:

- U ovom primeru pokazaćemo kako pomoću programa SIMULINK možemo naraviti model pomoću koga možemo rešavati sisteme jednačina.
- Potrebno je napraviti model za rešavanje sistema koji se sastoji iz 2 jednačine:

$$Z2+Z1=1$$

$$Z2-Z1=1$$

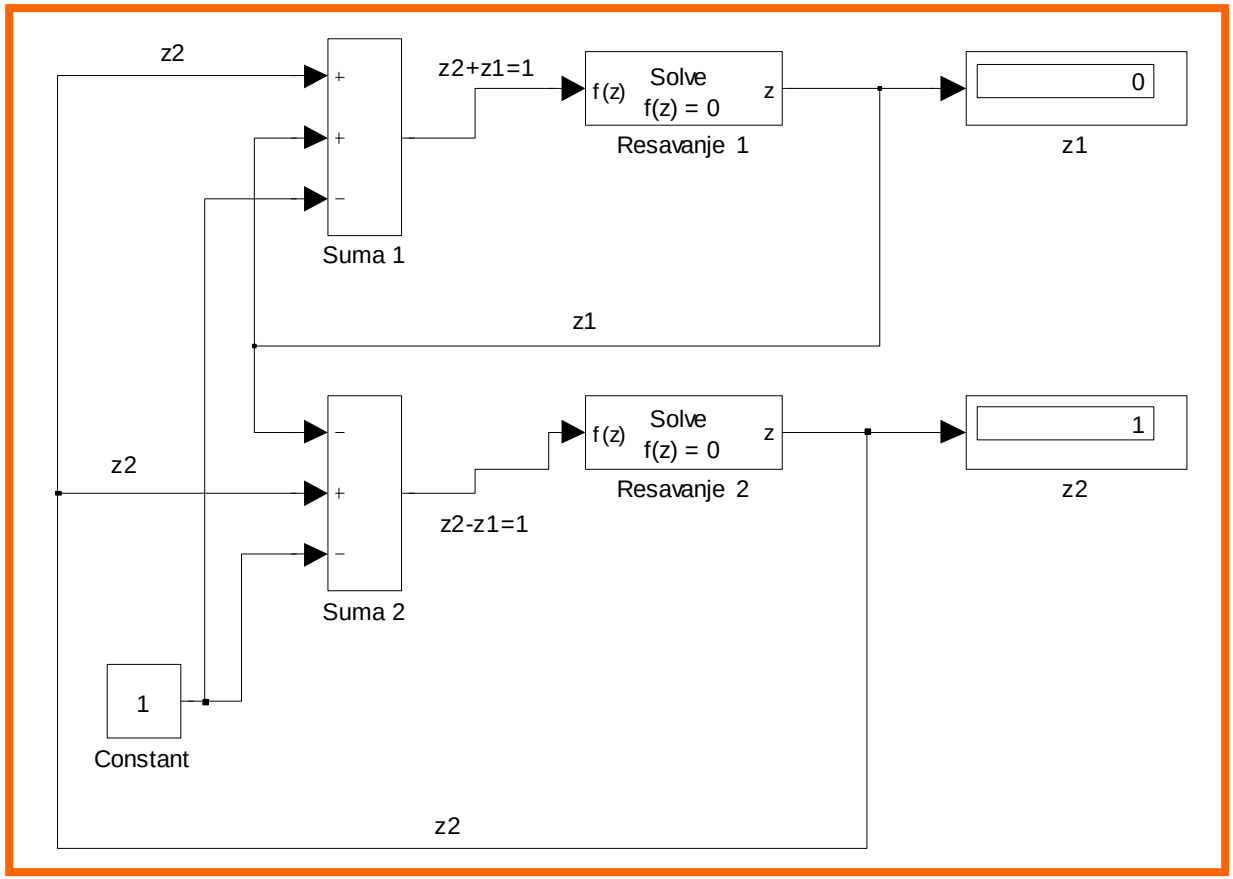
- Kao što se može lako zaključiti, rešenja ovog sistema su: Z2=1 i Z1=0. Napravimo univerzalni model koji nam može poslužiti za rešavanje sistema ovoga tipa.
- Da bi se to uradilo potrebno je pored poznatih uvesti još jedan od blokova a to je blok "Solve" (Simulink-Math Operations-Algebraic Constraint):



... koji se koristi za pronalaženje "0" fje.

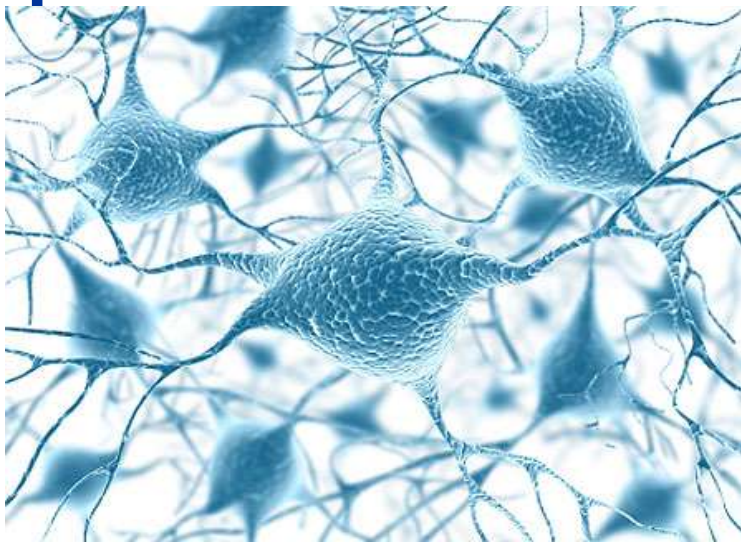
Rešavanje sistema algebarskih jednačina:

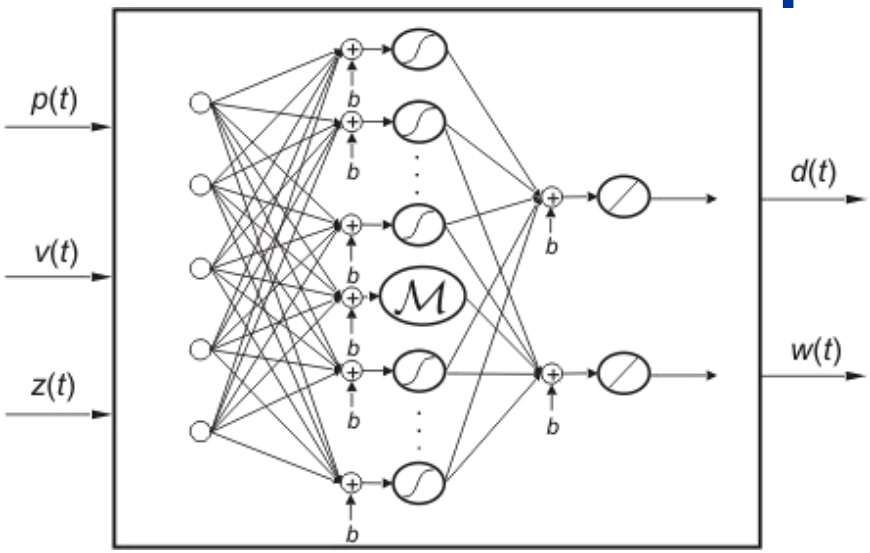
- Napravimo model za sistem: Z2+Z1=1 i Z2-Z1=1 jednacina1.mdl



Zadatak: Napraviti modele za nekoliko različitih sistema jednačina sa 3 ili više nepoznatih (Z2-2Z1=1; Z1+Z2=2.5) jednacina2.mdl

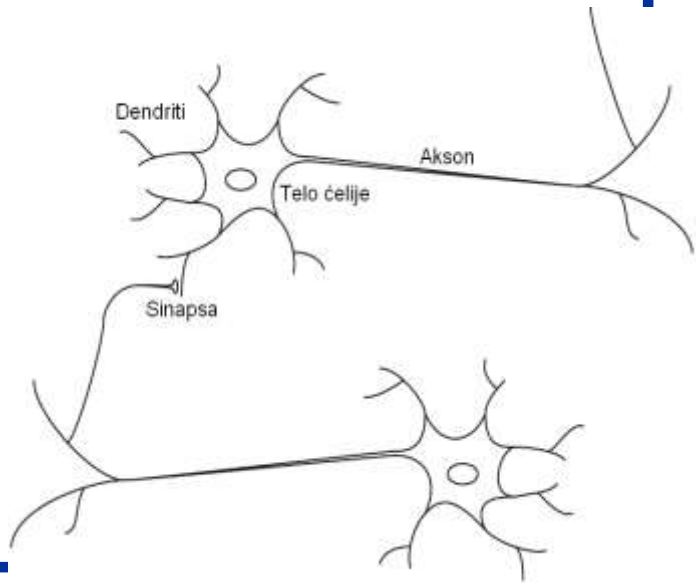
Neuronske mreže





Šta su neuronske mreže?

- Računarske neuronske mreže inspirisane su biološkim nervnim sistemom. Kao i u prirodi, veze između elemenata u velikoj meri određuju funkciju mreže. Neuronska mreža se može naučiti da obavlja određenu funkciju podešavanjem vrednosti veze (tzv. *težine*) između elemenata.
- U pravim biološkim sistemima, neke od neuronskih struktura sa nama su od rođenja, druge su pak, uspostavljene iskustvom. Opšte je prihvaćeno da se nervne funkcije, uključujući pamćenje, obavljaju u neuronima i vezama između njih. Učenje se posmatra kao uspostavljane novih veza i modifikacija starih.
- Neuroni koje ćemo ovde razmatrati nisu biološki. Oni su veoma prosta apstrakcija bioloških neurona, realizovani kao elementi u programu ili možda kao kola od silicijuma.
- Mreže ovakvih veštačkih neurona nemaju ni delić snage ljudskog mozga, ali mogu biti naučene da obavljaju razne korisne funkcije. Veštački neuron je zapravo računarski model inspirisan prirodnim neuronom.



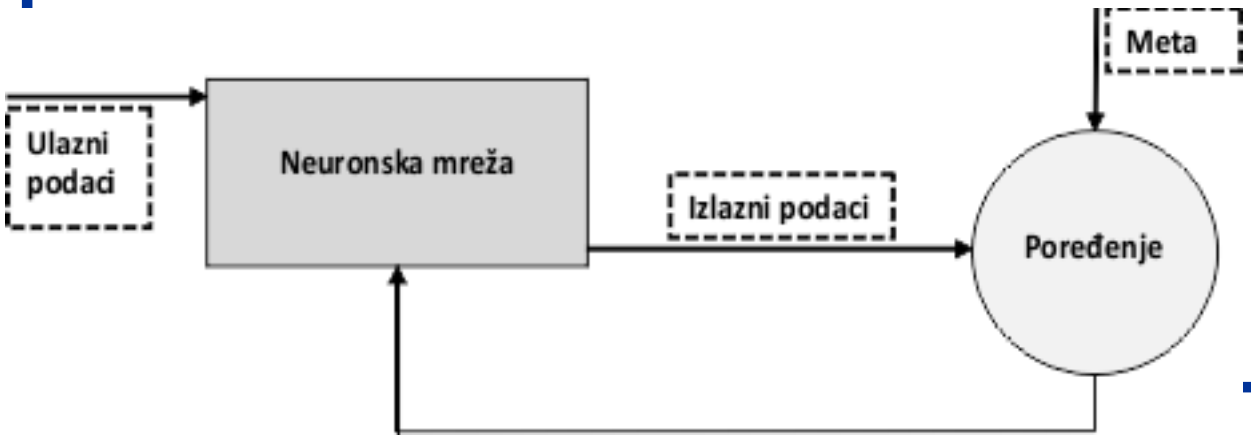
Šematski prikaz dva biološka neurona.

Opšti pregled neuronskih mreža

- Iako veštačke neuronske mreže ne prilaze kompleksnosti moždane strukture, postoje dve ključne sličnosti između bioloških i veštačkih neuronskih mreža.

  (1) Izgrađivački blokovi i jedne i druge mreže su prosti računarski uređaji (mada su veštački neuroni mnogo jednostavniji od bioloških) koji su izuzetno međusobno povezani.

  (2) Veze između neurona određuju funkciju mreže.
- Cilj koji se postavlja pred nama je da odredimo odgovarajuće veze da bi smo rešili određeni problem.
- Obično, neuronske mreže se usklađuju, ili treniraju, tako da određeni ulazni podaci vode do specifičnih izlaznih podataka – mete (očekivane vrednosti izlaznih podataka za date ulazne podatke).

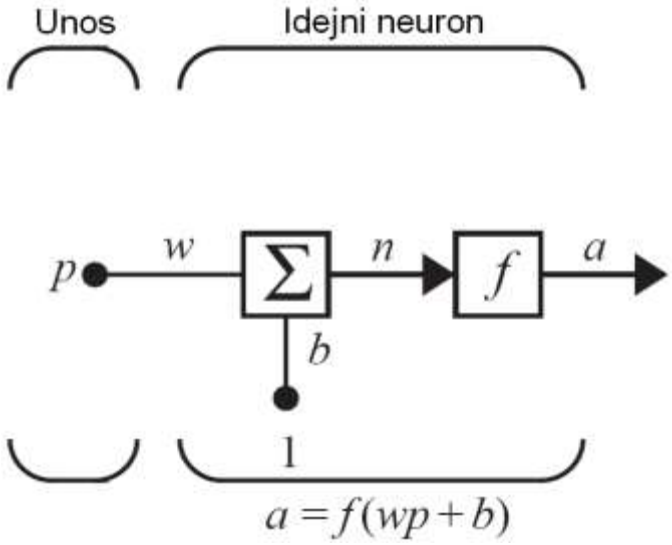


Treniranje mreže je zasnovano na poređenju izlaznih podataka i podataka mete.

Modeli neurona i mrežne strukture

Jednoulazni neuron

- Model jednoulaznog neurona izgrađen je iz više prostih delova od kojih je svaki prikazan i obeležen na slici.
- Skalarni *ulazni podatak* ili *unos p* množi se skalarnom *težinom w* (weight, eng. težina) da bi se formirao proizvod *wp* koji se dalje šalje na sumu.
- Drugi ulazni podatak, 1, umnožen je *pomerajem b* (bias, eng. pomeraj) i prosleđuje se do sume.
- Sumiranjem dobijamo tzv. *mrežni unos n*, koji ide do *funkcije prenosa f* koja proizvodi skalarni *izlazni podatak* ili *iznos a* neurona.



Jednoulazni neuron

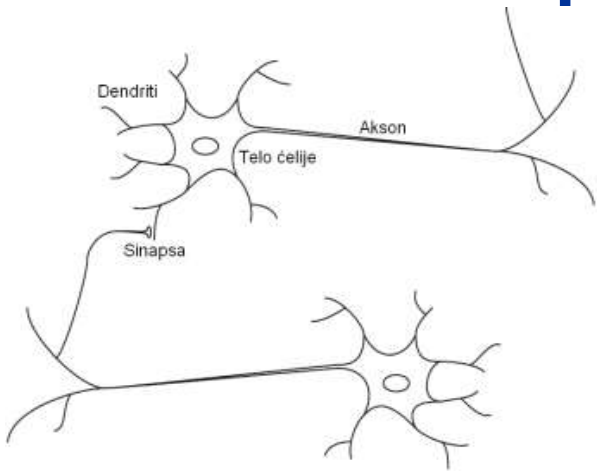
Modeli neurona i mrežne strukture

- Ako povežemo ovaj prosti model sa biološkim neuronom videćemo da težina *w* odgovara jačini sinapse, telo ćelije predstavljeno je sumom i funkcijom prenosa, a izlazni podatak *a* signalom na aksonu.
- Izlazni podatak računa se po formuli:

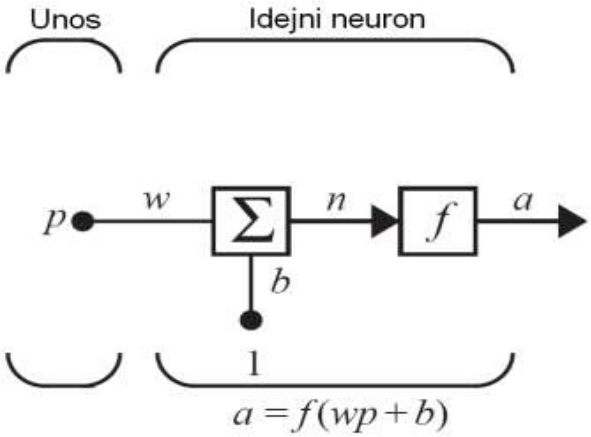
$$a = f(n) = f(wp + b)$$

(Ako je na primer: $w = 3$, $p = 2$ i $b = -1.5$, tada je $a = f(3 \cdot 2 - 1.5) = f(4.5)$

- Pomeraj *b* je vrlo sličan težini, osim što ima konstantan unos, 1. Međutim, ako ne želite da imate pomeraj u određenom neuronu, on može biti izostavljen.
- Treba imati na umu da su i *w* i *b* podesivi skalarni parametri neurona. Obično funkciju prenosa bira sam dizajner, a parametri *w* i *b* podešavaju se nekim *pravilom učenja* tako da odnos unos/iznos ispunjava neki specifičan cilj.
- Za različite svrhe koriste se različite prenosne funkcije.



Biološki neuroni



Jednoulazni neuron

Prenosne funkcije

- Prenosna funkcija može biti linearna ili nelinearna funkcija od n. Određena prenosna funkcija bira se tako da udovolji zahtevima problema koji neuron pokušava da reši. Postoji mnoštvo ovih funkcija od kojih ćemo navesti dve.
- ***Hard limit*** prenosna funkcija (ili *funkcija praga*) prikazana na slici levo data je sa:

$$a = 0 \text{ za } n < 0,\ a = 1 \text{ za } n \geq 0$$

Ova funkcija za izlazni podatak neurona postavlja 0 ako je argument manji od 0, ili 1 ako je argument veći ili jednak 0. Ova funkcija se koristi kada želimo da kreiramo neuron koji klasifikuje ulazne podatke u dve odvojene kategorije.

- Grafik na desnoj strani ilustruje ulazno/izlazne karakteristike jednoulaznog neurona koji koristi hard limit prenosnu funkciju (može se primetiti uticaj težine i pomeraja).

$a = hardlim(n)$

Hard limit prenosna funkcija

$a = hardlim(wp + b)$

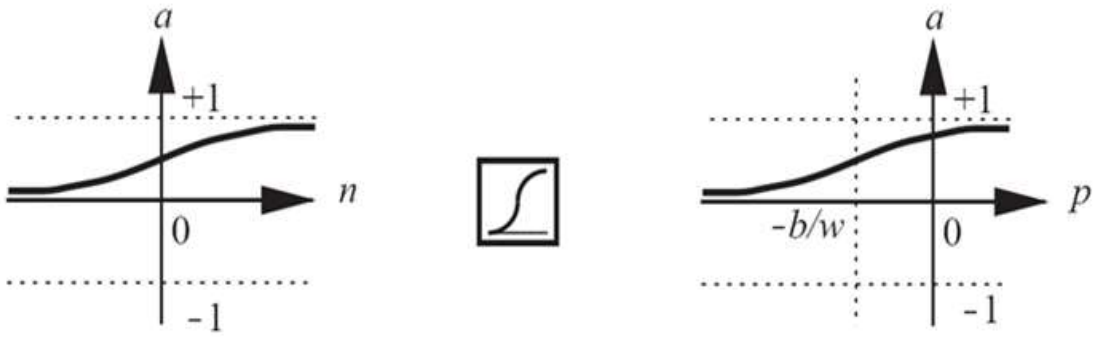
Jednoulazni *hardlim* Neuron

Prenosne funkcije

- ***Log-sigmoid*** (ili *sigmoidalna*) prenosna funkcija vrlo često je u upotrebi i data je sledećim izrazom:

$$a = \frac{1}{1 + e^{-n}}$$

- Ova funkcija svoj unos (koji može imati vrednosti između plus i minus beskonačno) pretvara u iznos sa vrednošću u intervalu od 0 do 1 (slika levo).
- Na desnoj strani slike opet se vidi kako odabir odgovarajućih vrednosti za težinu i pomeraj može uticati na vrednost izlaznog podatka *a*.
- Log-sigmoid prenosna funkcija najčešće se koristi u višeslojnim mrežama, a sa sobom nosi posebne pogodnosti zato što je diferencijabilna na čitavom svom domenu (od minus do plus beskonačno).



$a = logsig(n)$

Log-Sigmoid prenosna funkcija

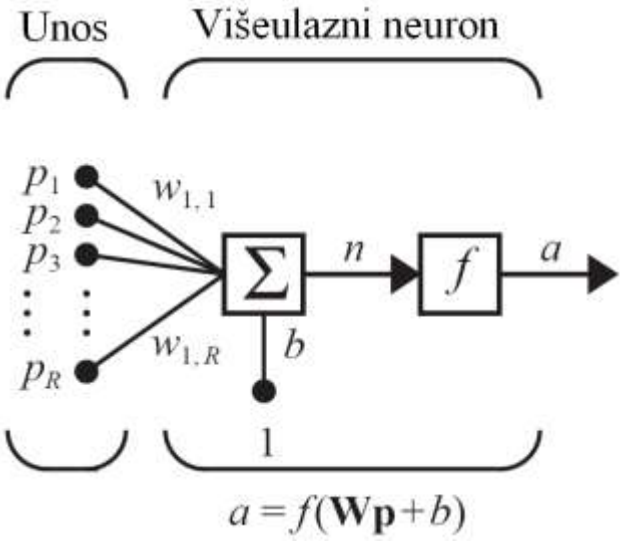
$a = logsig(wp + b)$

Jednoulazni *logsig* Neuron

Modeli neurona i mrežne strukture

Višeulazni neuron

- Neuron obično poseduje više od jednog unosa. Na slici je prikazan neuron sa R unosa.
- Pojedinačni ulazni podaci p_1, p_2,…, p_R umnoženi su odgovarajućim težinama koji predstavljaju elemente $w_{1,1}$, $w_{1,2}$,…, $w_{1,R}$ težinske matrice **W**.
- Neuron ima pomeraj b, koji se dodaje sumi unosa pomnoženih odgovarajućim težinama tako da formira mrežni unos n:



Višeulazni neuron

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b$$

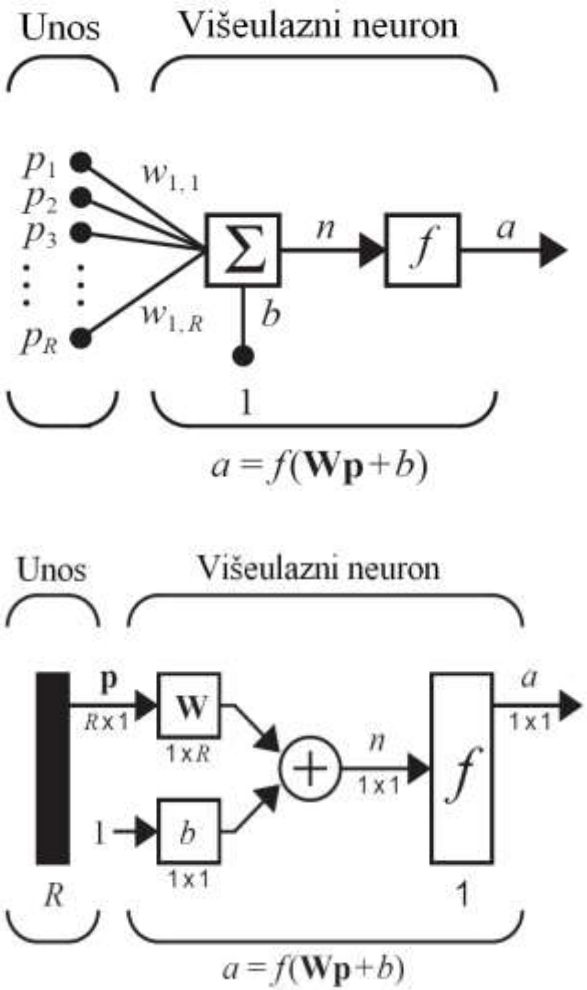
Ili u matričnoj formi: $n = \mathbf{W}\mathbf{p} + b$

gde matrica **W** za slučaj jednog neurona ima samo jedan red. Sada iznos iz neurona može biti napisan kao:

$$a = f(\mathbf{W}\mathbf{p} + b)$$

Prikaz neuronske mreže – skraćena notacija

- Ukoliko želimo da nacrtamo mreže od po nekoliko slojeva pri čemu svaki ima nekoliko unosa to bi bilo veoma kompleksno i konfuzno. Da bi smo to izbegli, koristi se skraćena notacija (slika desno).
- Kao što se vidi, *ulazni vektor* **p** predstavljen je punom vertikalnom trakom na levoj strani. Dimenzije vektora **p** prikazane su ispod njegove oznake kao R x 1, ukazujući da se radi o matrici koloni sa R elemenata.
- Ulazni vektor **p** množi se sa težinskom matricom **W**, koja u slučaju jednog neurona, ima R kolona, ali samo jedan red. Konstanta 1 ulazi u neuron kao unos i biva umnožena skalarnim pomerajem *b*.
- Mrežni unos prenosne funkcije *f* je *n*, što predstavlja sumu pomeraja *b* i proizvoda **Wp**.
- U ovom slučaju, iznos *a* celokupnog neurona je skalar. Kada bi imali više od jednog neurona mrežni iznos bio bi vektor.
- Na šemama skraćene notacije uvek su uključene i dimenzije promenljivih koje se u njima pojavljuju, tako da odmah možemo prepoznati da li je reč o skalaru, vektoru ili matrici.

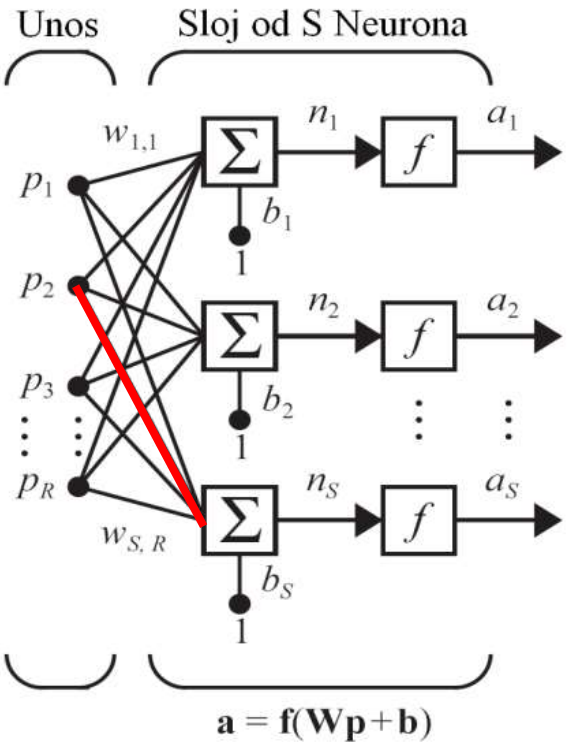


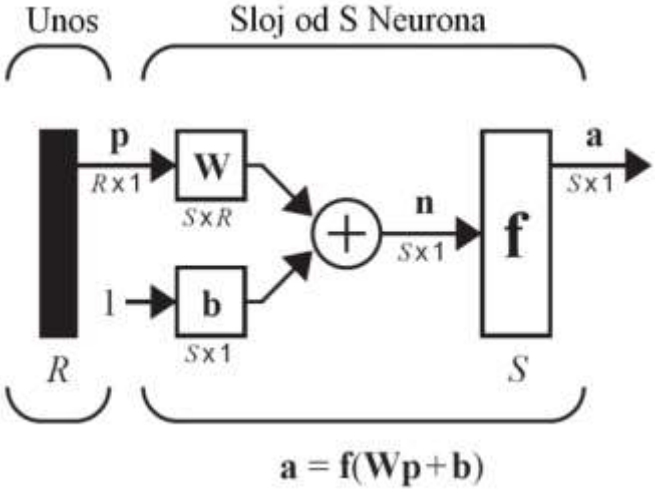
Neuron sa R unosa prikazan u skraćenoj notaciji

Mrežne strukture

Neuronski slojevi

- Jedan neuron, čak i sa više unosa, obično nije dovoljan. Zato koristimo više neurona koji rade paralelno u onome što se zove **“sloj”.**
- Jednoslojna mreža od S neurona data je na slici. Svaki od R unosa povezan je sa svakim od neurona, tako da težinska matrica **W** sada ima S redova.
- Sloj sadrži težinsku matricu, sume, vektor pomeraja **b**, prenosne funkcije i *izlazni vektor* **a**.
- Svaki element ulaznog vektora **p** povezan je sa svakim neuronom preko težinske matrice **W**. Svaki neuron poseduje pomeraj b_i, sumu, prenosnu funkciju f i iznos a_i. Svi iznosi zajedno formiraju izlazni vektor **a.**
- Broj unosa u sloj se obično razlikuje od broja neurona u sloju ($R \neq S$). Takođe, prenosne funkcije u sloju ne moraju biti iste za sve neurone. Može se definisati sloj koji će imati različite prenosne funkcije za različite neurone.





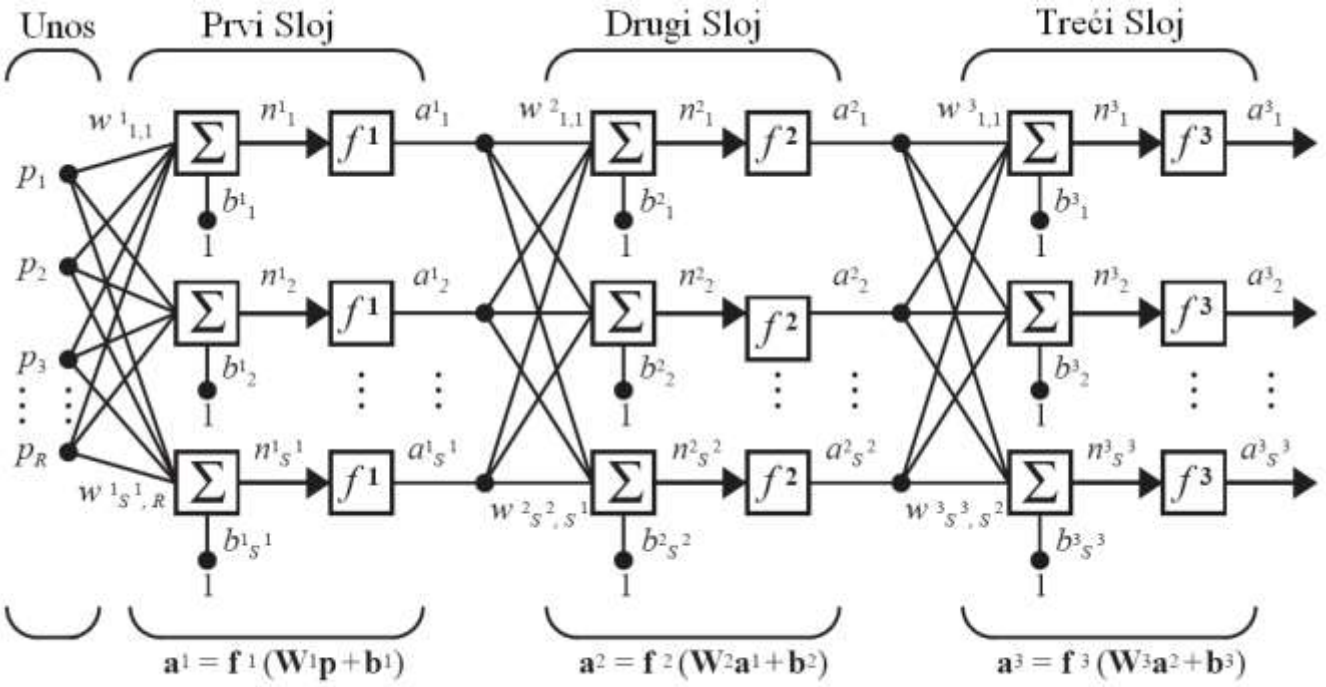
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & \ddots & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

Elementi ulaznog vektora ulaze u mrežu preko težinske matrice **W**.

Indeks reda elementa matrice označava neuron kome taj element (tj. težina) pripada, a indeks kolone označava izvor signala za tu težinu. Tako, indeksi elementa $w_{3,2}$ govore nam da ova težina predstavlja vezu između trećeg neurona i drugog izvora (unosa).

Višestruki neuronski slojevi

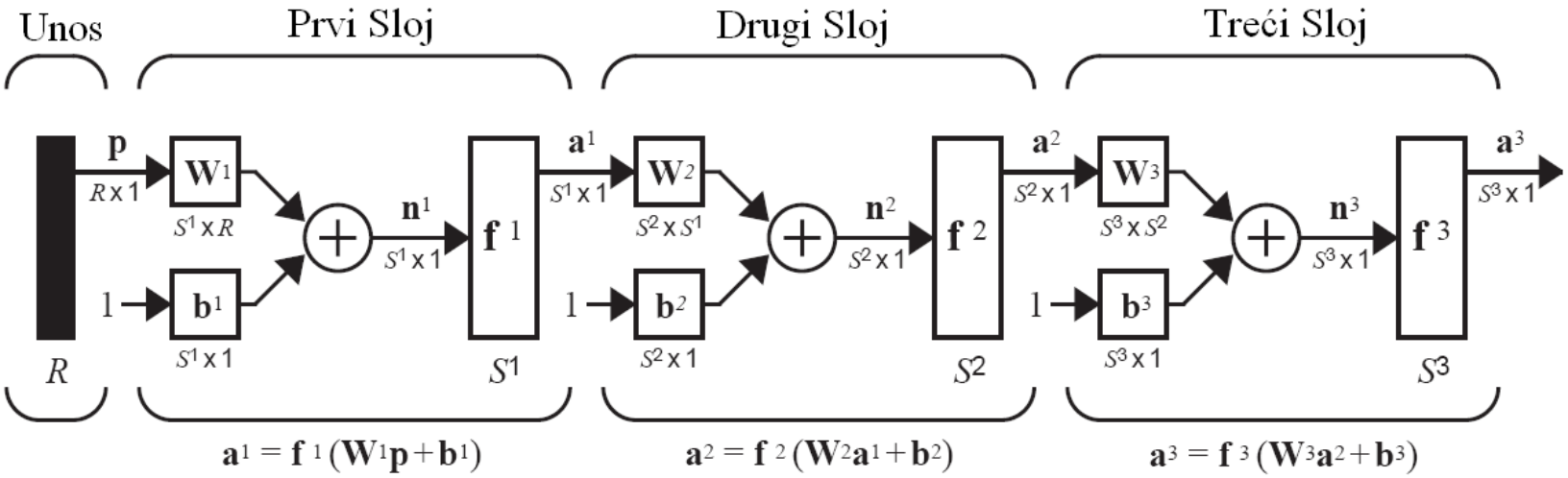
- Razmotrimo sada mrežu koja se sastoji od nekoliko slojeva. Svaki sloj ima svoju sopstvenu težinsku matricu **W**, svoj vektor pomeraja **b**, vektor mrežnog unosa **n**, kao i izlazni vektor **a**.
- Sada treba uvesti dodatne oznake kako bi razlikovali ove slojeve. Koristićemo desne gornje indekse za identifikaciju slojeva. Konkretno, priključićemo broj sloja kao desni gornji indeks oznaci svake promenljive koja pripada tom sloju.
- Tako, težinska matrica za prvi sloj biće napisana kao $\mathbf{W}^1$, a za drugi $\mathbf{W}^2$. Ovakva notacija korišćena je na slici, koja prikazuje mrežu od tri sloja.



$$a^3 = f^3(W^3 f^2(W^2 f^1(W^1 p + b^1) + b^2) + b^3)$$

Višestruki neuronski slojevi

- Vidimo da postoji R unosa u mrežu, S^1 neurona u prvom sloju, S^2 neurona u drugom, itd. Različiti slojevi mogu imati različit broj neurona. Izlazni podaci iz prvog sloja su ulazni podaci za drugi sloj, a izlazni podaci iz drugog su ulazni za treći sloj.
- Tako, drugi sloj može biti posmatran kao jednoslojna mreža koja ima $R = S^1$ unosa, $S = S^2$ neurona, i $S^1 \times S^2$ težinsku matricu $\mathbf{W}^2$.
- Unos u drugi sloj je $\mathbf{a}^1$, a iznos $\mathbf{a}^2$.
- Sloj čiji iznos je iznos za čitavu mrežu zove se ***izlazni sloj***. Ostali slojevi naziviaju se ***skriveni slojevi.***
- Na dole prikazanoj mreži (u skraćenoj notaciji) postoji jedan izlazni sloj (treći sloj) i dva skrivena (prvi i drugi).



Višestruki neuronski slojevi

- Višeslojne mreže moćnije su od monoslojnih (ali su zato komplikovanije).
- U praksi, ako imamo četiri spoljašnje promenljive, postavićemo četiri unosa u mrežu. Slično, ako želimo da mreža ima sedam iznosa, onda izlazni sloj mora imati sedam neurona.
- Željene karakteristike izlaznog signala pomoćiće nam da odaberemo prenosnu funkciju za izlazni sloj (npr. ako želimo da izlazni podaci budu 0 ili 1, u izlaznom sloju koristićemo *hard limit* prenosnu funkciju).
- Šta ako imamo više od dva sloja?

  Predviđanje optimalnog broja neurona u skrivenim slojevima nije nimalo lak zadatak i aktivna je oblast istraživanja. Većina praktičnih neuronskih mreža ima samo dva ili tri sloja. Četiri sloja ili više u upotrebi su vrlo retko.
- Na kraju, potrebno je odabrati da li će neuroni imati pomeraj ili ne. Pomeraj dodaje mreži jednu promenljivu više, tako da možemo očekivati da mreže sa pomerajem budu moćnije od mreža bez njega.

Učenje neuronske mreže

- Pod ***pravilom učenja*** podrazumevamo proceduru za modifikaciju težina i pomeraja mreže. Ovakva procedura takođe se može zvati i ***algoritam treninga***.
- Svrha pravila učenja je obučavanje mreže da izvede određeni zadatak. Postoji mnogo tipova pravila učenja koja se svrstavaju u tri široke kategorije: učenje sa nadzorom, učenje bez nadzora i učenje sa ocenjivanjem.

  (1) Učenje sa nadzorom, pravilo učenja snabdeveno je skupom primera (tzv. trenažnim setom) pravilnog ponašanja mreže:

  $$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

  gde je $\mathbf{p}_q$ unos u mrežu, a $\mathbf{t}_q$ odgovarajući tačni iznos koji se zove *meta* (q = 1, 2,…,Q). Kada neki unos primenimo na mrežu iznos se poredi sa metom, a pravilo učenja se onda koristi za podešavanje težina i pomeraja mreže tako da iznos bude što približniji meti.

  (2) Učenje sa ocenjivanjem slično je učenju sa nadzorom, osim što umesto tačnih iznosa za svaki unos, algoritam koristi sistem ocenjivanja (bodovanja) rada mreže. Ocena je mera učinka mreže za određenu sekvencu ulaznih podataka. Ovakav način učenja je znatno manje uobičajen od učenja sa nadzorom.

  (3) Učenje bez nadzora, težine i pomeraji modifikuju se samo u odnosu na ulazne podatke. Ne postoje očekivani izlazni podaci – mete. Ali kako trenirati mrežu, a da ne znamo šta će ona da radi? Većina ovih algoritama vrši određenu vrstu grupisanja ulaznih podataka. Oni se, zapravo, uče da kategorizuju ulazne podatke u konačan broj klasa. Ovaj vid učenja je manje zastupljen od učenja sa nadzorom, ali ipak pronalazi mnogobrojnu primenu.

Algoritam propagacije unazad

- Algoritam propagacije unazad uveden je od strane Rumelharta i Meklelanda 1986. i predstavlja jedno od najznačajnijuh otkrića u razvoju veštačkih neuronskih mreža.
- Ovaj algoritam spada u klasu učenja sa nadzorom i koristi se za treniranje višeslojnih neuronskih mreža.
- U procesu učenja, mreži se predstavlja skup primera (1) koji se sastoje od serije ulaznih podataka i njima pridruženih (očekivanih) izlaznih podataka – meta.

$$\{\mathbf{p}1, \mathbf{t}1\}, \{\mathbf{p}2, \mathbf{t}2\}, \ldots, \{\mathbf{p}Q, \mathbf{t}Q\} \qquad (1)$$

- Na samom početku treninga, mreži se zadaju početne vrednosti težina i pomeraja (najčešće male pozitivne vrednosti), a zatim se primenjuju ulazni podaci iz trenažnog skupa.
- Kada se primene svi ulazni podaci, za dobijene izlazne podatke računa se greška E(**x**) kao razlika između stvarnih izlaznih podataka i očekivanih izlaznih podataka – meta (2):

$$E(\mathbf{x}) = \sum_{q=1}^{Q} (t_q - a_q)^2, \qquad E(\mathbf{x}) = \sum_{q=1}^{Q} (\mathbf{t}_q - \mathbf{a}_q)^T (\mathbf{t}_q - \mathbf{a}_q), \qquad (2)$$

x je vektor koji sadrži sve težine i pomeraje, a T se odnosi na operaciju transponovanja

Greška ako postoji samo jedan izlazni podatak

Greška ako postoji više izlaznih podataka

Algoritam propagacije unazad

- Nakon prvog kruga treninga, mreži se zadaju nove (korigovane) vrednosti težina i pomeraja i čitav proces se ponavlja u iteracijama.
- Cilj algoritma propagacije unazad je traženje takve kombinacije težina (i pomeraja) koji odgovara minimumu funkcije E(**x**).
- Okosnicu algoritma čini tzv. ***delta pravilo*** (ili pravilo *gradijentnog spusta*), na kome se bazira korekcija težina i pomeraja:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \qquad w_{ij}(\mathbf{k}+1) = w_{ij}(\mathbf{k}) - \eta \frac{\partial E(\mathbf{k})}{\partial w_{ij}(\mathbf{k})};$$

$$\Delta b_i = -\eta \frac{\partial E}{\partial b_i}, \qquad b_i(\mathbf{k}+1) = b_i(\mathbf{k}) - \eta \frac{\partial E(\mathbf{k})}{\partial b_i(\mathbf{k})};$$

- Δw_{ij} je korekcija te težine u sledećoj iteraciji
- w_{ij}(k) predstavlja težinu veze između neurona *i* i neurona *j* u k-toj iteraciji
- Δb_i predstavlja korekciju pomeraja
- b_i(k) predstavlja pomeraj neurona *i* u k-toj iteraciji
- η naziva se ***brzina učenja*** i predstavlja pozitivnu konstantu.

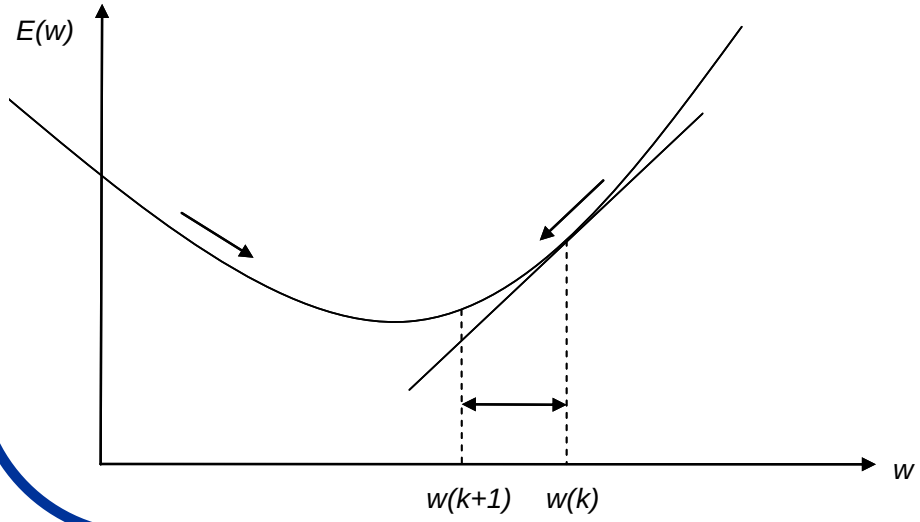
Vidimo da je korekcija težine jednaka negativnom gradijentu greške pomnoženom pozitivnom konstantom η.

Znak minus u relaciji govori nam da se korekcija težina odvija u smeru *gradijentnog spusta* ili niz gradijent greške

Algoritam propagacije unazad

- Na slici, u pojednostavljenoj formi, (2D slučaj) prikazana je korekcija težina koja se odvija u smeru ***gradijentnog spusta*** ili niz gradijent greške.
- Vrši se dakle, takva promena težina koja vodi do nižih vrednosti greške. Veličina korekcije zavisiće od vrednosti gradijenta ali i od vrednosti η, koja određuje hod (ili brzinu) učenja.

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \qquad w_{ij}(\mathbf{k}+\mathbf{1}) = w_{ij}(\mathbf{k}) - \eta \frac{\partial E(\mathbf{k})}{\partial w_{ij}(\mathbf{k})};$$



Pravilo gradijentnog spusta primenjeno na imaginarnu mrežu koja se sastoji od jednoulaznog neurona. Težine se koriguju u smeru smanjenja vrednosti greške.

Vrednost greške u ovom uprošćenom primeru predstavljena je krivom E(w).

Ovaj algoritam pokušava da nađe apsolutni (globalni) minimum ove krive.

Algoritam propagacije unazad

- U opštem slučaju E(**x**) predstavlja površ (ili hiperpovrš) u prostoru težina. Tehnikom gradijentnog spusta ovaj algoritam pokušava da nađe apsolutni (globalni) minimum ove površi.
- Težinama se najpre daju male nasumično odabrane vrednosti (ovo je ekvivalentno nasumičnom odabiru tačaka na površi greške).
- Algoritam zatim računa lokalne gradijente na površi i menja težine u smeru negativnog gradijenta (tj. u smeru smanjenja vrednosti E(**x**)).
- Ako površ greške sadrži više od jednog mimimuma, važno je da algoritam ne ostane zarobljen u lokalnom minimumu. Ovo se prevazilazi uvođenjem tzv. ***konstante momenta*** i uopštavanjem delta pravila:

$$\Delta w_{ij}(\mathbf{k}) = \alpha \Delta w_{ij}(\mathbf{k}-1) - \eta \frac{\partial E(\mathbf{k})}{\partial w_{ij}(\mathbf{k})} .$$

  - Konstanta momenta predstavljena je sa ***α*** i ima vrednost između 0 i 1.

- Dodavanjem dela predhodne korekcije težine, tekućoj korekciji težine (koja je vrlo mala u lokalnom mimimumu) moguće je da algoritam pobegne iz lokalnog minimuma.
- Zbog izračunavanja izvoda greške po težinama, prenosne funkcije neurona u mreži koja se trenira ovim algoritmom moraju da budu glatke i diferencijabilne na čitavom svom domenu.

Algoritam propagacije unazad

Da rezimiramo:

Algoritam propagacije unazad odvija se u sledećih nekoliko faza:

(1) Inicijalizacija težina i pomeraja;

(2) Primena ulaznih vektora (iz trenažnog seta) na mrežu i njihova propagacija kroz slojeve mreže da bi dobili izlazni vektor;

(3) Računanje signala greške poređenjem stvarnih izlaznih vektora sa željenim izlaznim vektorima (metama);

(4) Propagacija signala greške nazad kroz mrežu (po čemu je algoritam dobio ime) i podešavanje težina tako da se umanji signal greške.

(5) Faze (2-4) se ponavljaju dok se ne postigne zadovoljavajuća vrednost signala greške

Primena NN

- **NN imaju široku primenu u različitim oblastima života:**

- Prepoznavanju rukopisa (sada je to popularno kod tablet - računara)
- Kompresiju slika
- Predviđanja berzanskih kretanja
- Medicinskoj dijagnostici
- Analizi spektara
- Ostalo

http://tralvex.com/pub/nap/

Kreiranje NN simulacije u programu MATLAB

- NN je nacin uspostavljanja relacije izmedju bilo kog seta ulaznih podataka sa nekim izlazom kada se nista ne zna o njihovoj medjusobnoj relaciji.
- Da bi mogli da koristimo NN, potrebno je da imamo veci set ulaznih i izlaznih podataka
- Napravićemo NN model koji ima tri ulazna podatka: a,b,c i pravi izlaz y.
- Da bi mogli da testiramo model, uzmimo da je zavisnost izmedju ulaza i izlaza:

$$y=5*a+b*c+7*c$$

- Uzecemo ovaj model da bi generisali ulazne i izlazne podatke koje nemamo
- U realnom slucaju, necemo imati matematicku relaciju izmedju ulaza i izlaza
- Već ćemo imati setove realnih ulaznih vrednosti (a,b,c) kao i izlaznih (y)
- Ovde, napravicemo setove ulaznih podataka koriscenjem komande “rand“
- Neka bude 1000 vrednosti za (a,b,c) i samim tim 1000 vrednosti izlaza (y)

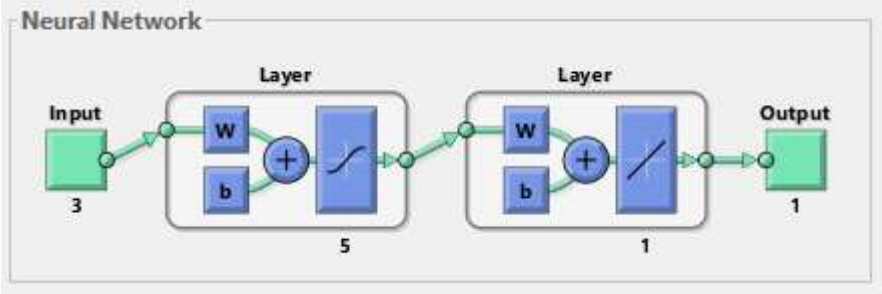
Kreiranje NN simulacije u programu MATLAB

```
% Generisanje matrice 1000 slucajnih vrednosti za a,b,c

a= rand(1,1000);      % Generisanje matrice a
b=rand(1,1000);       % Generisanje matrice b
c=rand(1,1000);       % Generisanje matrice c
n=rand(1,1000)*0.05; % Generisanje matrice za sum (n). Ovo je namerno
                        dodato da bi vrednosti za y izgledali kao
                        realni podaci
                      % Velicina suma je +-0.05 (tj. 0.1) i on je
                        uniforman

y=a*5+b.*c+7*c+n;     % Relacija koja racuna vrednosti y na osnovu
                        formule

I=[a; b; c];          % Pravimo jedinstvenu matricu redova i kolona
                      % ulaznih podataka I (I kao input)
O=y;                  % Pravimo matricu izlaznih podataka O (O kao
                        output i predstavlja zavisno promenljuvu "y")
```

Kreiranje NN simulacije u programu MATLAB

- NN treba da simulira mozak pun neurona koji su poredjani u vise slojeva.
- Prvi sloj uzima ulazne podatke koje sporvodi u medjuslojeve (skrivene slojeve).
- Poslednji (izlazni) sloj uzima izlaz iz skrivenih slojeva i prezentuje rezultat.
- Skrivenih slojeva moze biti proizvoljan broj.
- Svaki sloj je u stvari funkcija koja uzima neke promenljive (u obliku vektora u) i transformise ih u nove promenljive (vektor v) mnozeci ih sa nekim koeicijentima (matrica tezine w) i dodajuci im pomeraj b.
- Dimenzija vektora v poznata je kao v-dimenzija sloja: v=sum(w.*u)+b

- Napravimo NN sa 1 ulaznim i 1 izlaznim slojem.
- Dimenzija ulaznog sloja bice v=5 (a,b,c,w,b) tj. dimenzija ulaznog vektora koja je u=3 (tačke a,b,c) transformisaće se u vektor dimenzije 5 (v=5).
- Izlazni sloj imace dimenziju 1 i ima zadatak da ulazni vektor (u) dimenzije 5 transformise u vektor (v) dimenzije 1 (jer imamo samo jednu veličinu kao resenje)



Kreiranje NN simulacije u programu MATLAB

- Za kreiranje NN modela koristicemo komandu "newff"
- Najpre moramo napraviti matricu R za ulazne podatke. Ovde ce ona imati dimenzije 3x2 (3 je broj ulaznih promenljivih dok je 2 kolone potrebno za zadavanje min. i max. vrednosti ulaznih promenljivih.

```
R=[0 1; 0 1 ; 0 1]; % Prva kolona predstavlja minimalnu vrednost sva tri promenljive
                    a druga kolona njihovu maksimalnu vrednost. Posto su u pitanju
                    brojevi dobijeni rand komandom, opseg brojeva ce biti od 0 do 1
```

- Zatim pravimo matricu S koja odredjuje dimenzije ulazne matrice v tj. dimenzije svih slojeva. Rekli smo vec da je dim(v)=5

```
S=[5 1]; % Matrica S nam je potrebna da bi program znao koje su dimenzije slojeva
```

- Sada pravimo mrezu komandom "newff" sledecom sintaksom:

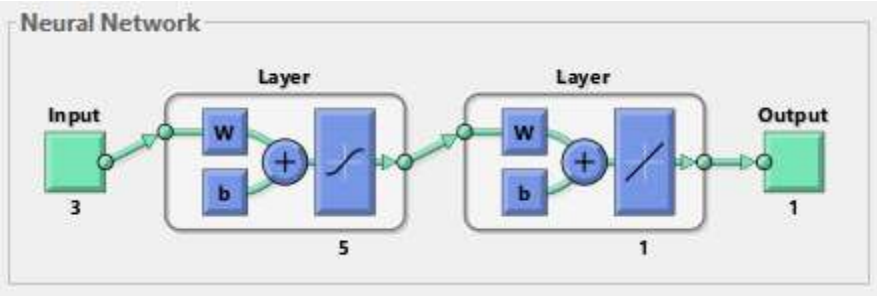
```
net = newff(R,S,{'tansig','purelin'}); % Promenljiva "net" je nas model NN
```

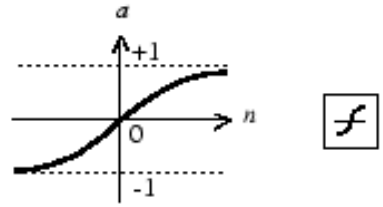
Kreiranje NN simulacije u programu MATLAB

- Zdržimo se na trenutak na prethodnom programskom redu. Tu se pominju dve nove komande: "tansig" i "purelin".

```
net = newff(R,S,{'tansig','purelin'});
```

- Prva komanda govori da nasa mreza za ulazni sloj koristi sigmoidalnu, a za izlazni sloj linearnu funkciju prelaza.





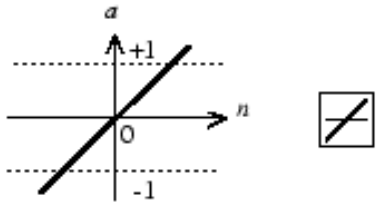
$a = tansig(n)$

Tan-Sigmoid Transfer Function

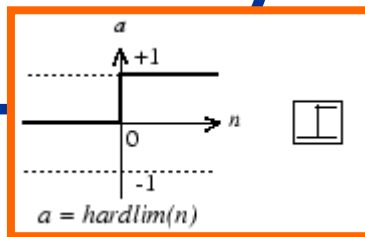
- Testirajmo kako ove dve funkcije izgledaju:

```
n = -5:0.1:5;
a = tansig(n);
plot(n,a)
```

```
n = -5:0.1:5;
a = purelin(n);
plot(n,a)
```



$a = purelin(n)$



$a = hardlim(n)$

Testirajte kako izgleda i hard-limit prelazna funkcija (hardlim) koristeći sličnu sintaksu

Kreiranje NN simulacije u programu MATLAB

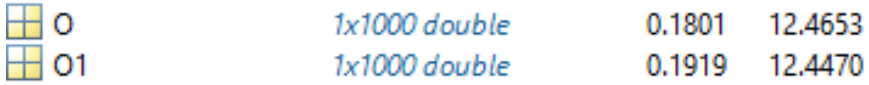
- Dobijena promenljiva "net" je naš model NN.
- Sada moramo da pustimo mrežu da uči. Mrežu ćemo trenirati podacima koje smo napravili korišćenjem "rand" komande. Koristimo sintaksu:

```
net=train(net,I,O);
```

- Pošto je mreža uspesno naučila, probajmo da simuliramo koju ćemo vrednost izlaza (O1) dobiti za "rand" generisan ulaz (I).

```
O1=sim(net,I);
```

- Ako uporedimo vrednost promenljive O i simulirane promenljive O1 videćemo da su praktično identične što znači da je naša mreža dobro naučila.



Kreiranje NN simulacije u programu MATLAB

- Ukoliko nacrtamo grafik vrednosti O i O1 za svih 1000 tacaka, možemo videti da su vrlo slične:

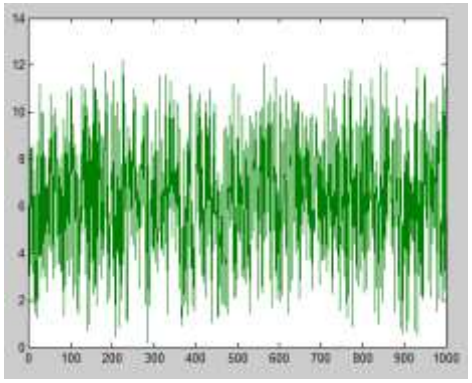
```
figure(1)
plot(1:1000,O,1:1000,O1);
```

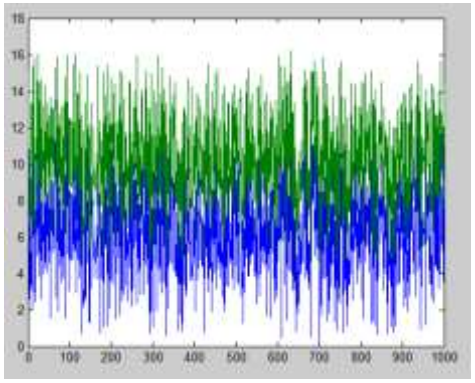
Da bi bolje videli, možemo vrednostima O1 dodati 4

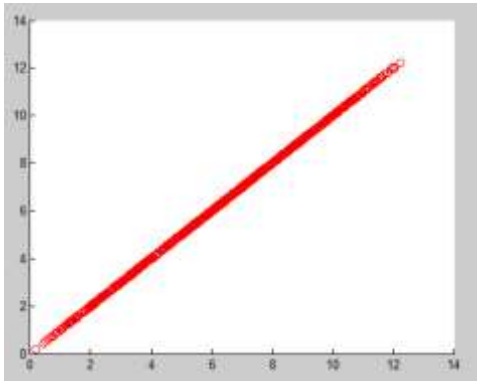
```
figure(1)
plot(1:1000,O,1:1000,O1+4);
```

Možemo nactati i uporediti realne (O) i simulirane (O1) tačke

```
figure(2)
scatter(O,O1);
```







Vidimo da su vrednosti O i O1 skoro identične

Kreiranje NN simulacije u programu MATLAB

- Sada bi mogli da testiramo našu naučenu mrežu za neke druge ulazne podatke.
- Recimo da su oni: a=1, b=1, c=1
- Za ove podatke, znamo da bi trebali da dobijemo vrednost y=13

```
y=a*5+b.*c+7*c
```

- Ulazna matrica će u ovom slučaju biti: I=[1 1 1]'
- Ovo " ' " je da bi dobili matricu sa 1 kolonom i 3 reda

```
y1=sim(net,[1 1 1]')
```

- Dobijamo broj koji je veoma blizak broju 13 (1*5+1*1+7*1)