Математички факултет, Београд

Преглед програмског пакета - QЛаб

кроз примере

QЛаб тим

Садржај:

QЛаб - идеја и циљеви

Опште је познат значај софтвера отвореног кода у модерној употреби рачунара. Снажна упоришта опен сорс иницијативе су разни универзитети широм света. Готово сваки познатији факултет на коме се изучава рачунарство развија неки софтвер отвореног кода по коме је препознатљив. QЛаб је амбициозни пројекат Математичког факулета да се развије софтер отвореног кода који би заменио комерцијални МАТЛАБ- један од најкоришћенијих програма који се примењује за разна математичка израчунавања. Идеја је да се кроз QЛаб пројекат искористе сви потенцијали Математичког факултета, како студената рачунарских смерова и модула, тако студената теоријске и примењене математике. Сви заинтересовани и мотивисани студенти имали су прилику да се придруже QЛаб тиму и учествују у развоју овог софтвера отвореног кода.

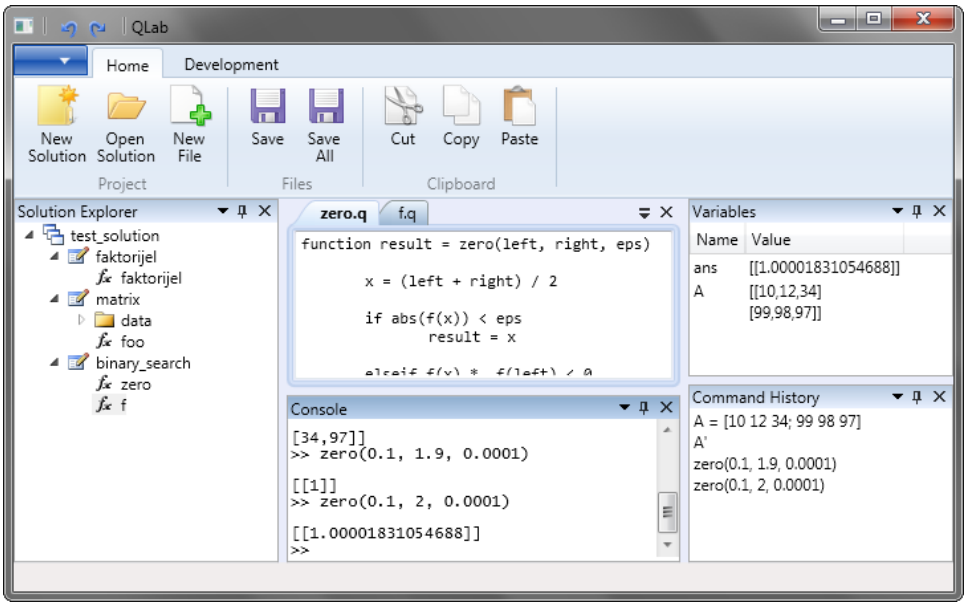
МАТЛАБ је врло често коришћен софтвер на факултетима у Србији. Користи се на свим природно-математичким и техничким факулетима као део наставе, где се многи прорачуни извршавају у МАТЛАБ-у. Поред факултета природно-математичке и техничке групације, МАТЛАБ је део наставе и на другим факултетима. Тако се, на пример, МАТЛАБ изучава и користи на факултету за туристички и хотелијерски менаџмент. Опште је познато да је МАТЛАБ врло моћан софтвер који, поред разних алгебарских, аналитичких, графичких могућности, у својим новијим верзијама (R2010) укључују алате који се примењују у физичком моделовању, финансијском рачунарству, дигиталној обради сигнала, симулацијама...Такође је опште познато да је МАТЛАБ врло скуп софтверски пакет, тако да многи факултети не могу да обезбеде одговарајући број лиценци и учине га доступним својим студентима и наставницима.

Планирано је да се прва верзија QЛаб-а користи у настави на Математичком факултету, као замена за МАТЛАБ (на практичним вежбама, колоквијумима, тестовима, испитима...). Кроз семинарске радове (обавезне или изборне) студенти би даље надограђивали QЛаб пакет и унапређивали га. Након тестирања на Математичком факултету, QЛаб се може понудити колегама са других факулета и Универзитета у Србији који би га допунили алатима из области које се изучавају на њиховим факултетима. На тај начин би се попунио скуп могућности QЛаб-а и на друге области, осим математике. Пошто је пројекат отвореног кода на њему ће моћи да учествују и академске установе које нису у Србији, тако да QЛаб може постати први избор математичког софтвера на факултетима широм света.

Циљ QЛаб-а јесте да благовремено испуни списак свих могућности МАТЛАБ-а и можда чак и да прошири, али остајући опен сорс, тј. не наплаћујући коришћење овог софтвера и омогућавајући свима да га користе, али и унапређују и објављују своја унапређења. Тако би се као и остали алати отвореног кода брзо ширио и растао и заузео значајно место међу расположивим математичким софтверима.

Графички кориснички интерфејс

При покретању QЛаб-а, појављује се кориснички интерфејс који садржи алате за манипулисањем пројектима, променљивама, претходно коришћених команди. Подразумевано, распоред компоненти у апликацији је такав да је главни мени на врху, solution explorer са леве стране, датотеке са кодом се отварају у средини, конзола је у дну, а списак променљивих и историја команди десно. Наравно, могуће је преместити, затворити,сакрити или променити величину компоненте по жељи, осим главног менија који је фиксиран на врху (њега је могуће смањити тако да се види само први ред са називима језичака).

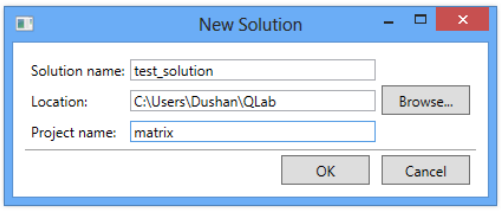


Слика 1- Изглед графичког корисничког интерфејса

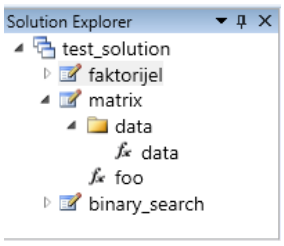
Креирање и учитавање пројеката и датотека

Рад у QЛаб-у заснива се на пројектима који су део једне целине - решења. Када корисник жели да напише неки програм потребно је прво да креирати ново решење (New solution) са пројектом или отворити постојеће а затим креира датотеку за писање свог програма.

Да би корисник дефинисао решење мора да унесе име пројекта и директоријум у коме се тај пројекат налази. Дефинисање директоријума је неопходно да би *QЛаб* знао где да тражи датотеке укључене у пројекат. Датотеке не морају да буду у истом директоријуму где и пројекат, тј. могу да се налазе произвољно дубоко у неком од поддиректоријума.



Слика 2 – Креирање новог пројекта



Слика 3 – Хијерархијски приказ пројекта

Компонента задужена за приказивање и управљање (промену имена, брисање, копирање) тренутно креираним пројекатима и датотекама је solution explorer.

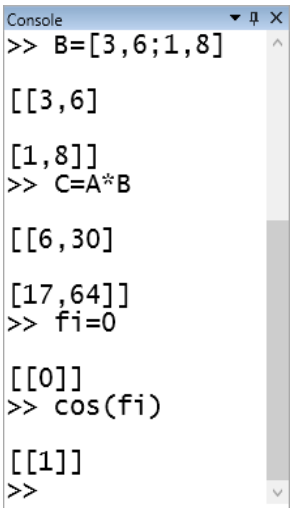
За управљање се користити мени који се отвора кликом на десни тасте миша на одређену датотеку. Такође су подржане пречице преко тастатуре.

Конзола

Конзола је начешће коришћена компонента корисничког интерфејса која се користи за дефинисање и иницијалзицају променљивих, за позивање уграђених функција као и функција које корисник сам напише.

Свака актуелна наредба почиње са знаком за унос „>>“, док корисник може да мења само текст који је после знака. Тренутна наредба се извршава притиском на тастер а нако њеног извршавања, испсује се резултат после којег се поново појави знак за унос.

Корисник може притиском на тастере горе (↑) и доле (↓) да пролази кроз историју унетих команди. Притиском на тастер „горе“ аутоматски се уписује последња унета команда. Поновним притиском на тастер „горе“ уписује претпоследња команда, итд., све док се не дође до прве команде. Слично за тастер „доле“.

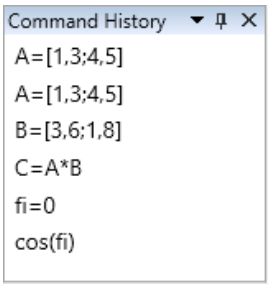


Слика 4 – Рад у конзоли

Историја команди

Поред тога што se кроз историју команд пролази стрелицама „горе“ и „доле“ на тастатури, QЛАБ садржи и посебну компоненту који приказује све извршене команде. На слици је је приказано како изгледа прозор са историјом команди.

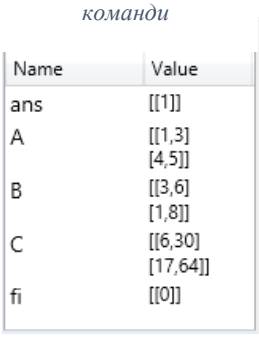
Команде се могу брисати, копирати и извршавати а такође је предвиђено и њихово уписивање у датотеку.



Слика 5 – Списак претходно коришћених команди

Листа променљивих

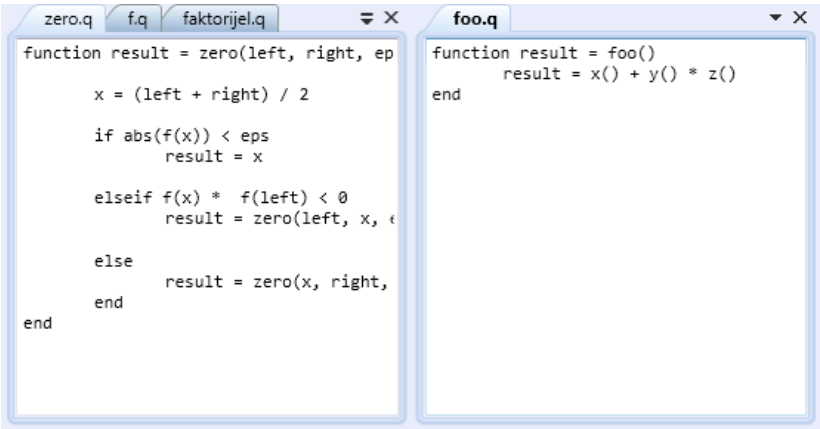
Све декларисане променљиве је неопходно видети на неки начин. За то ће да се стара компонента са списком свих декларисаних променљивих. У плану је могућност увоза података измене и дефинисања променљивих помоћу ове компоненте. Иначе унос променљивих је за сада могуће извршити на два начина. Први је декларацијом променљиве у конзоли а други начин је извршањем кода који у себи садржи декларацију променљиве.



Слика 6 – Списак променљивих које су креиране од стране корисника

Едитор

Ова компонента приказује отворене датотеке у виду језичака. Сваки језичак у себи садржи компоненту за уређивање кода. На слици се виде четири отворена документа распоређених тако да су три са лево а један десно. Распоред је могуће произвољно мењати.



Слика 7 – Едитор

Изрази у QЛаб-у

Као и већина програмских језика и QЛаб подржава математичке изразе, али за разлику од већине тих програмских језика ови изрази подразумевају и матрице а не само бројеве.

Градивни елементи израза су:

- променљиве
- бројеви
- оператори
- функције

Променљиве

У QЛаб-у није потребно вршити декларацију типа или димензију променљиве. (није строго типизиран). Када наиђе на доделу вредности новој променљивој QЛаб аутоматски креира ту променљиву и резервише одређени меморијски простор за њу. Уколико већ постоји тада се мења њен садржај и уколико је потребно алоцира нови меморијски простор.

Пример 1: *Додела вредности променљивој.*
>> X=5
[[5]]
>> X=[1,2,3]
[[1,2,3]]
Прво је креирана матрица димензије 1x1 и смештена вредност 5, затим је променљива X реалоцирана, на димензију 1x3, и смештене су вредности редмо 1,2,3.

Назив променљиве мора да почиње словом иза чега следи број,слово или доња црта. За приказ садржаја односно вредности довољно је навести име променљиве.

Бројеви

QЛаб за запис бројева користи конвенционалну децималну нотацију са опционим појављивањем знака и децималне тачке. Такође је подржана и научна нотација коришћењем ознака „E-“, „E+“, „e-“ или „e+“. за експонент.

Примери валидно записаних бројева:
5 -75 0.0008 3.14159265 1.24320e-20 4.0424e23

Оператори

У изградњи израза се могу користити следећи оператори:

Симбол	*Опис оператора*
+	Сабирање
-	Одузимање
*	Множење
/	Дељење
.*	Скаларно множење
./	Скаларно дељење
'	Транспоновање

Симбол	*Опис оператора*
>=	Веће или једнако
>	Веће
<=	Мање или једнако
<	Мање
==	Једнакост
~=	Неједнакост
&	Коњукција
|	Дисјункција

Функције

QЛаб поседује велик број имплементираних математичких функција као што су:

- abs
- acos
- asin
- atan
- acosh
- asinh
- atanh
- cos
- sin
- tan
- cosh
- sinh
- tanh
- exp
- log
- log

Семантика сваке функција је јасна из њеног назива.

Пример 2: *Израчунавање израза.*

```
>> fi = (1+sqrt(5))/2
[[1.61803398874989]]
>> X=[-1,-3,-4]
[[-1,-3,-4]]
>> Y=[2,3,1]
[[2,3,1]]
>> abs(X).*Y
[[2,9,4]]
>> i=sqrt(-1)
[[0 + -1i]]
>> i*i
[[-1]]
>> exp(log(5))
[[5]]
```

Матрице

У QЛабu матрица је низ бројева. Специјално значење је понекад придружено матрици димензије 1x1 која има улогу склара а такође и матрицама које имају само једну врсту или колону којима се представља вектор, тако да је најбоље о свему размишљати као о матрици. Док други програмски језици омогућавају рад са бројевима, QЛаб омогућава брзи и лак рад са матрицама на природан начин.

Креирање матрица

Обзиром да посматрамо све као матрицу, потребно је савладати начин руковања матрицама. Матрице се могу унети на више различитих начина:

- Експлицитном листом елемената
- Генерисање помоћу уграђених функција.
- Генерисање помоћу функција написаних од стране корисника.

Експлицитно уношење матрица

При експлицитном уношењу матрица потребно је испоштовати 3 правила:

- Елементе матрице је потребно одвајати белином или зарезом
- Крај врсте је потребно означити ;
- Целу листу елемената је потребно уметнути између угластих заграда.

Пример 3: *Као пример једне матрице посматрaћемо Дуреров магични квадрат.*
>>A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
[[16,3,2,13]
[5,10,11,8]
[9,6,7,12]
[4,15,14,1]]

Сада је променљиву A забележена, видљива је у компоненти за приказ променљивих и могуће је референсати на њу уз помоћ њеног имена – A.

На матрице је могуће примењивати велики број уграђених функција. Неке од фунцкија биће приказана на матрици A и управо ће открити зашто је ова матрица „магична“.

Познато је да су својства магичног квадарата везана за различите начине сумирања његових елемената, наиме збир по били којој врсти или колони или једној од две главне дијагонале је исти. Ово свосјтво матрице A можемо проверити уз помоћ QЛаба

Пример 4: *Израчунавање суме елемената по свакој колони.*
>>sum(A)
[[34,34,34,34]]

QЛаб и уграђене функције најчешће преферирају рад са колонама тако да је израчунавање суме елемената по свакој врсти могуће применом фунцкије за сумирање на транспоновану матрицу матрице A. Такође уколико се експлицитно не наведе променљива у којој се чува резултат, он ће бити сачуван у променљивој ans (скраћеница од answer).

Пример 5: *Транспоновањ матрице, израчунавање суме свка врсте.*
Транспоновање матрице се обавља операторм ' (сваки вектор врсте претвара у вектор колоне)

```
>>A'
[[16,5,9,4]
[3,10,6,15]
[2,11,7,14]
[13,8,12,1]]
>> sum(A')'
[[34]
[34]
[34]
[34]]
```

Остаје да проверимо суму елемната по дијагонали а то је могуће урадити уз помоћ уграђене функције diag која као резултат враћа елементе са главне дијагонале у једном вектору врсте.

Пример 6: *Рачунање суме елемената на главној дијагонали.*

```
>> sum(diag(A))
[[34]]
```

Елементима матрице је могуће приступити навођењем његове позиције у матрици уз помоћ два индекса или једног.Ако се наводе два индекса први означава врсту а други колону, док при навођењу једног индекса он означава редни број елемента бројећи по колонама. Тако елементу који се налази у пресеку 4 врсте и 2 колоне матрице A из претходних примера приступа се са A(4,2) или уз помоћ једног индекса са A(8).

Пример 7: *Рачунање суме елемената који се налазе у „угловима“ матрице.*

```
>> A(1,1)+A(1,4)+A(4,1)+A(4,4)
[[34]]
```

Покушај приступа елементу задавањем позиције која прекорачује димензију матрице резултује грешком док покушај уписивања на исту позицију резултује успешно, матрица се проширује до наведене димензије сви међу елементи се постављају на 0.

Пример 8: *Прављење копије матрице A и уписивање нове вредности ван димензије матрице.*

```
>> X=A
[[16,3,2,13]
[5,10,11,8]
[9,6,7,12]
[4,15,14,1]]
>> X(7,6)=34
[[16,3,2,13,0,0]
[5,10,11,8,0,0]
[9,6,7,12,0,0]
[4,15,14,1,0,0]
[0,0,0,0,0,0]
[0,0,0,0,0,0]
[0,0,0,0,0,34]]
```

Функције за генерисање матрица

QЛаб обезбеђује 5 основних функција за генерисање матрица као што су:
zeros која генерише матрицу са свим нулама, затим ones која генерише матрица са свим јединицама, eye која генерише јединичну матрицу и две функције за генерисање матрица случајних бројева који су униформно распоређени rand или нормално распоређени randn.
Све функције као аргумент прихватају димензију матрице коју је потребно генеристати.

Пример 9: *Генерисање матрице чији су елементи на главној дијагонали 7 а сви остали 2.*

```
>> B=5*eye(5)+2*ones(5)
[[7,2,2,2,2]
[2,7,2,2,2]
[2,2,7,2,2]
[2,2,2,7,2]
[2,2,2,2,7]]
```

Писање програма

Контрола тока

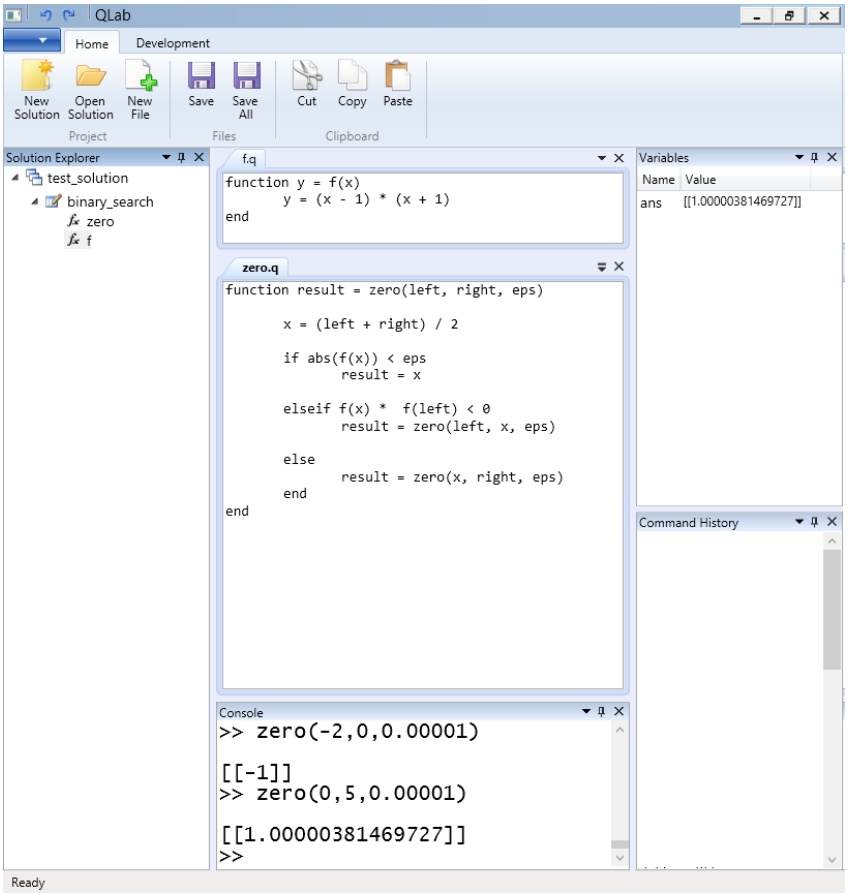
Решавање већине проблема често захтева извршавање одређеног дела програма у зависности од испуњености неког услова. Из тог разлога неопходно је постојање разгранатих структура.
QЛаб у тренутку писања овог текста подржава наредбу гранања – if, наредбу вишеструког гранања – switch и while петљу.

Наредба гранања if

Наредба условног гранања if омогућава извршавање одређеног дела програма у зависности од тога да ли је наведени услов испуњен или не. Уколико је услов испуњен, односно ако је вредност логичког исказа тачно (true), онда се извршава једна наредба или блок наредби наведених унутар команде if. Уколико је вредност логичког израза нетачно (false), може се извршити нека друга наредба, односно блок наредби.

Пример 10: *Метода половљења интервала за проналажење нуле функције на интервалу.*

```
function result = zero(left, right, eps)
  x = (left + right) / 2
  if abs(f(x)) < eps
            result = x
  elseif f(x) *  f(left) < 0
            result = zero(left, x, eps)
  else
            result = zero(x, right, eps)
  end
end
```



Слика 8 – Пример извршавања програма за проналажење нула функције

Switch наредба

Иако се вишеструко гранање може остварити коришћењем наредбе if, програм писан на такав начин је често веома непрегледан. Да би се то избегло, QЛаб подржава наредбу вишеструког гранања switch, која омогућава гранање програма у зависности од вредности неког израза.

Пример 11: *У зависности од параметра функције извршити одговарајуће множење матрица.*

```
function result = primer(p)
  x = [1 2 3; 4 5 6; 7 8 9]
  y = [9 8 7; 6 5 4; 3 2 1]
  z = [1 1 1; 2 2 2; 3 3 3]
  switch (p)
        case 0
              result = x * y;
        case 1
              result = x *  z;
        otherwise
              result = y * z;
  end
end
```

За разлику од switch наредбе у језику C нема пролажења кроз све case случајеве испод првог задовољеног. Уколико је први case услов задовољен, извршиће се само он а не и они који се налазе после њега, тако да break наредба није потребна.

While петља

Ова петља понавља блок наредби док год је неки услов испуњен.

Пример 12: *Функција за рачунање факторијела природног броја.*

```
function y = faktorijel(x)
  i=2;
  y=1;
  while(i<=x)
        y=y*i;
        i=i+1;
  end
end
```

Други природан начин за писање факторијел функције је употреба рекурзије која је такође подржана у QЛаб-у.

Пример 13: *Рекурзивна функција за рачунање факторијела природног броја.*

```
function y = faktorijel(x)
  if x == 1
        y = 1
  else
        y = x * faktorijel(x - 1)
  end
end
```

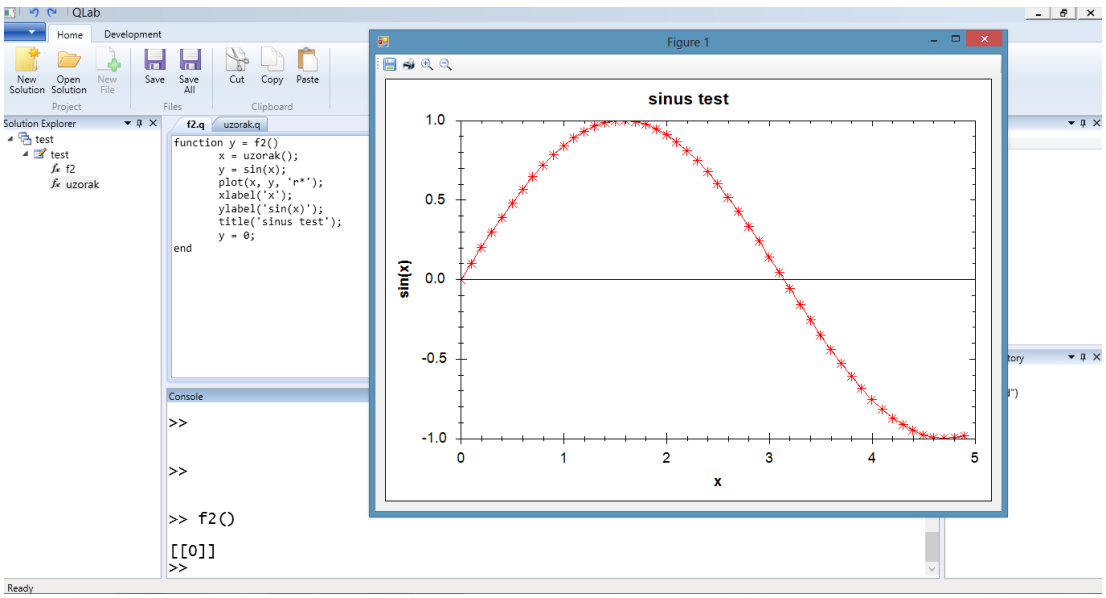
Исцртавање

Програмски пакет QЛаб пружа могућност исцртавања функција. Функција plot једна је од најчешће коришћених функција за графички приказ, има различите облике зависно од улазних аргумената.

Пример 14: *Исцртавање функције sin(x).*

```
function y=f2()
  x=uzorak();
  y=sin(x)
  plot(x,y,'r*');
  xlabel('x');
  ylabel('sin(x)');
  title('sinus test')
  y=0;
end
```

Након извршавања горе дефинисане фунцкије приказаће се исцртани график.



Слика 9 - Пример исцртавања

Подржане функције

Програмски пакет QЛаб омогућава коришћење следећих функција:

- plus
- minus
- uplus
- uminus
- times
- rdivide
- ldivide
- ctranspose
- transpose
- mtimes
- ge
- gt
- le
- lt
- eq
- and
- or
- not
- xor
- abs
- acos
- asin
- atan
- acosh
- asinh
- atanh
- cos
- sin
- tan
- cosh
- sinh
- tanh
- bitand
- bitor
- bitxor
- ceil
- floor
- round
- fix
- complex
- conj
- real
- imag
- exp
- log
- log2
- ischar
- isempty
- isfinite
- isnan
- isinf
- isfloat
- isinteger
- isletter
- isnumeric
- isreal
- isscalar
- isspace
- mod
- ndims
- nnz
- pow2
- power
- reallog
- realpow
- realsqrt
- rem
- sign
- sqrt
- isstr
- deblank
- findstr
- lower
- uper
- strrep
- strcmp
- strcmpi
- atan2
- hypot
- false
- true
- zeros
- ones
- eye
- isequal
- isequalwithequaln ans
- issorted
- cumsum
- cumprod
- all
- any
- size
- sort
- permute
- find
- reshape
- filter
- colon
- logical
- char
- int64
- uint64
- double
- length
- strncmp
- triu
- tril
- diag
- numel
- nzmax
- nonzeros
- isstrprop
- rand
- randi
- randn
- strncmpi
- setenv
- getenv
- regexp
- regexpi
- regexprep
- regextranslate
- transpose
- ctrnspose
- bitcmp

- bitget
- bitset
- bitshift
- Inf
- NaN
- isa
- isvarname
- isvector
- iscell
- isstruct
- figure
- hold
- clf
- axis
- legend
- grid
- xlabel
- ylabel
- title
- plot
- bar

Закључак

Свим студентима рачунарства и информатике, независно од тога на ком факултету студирају, пракса је изузетно потребна. Један од начина да се пракса реализује су управо пројекти које организују факултети. Од посебног значаја су пројекти развоја пакета отвореног кода, који ће осим самом факултету, бити корисни и целој академској заједници. Кроз такве пројекте студенти добијају преко потребну праксу, а факултет пројекат по коме постаје препознатљив. Сви чланови студентског тима сматрају да је рад на QЛаб-у једно велико искуство које му је употпунило редован ток студија и помогло у даљем развоју каријере.

Кроз претходне примере јасно је да је QЛаб достигао основни ниво употребљивости тако да се може преузети место МАТЛАБ софтверског пакета на основним курсевима. Такође је јасно да још увек није достигао потпуно функционалност као МАТЛАБ али уз помоћ студената могуће је достићи а чак и додати нове функционалности.