IMK-14 – Istraživanje i razvoj u teškoj mašinogradnji 26(2020)3, SR81-88
UDC 621 ISSN 0354-6829
doi: 10.5937/IMK2003081T

Slobodan softver kao alternativa komercijalnom u primeni numeričkih metoda i iterativnih postupaka

Marko Todorović[1*], Nebojša Bogojević[1]
[1]Fakultet za mašinstvo i građevinarstvo u Kraljevu, Univerzitet u Kragujevcu, Kraljevo (Serbia)

Uz pomoć odgovarajućeg softvera, računari omogućavaju brži rad i razvoj, olakšavaju izvođenje eksperimenata, simulacija, obradu rezultata na osnovu kojih je moguće izvući određene zaključke. Tokom proteklih decenija, razvijani su različiti programski jezici koji bi omogućili korisnicima računara da njima upravljaju i na taj način pristupe rešavanju određenih problema. Kako se ovaj proces proširio, i kako su mogućnosti ovih programskih jezika porasle usložavanjem istih, došlo se do mogućnosti kreiranja probnih razvojnih okruženja čiji je osnovni zadatak uprošćavanje i skraćivanje sintakse. Danas je u industriji najzastupljeniji softverski paket Matlab kao jedno od najkompleksnijih razvojnih okruženja sa velikim mogućnostima primene u različitim oblastima, od numeričkih proračuna do složenih simulacija. Međutim, to je komercijalni program za čiju je upotrebu potrebno izdvojiti poprilično dosta novca, te kao takav nije pristupačan učenicima, studentima, a ni obrazovnim institucijama. U okviru ovog rada biće razmotreno nekoliko paketa slobodnog softvera (free software) koji se mogu koristiti kao alternativa komercijalnim.

Keywords: GNU Octave, Matlab, numeričke metode, slobodan softver

1. UVOD

U nauci matematici, fizici, hemiji, inženjerstvu, ekonomiji, drugim prirodnim i društvenim naukama, posebno onim gde je izražena upotreba matematičkih izraza, algoritama, kvantitativnog opisivanja određenih pojava i formula, javlja se potreba za obradom prikupljenih podataka ili za sprovođenjem dugačkih i zamornih proračuna, koji se često i ponavljaju. Ranije, ti proračuni sprovođeni klasičnim računanjem, od strane jedne ili više osoba pri čemu bi jedna iteracija mogla trajati mesecima, a uz to je i postojala velika šansa za pojavu greške u proračunu pri čemu bi se, iz dobijenih, netačnih rezultata, izvlačili i netačni zaključci.

Opisan problem rešen je upotrebom računara - mašine koja može da izvši veliki broj proračuna u nekoliko iteracija za jako kratko vreme, neuporedivo kraće nego što bi to jedan tim sastavljen od ljudi mogao.

Evidentno je da je primena računara uspela da promeni svet, te izazove još jednu tehničku revoluciju. Danas, računari se nalaze svuda oko nas i, za razliku od prvobitnih koji su zauzimali čitave prostorije, računari se nalaze u džepovima većeg dela populacije.

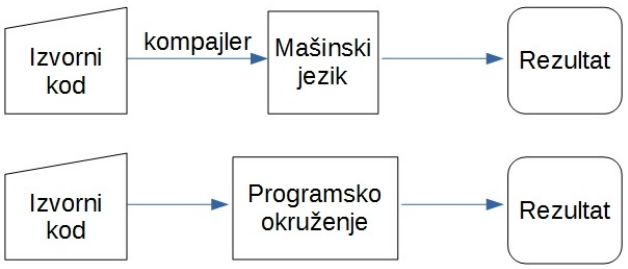
Tokom decenija razvoja računara, kako sa stanovišta fizičkih računarskih komponenti, tako i sa softverskog stanovišta, upravljanje ovim mašinama se razvijalo takođe. Računari su mašine i kao takvi oni razumeju mašinski jezik. Mašinski jezik je binarni jezik, tj. jezik koji se sastoji samo iz dve faze, faze *isključeno,*tj. nule (0) i faze *uključeno*, tj. jedinice (1). Pisati naredbe, tj. govoriti računaru, tj. konkretno računarskom procesoru šta želimo da on uradi na mašinskom jeziku jeste moguće, ali nije praktično. Bar nije praktično za čoveka. Iz tog razloga smišljeni su programski jezici koji se sastoje iz reči, mahom engleskog jezika, koje čovek može da razume, te na taj način može i lakše da organizuje naredbe koje daje računarskom procesoru na izvršenje. Međutim, kao što je rečeno ranije, računarski procesor razume samo mašinski jezik. Iz tog razloga, programski jezik se uz pomoć kompajlera u prevodi na mašinski jezik u procesu koji se naziva kompilacija. Računarski procesor može da razume samo komande koje su prošle kroz proces kompilacije. Postoji jako veliki broj programskih jezika, pri čemu svaki ima svoje prednosti i svoje mane.se

U zavisnosti od odabranog programskog jezika pri pisanju programa, u većoj ili manjoj meri, potrebno je obratiti pažnju na mnoge parametre, kao što su arhitektura procesora, platforma na kojoj će se program koristiti, upotreba memorije, tipove podataka i mnoge druge stvari koje samo kodiranje čine sporijim. U međuvremenu se javila ideja o kreiranju nekog zajedničkog okruženja koje bi omogućilo da se jedan te isti kod tumači identično na različitim platformama. Na taj način, potrebno je napraviti jedno okruženje u kojme se izvršava kod, pa je samo okruženje dovljno prilagoditi različitim platformama, a kod koji je jednom napisan će se u datom okruženju ponašati identično na svakoj platformi na kojoj je postavljeno dato okruženje. Na taj način, onaj koji piše kod ne mora da zna gde će se njegov program pokretati, niti koja je arhitektura uređaja korisnika programa, već mu je dovoljno da zna da uređaj na kojem se pokreće program ima dato programsko okruženje.

Jezici koji se izvršavaju u takvim okruženjima se često u literaturi nazivaju i*nterpretiranim programskim jezicima*, jer se kod napisan korišćenjem takvih jezika tumači i izvršava u okviru programskog okruženja. Ovakvi programski jezici pripadaju grupi programskih jezika visokog nivoa, kao i programski jezici čiji se kod prevodi direktno u mašinski jezik koji računarski procesor razume. Interpretirani programski jezici su višeg nivoa u odnosu na one koji se kompajliraju i karakteriše ih jednostavnija sintaksa i bolja čitljivost koda od strane ljudi, što pisanje i analiziranje koda napisanog od strane nekog drugog programera jednostavnijim. Međutim, samom programskom okruženju u kome se napisani kod tumači, potrebni su određeni fizički resursi, te su programi pisani u takvim programskim jezicima hardverski zahtevniji i izvršavaju se sporije od programskih jezika nižeg nivoa, tj. onih kod kojih se kod prevodi u mašinski jezik. Dijagram

*Kontakt adtresa autora: Srbija, 36000 Kraljevo, Naselje Moše Pijade 36/76, todorovic.m@mfkv.kg.ac.rs

koji oslikava razliku između ove dve porodice programskih jezika prikazan je na slici 1.



Slika 1: Šematski prikaz izvršenja koda kod kompajliranog i interpretiranog programskog jezika

Pri razvoju novih softverskih rešenja, autori softvera se susreću sa *Problemom dva jezika* [1], pri čemu moraju da biraju između jednostavnosti i čitljivosti koda jezika visokog nivoa i kompleksosti, ali boljoj iskorišćenosti računarskih resursa, programskih jezika nižeg nivoa. Često se u praksi novi principi rešavanja problema, novi algoritmi i ideje implementiraju jezicima visokog nivoa, da bi se kasnije, nakon što se dokaže da određeni algoritam ili određeno rešenje funkcioniše kako je planirano i daje zadovoljavajuće rezultate, prevodi u neki od programskih jezika nižeg nivoa.

Programska okruženja se obično razvijaju iz potreba određene grane industrije, pa su ona s toga prilagođena da odgovore na specifične probleme određene grane industrije, kao što je sprovođenje dugačkih proračuna ili simulacije kretanja fluida.

Specijalizovana programska okruženja našla su primenu i u različitim granama nauke, kao što su matematika, fizika, inženjerstvo, statistika, biologija i druge. Međutim, iako je pisanje programa u jezicima visokog nivoa jednostavnije i brže, oni nisu pogodni za pisanje programa za široku upotrebu, mada postoje alati koji omogućavaju da se sam kod pisan u, na primer Python programskom jeziku prevede u neki jezik nižeg nivoa, poput C ili C++ programskih jezika.

Jednostavnijom sintaksom u odnosu na ostale programske jezike, kompatibilnosti među različitim platformama i činjenicom da nije potrebno brinuti o tipu podataka niti o memoriji, ovakvi jezici su primamljiviji za upotrebu i u edukativne svrhe. Uz pomoć njih moguće je, na brz i jednostavan način bez udubljivanja u nauku kodiranja, programiranja i specifičnosti date arhitekture, uz pomoć svega par jednostavnih naredbi isprobati različita rešenja, izvršiti proračune, obraditi podatke i tako brže, i uz manje uloženog vremena doći do željenih rezultata. Ovo može omogućiti učenicima da se usresrede na rešavanje konkretnih problema umesto na pisanje koda, specifičnosti određene platforme i dr.

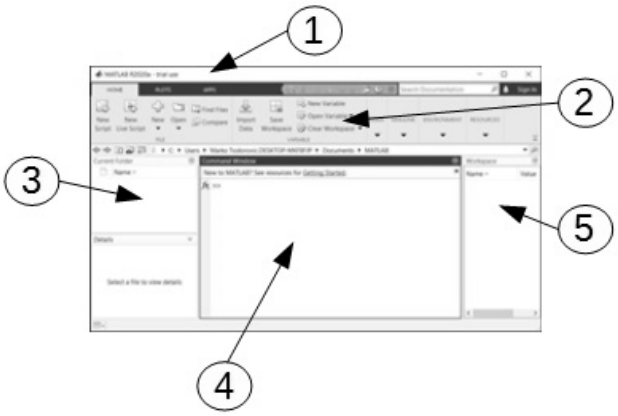
2. KOMERCIJALNI SOFTVERSKI PAKETI

2.1. MathWorks Matlab

Jednostavnije i brže rešavanje problema dovoljni su razlozi za primenu interpretiranih jezika u industriji. Kompanija MathWroks je razvila jedan od najpoznatijih i najkompletnijih rešenja, softverski paket i programski jezik Matlab, koji spada u grupu interpretiranih programskih jezika visokog nivoa, poput programskog jezika Python ili Julia, pri čemu je samo pisanje programa jednostavnije nego u drugim programskim jezicima visokog nivoa poput PASCAL-a ili C-a.

Matlab odlikuje sintaksa koja je većini ljudi koji imaju bilo kakvo iskustvo u pisanju softvera već poznata, a za one koji nemaju nikakvo prethodno iskustvo dovoljno jednostavna što omogućava da obuka za korišćenje ovog softverskog paketa ne traje dugo, što je omogućilo da Matlab brzo nađe svoje mesto u industriji. Ovaj programski jezik je specijalizovan za rad sa jednodimenzionalnim (vektori) i više dimenzionalnim (matrice) nizovima, a obilje alata i modula dodatno proširuju ili pojednostavljuju obavljanje različitih operacija na različitim poljima u nauci i industriji.

Grafičko korisničko okruženje, koje je prikazano na slici 2, sastoji se iz nazivne trake (1), trake sa paletom alata (2), leve bočne trake (3) sa prikazom sadržaja radne fascikle, komandnog prozora u sredini (ili konzole) u kojem se direktno mogu zadavati komande (4), dok se u desnoj bočnoj traci (5) nalazi spisak upotrebljenih promenljivih sa njihovim vrednostima.



Slika 2: Grafičko okruženje Matlab-a: *1 – nazivna traka, 2 – traka sa paletom alata, 3 – leva bočna traka, 4 - komandni prozor, 5 – desna bočna traka*

Izvršavanje naredbi u Matlabu je moguće na dva načina. Unošenjem naredbi u konzolu (izvršni deo interfejsa Matlaba – 4 na slici 2) i pritiskom tastera ENTER na tastaturi, moguće je izvršavati jedan po jedan red koda. U ovom slučaju matlab možemo posmatrati kao jedan napredni kalkulator. Drugi način je istovetan sa napednijim programskim jezicima – pisanjem odgovarajućeg skupa naredbi – skripti, koje se izvršavaju jedbna za drugom, premo unetom redosledu [2]. Pisanjem skripti, upotreba petlji i naredbi odlučivanja postaje moguća, što proširuje mogućnosti Matlab programskog jezika.

Ovaj softverski paket već poseduje ugrađen alat za grafičku reprezentaciju podataka i rezultata u vidu dvodimenzionalnih i trodimenzionalnih grafikona, kao i dijagrame različitih oblika. Za naprednije korisnike postoji i alat za kreiranje grafičkog korisničkog okruženja za programe pisane u Matlab programskom jeziku što takve programe čini jednostavnijim za upotrebu, naročito ako program koristi neko ko nije učestvovao u njegovom pisanju.

2.2. Wolfram Mathematica

Wolfram Mathematica je još jedan komercijalni softverski alat koji služi za obavljanje različitih matematičkih operacija.

Za razliku od MathWorks Matlab-a, Wolfram Mathematica Matematica je više usmerena na rad sa simboličkim promenljivima, odnosno sa varijabla koje nemaju numeričke vrednosti. Da ne bi bilo zabune, I Matlab ima vrlo dobre biblioteke za rad sa simboličkim promenljivima, ali u ovom trenutku, Matematica je daleko najnapredniji softverski paket za rad sa ovom vrstom promenljivih. Samim tim, kao izlaz iz proračuna ne dobija se uvek brojčani rezultat, već se može dobiti i izraz u opštim brojevima.

Sintaksa koju koristi jezik programskog paketa Wolfram Mathematica nije kompatibilna sintaksi programskog paketa MathWorks Matlab-a.

3. ALTERNATIVNI SOFTVER OTVORENOG KODA

Krajem dvadesetog veka, kada su računari već ušli u mnoge domove i ljudi su počeli da se navikavaju na njihovo sve veće prisustvo u svojim životima, pojavila se ideja da softver treba da bude slobodan, u smislu pisanja, distribucije i upotrebe. Zastupnici ove ideje smatraju da korisnici softvera treba da imaju pravo da isti softver koriste, dele, ali i menjaju po svojim potrebama, te izmenjen softver dele drugim ljudima na isti način i pod istim uslovima. Na taj način, zainteresovana osoba ima mogućnost da pogleda kod programa i na taj način razume kako je program napisan, na koji način je ostvarena neka funkcija i u mogućnosti je da izmeni nešto ukoliko smatra da je to potrebno, ili ukoliko to jednostavno želi, bez kršenja autorskih prava, a samim tim i zakona. Iz pomenutih razloga I uverenja, 1983. godine Ričard Stalman je osnovao GNU (GNU is not Unix) fondaciju pod čijom se licencom danas nalazi jako veliki broj računarskih alata, među kojima je najpoznatiji operativni sistem GNU Linux [3,4].

Svako softversko rešenje, program ili deo programa koji se nalazi pod GNU licencom je slobodno za upotrebu, modifikaciju i distribuciju, i kao takav je idealan u edukativne svrhe. Objavljivanjem izvornog koda programa zajedno sa samim programom, korisnici dobijaju uvid u način na koji je program napisan, i mogu sami da isprave greške nastale pri kodiranju ili programiranju, ili samostalno dopišu i na taj način modifikuju, poboljšaju ili prilagode program svojim potrebama. Osnivač pokreta, Ričard Stalman tvrdi da je jedna od glavnih prednosti ovakvog načina distribucije upravo mogućnost da se vidi kod, jer da bi programeri postali dobri programeri, oni moraju da čitaju dosta programa, da vide kod velikih programa i velikih sistema pri čemu njihov prvi samostalni korak treba biti uspešna izmena manjeg dela dugačkih kodova [4].

Pod otvorenom GNU licencom nalaze se i programska okruženja koja mogu da posluže kao alternative komercijalnim i u primeni numeričkih metoda i iterativnih postupaka. U okviru ovog rada biće prikazana dva softverska paketa izdata pod GNU licencom: GNU Octave i SciLab.

3.1. GNU Octave

GNU Octave je programski jezik visokog nivoa sa fokusom na numeričkim proračunima, a obično se koristi za rešavanje linearnih i nelinearnih jednačina, numeričke proračune i statističke analize [5].

Sam softverski paket se sastoji iz konzole u kojoj se mogu izvršavati komande njihovim direktnim unošenjem putem tastature i pritiskom na taster ENTER na kraju svake komande, što je prikazano na slici 3.

```
GNU Octave, version 5.2.0
Copyright (C) 2020 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.octave.or

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> disp("Hello world")
Hello world
octave:2>
```

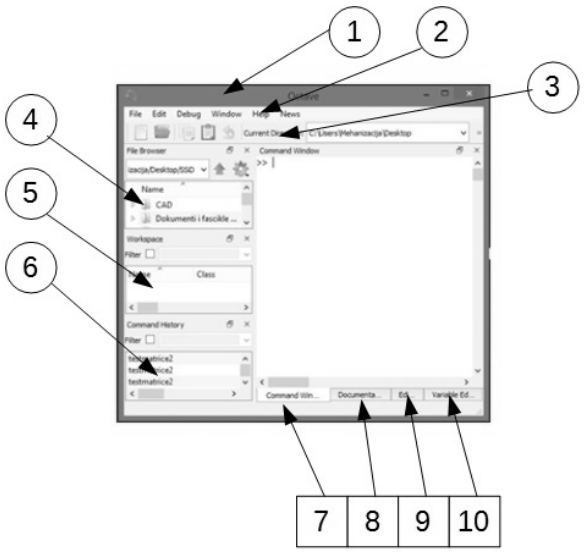
Slika 3: Ispisivanje „Hello world“ rečenice u GNU Octave konzoli

Osim konzole, tu je i grafičko okruženje prikazano na slici 4 koje olakšava korišćenje ovog programskog okruženja istovremeno mu proširujući i mogućnosti. Ono se sastoji iz nazivne trake (1), trake sa padajućim menijima (2), trake sa paletom osnovnih alata (3), dok se u levoj bočnoj traci nalazi prikaz radnog direktorijuma (4), spisak promenljivih (5) i istorija korišćenih komandi (6). Sa desne strane, u radnom delu prozora, nalazi se komandna linija (7), dokumentacija u kojoj je moguće potražiti tačne nazive komandi kao i uputstva za njihovu upotrebu (8), uređivač teksta u kome se mogu pisati skripte, tj. liste komandi koje je potrebno izvršiti, a te skripte je moguće sačuvati i pokretati iznova, po potrebi (9).

U radnom delu prozora nalazi se i Variable Editor (10) koji izlistava promenljive i njihove vrednosti u vidu tabele. Vrednosti promenljivih u okviru ove tabele moguće je izmeniti dvoklikom na polje koje treba izmeniti. Tako izmenjenu promenljivu, moguće je ponovo koristiti u daljim proračunima. Tabelu je, takođe, u celosti moguće izvest iz programa u .txt formatu. Prelazak iz jednog u drugi alat u središnjem delu grafičkog interfejsa moguće je izvršiti klikom na odgovarajuću karticu koja se nalazi u podnožju centralnog dela grafičkog korisničkog interfejsa.

Sintaksa GNU Octave jezika je slična i gotovo identična sintaksi Mathworksovog Matlab-a. Skripte koriste i isti sufiks (ekstenziju), tj. “.m“ sufiks, koji omogućava da skripte pisane u GNU Octave jeziku i programskom okruženju mogu biti pokrenute i u Mathworks Matlab-u.

Međutim, treba imati na umu i da „gotovo identična sintaksa“ ne znači isto što i „identična sintaksa“, jer ipak postoje određene razlike koje ih čine nekompatibilnim. Na primer, na slici 5 prikazan je primer jedne “for” petlje sa brojačem napisane u MathWorks Matlab-u, dok je na slici 6 prikazan primer iste petlje napisane u GNU Octave jeziku.



Slika 4: GNU Octave – grafičko okruženje: 1 – nazivna traka, 2 – meni traka, 3 – traka sa paletom alata, 4 - prikaz radnog direktorijuma, 5 – spisak promenljivih, 6 – istorija upotrebljenih komandi, 7 – komandni prozor, 8 – dokumentacija, 9 – uređivač teksta za pisanje skripti, 10 – Variable Editor

```
clc;clear;

s = 10;
H = zeros(s);

for c = 1:s
    for r = 1:s
        H(r,c) = 1/(r+c-1);
    end
end

disp(H)
```

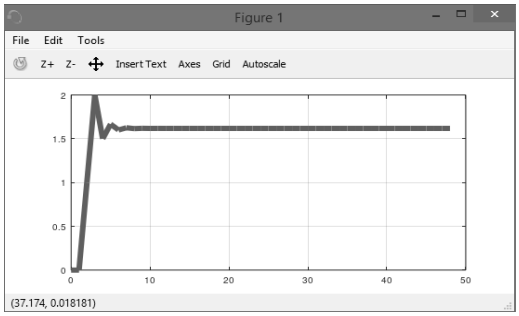
Slika 5: Primer „For“ petlje u Mathworks Matlab programskom jeziku [6]

```
clc; clear;

s=10
H=zeros(s);

for c=1:s
  for r=1:s
    H(r,c)=1/(r+c-1);
  endfor
endfor

disp(H)
```

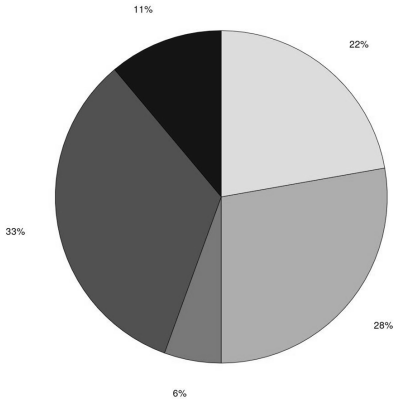
Slika 6: Primer „For“ petlje u GNU Octave programskom jeziku

Kao što se može primetiti na slikama 5 i 6, izlazak iz petlje u Mathworks Matlab programskom jeziku se označava rezervisanom reči “end”, dok u GNU Octave programskom jeziku istu funkciju ostvaruje reč “endfor”. Imajući u vidu ove male razlike u sintaksi, pre pokretanja većih skripti pisane u GNU Octave programskog jezika unutar Matlab okruženja potrebno je prilagoditi kod unutar same skripte, a isto važi i ako je potrebno pokrenuti skriptu pisanu u Matlab programskom jeziku unutar GNU Octave okruženja.

Kao i Matlab, GNU Octave programsko okruženje ima ugrađen alat za vizuelizaciju podataka u vidu dvodimenzionalnih i trodimenzionalnih grafikona, kao što je prikazano na slici 7, i dijagrama različe vrste i oblika, kao što je prikazano na slici 8, koji je nešto slabijih mogućnosti u odnosu na onaj ugrađen u Mathworks Matlab programsko okruženje.



Slika 7: Grafički prikaz zlatnog preseka Fibonačijevog niza u GNU Octave programskom okruženju



Slika 8: Kružni grafikon (Pie chart) u GNU Octave programskom okruženju

U nazivu samog GNU Octave jezika nalazi se akronim GNU, što znači da je celokupan projekat pod GNU otvorenom licencom, što sa sobom povlači sve one dobre, ali i loše osobine slobodnog softvera. Naime, sloboda GNU Octave čini besplatnim za upotrebu, kako u lične, edukativne, tako i u komercijalne svrhe. Programski jezik i programsko okruženje razvija grupa volontera, što znači da iza projekta ne stoji ni jedna velika korporacija koja ima direktan profit od razvoja ovog rešenja. S toga, razvoj GNU Octave je nešto sporiji, a celo okruženje odlikuje manja stabilnost u radu u odnosu na komercijalne alternative.
Takođe, svi dodaci koji postoje za GNU Octave su pisani od strane zajednice korisnika, tj. volontera, za razliku od, recimo Matlab-a čije dodatke pišu plaćeni programeri. To je ujedno i prednost GNU Octave projekta, jer i u slučaju da trenutna grupa volontera odustane od razvoja ovog jezika,

izvorni kod je javno dostupan, pa neka druga grupa volontera može da nastavi da razvija programski jezik, za razliku od komercijalnih varijanti gde, ako kompanija koja ih razvija odustane od projekta, onda su ti projekti trajno ugašeni, što se dešavalo u prošlosti. Na primer, različite varijacije programskog jezika Pascal su razvijane od strane brojnih komercijalnih kompanija, da bi na kraju opstale samo dve varijacije do danas od kojih je jedna slobodnog koda, FreePascal, a druga TurboPascal u okviru Delphi okruženja, danas pod okriljem kompanije Embracadero.

3.2. SciLab

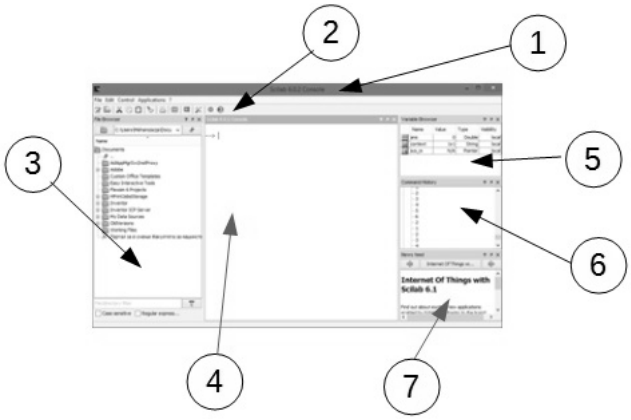
Scilab je još jedan programski paket namenjen numeričkim proračunima. Poput GNU Octave, i SciLab je otvorenog koda, pa je slobodan za upotrebu, menajnje i distribuciju. Međutim, osim grupe volontera, iza SciLab-a, od 2017. godine stoji i ESI Grupa koja se bavi rešenjima za pravljenje virtualnih prototipova.

Sintaksa je, takođe jako slična Mathworks Matlab programskom jeziku, ali takođe postoje razlike. Za razliku od GNU Octave programskog okruženja, SciLab ima sopstveni sufiks za svoje svoje skripte, i kao takve se ne mogu direktno pokrenuti u MathWorks Matlab niti GNU Octave programskim okruženjima. Jedna od razlika u sintaksi uključuje i broj pi koji se u SciLab-u poziva rezervisanom reči „%pi“, dok se u MathWorks Matlab i GNU Octave programskim jezicima poziva komandom „pi“.

Programski paket se sastoji od konzole u kojoj se mogu direktno, putem tastature unositi i izvršavati komande, kao što je prikazano na slici 9, uz koju dolazi i grafički interfejs prikazan na slici 10, koji po dizajnu grafičkog korisničkog interfejsa podseća na onaj koji ima MathWorks Matlab.

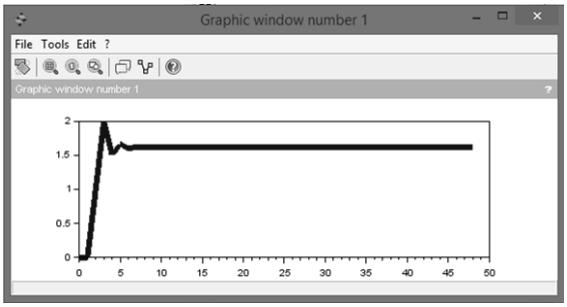
```
Scilab 6.0.2 (Feb 14 2019, 10:31:37)

Start CEMEF-UTN Analysis for Truss structures
       Load macros


--> A=round(rand(10,10)*100)
 A  =

   21.   56.   31.   50.   28.   41.   39.   54.   59.   65.
   76.   66.   93.   44.   13.   88.   92.   12.   48.   99.
   0.    73.   21.   27.   78.   11.   95.   23.   22.   5.
   33.   20.   31.   63.   21.   20.   34.   63.   84.   75.
   67.   54.   36.   41.   11.   56.   38.   76.   12.   41.
   63.   23.   29.   92.   69.   59.   73.   5.    29.   61.
   85.   23.   57.   4.    15.   69.   26.   67.   86.   85.
   69.   22.   48.   48.   70.   89.   50.   20.   85.   6.
   88.   88.   33.   26.   84.   50.   26.   39.   53.   83.
   7.    65.   59.   41.   41.   35.   53.   83.   99.   93.


-->
```

Figure 9: SciLab console; displaying 10x10 matrix



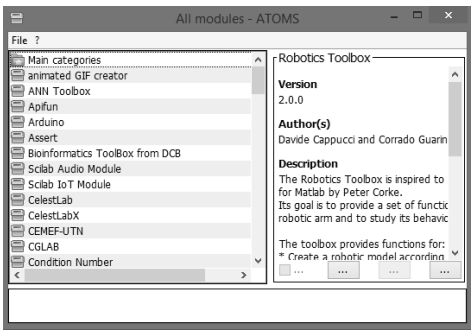
Slika 10: Grafičko korisničko okruženje SciLab programskog paketa: 1 – naslovna traka, 2 – traka sa osnovnim alatima, 3 – levi bočni panel sa prikazom sadržaja radnog foldera, 4 – komandni prozor, 5 – spisak promenljivih, 6 – istorija upotrebljenih komandi, 7 - obaveštenja o novim verzijama

U SciLab programsko okruženje takođe je ugrađen alat za vizualno predstavljanje podataka, koji je prikazan na slici 11, sličnih mogućnosti kao i onaj u GNU Octave programskom paketu, međutim komanda kao što je „grid on“ neće raditi u SciLab programskom okruženju, dok će u GNU Octave i Mathworks Matlab programskim okruženjima iscrtati mrežu koja se sastoji od bledih horizontalnih i vertikalnih linija na prikazanom grafiku. Umesto „grid on“ komande, u SciLab-u se koristi komanda „xgrid(2)“ u kojoj parematar 2 označava boju linija kreirane mreže.

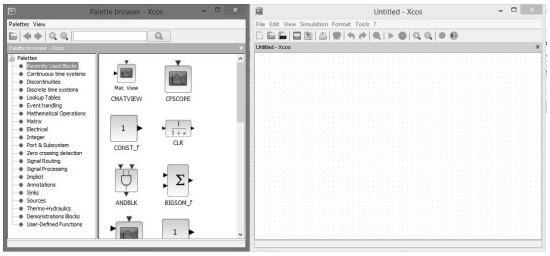


Slika 11: Grafički prikaz zlatnog preseka Fibonačijevog niza u SciLab programskom okruženju

Funkcionalnost programskog okruženja moguće je proširiti raznim dodacima koji su proizvod rada volonterskih timova i zajednice korisnika, a ovi dodaci objedinjeni su u ATOMS menadžeru paketa koji je prikazan na slici 12. S obzirom da su i ovi alati pod istom licencom kao i SciLab, i njih je moguće slobodno koristiti, distribuirati i menjati po želji ili potrebi. Unutar SciLab paketa postoji i modul za simulacije, koji predstavlja alternativu Mathworks Simulink-a otvorenog koda, pod nazivom Xcos (slika 13) čiji se grafičko korisničko okruženje sastoji od dva prozora, gde je jedan prozor paleta sa alatima su u grupama svrstani objekti potrebni za izgradnju modela, a drugi prozor predstavlja radni prozor gde se, iz palete sa alatima mogu prevući objekti i komponente čime se izgrađuje model za simulaciju.



Slika 12: ATOMS menadžer paketa



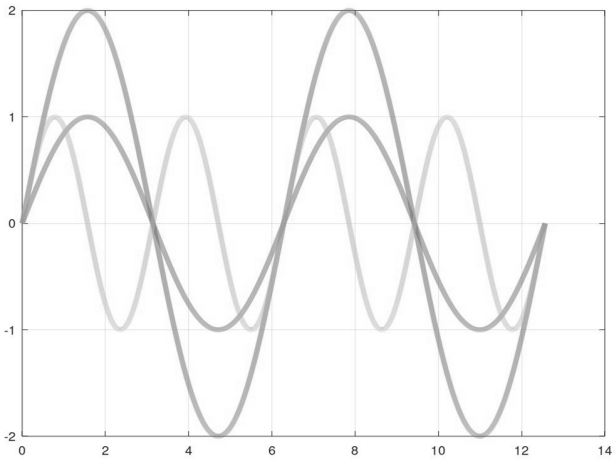
Slika 13: Xcos alat: levo – paleta sa alatima, desno - radni prostor

Za razliku od komercijalnih softverskih paketa, softver otvorenog koda nema ili ima veoma slabu podršku za rešavanje problema sa simboličkim promenljivima. S toga, ukoliko je potrebno da se rešavaju problemi sa simboličkim promenljivima, onda treba odabrati neka od komercijalnih rešenja, a u koliko nije značajna upotreba simboličkih promenljivih, onda je moguće osloniti se na softver otvorenog koda.

4. REŠAVANJE PROBLEMA UZ POMOĆ GNU OCTAVE I SCILAB SOFTVERSKIH PAKETA

Uloga pomenutih programskih jezika visokog nivoa je da skrate put do nalaženja najboljeg rešenja nekog problema i da skraćenjem vremena potrebnog da bi se došlo do rešenja poveća produktivnost u radu, što se, u odnosu na programske jezike nižeg nivoa čini skraćivanjem i pojednostavljivanjem sintakse programskog jezika što omogućava osobi koja piše program da napiše program uz pomoć manjeg broja komandi što dovodi do smanjenja utrošenog vremena na pisanje programa koji obavlja istu funkciju. Osobama koje počinju da uče programiranje je jednostavnije da razumeju suštinu programiranja umesto uz pomoć ovih programskih jezika zbog jednostavnije sintakse.

Osim u svrhe učenja programiranja, ova programska okruženja se mogu koristiti i za grafičku reprezentaciju, na primer, matematičkih funkcija jer učenici mogu videti kako se grafik funkcija menja promenom parametara unutar izraza. Alat za grafičko prikazivanje podataka se takođe može koristiti za objašnjavanje analitičke geometrije ili oblika talasa u fizici, kao što je prikazano na slici 14.



Slika 14: Grafički prikaz funkcija: sin(x),2sin(x) and sin(2x) u GNU Octave programskom okruženju

Jednostavnost upotrebe programskih jezika visokog nivoa, u ovom konkretnom slučaju SciLab-a i GNU Octave, najlakše je uočiti pri radu sa višedimenzionalnim nizovima, tj. u ovom konkretnom slučaju, matricama, što će biti pokazano u sledećem primeru.

Potrebno je napraviti program kojim se definiše matrica proizvoljnih dimenzija, a čiji su članovi nasumični brojevi od 0 do 1000 iz skupa prirodnih brojeva.

Postoji veliki broj načina na koji se ovaj problem može rešiti uz pomoć bilo kojeg programskog jezika. Jedno od rešenja napisano uz pomoć programskog jezika FreePascal, koji je programski jezik čiji se kod kompajlira, prikazano je na slici 15.

```
program Matrica_nasumicnih_brojeva;
type
  matrica = array of array of integer;
var
  n,m,i,j:integer;
  A:matrica;

begin
  write('Broj redova n: ');
  readln(n);
  write('Broj kolona m: ');
  readln(m);
    setlength(A,m,n);
  for i:=0 to m-1 do begin
    for j:=0 to n-1 do begin
      A[i][j]:=round(random*1000);
      end;
    end;
  writeln('Matrica A: ');
   for i:=0 to m-1 do begin
    for j:=0 to n-1 do begin

      write(A[i][j], '   ');
      end;
    writeln('');
    end;
    readln();
end.
```

Slika 15: Kod rešenja zadatka u programskom jeziku FreePascal

Posmatrajući sliku 15 može se videti da je za rešenje opisanog problema potrebno napisati 28 redova koda. Kompajliranjem i izvršenjem ovog koda, dobija se rezultat koji je prikazan na slici 16.

```
Broj redova n: 10
Broj kolona m: 10
Matrica A:
549   593   715   844   603   858   545   847   424   624
646   384   438   298   892   57   964   273   383   478
792   812   529   480   568   393   926   836   71   337
87   648   20   368   833   957   778   140   870   870
979   474   799   801   461   520   781   679   118   721
640   582   143   537   945   759   522   106   415   474
265   186   774   737   456   217   568   135   19   324
618   150   612   222   617   386   944   903   682   450
360   613   437   902   698   99   60   970   667   653
671   171   210   358   129   751   315   608   364   325
```

Slika 16: Izvršeni kod rešenja opisanog problema u programskom jeziku FreePascal

Kod prikazan na slici 15 se sastoji iz više segmenata. Zaglavlje programa, u kome se definiše naziv programa, kao i promenljive koje će se koristiti u samom programu, pri čemu je obavezno unapred odrediti njihov tip. Komande koje će se izvršiti unutar programa nalaze se između rezervisane reči **Begin**, koja označava početak programa, i rezervisane reči **end**, koja označava kraj programa. Unutar ove dve rezervisane reči, mogu se uočiti čak četiri **for** petlje, pri čemu su dve potrebne za definisanje članova dvodimenzionalnog niza, a dve za ispisivanje dvodimenzionalnog niza na zaslonu ekrana, kao što je prikazano na slici 16.

Prednost GNU Octave programskog jezika nad FreePascal programskom jeziku se može uočiti posmatrajući rešenje istog opisanog problema napisanog u GNU Octave programskom jeziku, prikazanom na slici 17.

```
1  clc; clear;
2
3  m=input('Broj redova matrice m: ')
4  n=input('Broj kolona matrice n: ')
5  A=round(rand(m,n)*1000)
```

Slika 17: Kod rešenja opisanog problema u GNU Octave programskom jeziku

Prvi red koda koji je prikazan na slici 17 je potreban da bi se očistio zaslon ekrana i memorija programskog okruženja radi sprečavanja uticaja prethodno izvršenih komandi ili prethodno pokrenutih skripti na rezultate koji se dobijaju pokretanjem ovog koda. Definisanje i ispisivanje dvodimenzionalnog niza postiže se uz pomoć komande ispisane u petom redu koda. Rezultat koji se dobija pokretanjem ove skripte prikazan je na slici 18.

Nakon izvršenja kompajliranog koda napisanog u FreePascal programskom jeziku nije moguće ponovo pristupiti definisanim promenljivim niti je moguće obavljati operacije sa njima koje nisu definisane u kodu pre kompajliranja, osim ako se ne izmeni izvorni kod i pokrene ponovo. Ako se pokrene skripta u okviru GNU Octave programskog okruženja, nakon izvršenja poslednjeg reda koda iz skripte, moguće je naknadno pristupiti svim promenljivim koje su definisane pokretanjem skripte uz pomoć komandne linije bez potrebe da se izvorni kod skripte menja i ponovo izvršava.

```
Broj redova matrice m: 10
m =  10
Broj kolona matrice n: 10
n =  10
A =

   575   118   750   280   672   262   730    73   538
   417   445   334    45   986   183   825   402   136
   138    77   832   434   338    58   372   743   930
   846   632   112   520   104   783   216   527   211
   696   483   751    36   117   639   722   609   448
   787   561   182   176   234   444   826    63   549
   291   228   423   193   110   967   379   426   760
   585   157   874   861   166   996   117   384   211
   653   424   637   907   454   705   408   686   844
   193   298   183   516   864   122   705   734   959
```

Slika 18: Dvodimenzionalni niz prikazan u GNU Octave programskom jeziku

Na primer, nakon izvršenja koda prikazanog na slici 17 moguće je izračunati determinantu dobijene matrice, kao i odrediti njenu inverznu matricu jednostavnim unošenjem komandi unutar komande linije, što je prikazano na slici 19.

```
>> det(A)
ans =    4.2775e+26
>> inv(A)
ans =

 Columns 1 through 6:

   0.001267460   0.001107409   0.015382939   0.013491466  -0.011966239   0.006872015
   0.003218100  -0.006655033  -0.051061015  -0.036960196   0.040101553  -0.026448631
   0.002271556  -0.003138084  -0.021520127  -0.015946273   0.017618313  -0.011940199
  -0.000387412  -0.001714031  -0.007255390  -0.005729931   0.005487590  -0.002512002
   0.001325292   0.000750979  -0.001767795  -0.000461202   0.000324521  -0.002123673
  -0.001582406   0.002443537   0.011053389   0.007617832  -0.009389396   0.006141345
  -0.001914811   0.000292451   0.006037775   0.003223588  -0.003972915   0.005329479
  -0.002040291   0.002675955   0.019253105   0.014222418  -0.014040219   0.009237252
   0.002853369  -0.003461647  -0.016185376  -0.010711530   0.012516597  -0.009218186
  -0.003414303   0.005182193   0.026044511   0.017091652  -0.020737761   0.014237262

 Columns 7 through 10:

   0.002431502   0.007633965  -0.024805008   0.001125676
  -0.009031795  -0.029653055   0.081159972  -0.004550744
  -0.004002827  -0.012334902   0.034288825  -0.002137358
  -0.002433946  -0.001981270   0.010104390   0.001103706
   0.000468244  -0.002074924   0.003464170  -0.000933350
   0.003211004   0.006692372  -0.017244681   0.000176128
   0.000105810   0.006199469  -0.012605434   0.002919241
   0.003006565   0.010923009  -0.030335115   0.002147329
  -0.002126016  -0.010398443   0.025283007  -0.000862839
   0.004791175   0.013902017  -0.037567698  -0.000092636
```

Slika 19: Izračunavanje determinante i inverzne matrice

Isti kod prikazan na slici 17 bez ikakvih izmena daje isti rezultat i kada se pokrene unutar MathWorks Matlab programskog okruženja, kao i unutar SciLab programskog okruženja, na bilo kojoj platformi na kojoj je moguće pokrenuti sama programska okruženja (Windows, Linux, MacOSX).

5. ZAKLJUČAK

Slobode koje nude open source alternative, poput SciLab ili GNU Octave programskih okruženja, najveće su prednosti ovih alternativa u odnosu na komercijalne, posebno kada je reč o upotrebi u edukativne svrhe. Nije potrebno kupiti potrebne licence jer je softver otvorenog koda slobodan za upotrebu u edukativne i komercijalne svrhe. Umesto kupovine licenci za softver, sredstva se mogu usmeriti na nabavku novih harverskih komponenti, ili neki drugi alat koji bi poboljšao kvalitet nastave unutar obrazovne institucije. Takođe, studenti mogu slobodno preuzeti i koristiti softver kod kuće bez ikakve novčane naknade, što školovanje čini pristupačnijim. Ako se uzme u

obzir da učenička licenca MathWorks Matlab programskog okruženja iznosi 69€ [7], da se SciLab i GNU Octave izučavaju u toku samo jednog semestra ili školske godine, kao i da učenici najčšće neće imati potrebe za naprednim alatima koji dolaze uz komercijalne alternative, opravdanost upotrebe komercijalnih rešenja postaje upitna.

Razlike između alternativa otvorenog koda i komercijalnih softverskih rešenja su, posebno u jednostavnijim slučajevima upotrebe koji se izučavaju u oviru kurseva u srednjim školama i na fakultetima, neznatni. Učenik koji ume da napiše skriptu u GNU Octave ili SciLab programskom jeziku neće imati većih problema prilikom prelaska na MathWorks Matlab programski jezik. Komercijalna rešenja su kompletnija i stabilnija pri korišćenju, te postoji velika verovatnoća da će se učenici i studenti susresti s njima u budućnosti, u poslovnom okruženju. Kako su razlike u upotrebi između pomenutih rešenja neznatne, prelazak sa jednog rešenja na drugo ne iziskuje puno truda i vremena. Mada, u manjim poslovnim okruženjima i manjim firmama, softver otvorenog koda igra sve veću ulogu, jer male firme nemaju razloga da ulože sredstva u skupa komercijalna rešenja ako postoje slobodne alternative koje obavljaju istu funkciju, ukoliko nemaju direktnu korist od dodataka i alata koje nude profesionalna rešenja.

Mogućnost izmene izvornog koda slobodnih alternativa takođe daje slobodu nastavnicima i profesorima da prilagode ova softverska rešenja svojim potrebama i na taj način postignu da njihova izlaganja budu kvalitetnija.

ZAHVALNICA

Autori žele da se zahvale na podršci Ministarstvu prosvete, nauke i tehnološkog razvoja Republike Srbije kroz ugovor br: 451-03-68/2020-14/200108

LITERATURA

[1] Intro to Julia (version 1.0), Jane Herriman

[2] Matlab sa primerima, Dragan Pršić, Kraljevo, 2015, Fakultet za mašinstvo i građevinarstvo u Kraljevu, ISBN 978-86-82631-78-1

[3] "What is Free Software?". (gnu.org).

[4] Richard Stallman on the nature of the Free software movement in 2008 on emacs-devel mailing list.

[5] GNU Octave (version 5.2.0), John W. Eaton, https://octave.org/doc/v5.2.0/

[6] GNU Octave documentation (v.4.2.1), https://octave.org/doc/v4.2.1/The-for-Statement.html

[7] Mathworks Matlab official web site: https://uk.mathworks.com/store/link/products/student/SV?s_tid=ac_buy_sv_but1

IMK-14 – Research & Development in Heavy Machinery 26(2020)3, EN81-89
UDC 621 ISSN 0354-6829
doi: 10.5937/IMK2003081T

Free Software as Alternatives to Commercial in Application of Numerical Methods and Repetitive Procedures

Marko Todorović[1*], Nebojša Bogojević[1]
[1]The Faculty of Mechanical and Civil Engineering in Kraljevo, University of Kragujevac, Kraljevo (Serbia)

Computers with the help of proper software usage enables efficiency in work as well as the conducting of experiments, developments, simulations and data processing which allows specific conclusions to be drawn. The programming languages that have been developed in past decades gave computer users more opportunity to think outside the box. Due to the rapid development and complexity of the programming languages and their syntax, the demand for a much faster and simpler problem-solving tool was created in order to cater situations which require the need to have the tool created in the first place. Today, the most popular and sophisticated programme used in industrial environments is Matlab as it is applicable in various fields of sciences – from numerical calculations to complex simulations. However, Matlab is a commercial programme. In order for it to be used, financial incentives must be prioritised which often is a disadvantage for students and educational institutions as it is costly. In this paper a few free software solutions will be considered as alternatives to commercial software solutions.

Keywords: GNU Octave, Matlab, Numerical methods, Open source software

1. INTRODUCTION

In science, mathematics, physics, chemistry, engineering, and other natural, technical and social sciences, especially in fields filled with mathematical expressions, algorithms, quantitative descriptions of certain natural phenomena and equations, the need for analysing collected data or for conducting long, repetitive and tiring calculations appears. Earlier, those calculations and data analyses were manually done by one or more people at the time. Each iteration could last for months, and there was a high probability of making unintentional errors during calculations which could lead to incorrect conclusions as well as results of conducted analyses.

The described problem is solved using computers – machines that can do huge number of calculations in several iterations for very short time. It's obvious that the use of computers has changed the world and it managed to force out another technological revolution. Today, computers can be found everywhere and, unlike the first computer that could barely fit inside a room, computers can be found in pockets of population majority.

During the past decades of computer development through software and hardware, ways of manipulating these devices were developing as well. Machine language is binary language that consists of two phases: phase *"off",* ie. zero (0), and phase *"on",* ie. one (1). Writing commands, ie. telling the computer processor what to do by using machine language is possible, but is not practical. At least not for humans. Programming languages consisted of, mostly, English words that humans could read and understand, so they could organize the commands have been developed as a solution to this problem. However, as it is described earlier, computer processor unit can only understand commands stated in machine language, therefore the commands written in programming languages are being translated with the use of compiler in the process called compiling. Only compiled commands can be run by the central processor unit.

There are many programming languages and each has its own strengths and weaknesses.

Depending on the programming language used for coding, it is required to pay attention to multiple parameters such as the architecture of central processor unit, platform on which the code will be executed, use of memory, types of data and many other things that make the process of coding slower. In the meantime, the idea of creating a programming environment that could enable the same code to be executed on different architectures, with little to no changes in it, was born. Following that idea, it would be necessary to make a single environment that would run the code, and then adapt the created programming environment to different platforms which would free the programmer from being obligated to know on which platform his program would be run and it would increase the portability. In that case, the programmer would have to write the code once and it would be executed identically within the environment on each platform where the environment had been previously installed, with similar performance, giving the same results. The programmer would know that the code would be executed as intended wherever the environment had been installed, meaning that the programmer would not have to predict on what kind of hardware his code would be executed.

Languages that are being executed in that kind of environments are often being referred as *interpreted programming languages* because the code written using them is being executed inside of the programming environment that interprets each line of code. They are in the family of high level programming languages as well as the compiled languages where the code is being translated directly into machine language that the central processor unit understands. Interpreted languages are higher level programming languages compared to compiled ones.

One of the main properties of interpreted languages are simpler syntax and higher readability of the written

*Corresponding author: Srbija, 36000 Kraljevo, Naselje Moše Pijade 36/76, todorovic.m@mfkv.kg.ac.rs

code which makes coding as well as the code analyses by other programmers easier.

However, the environment within the code of high level programming language is being executed requires resources on its own. This affects the performance of the written programs and it makes them run slower while requiring more resources compared to the lower level (compiled) programming languages. The diagram that illustrates the difference between compiled and interpreted languages is being shown in Figure 1.

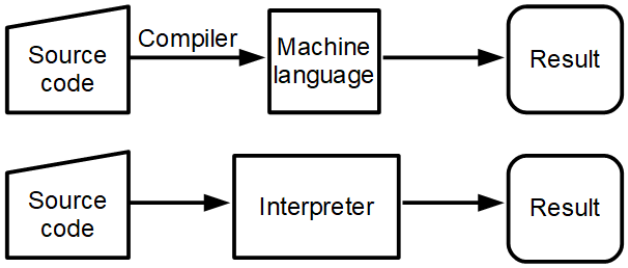


Figure 1: Scheme of code execution if it was written in compiled and interpreted language

When developing a new software solutions, developers face the *"The two language problem"* [1] where they have to choose between simplicity and readability of the code written using higher level programming languages, and complexity, but better use of resources and better performance gained by coding in lower level programming languages. Usually new ideas, approaches to the problem solving, new algorithms and solutions are first being tested by writing the code in higher level programming languages, and only if it is necessary and the results are satisfying, code is being translated into some lower level programming language for gaining extra performance.

The programming environments are usually developed out of the need of some branch of industry, therefore the environments are usually adjusted to meet the specific needs of different industries, such as conducting of long numerical calculations or advanced simulations of fluid movement.

Specialized programming environments are applied in wide variety of science fields such as mathematics, physics, engineering, statistics, biology, etc… Even though high level programming languages make coding faster, they are not suitable for doing general purpose programming. Though, there are tools that can automatically translate code from high to lower level programming language, i.e. from Python to C or C++.

Simpler syntax compared to other languages, compatibility across different platforms and considering that it is not needed to take the types of data or memory management into consideration, high level programming languages are appealing for use in educational purposes. Without the need for digging into the science of coding, programming and specifics of given architecture, with the use of a few simple commands, high level programming languages allow for multiple solutions of given problems to be tried out or data to be processed. This can allow students to focus on solving the given problem rather than wasting time on things like coding, platform specifics, etc.

2. COMMERCIAL SOFTWARE SUITS

2.1. MathWorks Matlab

Simpler and faster problem solving are good enough reasons for justifying the use of interpreted programming languages in industry. The company MathWorks developed one of the most famous and complete solutions – software suite and programming language Matlab which belongs to the group of interpreted high level programming languages, like Python or Julia, where the coding in it is much simpler compared to compiled ones like PASCAL or C for example.

Simple syntax that most of the people with any coding experience are already familiar with, and for those without any coding experience is simple enough, makes training of new users short is something that gave Matlab advantage over other solutions and made it spread fast through branches of industry. Matlab programming language is specialised for working with one dimensional (vectors) and multidimensional (matrices) arrays, and there are numerous tool packs and plugins that extend its capabilities and make working with it as easy as possible. Plugins and tool packs are specialized for different branches of science and industry.

Graphical user interface, as it is shown in Figure 2, consists of the title bar (1), tool bar (2), left side panel (3) where the content of work folder is being shown, command window in the middle (or console) where the commands can be directly typed and executed (4), while in the right side panel (5) the history commands is being placed, as well as the list of variables and their values.

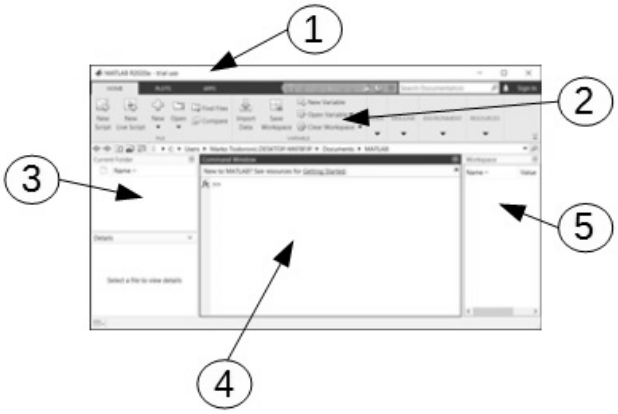


Figure 2: Graphical user interface of Matlab: 1 – title bar, 2 – tool bar, 3 – left side panel, 4 – command window, 5 – right side panel

Commands written in Matlab can be executed in two ways. By typing the command inside of a console (command window – 4 in Figure 2) and pressing the ENTER button on the keyboard user can execute one line of code at the time. This way of use makes Matlab seem like an advanced calculator. The other way, like when using any other programming language – is to write a bunch of commands within a text file – a script, that are being executed line by line, automatically, in the written order [2]. In the scripts, the use of loops and conditional

statments becomes available which extends the applicability of Matlab as programming language.

The package for easy and simple graphical representation of data using two dimensional or three dimensional graphs and charts of different shapes and forms. For more advanced uses, there are even tools for creating graphical user interface for the programs written in Matlab that makes the use of the programs for those who did not write it much easier.

2.2. Wolfram Mathematica

Wolfram Mathematica is another commercial software suite that can perform vast amount of different mathematical operations.

Unlike MathWorks Matlab, Wolfram Mathematica is primary used for doing calculations with symbolic operators – variables that do not have numerical values, but are represented with letters – symbols. Matlab also has very good libraries that enable the use of symbolic operators, but for now, Mathematica is more sophisticated in that field. By using symbolic operators, the output that Mathematica can give does not have to be a numerical value of some variable, but it can also be an expression consisted of pronumerals.

Syntax of the Wolfram Mathematica programming language is incompatible with MathWorks Matlab and it can only be run within the Wolfram Mathematica software suite.

3. FREE, OPEN SOURCE ALTERNATIVES

By the end of the 20th century, when the computers entered people's homes and general population started getting accustomed to their increasing presence, the idea that software should be free, in terms of writing, distribution and use, was born. Followers of this idea believe that the users of the software should be allowed to freely use, share, and adapt the software to their needs, and to also share the adapted form of the software by the same conditions. According to that, the source code should be acessable to all interested parties who want to know how the software was written and understand the way it works, or if capable, contribute by changing pieces of code, fixing errors, and/or adapting it for some other purpose without breaking the law by comitting copyright infregment. That idea led Richard Stallman to found The GNU (acronym for GNU is not Unix) foundation in 1983. that made the open source idea legal by licencing software under the GNU general public licences. Vast amount of software is created under the GNU general public licence, and among them, the most famous is the operating system GNU Linux [3,4].

Every piece of software under the GNU general public licence is free for use, modification and distribution, and as such it is ideal for educational purposes. By publishing the source code together with the software, users get a chance to see how the software was made, how it was coded, how certain problems had been solved, they can fix errors in code, bugs that appear while running the software, improve the software capabilities by writing new lines of code, or adapting it to the personal needs. The project founder, Richard Stallman claims that the main benefit of this way of software distribution is that young software engineers, programmers and coders have a chance to observe the code of small and big programs. In order for them to become good programmers and software engineers, they need to read a lot of code, to see and analyse the code of big programs and big systems, and their first step as independent programmers should be changing small pieces of big programs. [4]

Some of the programming environments used for applying numerical methods that can serve as alternatives to commercial ones are published under the GNU general public licence. Two such software suits shall be represented in this paper: GNU Octave and SciLab.

3.1. GNU Octave

GNU Octave is high level programming language focused on performing numerical calculations, and it is usually used for solving linear and non-linear equations, numerical calculations and statistical analyses [5].

The software suite is consisted of GNU Octave Client in which the commands can be typed inside of the console and, by pressing the ENTER button on keyboard, directly executed as it is shown in Figure 3.

```
GNU Octave, version 5.2.0
Copyright (C) 2020 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-w64-mingw32".

Additional information about Octave is available at https://www.octave.or

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1> disp("Hello world")
Hello world
octave:2>
```

Figure 3: Displaying the "Hello world" message using GNU Octave client

There is also GNU Octave GUI as part of the suite that gives the GNU Octave Client a graphical user interface that makes it more user friendly and expands capabilities of the environment. The graphical user interface, as it is shown in Figure 4, consists of the title bar (1), menu bar (2), tool bar (3), left side bar that contains file browser that shows content of the current directory (4), workspace that shows the list of variables as well as their values (5), and history of used commands (6). Central area of the graphical user interface is reserved for the command line (7) that can be used as GNU Octave client, documentation (8) that contains explanations of every command that can be used, text editor (9) used for writing series of commands organised in a single text document that can be saved and executed multiple times – scripts, as well as the Variable Editor (10) which shows the list of used variables and their values in spreadsheet format from where they can be easily changed by double clicking on the cells that contain values that needs to be changed.

The spreadsheet can be exported to .txt format, and the changed values of variables can be reused within the programming environment. Switching from one to another tool within the central area of the graphical user interface

can be performed by using tabs placed at the bottom of the window.

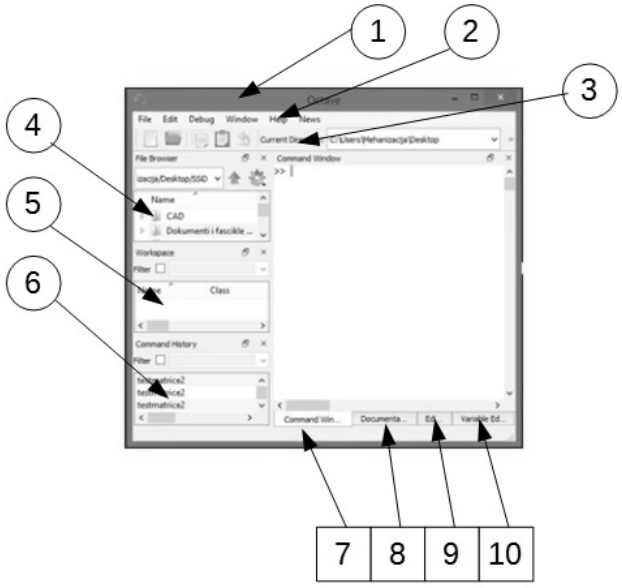


Figure 4: GNU Octave – graphical user interface: 1 – title bar, 2 – menu bar, 3 – tool bar, 4 – file browser, 5 - workspace, 6 – history, 7 – command window, 8 - documentation, 9 – Editor, 10 – Variable Editor

The syntax of the GNU Octave programming language is similar and nearly identical to the one used in MathWorks Matlab. Scripts use the same file extension ".m". That implies that code written in GNU Octave programming environment can be executed in MathWorks Matlab, and the other way around. However, nearly identical syntax does not imply that it is completely the same and compatible, and there are slight differences. For example, in Figure 5 it is shown how "for" loop with counter is written using Mathworks Matlab programming language, while the same thing, but written using GNU Octave programming language is shown in Figure 6.

```
1 -    clc;clear;
2
3 -    s = 10;
4 -    H = zeros(s);
5
6 -    for c = 1:s
7 -        for r = 1:s
8 -            H(r,c) = 1/(r+c-1);
9 -        end
10 -   end
11
12 -   disp(H)
```

Figure 5: Example of "for" loop with counter written in Mathworks Matlab programming language [6]

```
1   clc; clear;
2
3   s=10
4   H=zeros(s);
5
6   for c=1:s
7     for r=1:s
8       H(r,c)=1/(r+c-1);
9     endfor
10  endfor
11
12  disp(H)
```

Figure 6: Example of "for" loop with counter written in GNU Octave programming language

As presented on Figure 5 and Figure 6, MathWorks Matlab contains reserved word "end" for ending the loop, while GNU octave contains the reserved word "endfor" that performs the same task. Even though the difference seems slight, it is enough to stop the code written in one language to be successfully executed in the environment of the other out of box. In order for that to be accomplished, the code needs to be adapted to the environment where it is going to be executed.

Just like the Matlab, GNU Octave has built in, easy to use tool for displaying visual representation of data in form of two dimensional and three dimensional graphs , as it is illustrated in Figure 7, and differently shaped charts, as it is represented in Figure 8, even though it is not as rich with features as the one that can be found in the Mathworks Matlab environment.

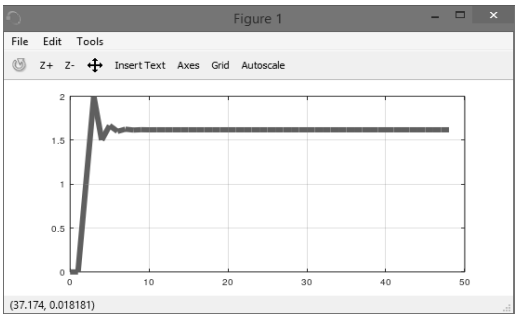


Figure 7: Graphical representation of the golden ratio of Fibonacci sequence in GNU Octave programming environment

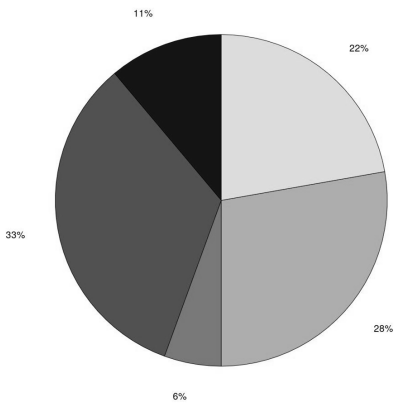


Figure 8: Pie chart created in GNU Octave programming environment

The title of the GNU Octave programming language contains the GNU acronym which means that the whole project is under the GNU general public licence, with all its good and bad properties of open source software. Freedom makes GNU Octave affordable for use in personal, educational and commercial purposes. Programming language and programming environment is a result of joint work of volunteers and it is not backed by any big corporation that would own the software and finance its development and collect all the commercial gain. Therefore, development of GNU Octave is somewhat slower and more unstable compared to the commercial solutions. Tools and add-ins that expend the abilities of the software suite and programming language are written by the volunteers as well, unlike those that can be found in commercial solutions where the add-ins and tools are being written by paid, professional programmers. Even though it seems like it is a weakness, it is actually one of the strengths of GNU Octave programming language, because it is powered and developed by community. If the group of volunteers gives up on the project and stops developing it, another group of volunteers can take it over and continue its development. This makes the open source projects less sensitive to changes on commercial market unlike the commercial software where if the company that is developing it stops doing so, the developed software stops being developed for good which did happened in the past.

The example for that are numerous variations of the Pascal programming language that had been developed by different commercial companies, and only two variations are being developed to this day, and one of which, FreePascal is open source. The other one is TurboPascal under the Delphi software suite owned by the company Embracadero.

3.2. SciLab

SciLab is another software suite for performing numerical calculations. Like GNU Octave, SciLab is open source which means it is free for use and distribution. However, from the year of 2017, ESI Group is backing the SciLab project up as well as the community of volunteers.

Syntax of the SciLab programming language is also similar to MathWorks Matlab, but there are also some differences. For example, reserved word for number pi in Matlab and GNU Octave is “pi”, while SciLab uses the word “%pi”. Unlike GNU Octave programming environment, SciLab has independent file format for text files in which written scripts are being saved, so scripts written in SciLab cannot be executed in MathWorks Matlab or GNU Octave programming environment out of box.

SciLab software suite is consisted of console, shown in Figure 9, in which the commands can be directly typed and executed, and graphical user interface, which is shown in Figure 10, with design that resembles older versions of MathWorks Matlab.

```
Scilab 6.0.2 (Feb 14 2019, 10:31:37)

Start CEMEF-UTN Analysis for Truss structures
        Load macros


--> A=round(rand(10,10)*100)
 A  =

   21.   56.   31.   50.   28.   41.   39.   54.   59.   65.
   76.   66.   93.   44.   13.   88.   92.   12.   48.   99.
   0.    73.   21.   27.   78.   11.   95.   23.   22.   5.
   33.   20.   31.   63.   21.   20.   34.   63.   84.   75.
   67.   54.   36.   41.   11.   56.   38.   76.   12.   41.
   63.   23.   29.   92.   69.   59.   73.   5.    29.   61.
   85.   23.   57.   4.    15.   69.   26.   67.   86.   85.
   69.   22.   48.   48.   70.   89.   50.   20.   85.   6.
   88.   88.   33.   26.   84.   50.   26.   39.   53.   83.
   7.    65.   59.   41.   41.   35.   53.   83.   99.   93.


-->
```

Figure 9: SciLab console; displaying 10x10 matrix

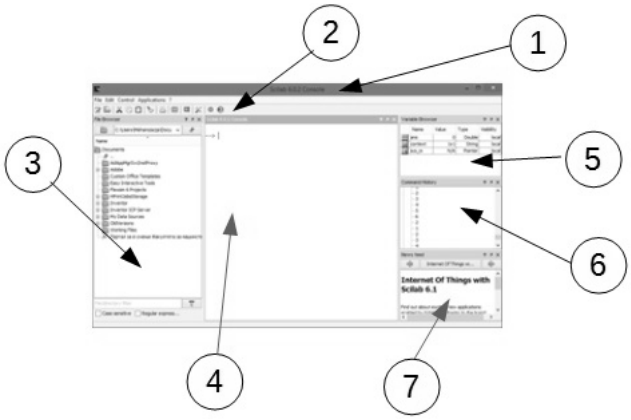


Figure 10: SciLab graphical user interface: 1 – title bar, 2 – tool bar, 3 – left side bar with displayed content of current work folder, 4 – command line, 5 – list of used variables, 6 – history of used commands, 7 – newsletter

Tool for graphical data representation that has similar capabilities as the one included in GNU Octave, that is shown in Figure 11, is built into the SciLab programming environment even though commands for controlling it are different. For example, grid made out of pale, equally distributed horizontal and vertical lines in MathWorks Matlab and GNU Octave can be created within the graph by using the command “grid on”, for doing the same in SciLab programming environment the command “xgrid(2)” is being used, where the number 2 in the command describes the colour of the grid lines.

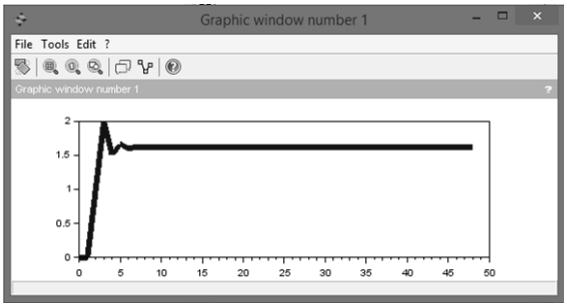


Figure 11: Graphical representation of the golden ratio of Fibonacci sequence in SciLab programming environment

Tools and plugins that expand the functionality of the programming language and programming environment are organised within ATOMS package manager, which is represented in Figure 12. They are result of joint effort between volunteers and user community and they are being published under the same licence as the SciLab, therefore all the freedoms of open source software apply to them as well. Within SciLab programming environment there is a module for performing simulations that represents alternative to MathWorks Simulink, and it is called Xcos (figure 13). Graphical user interface of Xcos is consisted of two separate windows where one window contains all the available tools for creating simulations sorted in bigger groups, and the second window that represents the workspace where simulation is being created out of the tools from the first window.

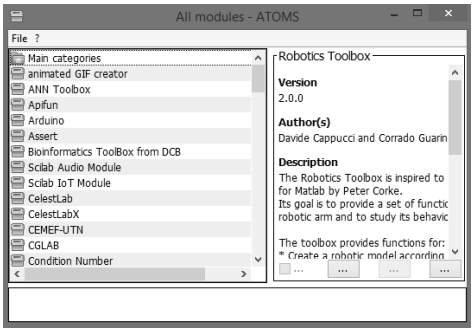


Figure 12: ATOMS package manager

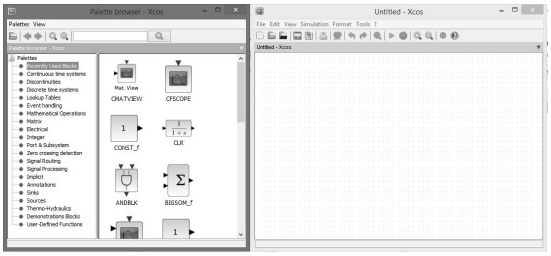


Figure 13: Xcos: left – window with tools, right - workspace

Support for working with pronumerals among the open source solutions is very low or it does not exist at all, therefore if working with pronumerals is primary focus of work, the commercial alternatives should be software solutions of choice.

4. SOLVING PROBLEMS USING GNU OCTAVE AND SCILAB SOFTWARE SUITS

The main goal of high level programming languages is to shorten the path that leads to the best and most efficient solution of some problem, or help picking the best solution out of many, and to increase the overall productivity of problem solver which is, compared to the lower level programming languages, being accomplished by making syntax of the language simpler and more compact allowing the programmer to write less amount of code in order to perform the same tasks. Such programming languages are easier for beginners to grasp, allowing them to focus on the logic of programming rather than syntax of a language.

These software suits are not only applicable for teaching programming and coding, but they can be also used for better explaining and visual representation of, for example, mathematical functions because students can see how changes in expressions influence the shape of graphs. Another example is using the tool for plotting graphs for explaining analytical geometry, or shape of waves in physics, which is illustrated in Figure 14.

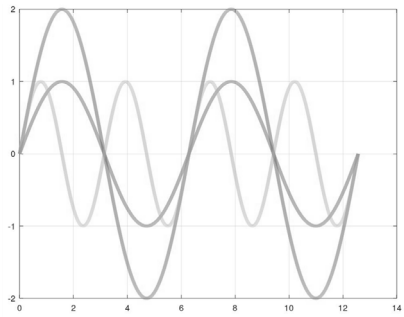


Figure 14: Graphical representation of sin(x),2sin(x) and sin(2x) functions created with GNU Octave

4.1. Example

Advantages of working with high level programming languages, such as SciLab and GNU Octave, are most noticeable while working with multidimensional arrays or matrices, which will be described in following example.

It is required to write a program that defines members of two dimensinal array of unknown size, where each member of the array takes the value of integer between 0 and 1000.

There are many solutions to this problem and it is possible to solve it in every programming language. One of the solutions, using FreePascal, a compiled programming language, is represented on figure 15.

```
 1  program Matrica_nasumicnih_brojeva;
    type
      matrica = array of array of integer;
    var
 5    n,m,i,j:integer;
      A:matrica;

    begin
      write('Broj redova n: ');
10    readln(n);
      write('Broj kolona m: ');
      readln(m);
        setlength(A,m,n);
      for i:=0 to m-1 do begin
15      for j:=0 to n-1 do begin
          A[i][j]:=round(random*1000);
        end;
      end;
      writeln('Matrica A: ');
20    for i:=0 to m-1 do begin
      for j:=0 to n-1 do begin

        write(A[i][j], '   ');
        end;
25    writeln('');
      end;
      readln();
28  end.
```

Figure 15: FreePascal code for defying and displaying dinamic twodimensional array

The conlusion that can be drawn out of observing Figure 15 is that it takes 28 lines of code to solve the described problem. When this code is compiled and run, it gives the output represented in Figure 16.

```
Broj redova n: 10
Broj kolona m: 10
Matrica A:
549   593   715   844   603   858   545   847   424   624
646   384   438   298   892   57   964   273   383   478
792   812   529   480   568   393   926   836   71   337
87   648   20   368   833   957   778   140   870   870
979   474   799   801   461   520   781   679   118   721
640   582   143   537   945   759   522   106   415   474
265   186   774   737   456   217   568   135   19   324
618   150   612   222   617   386   944   903   682   450
360   613   437   902   698   99   60   970   667   653
671   171   210   358   129   751   315   608   364   325
```

Figure 16: Compiled and executed program written in FreePascal programming language

The code shown in Figure 15 is consisted of multiple segments. First segment, before the keyword "Begin", deffines the title of the program and variables that are going to be used, as well as their types. Commands that are being executed are located between two reserved keywords: Begin – which marks the beginning of the code that has to be executed, and End – which marks the end of the code. The code that is placed between those two keywords contains two sets of two "for" loops. First set of two loops defines the members of two dimensional array while the second set of two loops display the array on the screen, as it is shown in Figure 16.

The obvious advantage of GNU Octave programming language over FreePascal can be noticed by observing the solution of the same task, written in just four lines of code, represented in Figure 17.

```
1  clc; clear;
2
3  m=input('Broj redova matrice m: ')
4  n=input('Broj kolona matrice n: ')
5  A=round(rand(m,n)*1000)
```

Figure 17: Solution written in GNU Octave programming language

The first line of the code displayed in Figure 17 is needed to clear the screen and the memory of the programming environment to prevent residues of previously executed code from some other script from interfering with the results, while defying and displaying of the two dimensional array is being done with the command that is placed in fifth line of the code. The result of running this scrip is being displayed in Figure 18.

After executing the compiled program written in FreePascal, it is not possible to easily access the defined variables and it is not possible to do additional tasks with them, unless the source code of the program is changed and executed again. If the script is run in GNU Octave, after executing the last line of code within the script, it is possible to easily access all variables that had been defined while running the script, within the command line, without changing the source code of the script and running it again.

```
Broj redova matrice m: 10
m =  10
Broj kolona matrice n: 10
n =  10
A =

   575   118   750   280   672   262   730    73   538
   417   445   334    45   986   183   825   402   136
   138    77   832   434   338    58   372   743   930
   846   632   112   520   104   783   216   527   211
   696   483   751    36   117   639   722   609   448
   787   561   182   176   234   444   826    63   549
   291   228   423   193   110   967   379   426   760
   585   157   874   861   166   996   117   384   211
   653   424   637   907   454   705   408   686   844
   193   298   183   516   864   122   705   734   959
```

Figure 18: Two dimensional array displayed in GNU Octave programming language

For example, after running the code from figure 17, it is possible to calculate determinant of the given matrices or their inverse matrices simply by typing the right commands in the command line, as it is shown in Figure 19.

```
>> det(A)
ans =    4.2775e+26
>> inv(A)
ans =

 Columns 1 through 6:

   0.001267460   0.001107409   0.015382939   0.013491466  -0.011966239   0.006872015
   0.003218100  -0.006655033  -0.051061015  -0.036960196   0.040101553  -0.026448631
   0.002271556  -0.003138084  -0.021520127  -0.015946273   0.017618313  -0.011940199
  -0.000387412  -0.001714031  -0.007255390  -0.005729931   0.005487590  -0.002512002
   0.001325292   0.000750979  -0.001767795  -0.000461202   0.000324521  -0.002123673
  -0.001582406   0.002443537   0.011053389   0.007617832  -0.009389396   0.006141345
  -0.001914811   0.000292451   0.006037775   0.003223588  -0.003972915   0.005329479
  -0.002040291   0.002675955   0.019253105   0.014222418  -0.014040219   0.009237252
   0.002853369  -0.003461647  -0.016185376  -0.010711530   0.012516597  -0.009218186
  -0.003414303   0.005182193   0.026044511   0.017091652  -0.020737761   0.014237262

 Columns 7 through 10:

   0.002431502   0.007633965  -0.024805008   0.001125676
  -0.009031795  -0.029653055   0.081159972  -0.004550744
  -0.004002827  -0.012334902   0.034288825  -0.002137358
  -0.002433946  -0.001981270   0.010104390   0.001103706
   0.000468244  -0.002074924   0.003464170  -0.000933350
   0.003211004   0.006692372  -0.017244681   0.000176128
   0.000105810   0.006199469  -0.012605434   0.002919241
   0.003006565   0.010923009  -0.030335115   0.002147329
  -0.002126016  -0.010398443   0.025283007  -0.000862839
   0.004791175   0.013902017  -0.037567698  -0.000092636
```

Figure 19: Calculating matrix determinant and inverse matrix

Exactly the same code from Figure 17, without any changes in it, would perform exactly the same and it would give the same output if run in MathWorks Matlab programming environment, as well as if it would be run in SciLab, on any platform where those environments can be installed (Windows, Linux, MacOSX).

5. CONCLUSION

Freedoms that open source alternatives, like SciLab and GNU Octave, allow are one of the biggest advantages over the commercial software especially in educational purposes. There is no need for purchasing licences for open source software since their licence allows free educational and commercial use. Instead of buying the software, funds can be directed to obtaining new hardware, or some other tools that would increase the quality of teaching within educational institution. Also, students are free to download and use the software at home free of charge which makes studying more affordable. Considering that the most affordable educational licence for MathWorks Matlab costs 69€ [7], and the fact that programming languages like Matlab, SciLab or GNU

Octave are being thought through only one semester or one school year where students usually do not even use more advanced tools that come within commercial software solutions, it becomes questionable if the cost is justified.

The difference between open source and commercial alternatives are slight, at least in entry level that is being thought in high schools and faculties. A student that can write a script in GNU Octave or SciLab should not experience bigger issues when writing in MathWorks Matlab. Commercial solutions are more complete and stable in use, and in real industrial environment there is higher probability that students would, in future, work with them, but mentioned slight difference makes the transition from one solution to another fast and easy. Though, especially in smaller businesses the free software plays bigger and bigger role, since these busnisses have no reason to invest into expensive professional software solutions if the free software alternatives do the same job just as well, unless they directly benefit from add-ins and plugins that professional tools offer.

The freedom to change and adapt the open source software also gives a chance to professors and teachers to adapt the software to their needs so it can become more suited for presenting and teaching which can improve the quality of education.

ACKNOWLEDGMENT

The authors would like to acknowledge support of the Serbian Ministry of education, science and technology development through contract no: 451-03-68/2020-14/200108

REFERENCES

[1] Intro to Julia (version 1.0), Jane Herriman

[2] Matlab sa primerima, Dragan Pršić, Kraljevo, 2015, Fakultet za mašinstvo i građevinarstvo u Kraljevu, ISBN 978-86-82631-78-1

[3] "What is Free Software?". (gnu.org).

[4] Richard Stallman on the nature of the Free software movement in 2008 on emacs-devel mailing list.

[5] GNU Octave (version 5.2.0), John W. Eaton, https://octave.org/doc/v5.2.0/

[6] GNU Octave documentation (v.4.2.1), https://octave.org/doc/v4.2.1/The-for-Statement.html

[7] Mathworks Matlab official web site: https://uk.mathworks.com/store/link/products/student/SV?s_tid=ac_buy_sv_but1