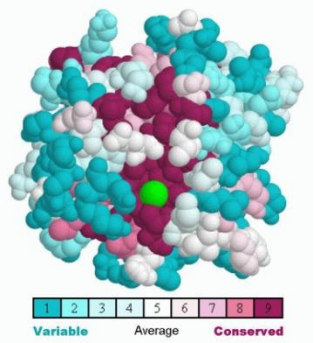
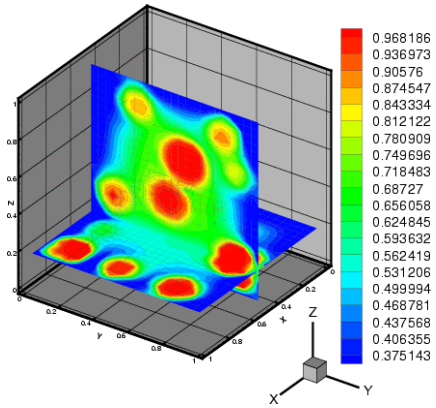
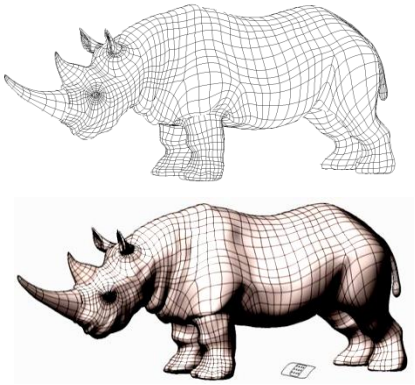
Uvod u računarske simulacije:







Miloš Mojović, v. prof.

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:

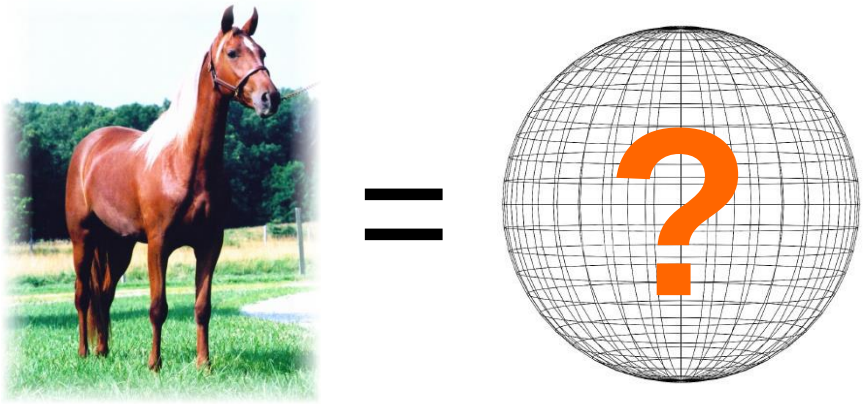
- **Prototip** predstavlja deo (podsistem) realnog sistema čiji model želimo da simuliramo. On obično predstavlja kompleksni sistem koji se sastoji od više odvojenih podsistema koji međusobno interaguju.
- Naš model bi trebao da predstavlja sve podsisteme, ali ponekad je korisno da se neki od njih smatraju 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).

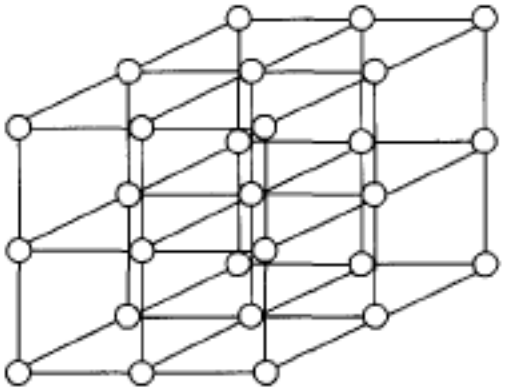


Model i simulacija:

- 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.

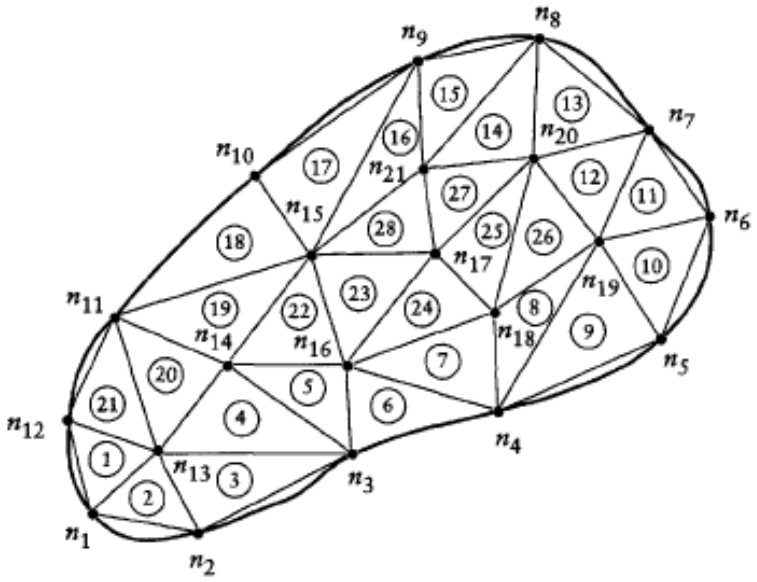
Ova metoda se sastoji u tome da se promena osobine koja se ispituje (neka je to npr. koncentracija *n*), definiše samo u okviru regije od interesa.

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

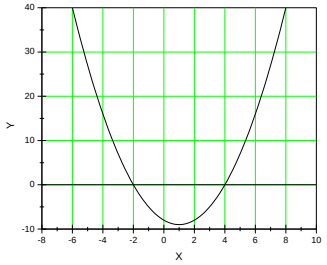
$$f(x) = x^2 - 2x - 8$$

- 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

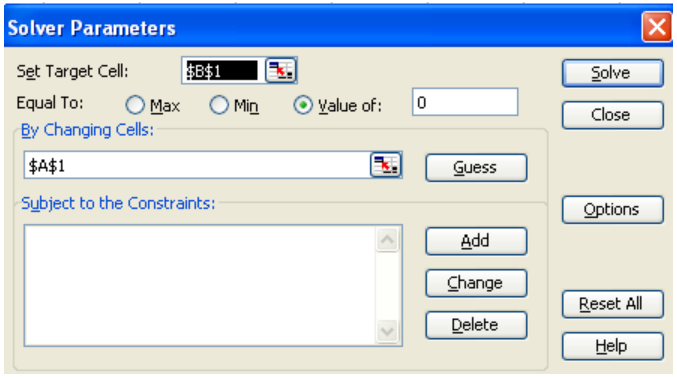
- Kao što se vidi, računar je pronašao rešenje, ali sa malom greškom. Takođe, ni rešenje koje je računar pronašao nije baš tačno nula, već broj koji je veoma blizu nje (ova tačnost je najčešće dovoljna za većinu zadataka).
- Ukoliko želimo da budemo manje "tolerantni" prema greški koju računar pravi, 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:

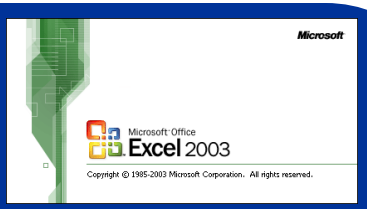
- Isti princip rešavanja problema možemo primeniti za rešavanje nekih fizičkohemijskih problema.
- Npr. zadatak je <u>naći specifičnu zapreminu</u> n-butana na 500K i 18 atmosfera korišćenjem Redlih Kwong-ove jednačine realnog gasnog stanja.

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

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

- 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?

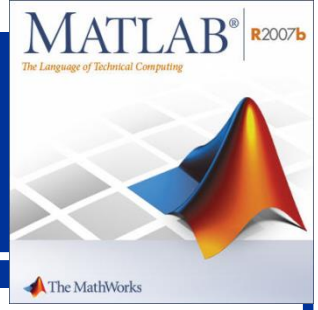
Provera dobijenog rezultata:

- 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.

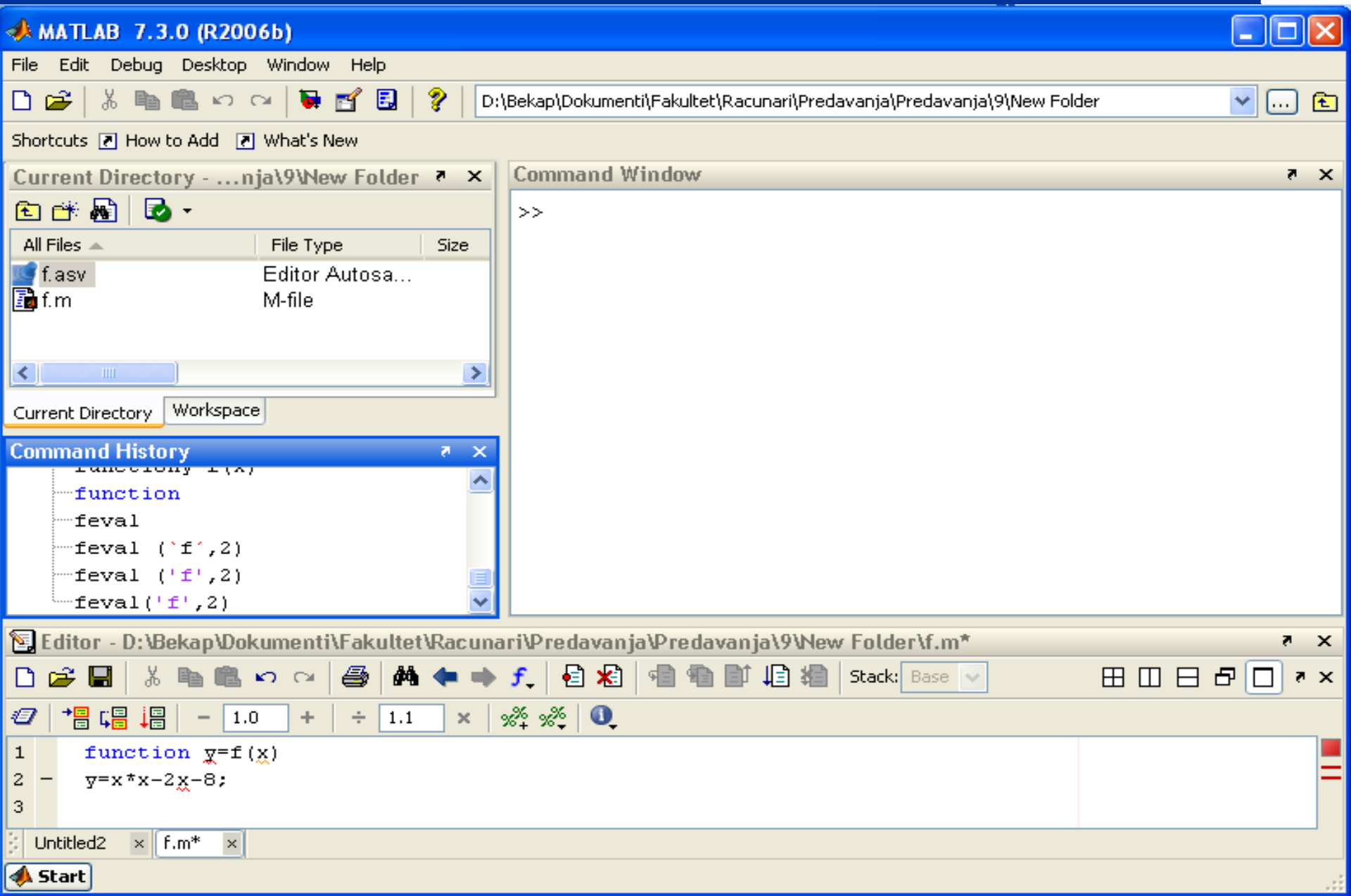
Rešavanje jednačina korišćenjem programa MATLAB:



- 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:

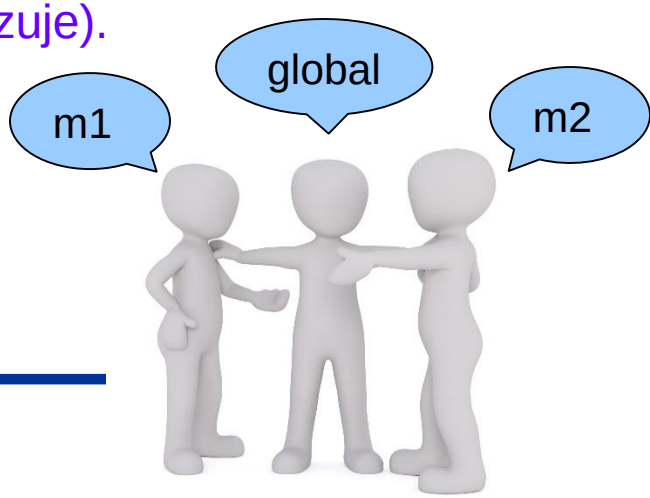


MATLAB - neke osnovne stvari:

- 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).

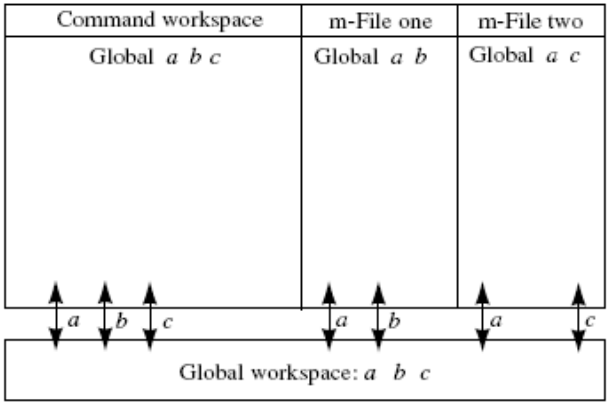
m-fajlovi:

- 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:
- Znakovi deljenja u MATLAB-u: (postoji "/" i "\") i ne znače isto tj. "\" predstavlja inverziju tj. INV (A)*B
- Ukucavanje dugačkih komandnih linija se vrši pomoću "..."

```
>> i

ans =

        0 + 1.0000i
```

```
>> 10/2

ans =

     5

>> 10\2

ans =

    0.2000
```

```
>> 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)
```

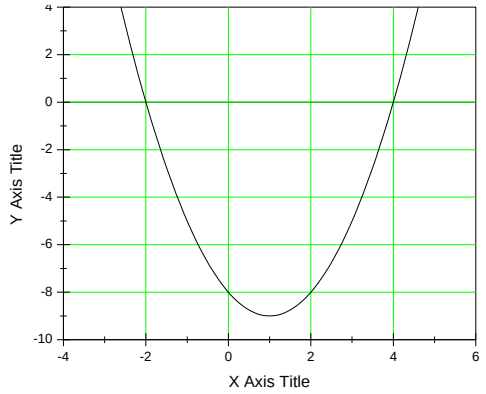
- Ukoliko želimo da dobijemo i drugu "nulu", moramo MATLAB-u naložiti da varira druge vrednosti za "x", različite od onih malopre. Upišemo:

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

Kako dobiti sva rešenja jednačine?

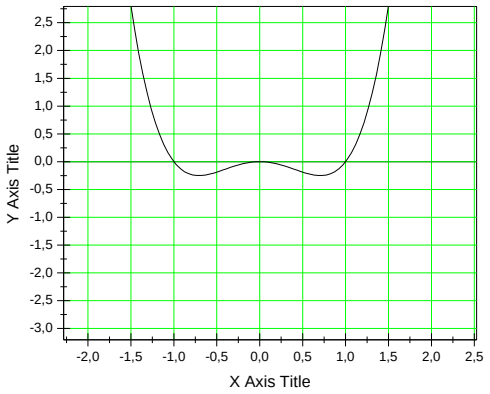
- Ipak, MATLAB ima komandu pomoću koje možemo jednostavno dobiti sve nule neke funkcije:

```
>> 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 =

(C4*i)/exp(i*t) - C3*i*exp(i*t)

>> z.y

ans =

C3*exp(i*t) + C4/exp(i*t)
```

- 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).

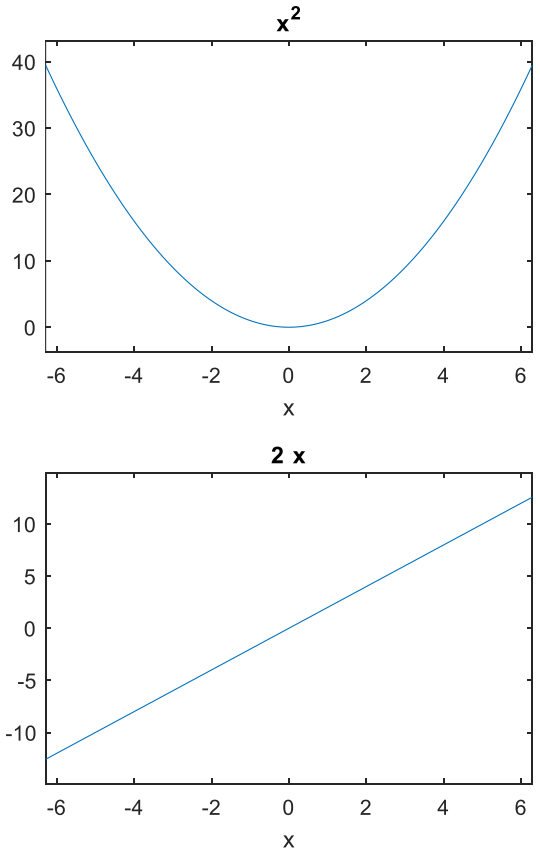
```
>> [x,y] = dsolve('Dx = y', 'Dy = -x')

x =

(C4*i)/exp(i*t) - C3*i*exp(i*t)


y =

C3*exp(i*t) + C4/exp(i*t)
```

- 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
```

```
f(x) =

x^2

y(x) =

2*x
```



- Zadatak: Uraditi isto ovo, ali ovoga puta definsati funkciju na dva načina koja su ranije obrađena.

Definisati fju na drugi način ...

Upotrebite "inline" komandu

```
>> f = inline('(x.^2)','x')
f =
     Inline function:
     f(x) = (x.^2)
>> syms x %probati bez syms x
>> y = diff(f)
Undefined function 'diff' for input arguments of type 'inline'.
>> y = diff(f(x))
y =
2*x
```

Upotrebite "@" function handle

```
> f =@(x)(x.^2)
f =
    @(x)(x.^2)
>> y = diff(f)
Undefined function 'diff' for input arguments of type 'function_handle'.
>> syms x
>> y = diff(f)
Undefined function 'diff' for input arguments of type 'function_handle'.
>> y = diff(f(x))
y =
2*x
```

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
```

Ako želimo da dobijemo brojevnu vrednost upisaćemo: double(ans)

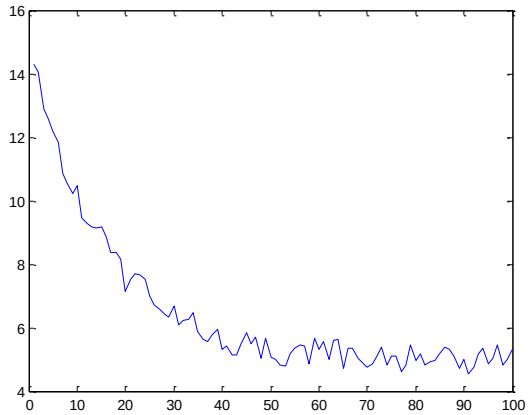
```
>> syms x z;

>> 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.

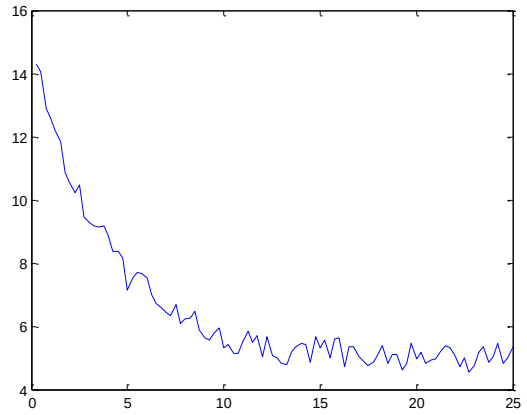
- Zadatak: Rešiti nekoliko neodređenih integrala u odnosu na različite promenljive. Probati računanje integrala bez definisanja promenljivih (ali prethodno obrisati postojeće promenljive komandom "clear all".

Crtanje 2D grafika u MATLAB-u

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




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

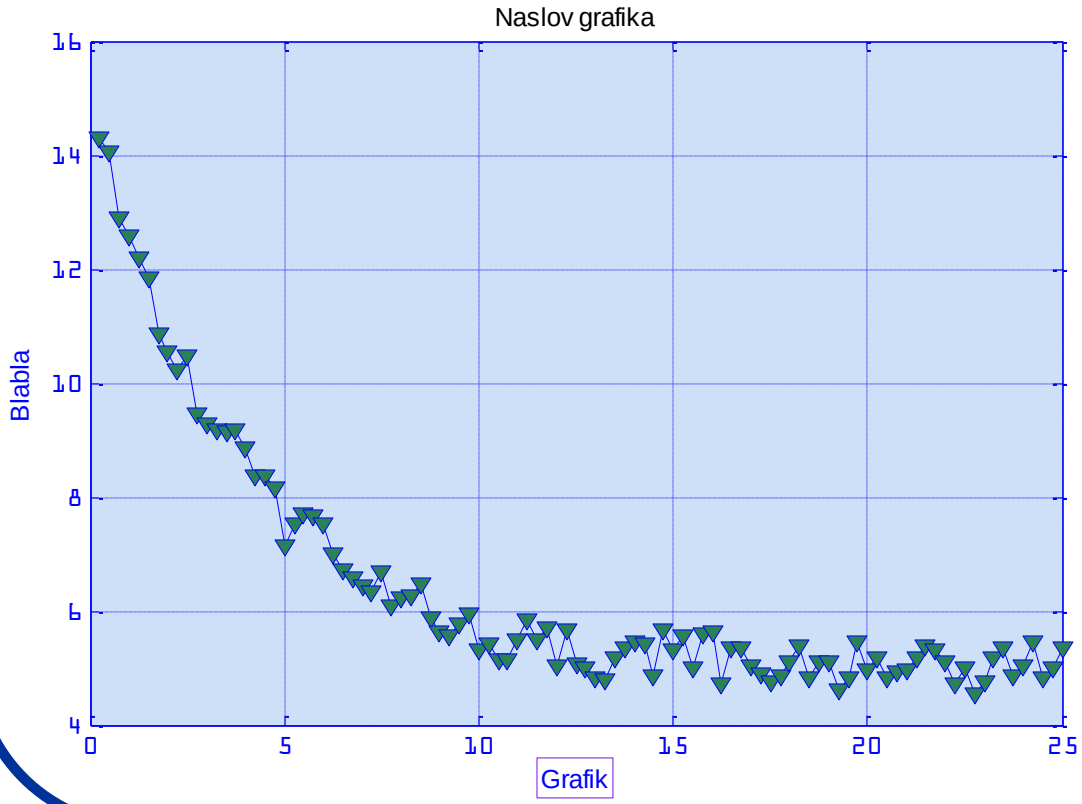


- Najpre naučimo kako nacrtati grafik ukoliko imamo zadate tačke samo na y-osi.
- U "Workspace" prozoru desnim klikom izaberemo opciju "New" pri cemu formiramo novu promenljivu (nazovimo je "y").
- Dvoklikom na promenljivu "y" otvoriće se "Variable editor" prozor (matrica) u kome možemo upisati tačke (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 ...

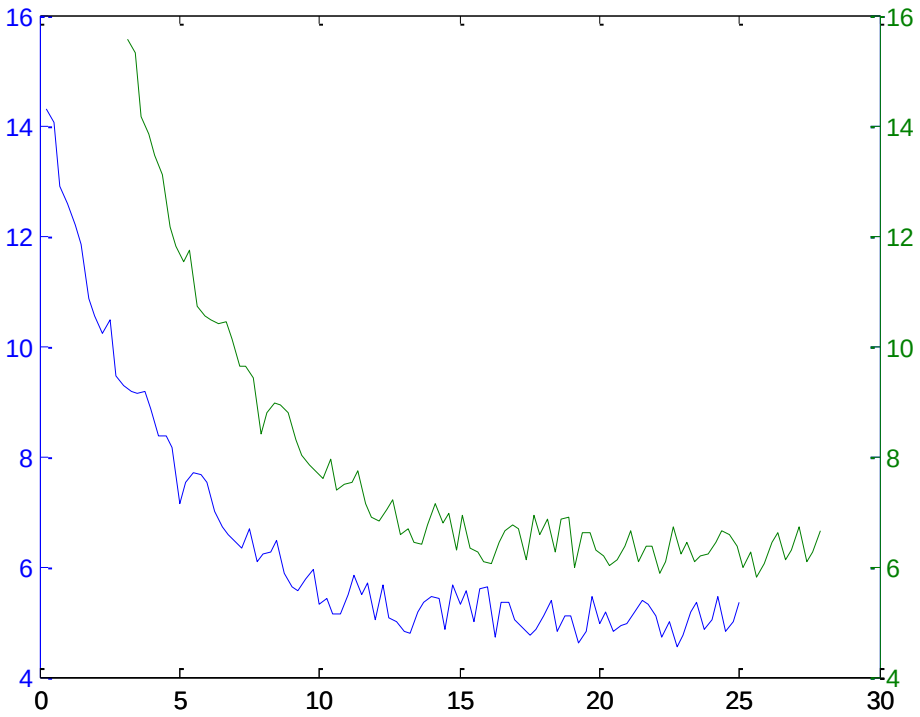


- Jednostavno na grafiku u gornjem meniju ići na. View-Proprety Editor
- Ne zaboraviti opciju: More Properties u okviru ovog menija.
- Upotrebiti i opciju: Tools za Zoom, 3D prikaz, markiranje i brisanje tačaka ...

- 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.

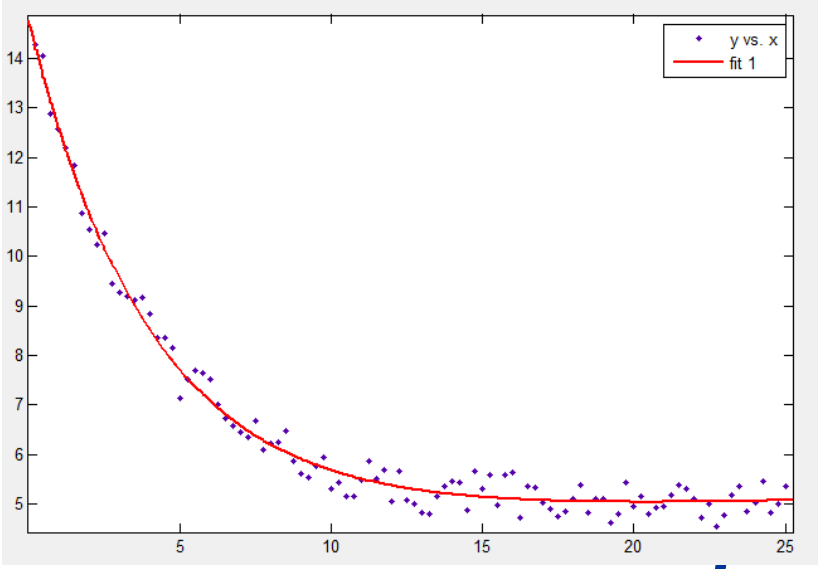
- Zadatak: Na što veći broj načina prikazati dobijene 2D grafike sa više setova tačaka. Upotrebiti što više opcija koje nudi MATLAB. U "Help" meniju ukucati "plot" i "plotyy" pa pročitati i upotrebiti ostale sintakse koje su na raspolaganju.

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.

Ekstrahovanje rezultata fitovanja

- Fit-Save to Workspace
- U konzoli upisati: fittedmodel

```
>> fittedmodel

fittedmodel =

     General model Exp2:
     fittedmodel(x) = a*exp(b*x) + c*exp(d*x)
       where x is normalized by mean 12.63 and std 7.253
     Coefficients (with 95% confidence bounds):
       a =        0.47  (0.2972, 0.6427)
       b =      -1.766  (-1.961, -1.57)
       c =       4.836  (4.592, 5.08)
       d =     0.02543  (-0.01578, 0.06665)
```

- Ali, kako ekstrahovati pojedine koeficijente i funciju odavde ... ???
- Jedan nacin je ... u konzoli upisati: fittedmodel.a, fittedmodel.b, fittedmodel.c
- Ili jednostavnije: coeffvalues(fittedmodel)
- Formulu ekstrahujemo komandom: f=formula(fittedmodel)
- Nazive koeficijenata ekstrahujemo komadnom: koefn=coeffnames(fittedmodel)

Kako nacrtati funkciju?

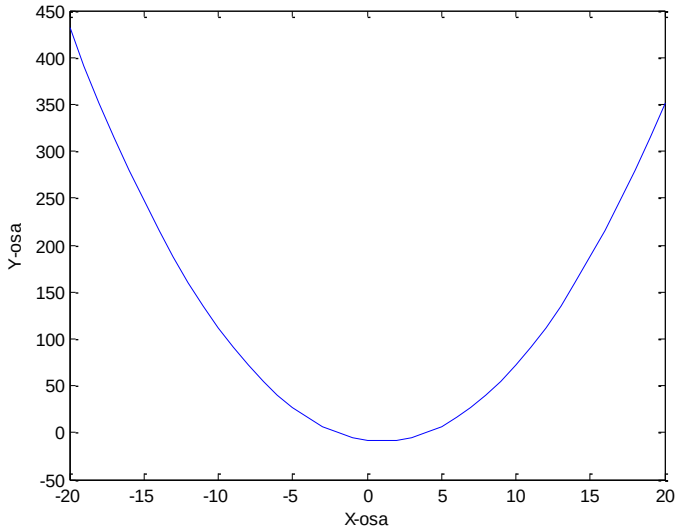
• Ukoliko definišemo našu funkciju kao:

Fajl: “fplot.m”

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

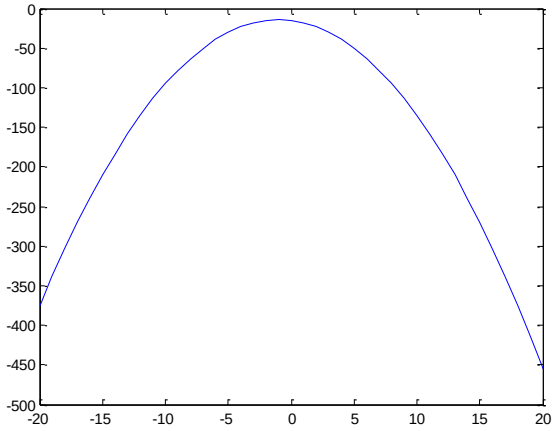
Definicija neke funkcije, određivanja opsega za x-vrednost sa korakom 1, crtanje funkcije i obeležavanje osa.

• 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“

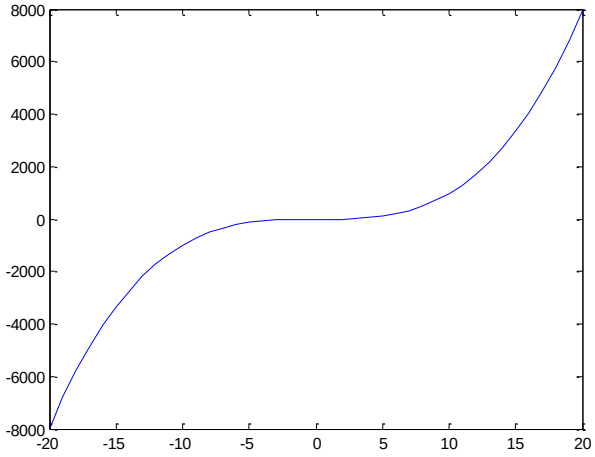
Definisanje intervala i kome želimo da izračunamo minimum funkcije

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.

- Pitanje: Kao što vidmo, rešenje je samo približno tačno. Kako bi mogli da dobijemo tačno rešenje?

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".

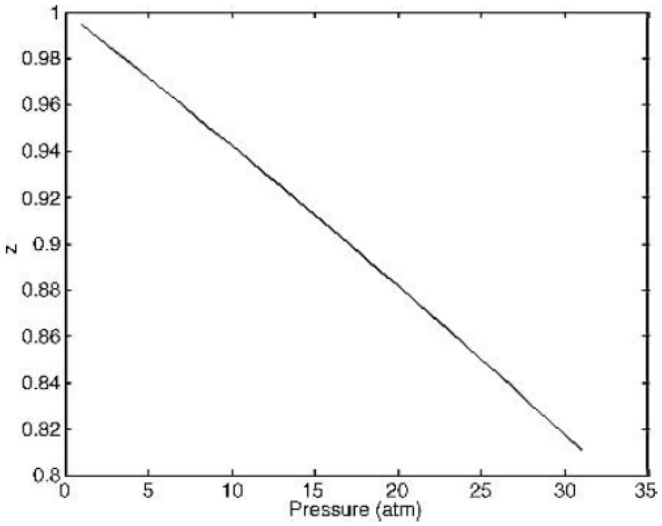
Programski kod (objašnjenje):

```
% 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:
- Ne zaboravite da 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.

Tapiserija:

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

ili još lepše:

Svemir:

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

Topologija:

```
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|' )
```

... 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

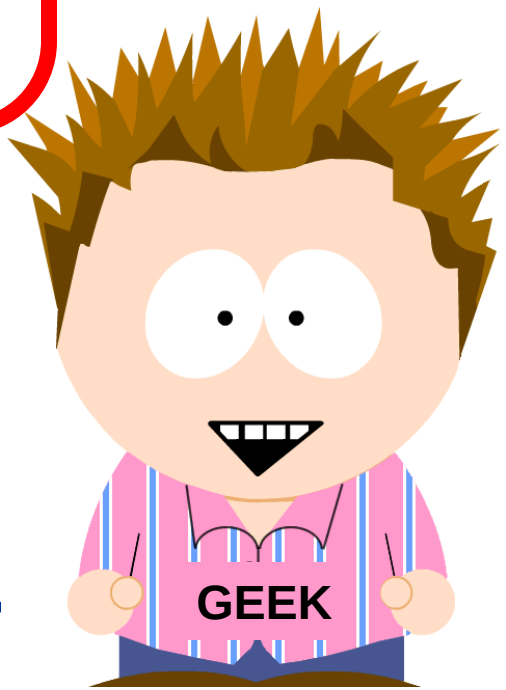
```
if rand < 0.5 fprintf(  'G'  ), end
if rand >= 0.5 fprintf(  'P'  ), end
```

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 i vreme izvršenja programa.

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 :)

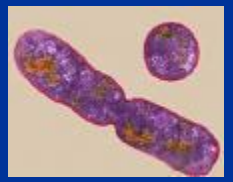
- Tako, ako je slučajno generisan broj **rand** u opsegu [0, 1), tada je **6 * rand** broj koji će biti u opsegu [0, 6), dok ce **6 * rand + 1** biti u opsgu [1, 7) ili preciznije od 1 do 6.999...
- Ukoliko odbacimo decimalni deo broja (tako što dobijenu vrednost zaokružimo na manji broj komandom **floor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

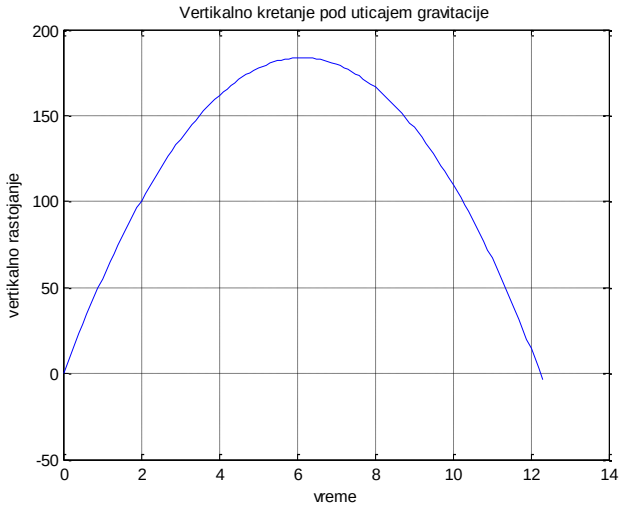
$$s=ut-gt^2/2$$

- 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: Definisati fju kretanja na jos dva nacina i uraditi isti zadatak.

Vežba: Function handle

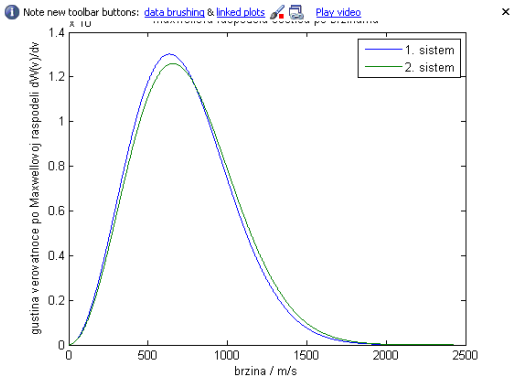
Zadatak: Igrati se malo sa upotrebom ove korisne sintakse.
Na primer:

```
>> kvadrat(4)

Undefined function or variable 'kvadrat'.

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

kvadrat =

    @(x)x.^2

>> kvadrat (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
```



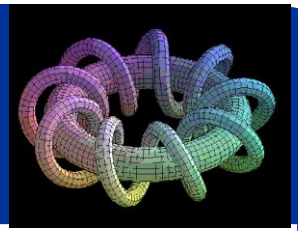
Program:
maxwell.m (sajt ffh)

Potprogrami:

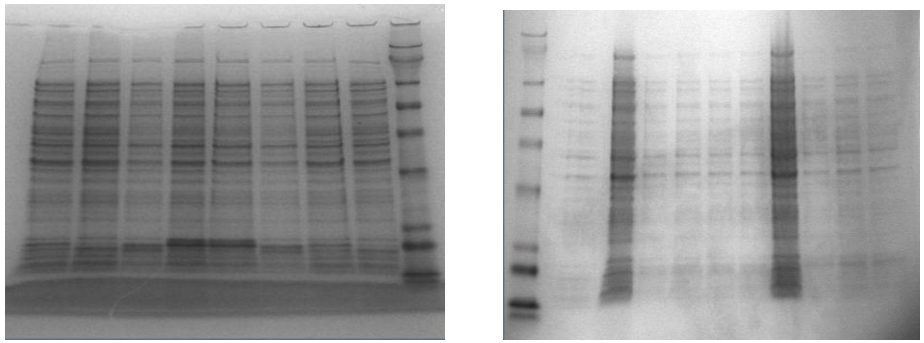
unos.m	vmaxi.m
brzine.m	vsred.m
grafik.m	vsrkv.m
gvmaxi.m	
gustinaverovatnoce.m	

Vežba: Obraditi i ostale programe koji se nalaze na: *http://www.ffh.bg.ac.rs/osnovne_primena_racunara.html*

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)

Isprobati i x = [1:10];

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.

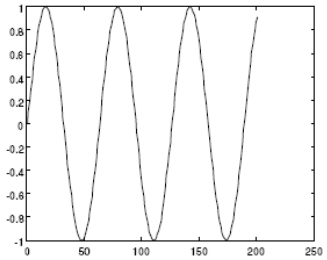
```
>> s1 = sin (z);
```

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:

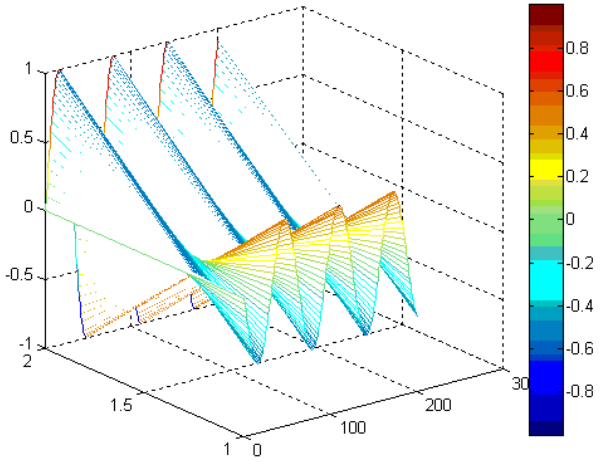
```
>> plot(s1);
```



- 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"

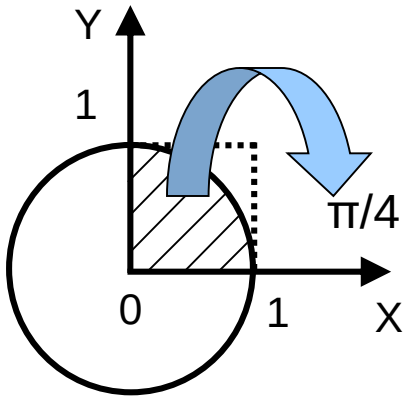
- Na sličan način možemo obrađivati i audio-signale ali ovo prevazilazi okvire ovog kursa (ili bar nema direktne veze sa fizičkom hemijom :)

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:

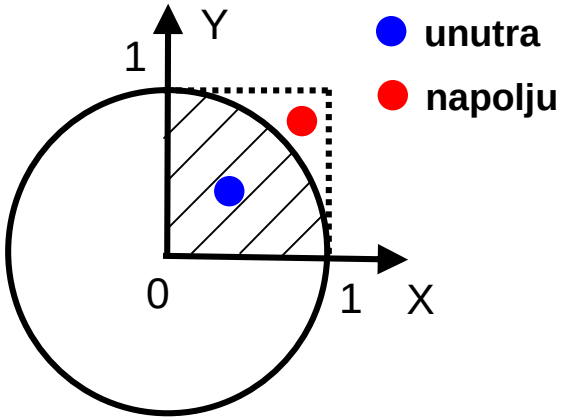
- Kao što je prethodno već bilo napomenuto Monte Karlo simulacije su imale veliki uticaj na razvoj računarskih simulacija u različitim oblastima nauke.
- Da ponovimo, ova tehnika koristi slučajne brojeve da bi modelovala različite vrste fizičkohemjskih procesa.
- Monte Karlo simulacija posebno dobro 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

SREĆAN RAD !!!

ZADATAK:

Pokrenuti nekoliko (4-5) programa po želji i videti kako funkcionišu. Pregledati progamske kodove i uočiti komande koje ste naučilli na ovom kursu.

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:

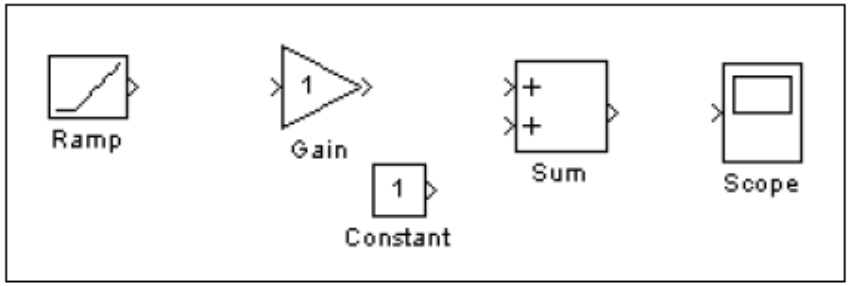
$$T_F = 9/5(T_C) + 32$$

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)

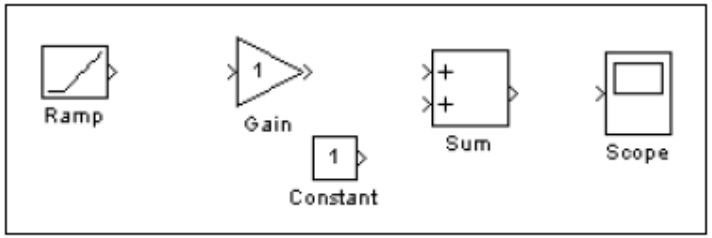
$$T_F = 9/5(T_C) + 32$$



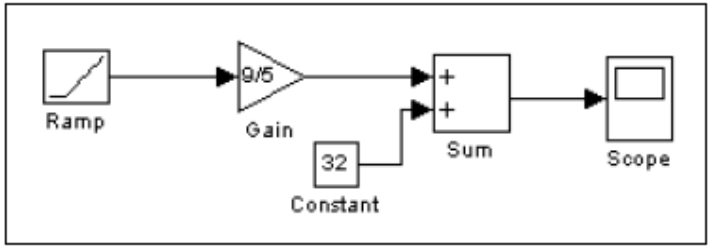
Realizacija Modela:

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

- 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.



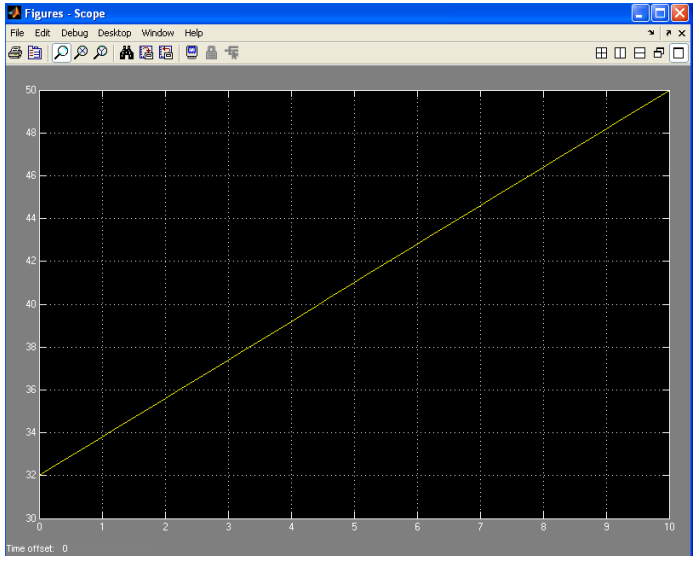
pre



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

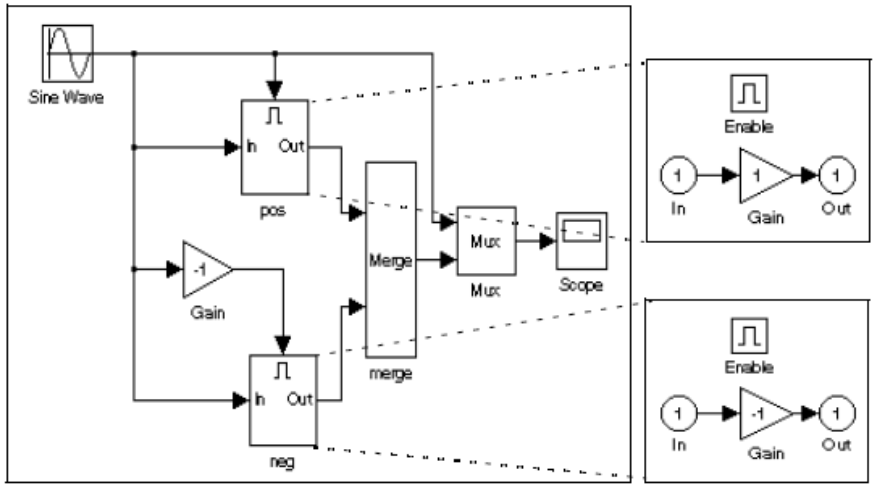
  - Merge (spajajući) blokovi
- **Podsistem (Subsystem)** predstavlja deo sistema čiji elementi međusobno interaguju nezavisno od sistema kao celine. **Uslovno izvršavajući podsistemi** (pod Ports & Subsystems), predstavljaju podsisteme čije izvršenje zavisi od vrednosti ulaznog signala (recimo, puštaju 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.

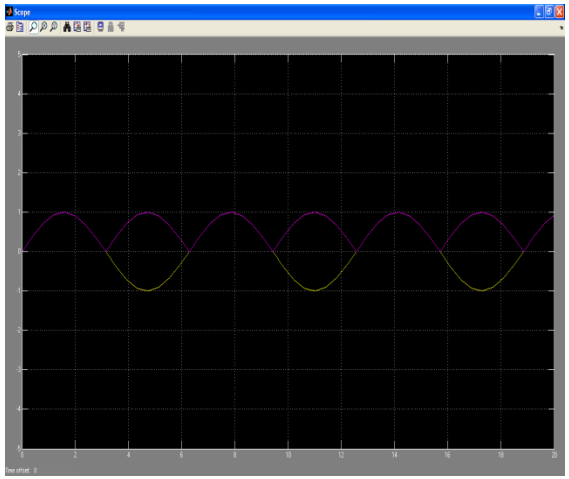
Opet, najbolje je ovo shvatiti na primeru:

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"

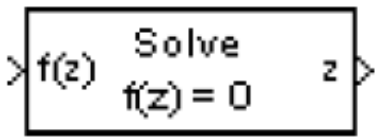
Rešavanje sistema algebarskih jednačina:

- 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:

$$Z2+Z1=1$$

$$Z2-Z1=1$$

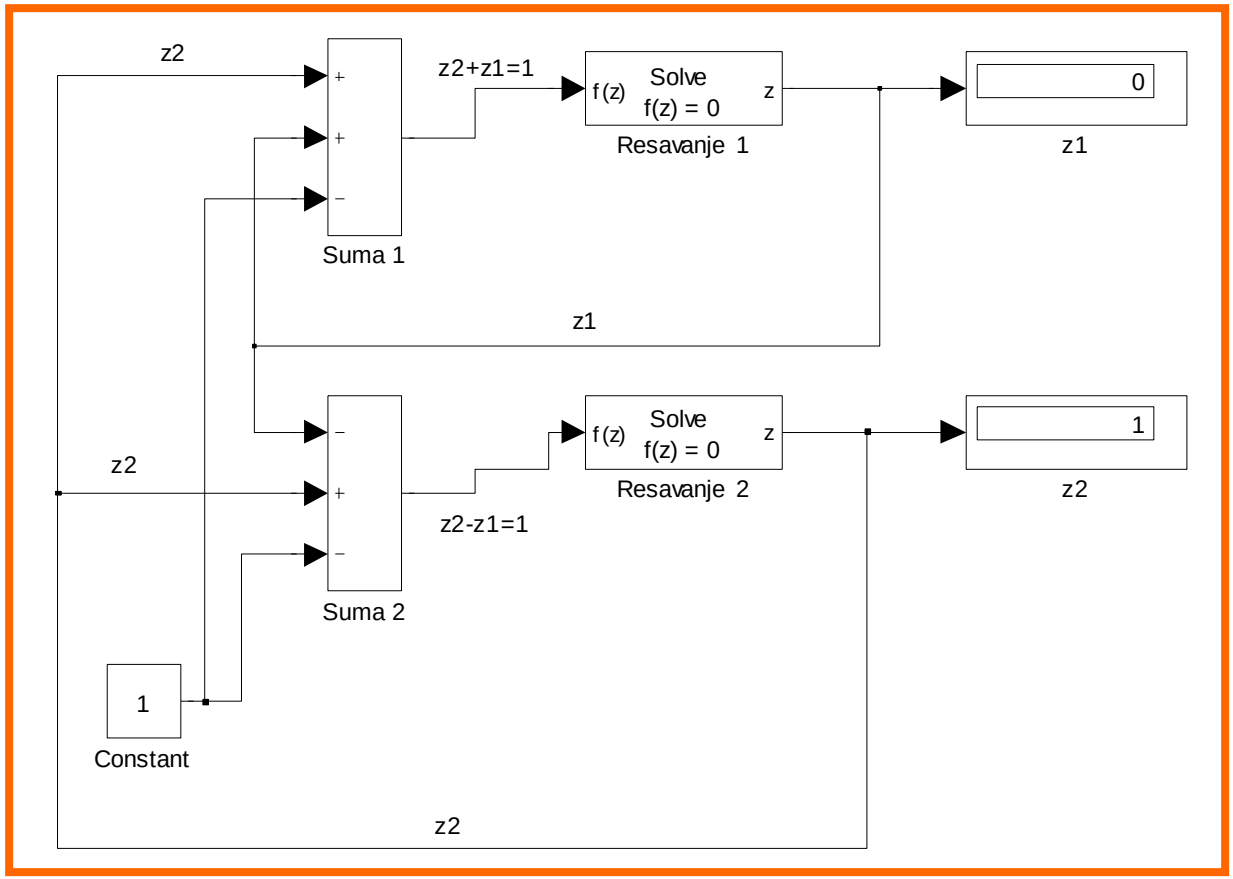
- Kao što se može lako zaključiti, rešenja ovog sistema su: Z2=1 i Z1=0. Napravimo univerzalni model koji nam može poslužiti za rešavanje sistema ovoga tipa.
- Da bi se to uradilo potrebno je pored poznatih uvesti još jedan od blokova a to je blok "Solve" (Simulink-Math Operations-Algebraic Constraint):



... 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